

1 爬虫框架实现

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。

1.1 Nutch 框架搭建

1.1.1 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 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 2 建立 JDK 安装目录

3. # chmod +x jdk-6u65-linux-x64.bin (获得执行权限)。

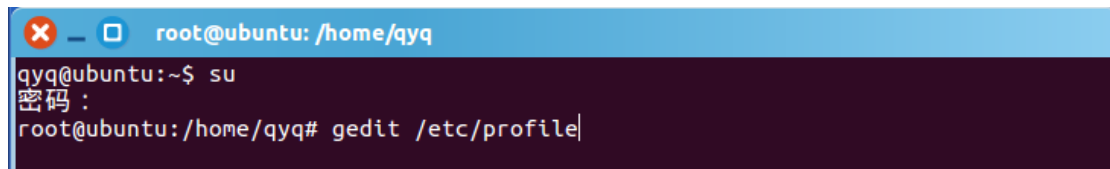
```
root@ubuntu: /home/qyq
root@ubuntu:/home/qyq# chmod +x /etc/profile
```

图 3 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

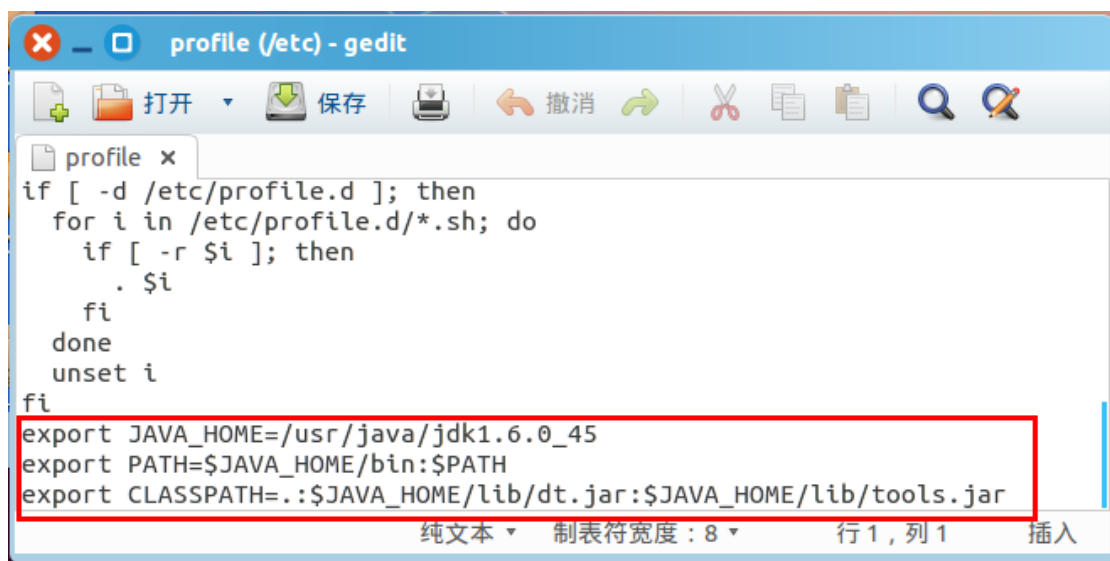
gedit /etc/profile



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

图 4 打开配置文件

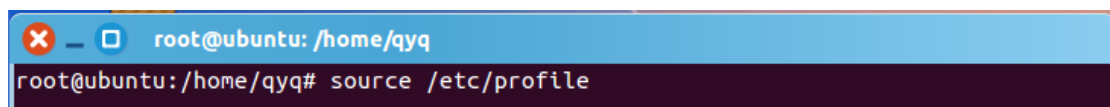
8. 添加环境变量



```
profile x
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 5 添加环境变量

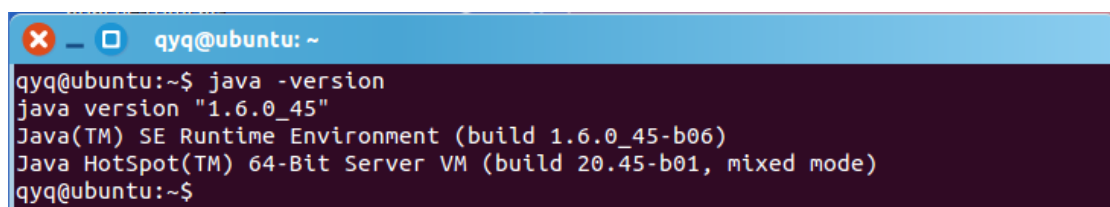
9. # chmod +x /etc/profile (执行权限)。
10. # source /etc/profile (此后设置有效)。



