



北京大学

JAVA编程技能训练课程作业

项目名称：网络爬虫（技术类博客）

组名：第21组

组长：邱玉钦

组员：罗锦文

组员：杨宗睿

目录

1.	项目介绍.....	1
1.1	项目题目.....	1
1.2	项目概述.....	1
1.3	项目分工.....	1
1.4	项目优势.....	1
1.5	Git 代码访问方式.....	2
2.	系统设计.....	2
2.1	系统整体设计.....	2
2.2	系统的业务流程.....	3
2.3	所采用的技术.....	4
2.3.1	Nutch.....	4
2.3.2	Stanford Postagger.....	5
3.	系统模块划分和开发步骤.....	5
3.1	系统模块划分.....	5
3.2	系统开发步骤.....	6
4.	系统实现.....	6
4.1	NUTCH 框架搭建.....	6
4.1.1	Ubuntu 配置 JDK1.6.....	6
4.1.2	安装 Tomcat.....	8
4.1.3	配置 Nutch.....	10
4.2	数据抓取的实现.....	14
4.2.1	数据爬取.....	14
4.2.2	爬取结果处理.....	15
4.2.3	数据解析.....	17
4.3	数据处理与存储的实现.....	18
4.3.1	数据读取的实现.....	18
4.3.2	数据处理的实现.....	18

4.3.3 数据存储的实现.....	19
4.4 数据挖掘与分析的实现.....	20
4.4.1 词性标注的介绍.....	20
4.4.2 算法的选取与舍弃.....	21
4.4.3 算法思想的实现.....	21
4.5 展示数据分析结果的实现.....	22
5. 程序运行结果.....	28
5.1 系统初始化界面.....	28
5.2 输入关键词并点击查询.....	28
5.3 查询结果显示.....	29
6. 本项目的收获.....	31

1. 项目介绍

1.1 项目题目

基于 Nutch 爬虫框架的技术类博客数据抓取与分析

1.2 项目概述

爬取技术类博客，使用 nutch 爬虫框架，爬起技术类博客的标题、标签、主要内容。将技术类博客进行汇总分类，如 java, python, Hadoop, spark 等。java 目录又分为算法，多线程，分布式等。最后编写前台界面，将技术类博客的分类结果展示出来，形成一个爬虫结果的比例分析。

1.3 项目分工

邱玉钦（1501210972）——Nutch 爬虫框架的搭建，爬取数据。

罗锦文（1501210635）——数据预处理，分词，数据挖掘。

杨宗睿（1501211017）——界面实现，数据库搭建及数据存储，数据分类。

1.4 项目优势

Nutch 是一种开源网络搜索引擎，初建时用 Java 编写，可以对 HTML 等文档格式的文件进行抓取、存储、解析等一系列的页面细节处理。Nutch 最初由 Doug Cutting（Lucene 和 HaDoop 的构建者）和 Mike Cafarella 合作构建。Nutch 系统是一个完整的搜索引擎，利用 Nutch 经过若干步骤的简单设置就可以建立自己内部网的搜索引擎，也可以针对 Internet 建立网络搜索引擎。在通用搜索引擎基础上构建主题搜索引擎的方法有三种：一是控制信息采集更新的网站范围，将索引和检索信息限制在特定的几个主题网站之内；二是在通用搜索引擎信息采集的基础上进行文本分类或过滤，提取主题信息进行检索和索引；三是实现主题 Crawler 来控制信息的采集，仅仅采集，索引主题相关的信息。

目前，国内基于 Nutch 构建主题搜索引擎的方法也是以上三种。一是实现主题 Crawler，主要有两种方法：一种是对网络爬虫进行算法改造使其成为主题爬虫；另一种是制定主题词库帮助网络爬虫发现主题资源。二是建立主题词库进行主题过滤，主要有两种方法：一是在爬虫模块与文本解析模块之间制定主题词库对网页信息进行主题过滤；另一种方法是改进分词技术。国外基于 Nutch 构建主题搜索引擎中可上线访问的主题搜索引擎较多，如 Ask About Oil 是关于石油产业的主题搜索引擎。

国内采用主题爬虫和构建主题词库的研究成果较为丰富，但是在控制信息采

集范围方面，以往成果多是直接列出一到两个目标网站作为采集范围，并未进行系统的主题资源搜集。

在本项目中主要采取的实现策略是对爬虫抓取的数据进行主题过滤，即在建立特征词库的基础上实现新的分词策略，对文本数据进行分词，进行词性标注，提取特征词汇，根据提取的特征词对技术类博客进行分类。分词是进行语义分析的前提。我们知道，在英文的行文中，单词之间是以空格作为自然分界符的，而中文只是字、句和段能通过明显的分界符来简单划界，唯独词没有一个形式上的分界符，虽然英文也同样存在短语的划分问题，不过在词这一层上，中文比之英文要复杂的多、困难的多。本文的优势是采用基于上下文无关文法的句法分析。本文采用的训练集是宾夕法尼亚大学的树库 **treebank** 作为语料库，对句子进行切分，得到了较好的结果。

1.5 Git 代码访问方式

<https://github.com/QYqin/PkuJava21/>

2. 系统设计

2.1 系统整体设计

本次项目主要是抓数据、数据处理、数据分析和挖掘、展示分析和挖掘的成果。根据这些需求确定出了系统的整体设计图和整体功能结构图。

本次项目主要是搭建 **Nutch** 爬虫框架、抓取数据、数据处理、数据分析和挖掘、展示分析和挖掘的成果。根据这些需求确定处理系统的整体设计图和整体功能结构图。

系统的整体设计图如图 2.1:



图 2.1 整体设计图

2.2 系统的业务流程

在上面需求分析和系统整体设计的基础上，需要对系统进行详细设计。详细设计阶段首先要确定出系统的业务流程图。系统的业务流程图如图 2.2 所示：

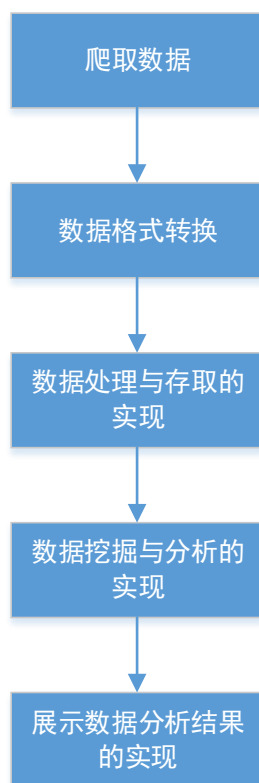


图 2.2 系统的业务流程图

2.3 所采用的技术

2.3.1 Nutch

Nutch 是一个开源 Java 实现的搜索引擎。它提供了我们运行自己的搜索引擎所需的全部工具。包括全文搜索和 Web 爬虫。

使用 nutch 的原因：

(1) 透明度：Nutch 是开放源代码的，因此任何人都可以查看他的排序算法是如何工作的。

(2) 扩展性：Nutch 是非常灵活的：他可以被很好的客户订制并集成到你的应用程序中，使用 Nutch 的插件机制，Nutch 可以作为一个搜索不同信息载体的搜索平台。

在 Nutch 中，Crawler 操作的实现是通过一系列子操作的实现来完成的。这些子操作 Nutch 都提供了子命令行可以单独进行调用。下面就是这些子操作的功能描述以及命令行，命令行在括号中。

1. 创建一个新的 WebDb (admin db -create).
2. 将抓取起始 URLs 写入 WebDB 中 (inject).

3. 根据 WebDB 生成 fetchlist 并写入相应的 segment(generate).
4. 根据 fetchlist 中的 URL 抓取网页 (fetch).
5. 根据抓取网页更新 WebDb (updatedb).
6. 循环进行 3—5 步直至预先设定的抓取深度。
7. 根据 WebDB 得到的网页评分和 links 更新 segments (updatesegs).
8. 对所抓取的网页进行索引(index).
9. 在索引中丢弃有重复内容的网页和重复的 URLs (dedup).
10. 将 segments 中的索引进行合并生成用于检索的最终 index(merge).

2.3.2 Stanford Postagger

Stanford Postagger, 由斯坦福大学自然语言处理小组开发的词性标注工具, 最新发布于 2006 年, 该词性标注工具主要分为标注、测试和训练三个模式。本文所用到其主要功能为对文本中的单个词汇进行词性判断并在每个单词后进行词性标注, 例如形容词 (/JJ)、副词 (/RB)、动词 (/VB) 等词性, 以方便开发人员对相关词汇进行筛选, 减小数据处理时间, 提高工作效率。该开源词性标注词典对英文单词的标注准确率达到 97%。除此之外, 该词性标注工具还可对中文、德文以及法文等进行词性标注。

3. 系统模块划分和开发步骤

3.1 系统模块划分

本项目分为四个模块:

1. 抓取数据

用 Nutch 框架爬取数据, 将 CSDN 技术类博客作为主页, 设置爬去深度、并发进程数量、每层抓取的最大页面数、存放目录等。将爬取的技术类博客内容存放至指定文件。

2. 爬虫数据处理和存储

把获取到的数据通过除杂和统一格式等处理成能直接使用的格式并存到文本文件。抓取的数据主要供 Tomncat 显示, 通过搭建的 Apache 服务器, 通过网页访问, 利用 Nutch 框架的集成 readseg 功能, 将抓取的数据转换为可读的文本格式。

3. 数据处理

使用 stanford.nlp.tagger 和 stanford.nlp.ling 包对文本内容进行分词和词性标

注处理，过滤掉非关键信息，再通过正则表达式匹配文本内容，得到关键词和网址对

4.数据处理结果展示

将从爬虫数据中提取的 URL 及标签按条存入数据库中，并设计开发前端界面。在界面中获取用户希望获取的博客类型的关键字，当用户点击 Search 按钮后进行数据库关键字查询，并将查询结果（网页的 URL）返回界面显示

