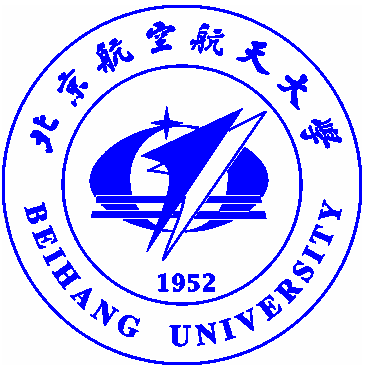
**成绩**

**北 京 航 空 航 天 大 学**

**人工智能实验报告**

**主题爬虫设计与实现**

****

**学 院 浙江工业大学交换生**

**专业方向 计算机科学与技术**

**学生姓名 姜楠**

**指导教师 巢老师**

**北京航空航天大学 计算机学院**

**2014 年6月**

### 主题爬虫的设计与实现

#### 引言

互联网是一个庞大的非结构化的数据库，将数据有效的检索并组织呈现出来有着巨大的应用前景。搜索引擎作为一个辅助人们检索信息的工具，但是这些通用性搜索引擎存在着一定的局限性。不同领域、不同背景的用户往往具有不同的检索目的和需求，通用搜索引擎所返回的结果，包含大量用户不关心的网页。所以需要一个能基于主题搜索的，满足特定需求的网络爬虫。

为了解决上述问题，参照成功的网络爬虫模式，对网络爬虫进行研究，提供满足特定搜索需求的网络爬虫。

#### 实验设计

##### 实验要求

1. 可以进行多线程进行抓取；
2. 可以进行面向主题进行抓取；
3. 可以分辨重复的的网页内容；
4. 可以计算主题相关性；
5. 可以处理网络延时等待的处理；

##### 实验环境配置

1. 系统硬件环境：LENOVO-G470 Intel Core(TM) i3-2330M@2.20GHz
2. 操作系统环境：Windows 8.1 专业版
3. 实验配置环境：Java SE Development Kit 7 Update 45；

Eclipse Kepler ；

MySQL Server 5.6；

##### 实验方案设计

1. **广度优先搜索策略**

广度优先搜索策略是指在抓取过程中，在完成当前层次的搜索后，才进行下一层次的搜索。该算法的设计和实现相对简单。在目前为覆盖尽可能多的网页，一般使用广度优先搜索方法。

也有很多研究将广度优先搜索策略应用于主题爬虫中，基本思想是认为与初始URL在一定链接距离内的网页具有主题相关性的概率很大。

另外一种方法是将广度优先搜索与网页过滤技术结合使用，先用广度优先策略抓取网页，再将其中无关的网页过滤掉。

这些方法的缺点在于，随着抓取网页的增多，大量的无关网页将被下载并过滤，算法的效率将变低。

1. **深度优先搜索策略**

深度优先搜索策略是一种在开发网络爬虫早期使用得较多的方法，是指网络蜘蛛会从起始页开始，一个链接一个链接跟踪下去，处理完这条线路之后再转入下一个起始页，继续跟踪链接。当不再有其他超链可选择时，说明搜索已经结束。

1. **最佳优先搜索策略**

最佳优先搜索策略按照一定的网页分析算法，预测候选URL与目标网页的相似度，或与主题的相关性，并选取评价最好的一个或几个URL进行抓取。它只访问经过网页分析算法预测为“有用”的网页。

存在的一个问题是，在爬虫抓取路径上的很多相关网页可能被忽略，因为最佳优先策略是一种局部最优搜索算法。因此需要将最佳优先结合具体的应用进行改进，以跳出局部最优点。研究表明，这样的闭环调整可以将无关网页数量降低30% ~ 90% 。

1. **主题相关度建模 – 向量空间模型**

向量空间模型是由Salton等人于20世纪60年代末提出，是一种简便、高效的文本表示模型，其理论基础是代数学[1]。

向量空间模型把用户的查询要求和数据库文档信息表示成由检索项构成的向量空间中的点，通过计算向量之间的距离来判定文档和查询之间的相似程度。然后，根据相似程度排列查询结果。