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

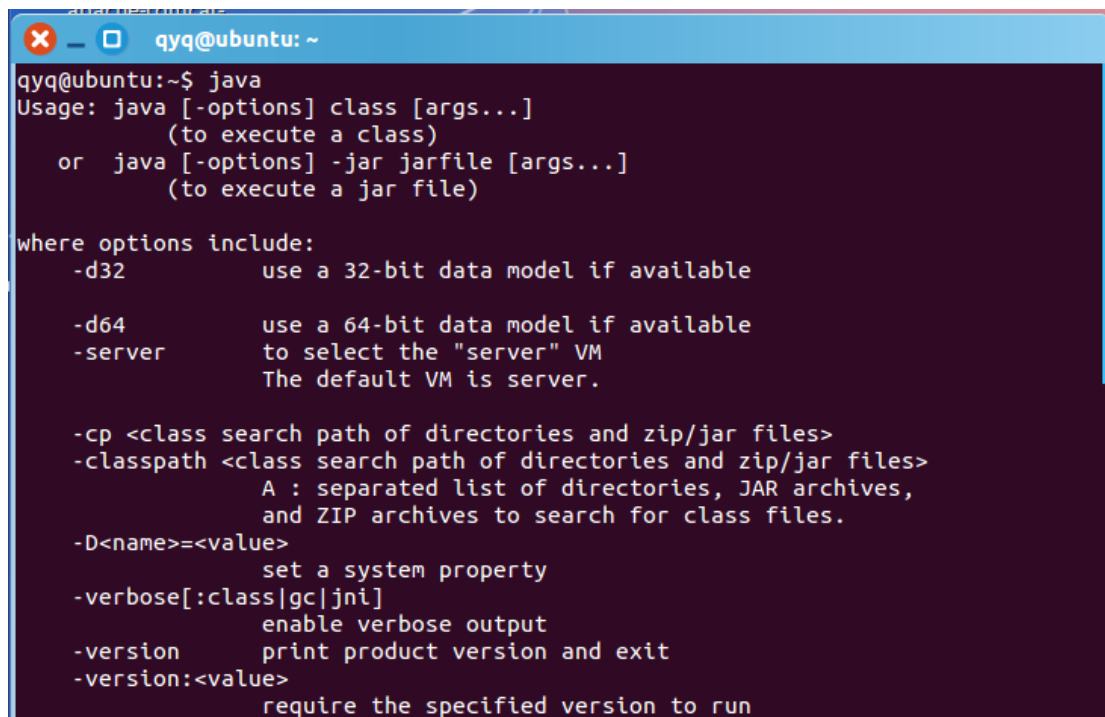
图 6 更新 profile 文件

11. 测试 JDK 是否安装成功



```
qyq@ubuntu: ~
qyq@ubuntu:~$ java -version
java version "1.6.0_45"
Java(TM) SE Runtime Environment (build 1.6.0_45-b06)
Java HotSpot(TM) 64-Bit Server VM (build 20.45-b01, mixed mode)
qyq@ubuntu:~$
```

图 7 测试 java -version



```
qyq@ubuntu: ~  
qyq@ubuntu:~$ java  
Usage: java [-options] class [args...]  
          (to execute a class)  
    or java [-options] -jar jarfile [args...]  
          (to execute a jar file)  
  
where options include:  
    -d32          use a 32-bit data model if available  
  
    -d64          use a 64-bit data model if available  
    -server       to select the "server" VM  
                  The default VM is server.  
  