3.2 系统开发步骤

在上面划分模块的基础上，按着各个模块的逻辑关系和每个模块内部的依赖关系，确定的开发步骤如下：

1. 搭建 Linux 的 java 环境，安装 JDK
2. 安装 Tomnact，配置 Apache 服务器，用于显示爬去的网页，并能够按照爬虫框架预先设定好的索引进行分类。
3. 配置和应用 Nutch，安装 Nutch，配置爬虫相关参数，并将爬去的内容存入指定文件夹。
4. 将爬去的内容通过 Tomnact 验证是否成功，并将爬去内容格式转化为文本文件。
5. 先通过 stanford 自然语言工具包，过滤文本信息，再通过正则表达式匹配关键词，形成 URL 和标签对。
6. 前段界面开发，使用 Swing 编程，开发前端界面，设置鼠标点击的监听事件，并使用 JDBC 连接 MySQL 数据库，使用 Insert 命令执行插入操作，并使用 Select 命令执行查询。

4. 系统实现

本次项目主要包括四个模块：Nutch 框架搭建、数据爬取、数据存储与处理、数据分析与挖掘和展示数据分析和挖掘的结果。接下来按着模块来介绍系统的详细实现过程。

4.1 Nutch 框架搭建

4.1.1 Ubuntu 配置 JDK1.6

1. 点击 Download，到官网下载 Linux 版本的 JDK。选择自己对应的操作系统及 32 或 64 位版本，这里我下载的是 64 位版本的 jdk-6u65-linux-x64.bin。

Java SE Runtime Environment 6u45		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.		
Product / File Description	File Size	Download
Linux x86	20.24 MB	jre-6u45-linux-i586-rpm.bin
Linux x86	20.76 MB	jre-6u45-linux-i586.bin
Linux x64	19.82 MB	jre-6u45-linux-x64-rpm.bin
Linux x64	20.39 MB	jre-6u45-linux-x64.bin
Solaris x86	20.4 MB	jre-6u45-solaris-i586.sh
Solaris x64	7.54 MB	jre-6u45-solaris-x64.sh

图 3.1 JDK 下载

2. 创建一个目录/usr/java/以便于把下载的安装包放到这个目录下。

```
root@ubuntu: /usr/java
root@ubuntu:/usr/java# mkdir /usr/java
```

图 3.2 建立 JDK 安装目录

3. # chmod +x jdk-6u65-linux-x64.bin (获得执行权限)。
4. # ./jdk-6u65-linux-x64.bin (生成 rpm 安装包)。
5. # rpm -ivh jdk-6u65-linux-x64.rpm (安装 JDK)。
6. 安装完毕后, JDK 默认安装在/usr/java/目录下。
7. 配置 JAVA 环境变量。在/etc/profile 中设置环境变量

```
# gedit /etc/profile
```

```
root@ubuntu: /home/qyq
qyq@ubuntu:~$ su
密码:
root@ubuntu:/home/qyq# gedit /etc/profile
```

图 3.3 打开配置文件

8. 添加环境变量

```
profile (/etc) - gedit
if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
unset i
fi
export JAVA_HOME=/usr/java/jdk1.6.0_45
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
```

图 3.4 添加环境变量

9. # chmod +x /etc/profile (执行权限)。

10. # source /etc/profile (此后设置有效)。

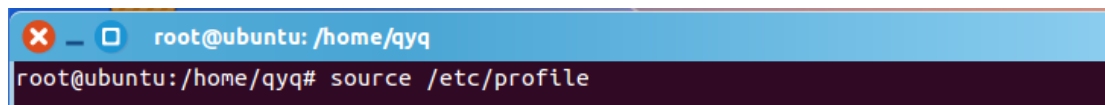


图 3.5 更新 profile 文件

11. 测试 JDK 是否安装成功

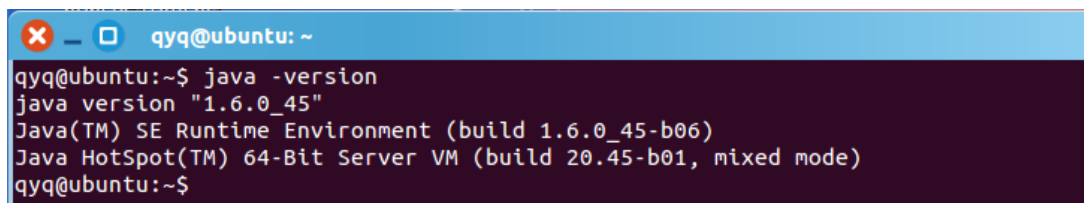


图 3.6 测试 java -version

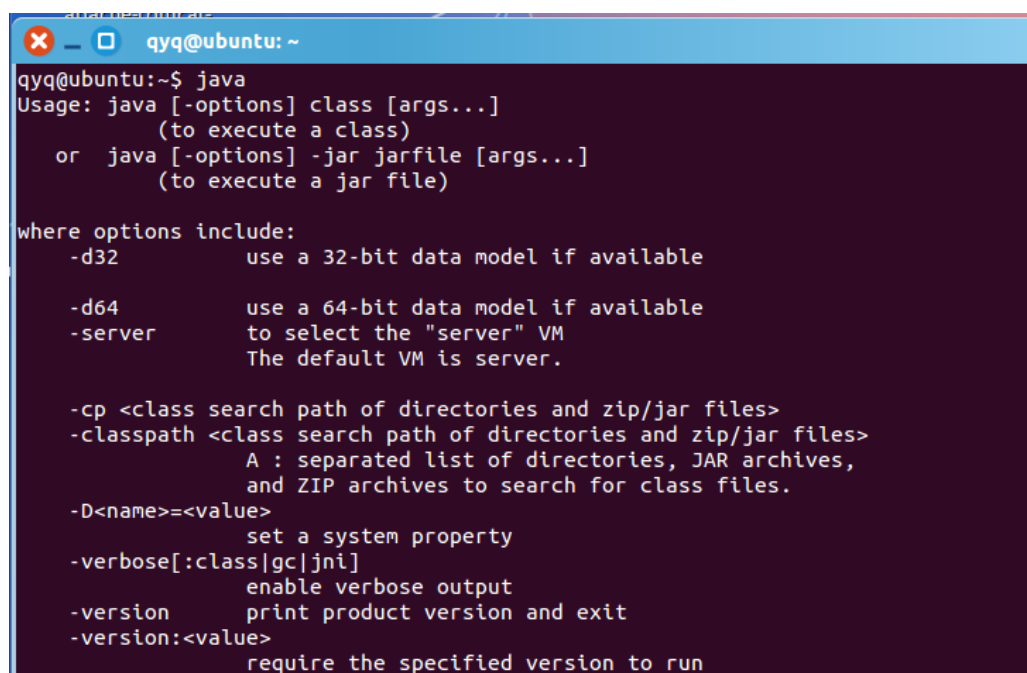


图 3.7 测试 java 命令

JDK 安装成功。

4.1.2 安装 Tomcat

1. 设置环境变量

root@ubuntu:/home/qyq# gedit /etc/profile



图 3.8 配置环境变量

```
root@ubuntu:/home/qyq# source /etc/profile
```

2. 安装 Tomcat，解压到主文件夹下

```
tar xf apache-tomcat-6.0.18.tar.gz
```

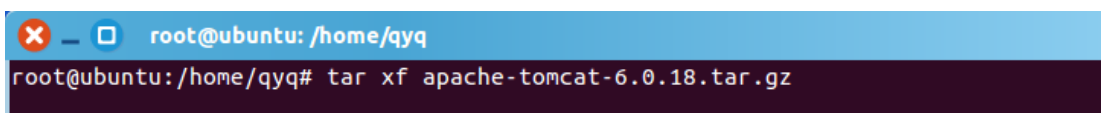


图 3.9 解压 Tomcat 到主文件夹

3. 启动 Tomcat

```
#cd /home/qyq/apache-tomcat-6.0.44
```

```
#./bin/startup.sh
```

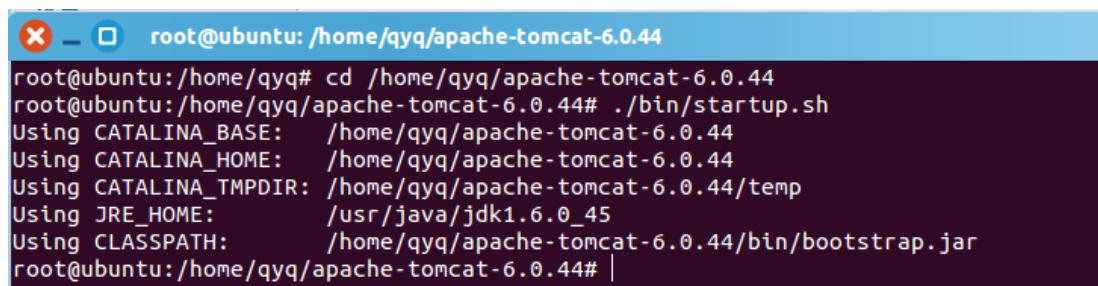


图 3.10 启动 Tomcat

- Tomcat 的网页主目录是 `/home/qyq/apache-tomcat-6.0.44/webapps/`，只需在 `webapps` 目录中添加相应网页即可在浏览器访问，Tomcat 默认目录是 `webapps` 下的 `ROOT` 目录。
- `http://localhost:8080` 访问 tomcat 默认主目录，`ROOT`。Apache http 服务器的端口是 8080，`http://localhost:8080` 访问的是 Apache 主目录。Apache Tomcat 服务器端口是 8080，二者不冲突，若有冲突，则可以修改 tomcat 配置文件 `server.xml`。

```
#gedit /home/qyq/apache-tomcat-6.0.44/conf/server.xml
```