向量空间模型的关键在于特征向量的选取和特征向量的权值计算两个部分。

对于任一文档，我们可以把它表示为如下t维向量的形式：



其中，向量分量代表第i个标引词在文档中所具有的权重，t为系统中标引词的总数。在布尔模型中，的取值范围是；在向量空间模型中，由于采用“部分匹配”策略，的取值范围是一个连续的实数区间[0，1]。

在检索的前处理中，一篇文档中会标引出多个不同的标引词，而这些标引词对表达该篇文档主题的能力往往是不同的。也就是说，每个标引词应该具有不同的权值。如何计算文档向量中每个标引词的权值，不仅关系到文档向量的形成，也关系到后续的检索匹配结果。

　　标引词权重的大小主要依赖其在不同环境中的出现频率统计信息，相应的权重就分成局部权重和全局权重。

　　局部权重(Local Weight) 是按第i个标引词在第j篇文档中的出现频率计算的权重。它以提高查全率为目的，对在文档中频繁出现的标引项给予较大的权重。

全局权重(Global Weight) 则是按第i个标引词在整个系统文档集合中的分布确定的权重。它以提高查准率为目的，对在许多文档中都出现的标引项给予较低的权重，而对仅在特定文档中出现频次较高的标引项给予较大的权重。

计算全局权重的典型方法就是逆文档频率IDF(Inverse Document Frequency)加权法：



其中，N为系统文档总数，为系统中含有标引词  的文档数。

向量间相似程度的度量方法有内积法(Inner Product)、Dice法(Dice Coefficient)、Jaccard法(Jaccard Coefficient)和余弦法(Cosine Coefficient)。

　　较常用的度量方法是提问向量和文档向量间的内积法，其计算公式如下：



　　其中，是检索提问中检索项i的权值，是文档中标引项i的权值，N为总的项数。

　　当每个向量都通过余弦法进行加权后，则内积法转换为余弦法。余弦法采用的相似度计算指标是两个向量夹角的余弦函数。

1. **主题爬虫模型建立**



图1-1 主题爬虫模型结构

1. 种子以及关键词的选取

在初始化阶段，我们需要给爬虫提供初始的网址和一组与主题相关的关键词。关键词的提取，我采用的是凭经验手动给出。同时，每个关键词都需要指定不同的权值，权值的设置方法有两种：手工设置和特征提取。特征提取是指给定一个跟主题有关的网页集合,由程序自动提取这些网页里面共同的特征,并根据频率确定权值。

遵循简洁有效的原则，我选取几组特定的网页，提取词频较高的前20个关键词，所有词的权值均相同。

1. URL队列的运行与维护

对于单线程爬虫，可以使用简单的优先队列实现最佳有限算法，每次线程从队列中提取相关度最高的URL链接，运行结束后，将有效的链接插入队列中。

对于多线程爬虫，由于线程之间的耦合度很低，属于并行计算模型。假若多个线程同时访问URL队列，就会发生异常情况，可能多个线程取到的URL均相同，这样就会间接浪费线程资源。因为我们希望相同的URL只需要被处理一次就可以了。同时，在多个线程将有效URL插入队列时，URL队列在同一时间只能为一个线程服务，其余线程送入的URL将会被抛弃。

基于上述考量，我们需要采用给URL队列分配互斥锁，一旦某个线程获得URL队列的使用权，其余线程就忙等待，直到该互斥锁被撤销。

在设计过程中，我只采用一个等待队列，在这个队列中，URL等待被处理，新发现的URL被加入到该队列中。不相关的URL均被抛弃，完成抓取的URL均存入数据库，不再建立完成队列记录访问的链接。

在Java Virtual Machine中，我们可以使用关键字synchronized，对需要访问的资源加锁。当两个并发线程访问同一个对象object中的这个synchronized同步代码块时，一个时间内只能有一个线程得到执行。另一个线程必须等待当前线程执行完这个代码块以后才能执行该代码块。

|  |
| --- |
| //提取的URL入队  **public** **synchronized** **void** pushUrl() {}  //线程从队列中读取一个URL  **public** **synchronized** String popUrl() {} |