    -cp <class search path of directories and zip/jar files>  
    -classpath <class search path of directories and zip/jar files>  
                  A : separated list of directories, JAR archives,  
                  and ZIP archives to search for class files.  
    -D<name>=<value>  
                  set a system property  
    -verbose[:class|gc|jni]  
                  enable verbose output  
    -version       print product version and exit  
    -version:<value>  
                  require the specified version to run
```

图 8 测试 java 命令

JDK 安装成功。

1.1.2 安装 Tomcat

1. 设置环境变量

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



```
*profile (/etc) - gedit  
...  
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  
export JDK_HOME=$JAVA_HOME
```

图 9 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`

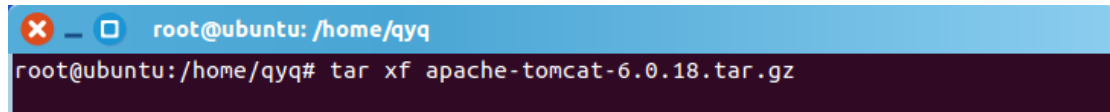


图 10 解压 Tomcat 到主文件夹

3. 启动 Tomcat

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

`#./bin/startup.sh`

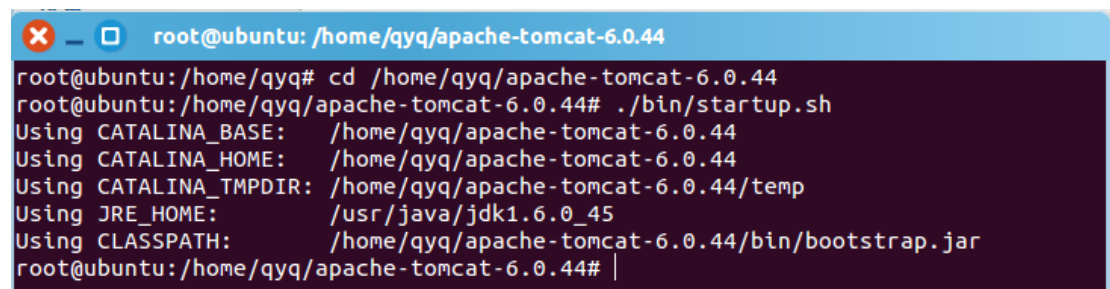


图 11 启动 Tomcat

- 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 12 端口号配置

6. 测试 Apache Tomcat 是否启动成功。

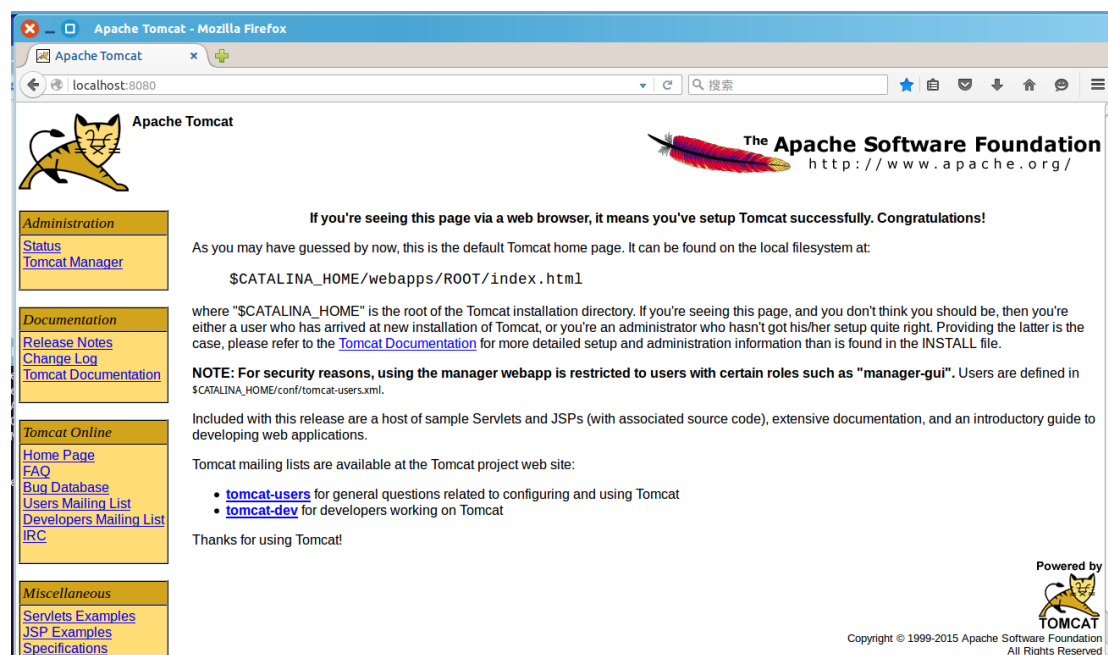


图 13 测试 Tomcat 启动成功

1.1.3 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

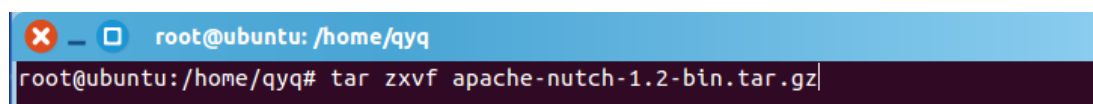
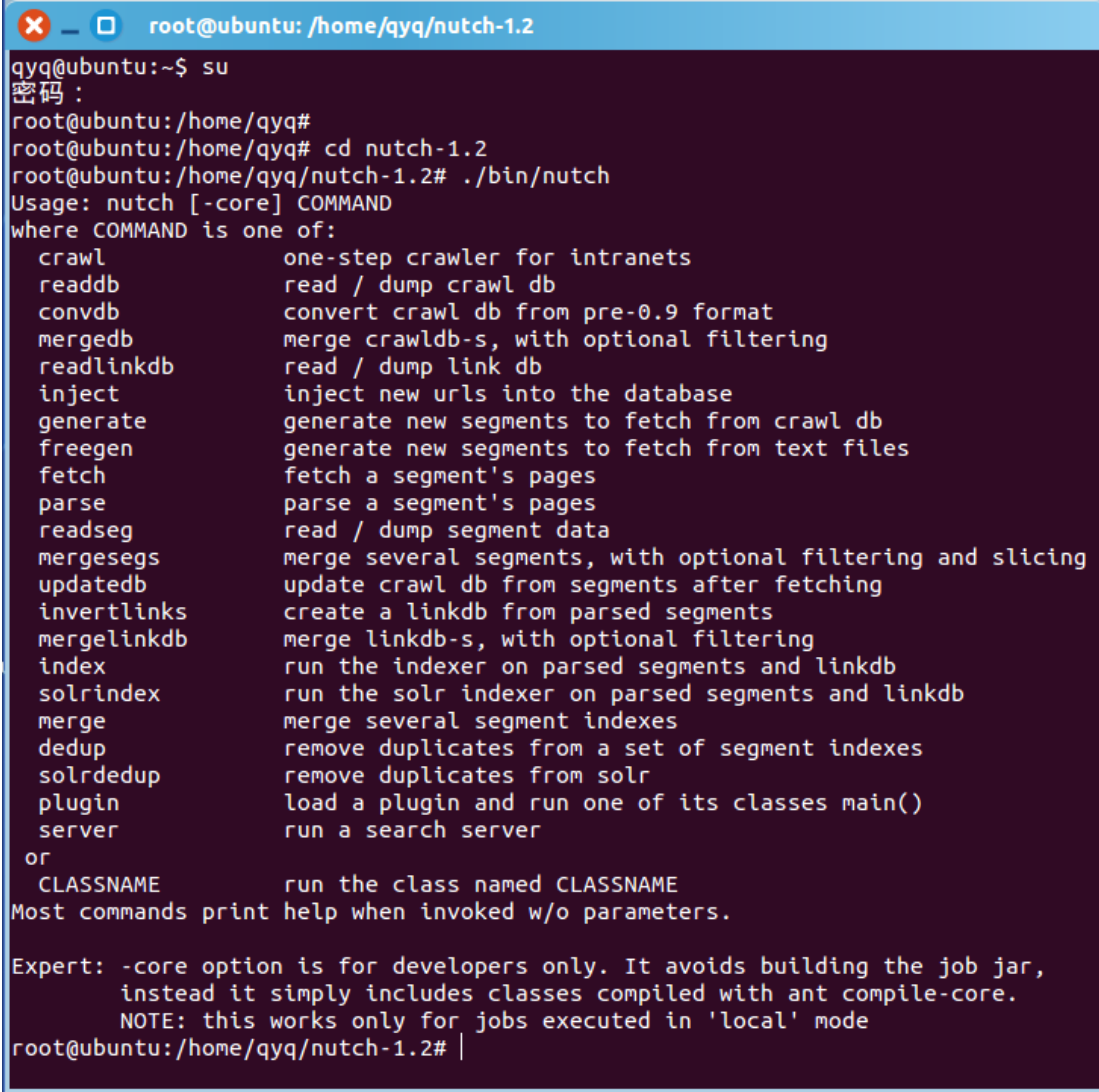


图 14 解压 nutch1.2

3. 测试 Nutch

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