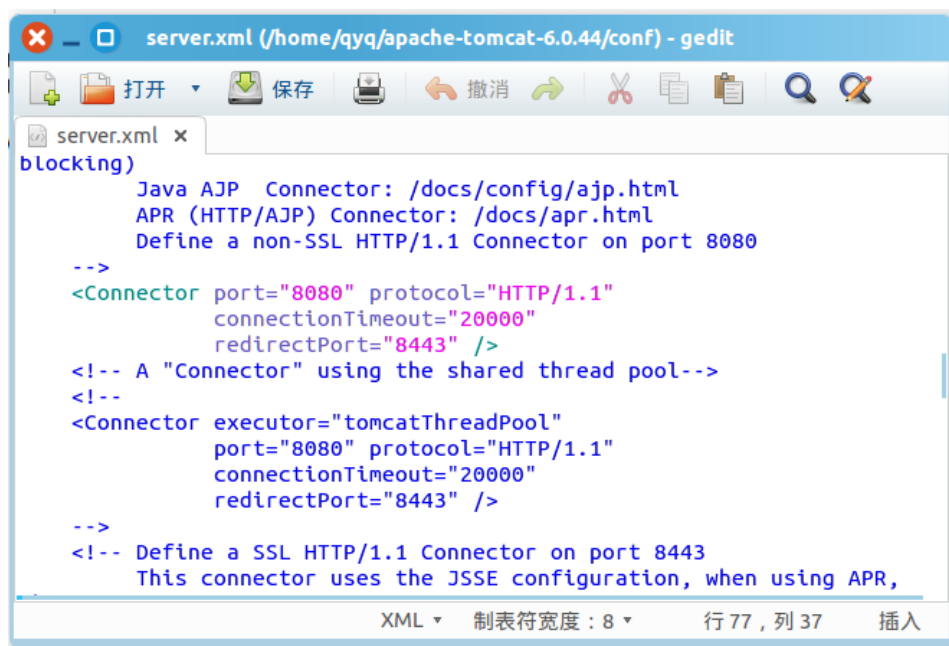


图 3.11 端口号配置

6. 测试 Apache Tomcat 是否启动成功。

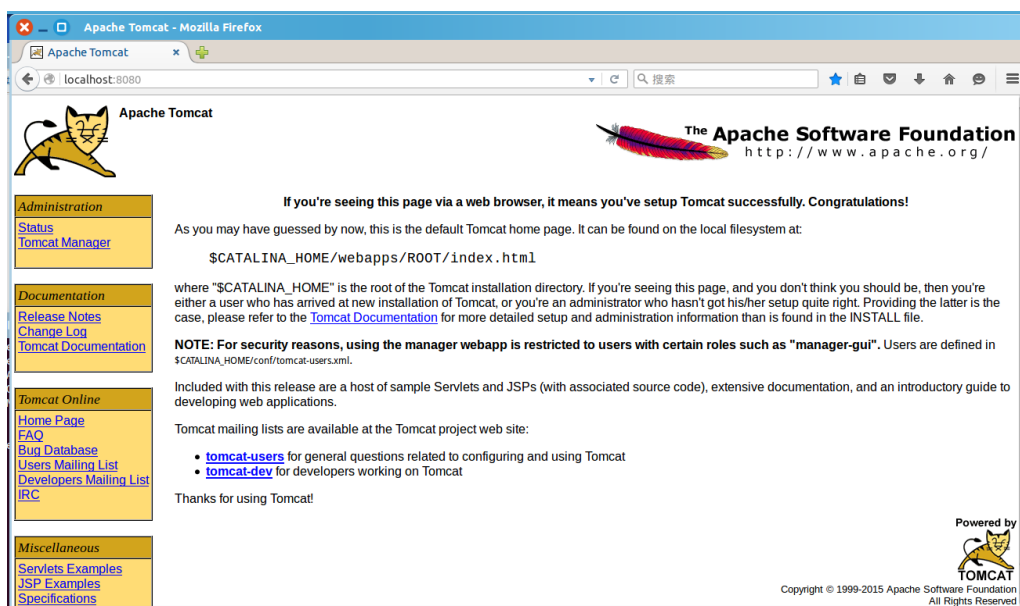


图 3.12 测试 Tomcat 启动成功

4.1.3 配置 Nutch

1. 下载 apache-nutch-1.2-bin.tar.gz 至/home/qyq 文件夹下。
2. 解压 nutch-1.2

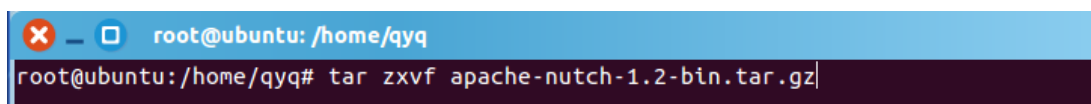


图 3.13 解压 nutch1.2

3. 测试 Nutch

```
#./home/qqq/nutch-1.2/bin/nutch
```

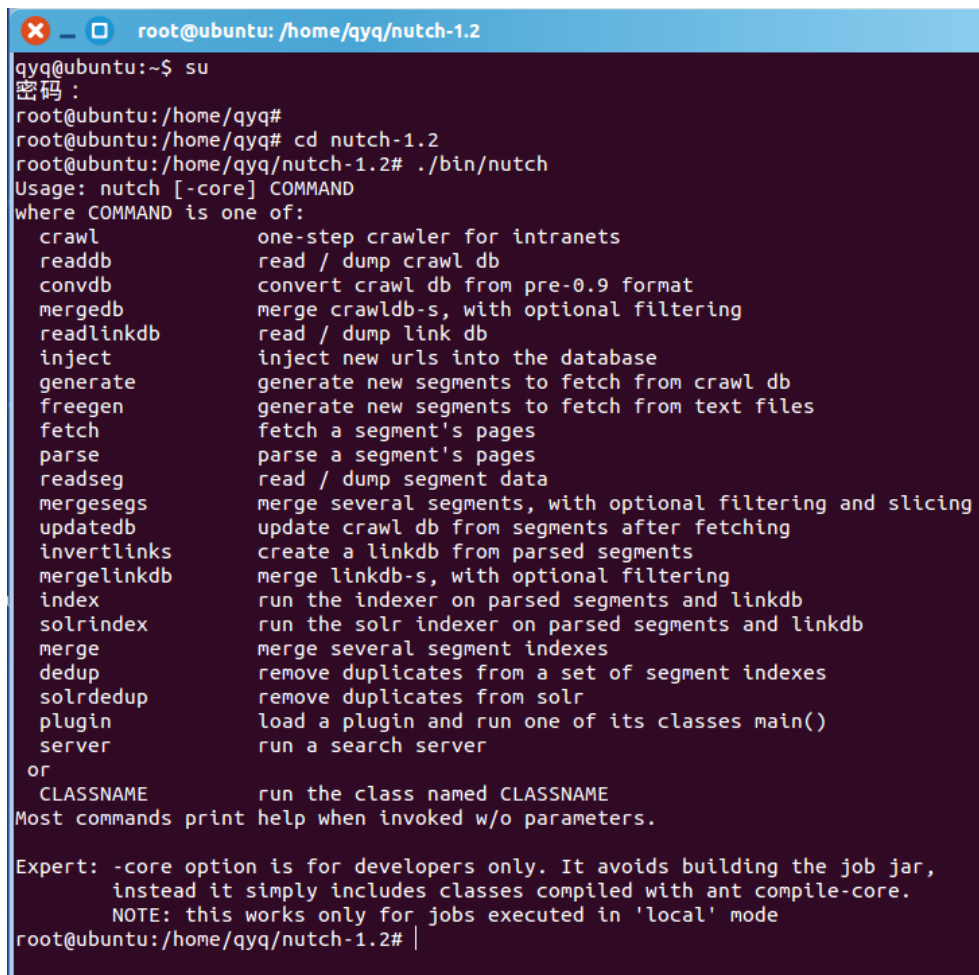


图 3.14 测试 nutch 启动成功

4. 设置 Nutch 抓取网站的入口网址

```
# cd /home/qqq/nutch-1.2/
```

```
# mkdir urls
```

```
# gedit urls/urls_crawl.txt
```

或者不用创建目录，直接创建一个文件 urls_crawl.txt，我们采用此法，

```
# gedit urls_crawl.txt
```

写入要抓取(crawl)网站的入口网址，即从此入口开始抓取当前域名下的任何 URL 页面，例如：

```
http://blog.csdn.net/column.html/
```

<http://blog.csdn.net/code/column.html>

5. 指定爬取过滤规则

编辑 nutch 的 URL 过滤规则文件 conf/crawl-urlfilter.txt

```
[root@red-hat-9 nutch-0.9]# vi conf/crawl-urlfilter.txt
```

修改

```
# accept hosts in MY.DOMAIN.NAME
```

```
# +^http://([a-z0-9]*.)*MY.DOMAIN.NAME/
```

这是你想要爬取网站的域名，表示爬取当前网站下的所有 URL 页面，如果爬取网站的 url 含有以下过滤字符，如?和=，而你又需要这些访问，可以更改过滤表

```
# skip URLs containing certain characters as probable queries, etc.
```

```
-[?!@=]
```

改为-[*!@]

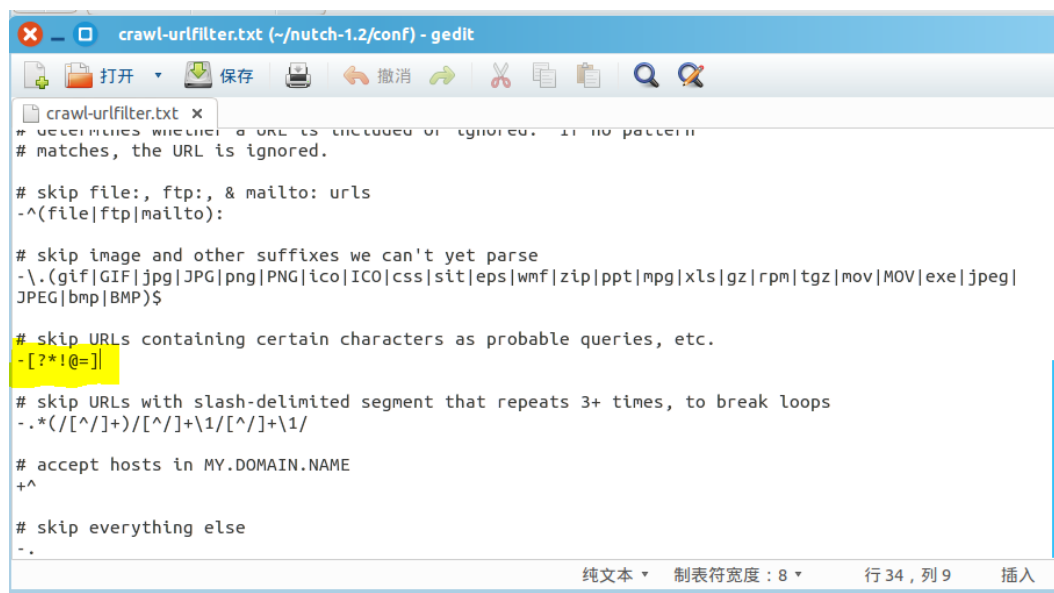


图 3.15 配置过滤规则

6. 修改 conf/nutch-site.xml

修改为

```
<configuration>
```

```
  <property>
```

```
    <name>http.agent.name</name> http.agent.name 属性
```

```
    <value>gucas.ac.cn</value>被抓取网站的名称，当抓取多个网站时，用“*”
```

```
  </property>
```

```
</property>
```

```
  <name>http.agent.version</name>
```

```
  <value>1.0</value>
```

```
</property>
```

```

<property>
  <name>searcher.dir</name>
  <value>/home/qqq/nutch-1.2/gucas</value>
  <description> Path to root of crawl</description>
</property>
</configuration>