为了实现最佳优先策略，我们需要扩展URL，集成系统提供的接口Comparable，为其添加优先级的属性，自定义比较大小方法。如果不采用该策略，系统默认的比较方法是，比较两个URL（String）的字典序大小，这显违背我们的最佳优先策略。

|  |
| --- |
| /\*\*  \* 具有优先比较的URL类  \*/  **public** **class** PriorityURL **implements** Comparable<PriorityURL>{  ... ....  **public** **double** priority;  @Override  /\*\*  \* 实现两个URL之间大小的比较方法  \*/  **public** **int** compareTo(PriorityURL p) {  **if**(priority>p.priority) **return** 1;  **else** **if**(priority<p.priority) **return** -1;  **else** **return** 0;  } |

1. 主题相关度计算模块

为了保证爬取的网页与我们的主题有高度相关性，我们需要过滤相关度较低的网页链接。前面我们已经讨论至“向量空间模型”，在这里我们采用余弦度量法。

具体做法是，统计网页中关键词出现的频率，然后与我们初始的关键词按照公式：，求解得余弦值，即可得到该网页的相关度。指定一个阈值threshold，当，可以认为该页面和主题是比较相关的, r的取值需要根据经验和实际要求确定,如果想获得较多的页面,可以把r设小一点,要获得较少的页面可以把r设的大一点。在我们的程序中，默认将阈值r设置为0.9。

1. 网页下载与链接提取模块

Java提供了一组类用于网络通信java.net.URL，但是需要繁复的设置来维护一个socket通信。本次实验只需要下载html纯文本文件，如果能尽量简单的设置网络通信规范，屏蔽基层类的复杂性，能有效减少程序编制时间，提高程序鲁棒性。因此，我们采用开源Jsoup.jar插件，这个类库提供一组便捷的下载并解析HTML的API。只要预先给出URL链接，Jsoup库能下载该HTML文件。

事实上，我们还需要注意HTTP Status code,只有当链接返回的状态值为200，才表明网页响应正常；若是返回的状态值为404 Not Found，表示无法找到指定位置的资源，这时候Jsoup类库就会显示Exception，程序会异常终止。

|  |  |
| --- | --- |
| HTTP常用状态表 | |
| 返回值 | 含义 |
| 200 | 一切正常，响应报文跟在后面 |
| 400 | Bad Request，请求出现语法错误 |
| 403 | Forbidden，资源不可用 |
| 404 | Not Found 无法找到指定位置的资源 |
| 502 | Bad Gateway，服务器收到无效响应 |

* 为了维持一个正常HTTP的链接，响应正确的链接，我们必须做必要的设置，可以参考：

|  |
| --- |
| //忽略链接中出现的任何异常，保持程序能持续运行  Jsoup.*connect*(url).ignoreHttpErrors(true); |

* 若是通过代理服务器上网的需要设置系统代理，而且必须在软件中用代码定义，否则仅仅是在Windows操作系统网络设置修改代理，仍会出现

java.net.UnknownHostException这个错误，至于设置方法可以参考：

|  |
| --- |
| //设置代理网络地址  System.*setProperty*("http.proxyHost", "127.0.0.1");  //设置代理端口号  System.*setProperty*("http.proxyPort", "8182"); |

* 由于网络的异构性，与实验环境网络不稳定型，导致一个socket极易因为连接超时，导致java.net.SocketTimeoutException异常，为此我们需要修改默认设置的超时时间，可以参考：

|  |
| --- |
| //设置超时时间为10s  Jsoup.*connect*(url).timeout(10000); |

* 为了使服务器返回的页面与浏览器中看到的页面一致，我们还需要给出浏览器的UserAgent信息（是一个浏览器的身份标识），例如，google Chrome浏览器的UserAgent信息：

“Mozilla/5.0 (Windows NT 5.2) AppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.122 Safari/534.30”