```
root@ubuntu: /home/qyq/nutch-1.2
qyq@ubuntu:~$ su
密码:
root@ubuntu:/home/qyq#
root@ubuntu:/home/qyq# cd nutch-1.2
root@ubuntu:/home/qyq/nutch-1.2# ./bin/nutch
Usage: nutch [-core] COMMAND
where COMMAND is one of:
  crawl          one-step crawler for intranets
  readdb          read / dump crawl db
  convdb          convert crawl db from pre-0.9 format
  mergedb         merge crawl-db-s, with optional filtering
  readlinkdb      read / dump link db
  inject          inject new urls into the database
  generate        generate new segments to fetch from crawl db
  freegen         generate new segments to fetch from text files
  fetch           fetch a segment's pages
  parse           parse a segment's pages
  readseg         read / dump segment data
  mergesegs       merge several segments, with optional filtering and slicing
  updatedb        update crawl db from segments after fetching
  invertlinks     create a linkdb from parsed segments
  mergelinkdb     merge linkdb-s, with optional filtering
  index           run the indexer on parsed segments and linkdb
  solrindex       run the solr indexer on parsed segments and linkdb
  merge           merge several segment indexes
  dedup           remove duplicates from a set of segment indexes
  solrdedup       remove duplicates from solr
  plugin          load a plugin and run one of its classes main()
  server          run a search server
or
  CLASSNAME      run the class named CLASSNAME
Most commands print help when invoked w/o parameters.