```

如果没有配置此 agent，爬取时会出现 Agent name not configured! 的错误。

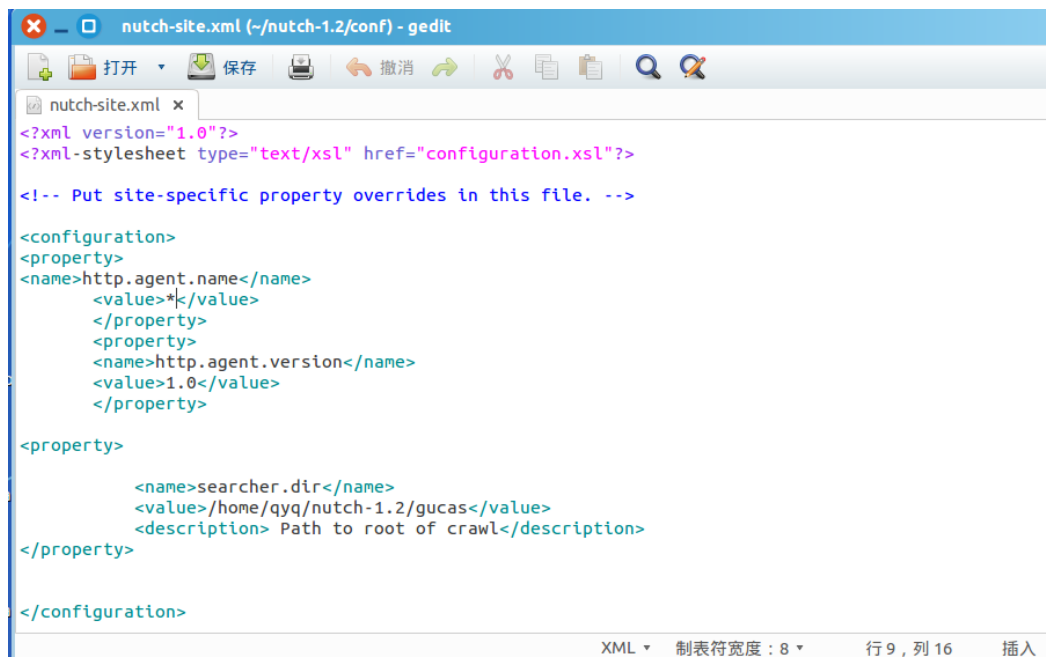


图 3.16 配置 conf/nutch-site.xml

7. 部署 web 前端

将 nutch 主目录下的 nutch-1.2.war 包拷贝到 tomcat 的 webapps 目录下

```
# cp nutch-1.2.war /home/qqq/apache-tomcat-6.0.44/webapps/
```

然后浏览器网址 <http://localhost:8080/nutch-1.2/>。此时 war 包会自动解压，在 tomcat 的网页主目录 webapps 下会出现一个 nutch-1.2 文件夹。



图 3.17 测试 web 前端界面

8. 修改 tomcat 中 nutch 的 web 配置

```
# gedit /home/qqq/apache-tomcat-6.0.44/webapps/nutch-1.2/WEB-INF/classes/nutch-site.xml
```

将 searcher.dir 属性值更改为索引生成的目录。

```
<configuration>
<property>
  <name>searcher.dir</name>
  <value>/home/qqq/nutch-1.2/gucas</value>gucas 为爬虫爬取后的文件
</property>
```

4.2 数据抓取的实现

4.2.1 数据爬取

运行 crawl 命令抓取网站内容

```
#bin/nutch crawl multiurls.txt -dir gucas -depth 50 -threads 5 -topN 5
```

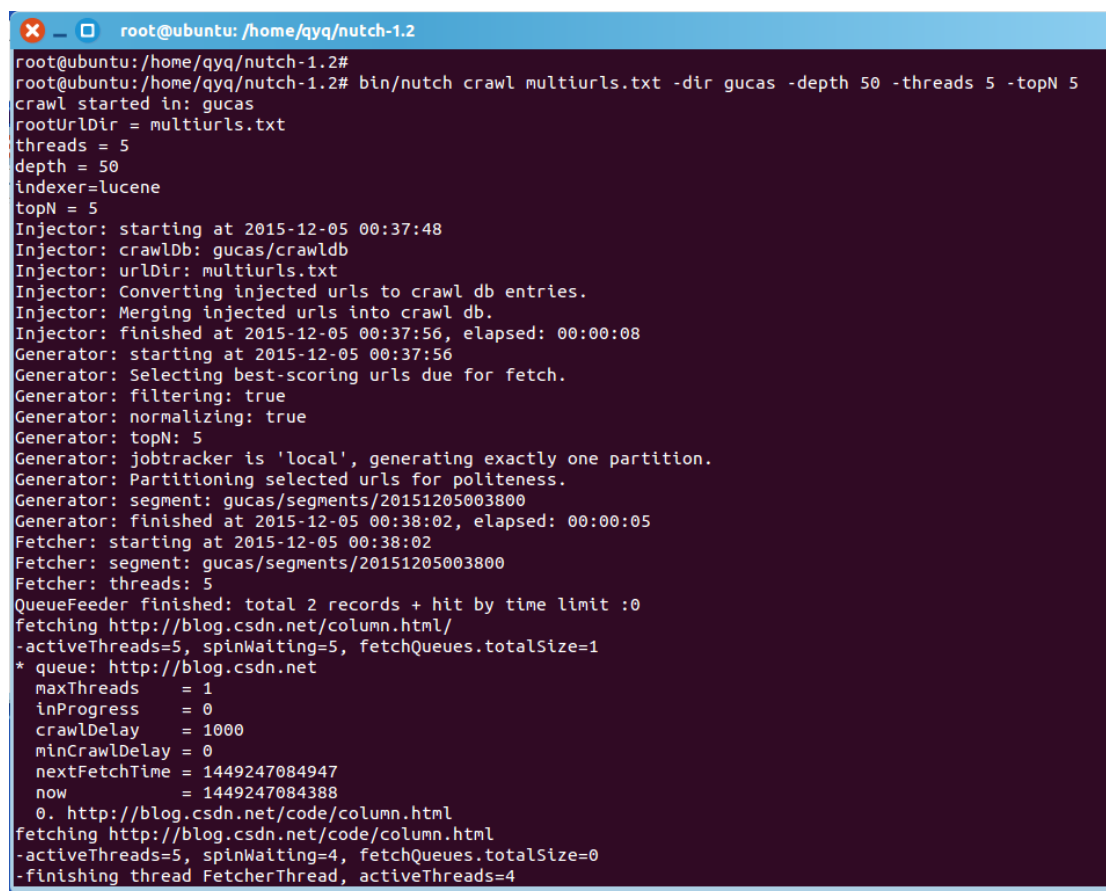
表 4.1 参数说明

-dir	dirname	设置保存所抓取网页的目录
-depth	depth	表明抓取网页的层次深度
-delay	delay	表明访问不同主机的延时，单位为“秒”
-threads	threads	表明需要启动的线程数
-topN	1000	表明只抓取每一层的前 N 个 URL

在上述命令的参数中，urls_crawl.txt 为刚才存储抓取网址的文件 urls_crawl.txt；dir 指定抓取内容所存放的目录，此处为 gucas；depth 表示以抓取

网站顶级网址为起点的爬行深度；threads 指定并发的线程数；topN 表明只抓取每一层的前 N 个 URL。

爬取过程如图 3.18 所示。

A terminal window titled 'root@ubuntu: /home/qyq/nutch-1.2' showing the execution of the Nutch crawler. The command is 'bin/nutch crawl multiurls.txt -dir gucas -depth 50 -threads 5 -topN 5'. The output shows the crawler starting, injecting URLs, generating segments, and fetching data from 'http://blog.csdn.net/column.html'. The terminal text is as follows:

```
root@ubuntu: /home/qyq/nutch-1.2#
root@ubuntu: /home/qyq/nutch-1.2# bin/nutch crawl multiurls.txt -dir gucas -depth 50 -threads 5 -topN 5
crawl started in: gucas
rootUrlDir = multiurls.txt
threads = 5
depth = 50
indexer=lucene
topN = 5
Injector: starting at 2015-12-05 00:37:48
Injector: crawlDb: gucas/crawldb
Injector: urlDir: multiurls.txt
Injector: Converting injected urls to crawl db entries.
Injector: Merging injected urls into crawl db.
Injector: finished at 2015-12-05 00:37:56, elapsed: 00:00:08
Generator: starting at 2015-12-05 00:37:56
Generator: Selecting best-scoring urls due for fetch.
Generator: filtering: true
Generator: normalizing: true
Generator: topN: 5
Generator: jobtracker is 'local', generating exactly one partition.
Generator: Partitioning selected urls for politeness.
Generator: segment: gucas/segments/20151205003800
Generator: finished at 2015-12-05 00:38:02, elapsed: 00:00:05
Fetcher: starting at 2015-12-05 00:38:02
Fetcher: segment: gucas/segments/20151205003800
Fetcher: threads: 5
QueueFeeder finished: total 2 records + hit by time limit :0
fetching http://blog.csdn.net/column.html/
-activeThreads=5, spinWaiting=5, fetchQueues.totalSize=1
* queue: http://blog.csdn.net
  maxThreads = 1
  inProgress = 0
  crawlDelay = 1000
  minCrawlDelay = 0
  nextFetchTime = 1449247084947
  now = 1449247084388
  0. http://blog.csdn.net/code/column.html
fetching http://blog.csdn.net/code/column.html
-activeThreads=5, spinWaiting=4, fetchQueues.totalSize=0
-finishing thread FetcherThread, activeThreads=4
```

图 3.18 运行爬虫指令

4.2.2 爬取结果处理

如图 3.19 所示，为爬取后 gucas 中的文件。

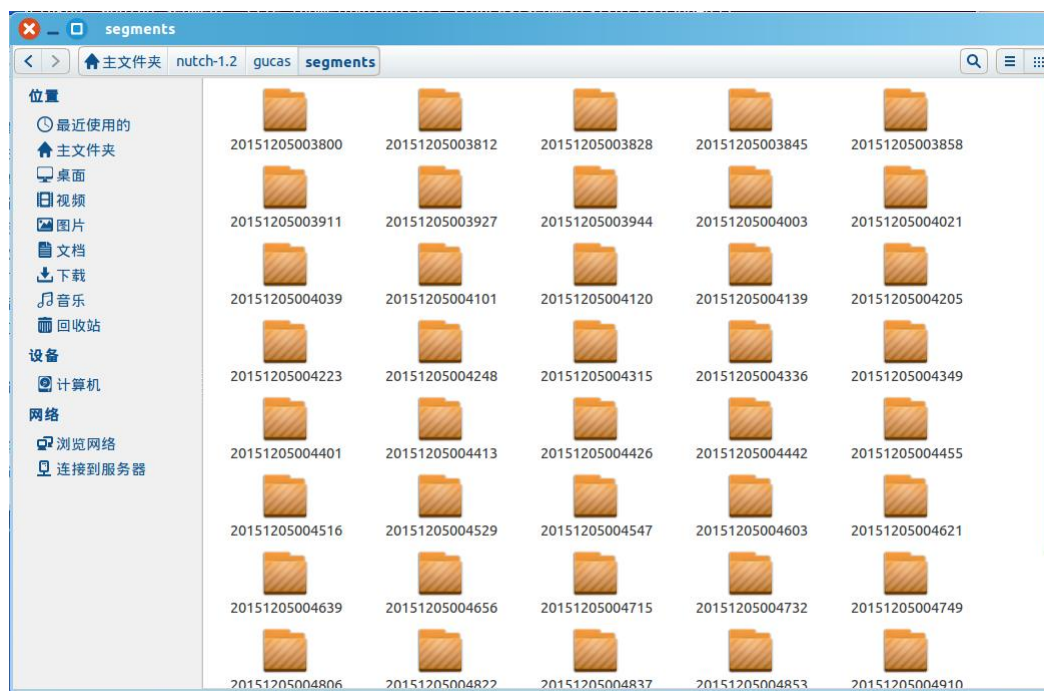


图 3.19 爬取结果

选取其中任意数据的 parse-text 文件。

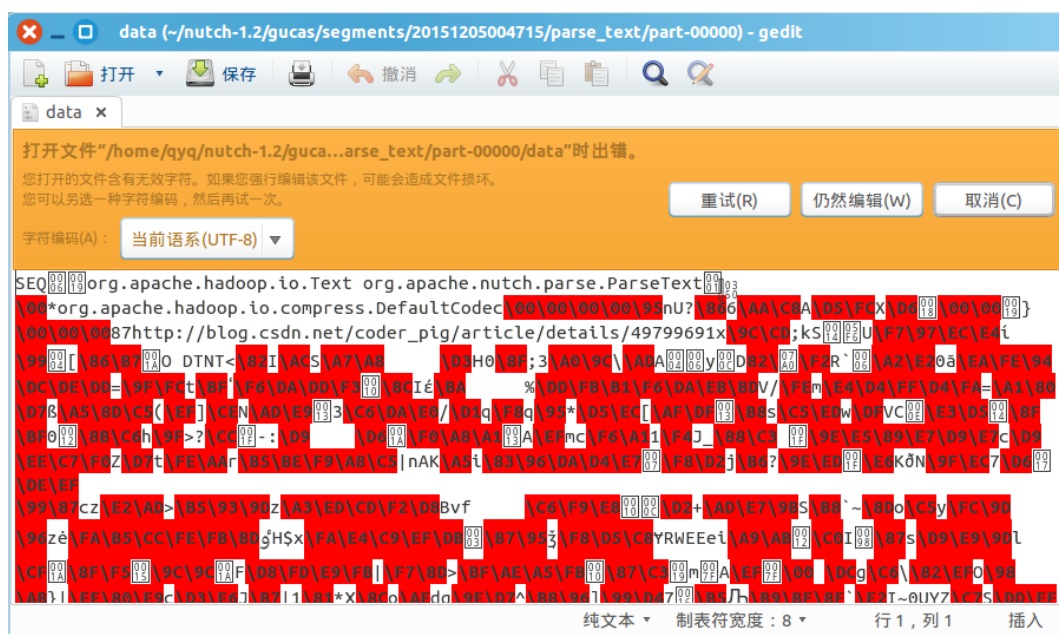


图 3.20 爬取结果

可以看到爬取结果不可读，但是根据 nutch 生成的索引文件，可以通过 web 界面访问。如图 3.21 所示为输入 java 后显示的爬取数据分类结果。

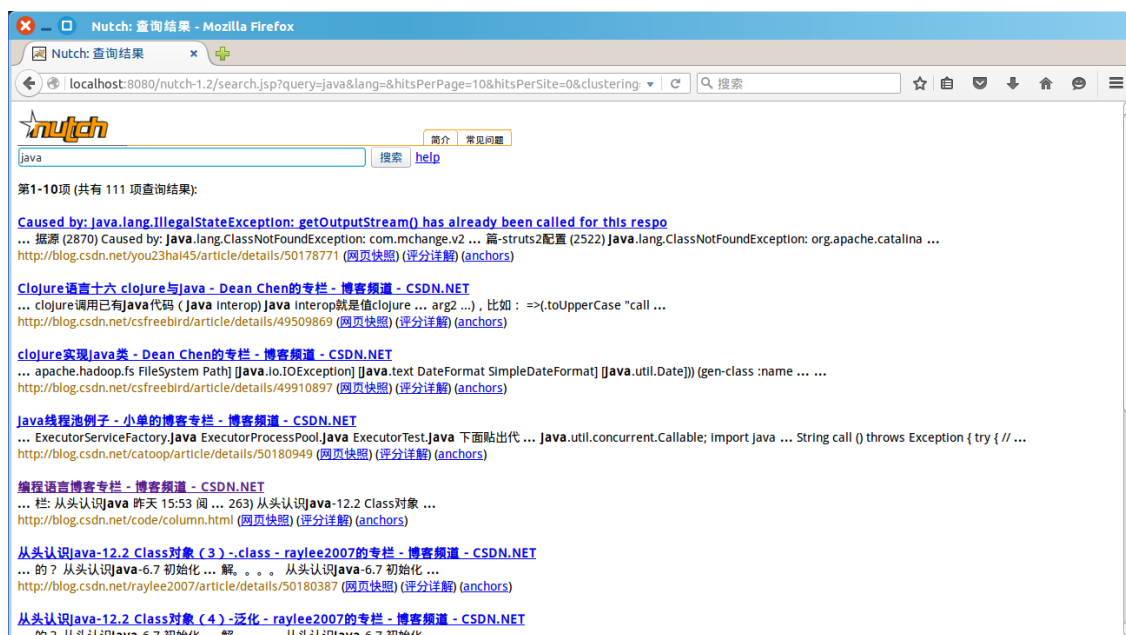


图 3.20 爬取结果 web 索引

4.2.3 数据解析

输入命令，使用 nutch 函数将爬取的数据转换为文本文件。

```
# bin/nutch readseg -dump gucas/segments/20151205004853 segdb -nocontent -nofetch -nogenerate -noparse -noparsedata
```

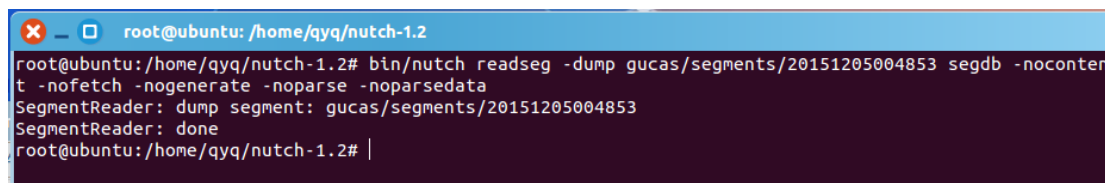


图 3.21 数据解析

由图 3.21 可知，数据解析的文件存入 segdb 中的 dump 文件中，解析结果如图 3.22 所示。可以看到，所有爬取的网页内容已经按照 URL 进行分类，条目清晰。