Expert: -core option is for developers only. It avoids building the job jar,
        instead it simply includes classes compiled with ant compile-core.
NOTE: this works only for jobs executed in 'local' mode
root@ubuntu:/home/qyq/nutch-1.2# |
```

图 15 测试 nutch 启动成功

4. 设置 Nutch 抓取网站的入口网址

`# cd /home/qyq/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-[*!@]

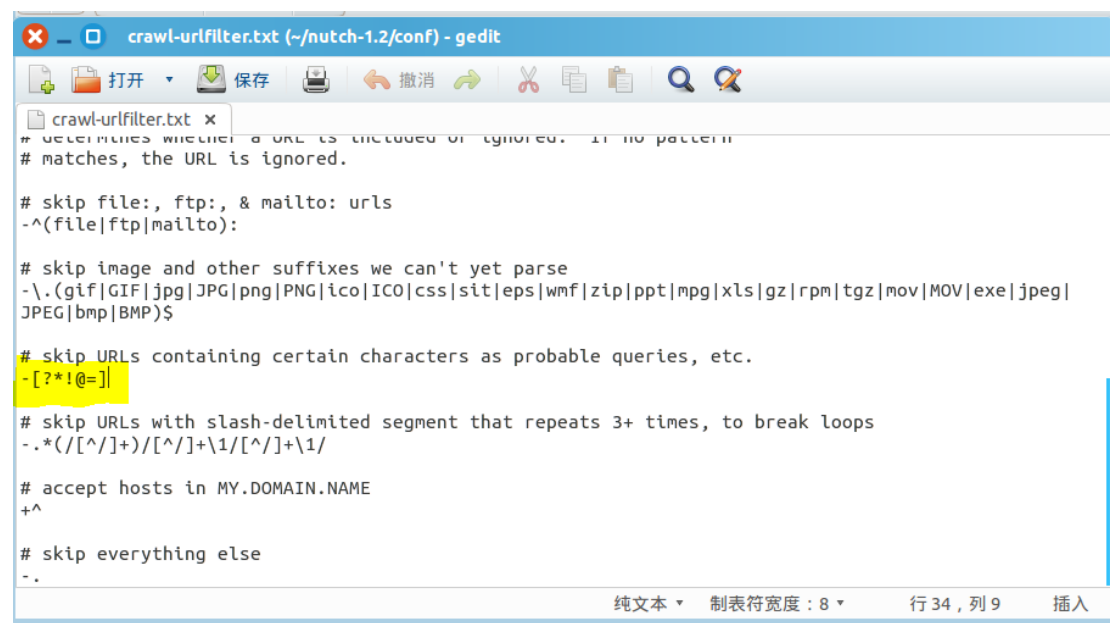


图 16 配置过滤规则

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/qyq/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。

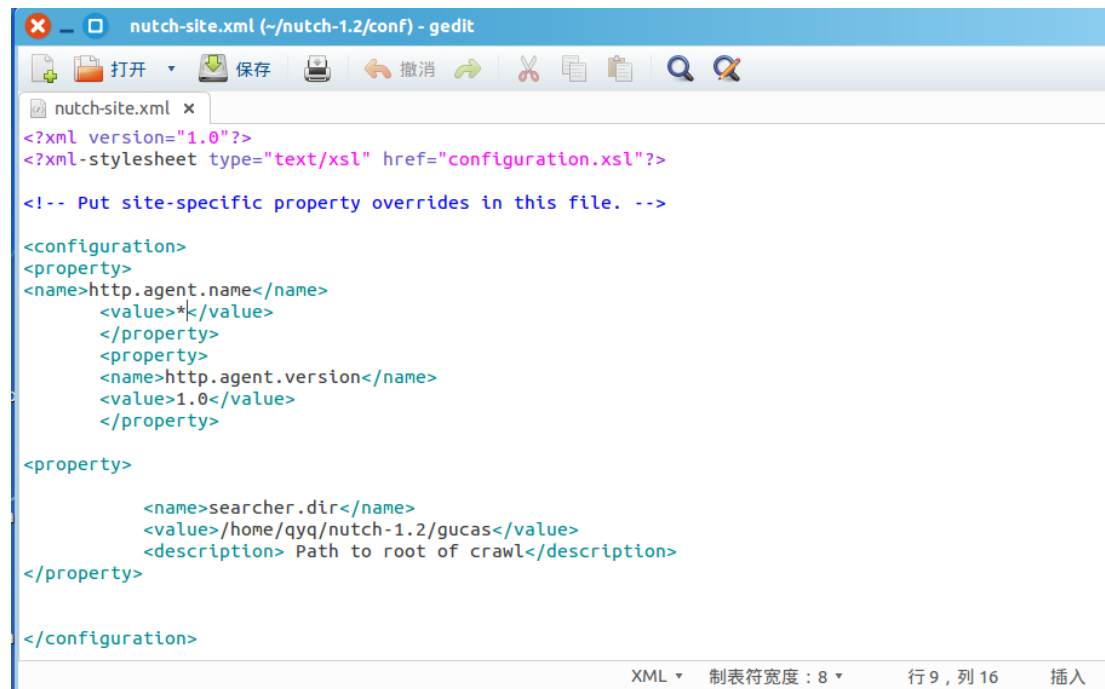


图 17 配置 conf/nutch-site.xml

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/qyq/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 18 测试 web 前端界面

2 数据处理与存储的实现

2.1 数据处理的实现

在数据预处理阶段，我们主要通过 java 的正则表达式功能去筛选掉由网页上无用的符号信息，匹配出有用的关键词信息作为以后文章语义分析的基础。这样能够更有目标的筛选出网页的有用信息。