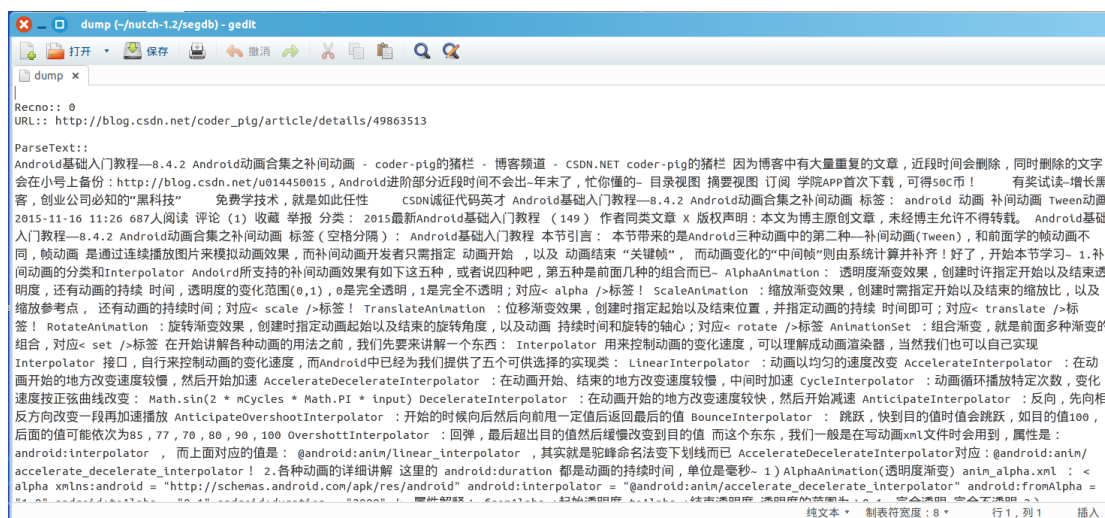


图 3.22 解析数据文件

4.3 数据处理与存储的实现

4.3.1 数据读取的实现

表 4.2 数据读取

```
String encoding = "utf-8";
String path = "spider";
File file = new File(path); // 从文件中读取所有文件
File[] tempList = file.listFiles(); // 将文件名称存储成一个链表
System.out.println("该目录下对象个数: " + tempList.length);
ArrayList<String> output = new ArrayList<String>();
for (int i = 0; i < tempList.length; i++) {
    if (tempList[i].isFile() && tempList[i].exists()) {
        // 如果文件存在，这通过 InputStream 方法读取文件内容
        InputStreamReader read = new InputStreamReader(
            new FileInputStream(tempList[i]), encoding);
        BufferedReader bufferedReader = new BufferedReader(read);
```

如表 4.1 所示将 Nutch 爬出来的文件放入同一个文件夹中，通过如上按行命令读取文件内容。下面的分析算法都是建立在此基础之上的。

4.3.2 数据处理的实现

数据处理方面我们使用了斯坦福大学的自然语言处理工具包

```

▷ jsoup-1.7.1.jar - C:\Users\LuoJw\Documents\nlp\jaxen-1.0-FCS
▷ stanford-segmenter-3.5.2.jar - C:\Users\LuoJw\Documents\nlp\stanford-segmenter-2015-04-20
▷ stanford-corenlp-3.5.2.jar - C:\Users\LuoJw\Documents\nlp\stanford-corenlp-full-2015-04-20
▷ stanford-corenlp-3.5.2-sources.jar - C:\Users\LuoJw\Documents\nlp\stanford-corenlp-full-2015-04-20
▷ stanford-postagger-3.5.2.jar - C:\Users\LuoJw\Documents\nlp\stanford-postagger-full-2015-04-20
▷ stanford-postagger.jar - C:\Users\LuoJw\Documents\nlp\stanford-postagger-full-2015-04-20
▷ stanford-postagger-3.5.2-javadoc.jar - C:\Users\LuoJw\Documents\nlp\stanford-postagger-full-2015-04-20
▷ stanford-postagger-3.5.2-sources.jar - C:\Users\LuoJw\Documents\nlp\stanford-postagger-full-2015-04-20

```

图 4.23 分词工具

如图 4.23 所示作为分词的工具

通过正则表达式得到网站的标签信息，大致定义网站的分类方向，出入本地磁盘，在对文章的内容进行分词，分词是进行语义分析的前提，我们知道，在英文的行文中，单词之间是以空格作为自然分界符的，而中文只是字、句和段能通过明显的分界符来简单划界，唯独词没有一个形式上的分界符，虽然英文也同样存在短语的划分问题，不过在词这一层上，中文比之英文要复杂的多、困难的多。所以这一步是语义分析的重点，我们采用的训练集是宾夕法尼亚大学的树库 treebank 作为语料库，对句子进行切分，得到了较好的结果。

```

public static void main(String[] args) throws Exception {
    Properties props = new Properties();
    props.setProperty("sighanCorporaDict", "data");
    props.setProperty("serDictionary", "data/dict-chris6.ser.gz");
    props.setProperty("inputEncoding", "UTF-8");
    props.setProperty("sighanPostProcessing", "true");
    CRFClassifier classifier = new CRFClassifier(props);
    classifier.loadClassifierNoExceptions("data/ctb.gz", props);
    classifier.flags.setProperties(props);

    String sentence = "登录注册收藏成功确定收藏失败请重新收藏确定标题标题不能为空网址标签摘要公开";
    String ret = doSegment(sentence, classifier);
    System.out.println(ret);
}

```

图 4.24 分词实现

分词结果为如下图 4.25:

```

Done. Unique words in ChineseDictionary is: 423200.
done [7.8 sec].
serDictionary=data/dict-chris6.ser.gz
sighanCorporaDict=data
inputEncoding=UTF-8
sighanPostProcessing=true
INFO: TagAffixDetector: useChPos=false | useCTBChar2=true | usePKChar2=false
INFO: TagAffixDetector: building TagAffixDetector from data/dict/character_list and data/dict/in.ctb
Loading character dictionary file from data/dict/character_list
Loading affix dictionary from data/dict/in.ctb
登录注册收藏成功确定收藏失败请重新收藏确定标题标题不能为空网址标签摘要公开取消收藏分享资讯传PPT文档提问写博客传
资源创建项目创建代码片设置昵称编辑自我介绍让更多人了解你帐号设置退出

```

图 4.25 分词结果

虽然结果中仍有少量的结果不太理想，但是不会影响最终结果。

4.3.3 数据存储的实现

由 4.2.3 结果可知，Nutch 将爬去的数据文件进行解析后存储到了存入 segdb 中的 dump 文件中，在进行数据处理操作后，将数据按照 url、tabword 的顺序按

条存入 `ArrayList<>` 中，本文将采用 MySQL 数据库存储 `ArrayList<>` 中的数据，数据库表 `text` 设计如下所示：

表 4.3 text 表设计

字段名	字段类型	字段长度	备注
id	int	32	Primary key
url	varchar	255	Not null
tabword	varchar	255	Not null

JDBC 连接数据库代码如下图所示：

```
//jdbc链接数据库
Class.forName("com.mysql.jdbc.Driver");
String url = "jdbc:mysql://localhost:3306/nlp?user=root&password=123456";
Connection cn = (Connection) DriverManager.getConnection(url);
```

图 4.26 JDBC 连接数据库

在成功连接数据库后，将存放在数组链表中的数据按条存入数据库的 `text` 表中，代码如下图所示：

```
// 执行插入
PreparedStatement ps = (PreparedStatement) cn
    .prepareStatement("insert into text (url,tabword) values (?,?)");
for (int i = 1; i <= output.size(); i = i + 2) {
    ps.setString(1, output.get(i - 1));
    ps.setString(2, output.get(i));
    ps.executeUpdate();
}
```

图 4.27 执行插入操作

在上述代码中，下标 `i` 为偶数的位置存放的是博客的标签，其前一个位置存放的是该博客的 URL。

数据库表 `text` 如下图所示：

id	url	tabword
1	http://blog.csdn.net/bdmh/article/details/39295795	Android
2	http://blog.csdn.net/bdmh/article/details/40425213	Android
3	http://blog.csdn.net/bdmh/article/details/41891823	Android
4	http://blog.csdn.net/bdmh/article/details/49927787	Java
5	http://blog.csdn.net/bdmh/article/details/49928197	Java

图 4.28 插入数据结果

4.4 数据挖掘与分析的实现

4.4.1 词性标注的介绍

现阶段的主要内容是对词性进行标注。所谓词性标注就是根据句子的上下文信息给句中的每个词确定一个最为合适的词性标记。词性标注的难点主要是由词

性兼类所引起的。词性兼类是指自然语言中一个词语的词性多余一个的语言现象。词性兼类是自然语言中的普遍现象，例如下面的句子：S1=“他是北京大学的教授。”；S2=“他在北京大学教授计算语言学。”句子 S1 中，“教授”是一个表示职称的名词，而句子 S2 中“教授”是一个动词。词性兼类的消歧常采用概率的方法，如隐马尔科夫模型。

因此在这里我们使用了斯坦福的语义标注包去实现这个功能，实现代码如下
图 4.29

```
import java.io.StringReader;
/*
 *
 * 对原文本做词性标注.
 * 输入格式为: The list of prisoners who may be
 * 得到的结果格式为: The/DT list/NN of/IN prisoners/NNS who/WP may/MD be/VB
 */
public class SeparateWord {
    public String SeparateWords(String line) {
        String str = line; // StringBuffer buf = new StringBuffer();
        String result = "";
        String str =
        //
        // "The list of prisoners who may be released in coming days includes militants"
        //
        // +
        // " who threw firebombs, in one case at a bus carrying children; stabbed and shot"
        //
        // +
        // " civilians, including women, elderly Jews and suspected Palestinian collaborators; "
        //
        // +
        // "and ambushed and killed border guards, police officers, security agents and soldiers. "
        //
        // +
        // "All of them have been in prison for at least two decades; some were serving life sentences.";
        MaxentTagger tagger = new MaxentTagger(
            "C:/Users/Luojw/Documents/nlp/stanford-postagger-full-2015-04-20/models/chinese-distsim.tagger");
        Long start = System.currentTimeMillis();
        List<List<HasWord>> sentences = MaxentTagger
            .tokenizeText(new StringReader(str));
```

图 4.29 实现代码

运行的结果为：

2015最新Android基础入门教程/NT 完结散花/VV ~/PU -/PU coder-pig的猪栏/NR -/PU 博客频道/NR -/PU CSDN.NET/NR
coder-pig的猪栏/NR 目录视图/VV 摘要视图/VV 订阅/VV 想听课/NN ? /PU 来发话题吧/VV 云服务器使用初体验获奖通知/NR
技术干货/NN 、 /PU 还免费/NN ? /PU 来这儿/AD CSDN博客同类文章/NR , /PU 一键即达/NR 2015最新Android基础入门教程/NT
完结散花/VV ~/PU 标签/NN : /PU android/JJ 基础入门教程/NN 完结散花/VV 自学心得/VV 资源分享/VV 2015-11-23/CD 20:49/NT
1259人阅读/NT 评论/VV -LRB-/NR 28/CD -RRB-/M 收藏/VV 举报/NN 分类/NN : /PU 2015最新Android基础入门教程/NT (/PU 149/OD
) /PU 作者同类文章/VV X/NN 版权声明/VV : /PU 本文为博主原创文章/NR , /PU 未经博主允许不得转载/VV 。/PU
2015最新Android基础入门教程/NT 完结散花/VV ~/PU 标签/NN (/PU 空格分隔/NN) /PU : /PU 反思小结/NR 引言/NN : /PU
从六月底就开始编写这套教程/AD , /PU 历时将近五个多月/NR , /PU 今天终于写完了/NR , /PU 全套教程正文部分148篇/NR , /PU
十大章/NR , /PU 从基本UI控件到四大组件/NR , /PU Intent/NR , /PU Fragment/NR , /PU 事件处理/VV , /PU 数据存储/VV , /PU
网络编程/VV , /PU 绘图与动画/VV , /PU 多媒体/NN , /PU 系统服务等都进行了详细的讲解/VV ! /PU 代码都是都是在Android/VV
Studio上进行编写的/NN , /PU 全文/NN 采用Markdown/NN , /PU 行文结构清晰/VV , /PU

图 4.30 运行结果

4.4.2 算法的选取与舍弃

我们对文章做了分词和词性标注，去掉其中会对文章内容匹配有影响的词性，如标点符号/PU 动词/VV 和数词/NT 等，再通过正则表达式匹配得到标签，得到了较好的匹配结果，为算法的展示提供关键字标签。

4.4.3 算法思想的实现

表 4.4 关键字匹配代码

```
String reg1 ="(URL:: )(http|www|ftp|)?(://)?(\\w+(-\\w+)*)$";
```



```

String reg2 = "((分类: |标签: |//NN)[a-zA-Z\\s]*)";
//这是正则表达式，再次对文章的关键字和网址信息做匹配。
while((lineTxt = bufferedReader.readLine()) != null){
    String out = "";
    Pattern p= Pattern.compile(reg1);
    Matcher macher =p.matcher(lineTxt);
    if (macher.find())
        out +=macher.group(0).replaceAll("(URL:: )", "");
    Pattern pp= Pattern.compile(reg2);
    Matcher macher1 =pp.matcher(lineTxt);
    if (macher1.find())
        out = out+macher1.group(0).replaceAll("(分类: |
标签: )", "");

    if (out.equals(""))
        continue;
    output.add(out);
}
for(int s=0;s<6;s++)
    for (int j=0; j<output.size()-1;j++) {
//如下为格式对齐，使得网址和文章关键词并排写入到链表中，方便数据库的读写。
        if (output.get(j).length()>5&&output.get(j+1).length()>5)
        {
            if(output.get(j).subSequence(0,
5).equals(output.get(j+1).subSequence(0, 5)))
                output.remove(j);
        }

    }
    for (String string : output) {
        System.out.println(string);
    }
    boolean s=Save(output, tempList[i].getPath());

```

通过如表 4.4 所述的算法，可以得到文章的关键字和网址对存 ArrayList<>，为结果中网址推荐提供你数据。达到了较好的结果。

4.5 展示数据分析结果的实现

编写窗口代码如表 4.5 所示，使用 BorderLayout 和 GridLayout 两种布局方式。

表 4.5 前段界面代码

```

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

```

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.sql.DriverManager;
import java.sql.ResultSet;

import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

import com.mysql.jdbc.Connection;
import com.mysql.jdbc.PreparedStatement;

public class WebCrawler extends JFrame implements ActionListener {
    // 定义标题条
    JLabel titletop;
    // new 新组建
    JButton btnsearch;
    JTextArea txtDate;// 输出框
    JScrollPane jsp1;// 滚动条
    JTextField txtkeyFirst = new JTextField();// 关键词输入框
    //JTextField txtkeyFirst;
    JFileChooser fileChoser;// 文件选择
    File file;// 定义文件
    String path, keywordFrist, keywordSecond;// 文件路径及关键词
    //String t = new String();
    // 构造界面
    public WebCrawler() {
        this.setSize(800, 700);
        // 定义标题
        titletop = new JLabel("网络爬虫 PkuJava21", JLabel.CENTER);

        // 定义按钮
        // btnOpenFile = new JButton("Open File");
        btnsearch = new JButton("Search");

        // 定义文本框
        txtDate = new JTextArea();
        jsp1 = new JScrollPane(txtDate);// 滚动条
    }
}
```

```
// 定义关键词
JLabel labkeyFirst = new JLabel("First key words");

// 定义关键词文本框
//JTextField txtkeyFirst = new JTextField();

// 布置按钮画板
JPanel panBtn = new JPanel();
panBtn.setLayout(new GridLayout(1, 2));
panBtn.add(btnsearch);

// 布置关键词画板
JPanel panInput = new JPanel();
panInput.setLayout(new GridLayout(1, 2));
panInput.add(labkeyFirst);
panInput.add(txtkeyFirst);

// 定义关键词与按钮画板
JPanel panInputBL = new JPanel();
panInputBL.setLayout(new GridLayout(1, 1));
panInputBL.add(panInput);
panInputBL.add(panBtn);

// 布置小画板
JPanel panChoose = new JPanel();
panChoose.setLayout(new BorderLayout());
panChoose.add(jsp1, BorderLayout.CENTER);
panChoose.add(panInputBL, BorderLayout.SOUTH);

// 布置窗体
this.setLayout(new BorderLayout());
this.add(titletop, BorderLayout.NORTH);
this.add(panChoose, BorderLayout.CENTER);

// 注册监听
btnsearch.addActionListener(this);
// btnOpenFile.addActionListener(this);
// btncluster.addActionListener(this);
}

public static void main(String[] args) throws Exception {
    WebCrawler w = new WebCrawler();
    w.setVisible(true);
}
```

```

    }
}

```

设置触发事件，在用户输入关键希望查询的关键词后，点击 **Search** 按钮，触发查询事件，系统根据关键词查询 **tabword** 字段，并将符合条件的标签对应的 URL 显示在界面上，设置触发事件代码如表 4.6 所示。

表 4.6 事件响应

```

package jdbcTest;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.sql.DriverManager;
import java.sql.ResultSet;

import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

import com.mysql.jdbc.Connection;
import com.mysql.jdbc.PreparedStatement;

public class WebCrawler extends JFrame implements ActionListener {
    // 定义标题条
    JLabel titletop;
    // new 新组建
    JButton btnsearch;
    JTextArea txtDate;// 输出框
    JScrollPane jsp1;// 滚动条
    JTextField txtkeyFirst = new JTextField();// 关键词输入框
    //JTextField txtkeyFirst;
    JFileChooser fileChoser;// 文件选择
    File file;// 定义文件
    String path, keywordFrist, keywordSecond;// 文件路径及关键词
    //String t = new String();
    // 构造界面

```

```
public WebCrawler() {
    this.setSize(800, 700);
    // 定义标题
    titletop = new JLabel("网络爬虫 PkuJava21", JLabel.CENTER);

    // 定义按钮
    // btnOpenFile = new JButton("Open File");
    btnsearch = new JButton("Search");

    // 定义文本框
    txtDate = new JTextArea();
    jsp1 = new JScrollPane(txtDate); // 滚动条

    // 定义关键词
    JLabel labkeyFirst = new JLabel("First key words");

    // 定义关键词文本框
    //JTextField txtkeyFirst = new JTextField();

    // 布置按钮画板
    JPanel panBtn = new JPanel();
    panBtn.setLayout(new GridLayout(1, 2));
    panBtn.add(btnsearch);

    // 布置关键词画板
    JPanel panInput = new JPanel();
    panInput.setLayout(new GridLayout(1, 2));
    panInput.add(labkeyFirst);
    panInput.add(txtkeyFirst);

    // 定义关键词与按钮画板
    JPanel panInputBL = new JPanel();
    panInputBL.setLayout(new GridLayout(1, 1));
    panInputBL.add(panInput);
    panInputBL.add(panBtn);

    // 布置小画板
    JPanel panChoose = new JPanel();
    panChoose.setLayout(new BorderLayout());
    panChoose.add(jsp1, BorderLayout.CENTER);
    panChoose.add(panInputBL, BorderLayout.SOUTH);

    // 布置窗体
    this.setLayout(new BorderLayout());
}
```

```

        this.add(titletop, BorderLayout.NORTH);
        this.add(panChoose, BorderLayout.CENTER);

        // 注册监听
        btnsearch.addActionListener(this);
        // btnOpenFile.addActionListener(this);
        // btncluster.addActionListener(this);
    }

    public static void main(String[] args) throws Exception {
        WebCrawler w = new WebCrawler();
        w.setVisible(true);
    }
//设置触发事件
@Override
    public void actionPerformed(ActionEvent arg0) {
        // TODO Auto-generated method stub
        if (arg0.getActionCommand().equals("Search")) {
            // 执行查询并输出显示结果
        }
    }
}
}

```

获取用户关键词并在文本框上显示代码如下图所示：

```

// 获取文件路径及关键词输入框的内容
String textfirst = new String();
textfirst = txtkeyFirst.getText();
txtDate.append(textfirst+"\r\n");

```

图 4.31 获取关键词

执行查询操作代码如下图所示：

```

//查询语句
String keywords = textfirst;
String sqlSelect = "select * from text where tabword='"+keywords+"'";
//执行查询
ResultSet rs = ps.executeQuery(sqlSelect);

```

图 4.32 执行查询

获取查询结果并在文本框上显示代码如下图所示：