```
public static void main(String[] args) {
    try{
        // TODO Auto-generated method stub
        String encoding = "utf-8";
        String path = "1321";
        File file = new File(path);
        ArrayList<String> output = new ArrayList<String>();
        if (file.isFile() && file.exists()) {
            InputStreamReader read = new InputStreamReader(
                new FileInputStream(file), encoding);
            BufferedReader bufferedReader = new BufferedReader(read);
            String lineTxt = null;
            String reg1 = "(URL: )(http|www|ftp)?(?:/)?(\\w+(-\\w+)*)(\\.\\w+(-\\w+)*))*((:\\d+";
            String reg2 = "((分类: |标签: )[a-zA-Z\\s]*)";
            while((lineTxt = bufferedReader.readLine()) != null){
                System.out.println(bufferedReader.readLine());
                String out = "";
                Pattern p= Pattern.compile(reg1);
                Matcher macher = p.matcher(lineTxt);
                if (macher.find()){
                    out = out+macher.group(0);
                    Pattern pp= Pattern.compile(reg2);

```

图 19 正则表达式实现

得到的匹配结果如下：

URL:: <http://blog.csdn.net/csfreebird/article/details/49307935>
标签: linux

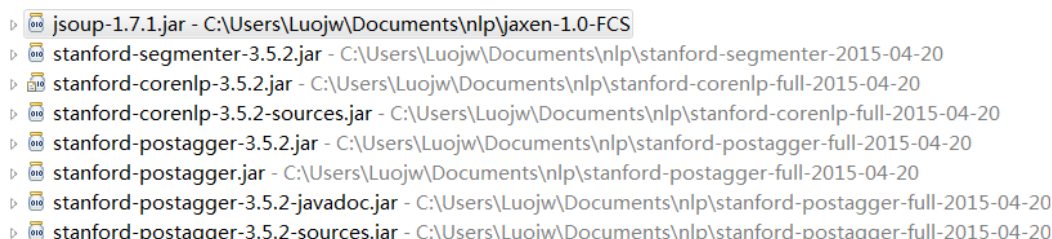
URL:: <http://blog.csdn.net/csfreebird/article/details/49721235>
标签: spark

URL:: <http://blog.csdn.net/xiaoxian8023/article/details/49619777>
标签: ssl tomcat https

如上所述，得到的结果主题目标能够更好指导我们进行文章的信息提取，可以从文章内容中得到更多有用的结果，丰富标签的内容，更希望建立针对各个方向的知识库。

2.2 数据分析的实现

数据分析方面我们使用了斯坦福大学的自然语言处理工具包



The image shows a file explorer window with the following files listed:

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

图 20 调用的 java 工具包

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。

实现代码为：

```
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取消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 22 分词工具实现

分词结果为：

```
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/d
ict/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 题 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 23 简单的分词结果

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。

3 数据存储的实现

在当前数据量不大的情况下，处理出来的数据可以直接存到本地磁盘上，方便调试时读取数据，分析结果。

```
/*
 *
 * 内存溢出不属于本程序问题，是separateWord内种出现问题，而separate类中的标注方法由斯坦福官方提供。
 * 每台计算机的内存溢出的数量也不同
 * 在完全不调用写文件代码时也会出现内存溢出情况，即原因与写文件无关。
 */
public class ToolReadAndWriteForSW {
    //读文件时候的一个文本缓冲区
    Map<Integer, String> map = new HashMap<Integer, String>();
    //准备数据结果，用于写出文本
    Map<Integer, String> newmap = new HashMap<Integer, String>();

    public int readTxtFile(String fileReadPath,String fileWritePath) {
        try {
            String encoding = "utf-8";
```

图 24 数据存储