```

// 执行查询后取值
while (rs.next()) {
    System.out.println(rs.getString(2));
    txtDate.append(rs.getString(2)+"\r\n");
}

```

图 4.33 获取查询结果

5. 程序运行结果

5.1 系统初始化界面

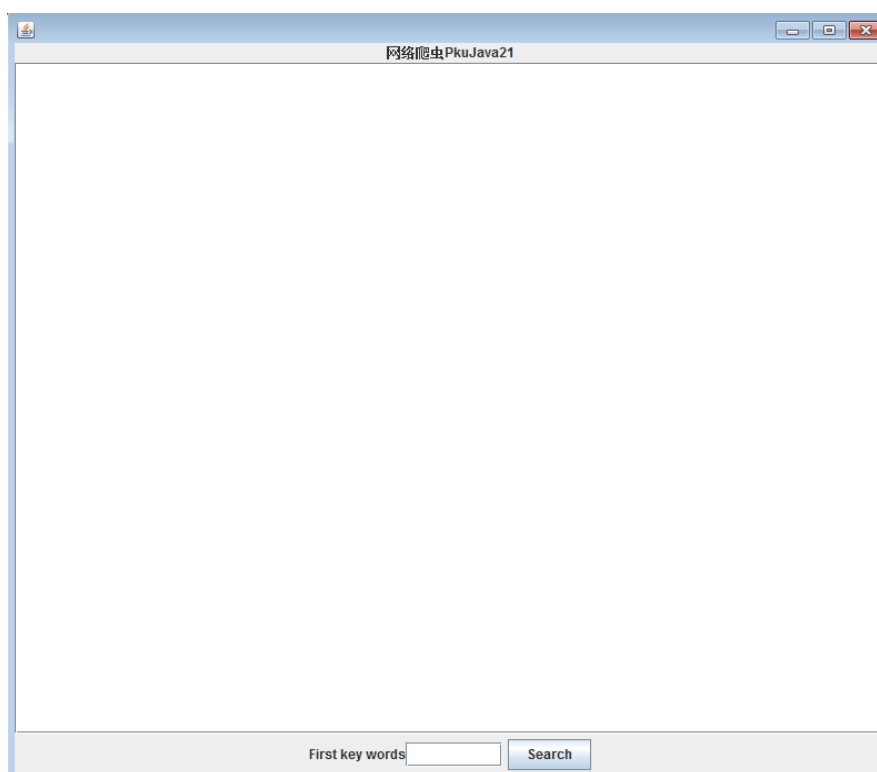


图 5.1 初始化界面

由上图所示，初始化界面包含关键词输入框，结果显示界面，以及 Search 按钮。

5.2 输入关键词并点击查询

在文本框中输入关键词并在界面显示，结果如图 5.2 所示：

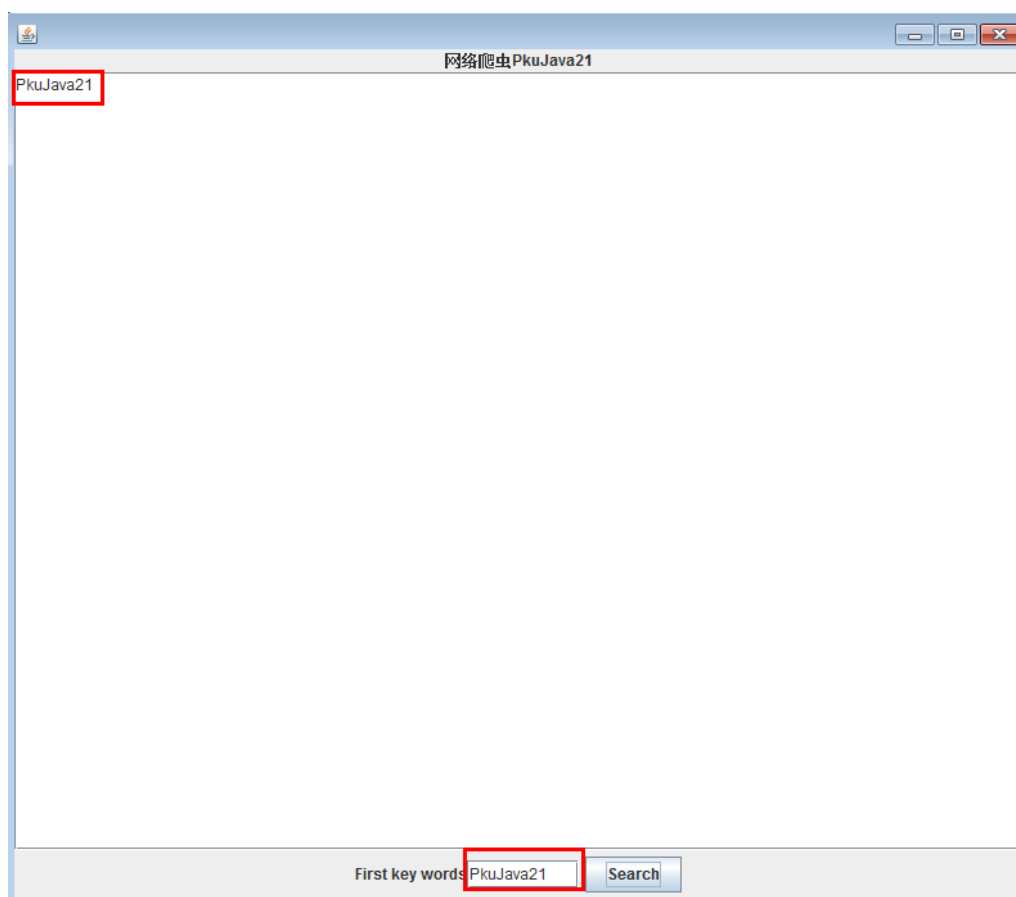


图 5.2 关键词输入并获取显示结果

由图 5.2 可知，当用户输入 PkuJava21 后，系统会获取关键词并显示在输出界面上。

5.3 查询结果显示

输入关键词 Android:

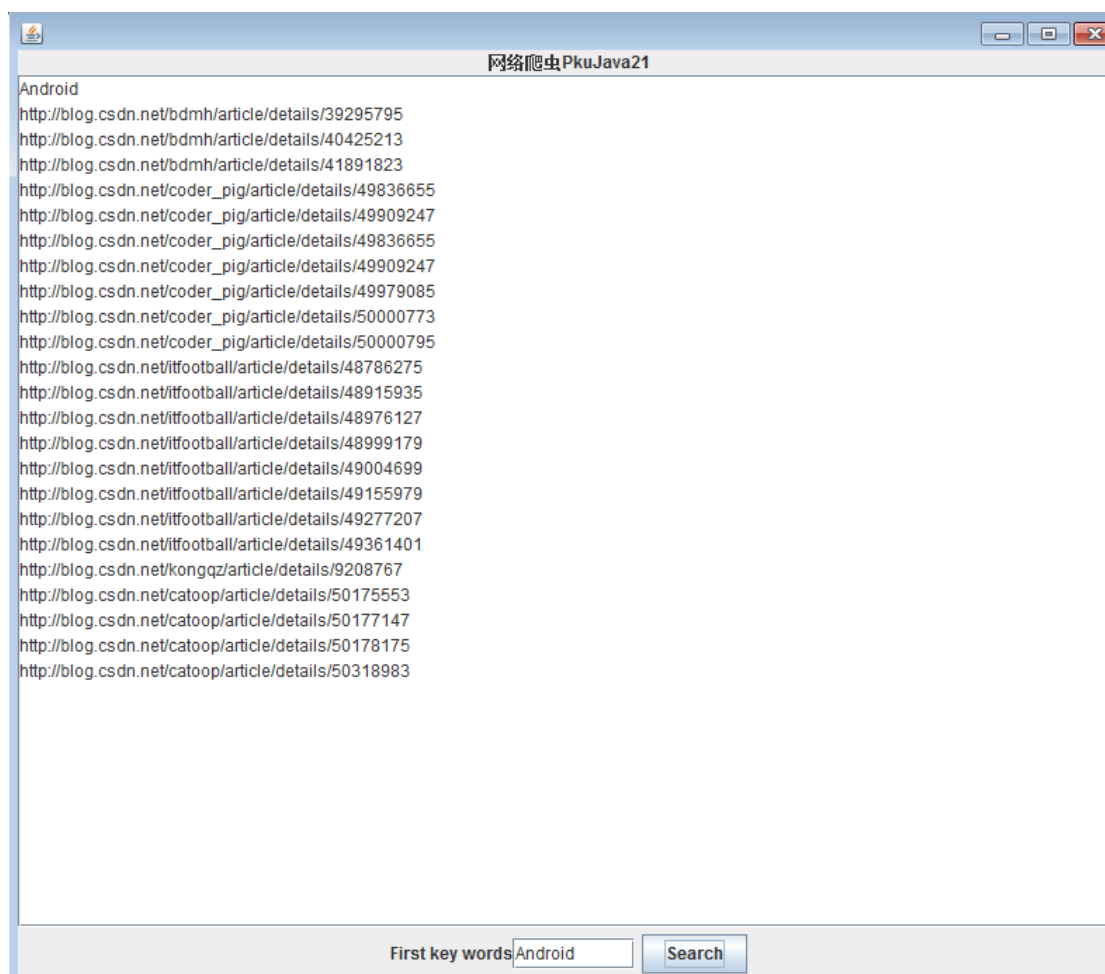


图 5.4 关键词 Android 查询结果

在测试部分，用户输入关键词 Android 后，点击查询按钮，系统输出结果如上图所示。

输入关键词 Ruby 结果如下：



数据库对比结果如下：

272	http://blog.csdn.net/nsrainbow/article/details/49834097	ruby
273	http://blog.csdn.net/nsrainbow/article/details/50058293	javascript
274	http://blog.csdn.net/nsrainbow/article/details/50375578	ruby

图 5.5 关键词 Ruby 查询结果

将网页输入浏览器进行验证：

输入网址：<http://blog.csdn.net/nsrainbow/article/details/49834097> 进行验证，结果如图 5.6 所示：



图 5.6 URL 正确性验证

输入网址：<http://blog.csdn.net/nsrainbow/article/details/50375578> 进行验证，结果如图 5.7 所示：



图 5.7 URL 正确性验证

6. 本项目的收获

通过本次项目，我们组收获很大。首先是更加注重团队合作了，团队意识更强。其次学会了版本控制的使用，更有利于团队开发。除此之外还在技术上收获很大。学习了用 Nutch 抓取数据，学习了用正则表达式处理数据，进一步熟悉了使用 sql 语句存储数据和查询数据。非常感谢老师布置的这次作业。