在项目中我们只需要添加下列代码即可

|  |
| --- |
| //useragent代表上述的String字符串  Jsoup.*connect*(url).userAgent(useragent); |

* 我们还需要注意到，不同网页之间的编码格式是不同的，中文网页普遍使用GBK编码，国外网站普遍使用UTF-8编码。虽然，针对每个HTML文件，在其源代码中均会标明编码格式，例如：

<meta ... ...charset=utf-8" />

在实验过程中，我们利用正则表达式，提取出HTML标明的charset，利用Java提供的BufferedWriter，设置写入字符的格式为指定格式，用浏览器查看保存的文件，显示正常，使用方法可以参考：

|  |
| --- |
| BufferedWriter bw = **new** BufferedWriter(**new** OutputStreamWriter (**new** FileOutputStream(filename),"gbk")); |

* 关于何时保存文件的考量

首先，由于一张网页就有100KB的大小，将该庞大的网页内容存入数据库显然是不合理的，同时从用户角度考虑，采用文本方式保存到本地文件夹下更为合理。因此，我们可以采用两种方式保存文件，一种是基于异步线程，将下载的网页存入一个队列中，一个异步线程每隔一段时间（例如：2 s）读取队列中的内容，保存至本地，这样做的好处是，主线程不需要等待缓慢的文件保存过程，通过托管的方式让其他线程完成，节省了宝贵的时间。这样做明显的缺点，就是占用大量内存，文本文件均暂留在内存中会妨碍其他程序的运行；另一种方案是采取同步模式，直接在主线程中完成文件的读写操作，这样做虽然会增加耗费的时间，但会有效节省空间的占用。本次实验，我们采取第二种方法。

1. 网页去重模块

在网络爬虫中，有一种被称为“黑洞”的现象，即在抓取网页链接是，链接本身是一个无限循环。因此，爬虫抓取的时候也会循环，以致资源被白白浪费而没有任何效果。并且有些URL看起来不同，但实际值相同一张网页，也会是爬虫陷入重复循环之中。

为此，我们可以建立一张表格记录抓取的URL链接，每次新提取的URL插入队列之前需要检查表格记录中是否已经存在该项，如果是的话，直接舍弃。我们可以采用HashSet来实现创建记录表格，每次查找所需的时间均为O(1)，查询效率非常高。

我们已经知道仅仅通过比较URL链接是无法有效判断重复性的，为此我们需要提取网页中正文段内容，通过生成消息内容摘要的方式，作为文本比较的依据。关于消息报文摘要的生成方法有SHA-1算法，MD5算法，我们采取MD5方案，Java类库中已经为我们实现了该算法，只需要调用即可。

|  |
| --- |
| //调用MD5算法  MessageDigest md = MessageDigest.*getInstance*("MD5"); |

具体操作方案可以如下实施：先下载网页，生成MD5代码，再在visited表格中，查询该MD5是否已存在，若是的话，则从该网页中获取的新的URL均会被抛弃。否则，接受该网页中的所有链接。

这种方案最大的优点就是不需要比较整篇文章，效率高，比较方式简洁。也存在一些缺点，例如对于转载文章，该策略就失效了。

1. 数据库存储模块

为了合理管理下载下来的HTML文件，我们需要在数据库中存储相关记录项，下面给出了数据库表格的设计方案：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据库表格设计方案 | | | | |
| 数据项名 | 数据类型 | 长度 | 完整性约束 | 备注 |
| ID | Int | 4 | 主键,非空,唯一 | 自增编号 |
| Title | varchar | 200 |  | 网页标题 |
| url\_link | varchar | 200 | 非空 | URL链接 |
| contentMD5 | varchar | 45 | 非空，唯一 | MD5计算值 |
| cos\_relavancy | double | 8 |  | 余弦相关度 |

为了便于管理数据库操作，我们将数据库连接建立，查询操作，插入操作均封装到一个类中，减少功能模块之间的耦合度。基于数据库安全性的考量，我们抛弃非安全的Statement类，选用PreparedStatement类，撰写查询，插入语句时，会带来一些复杂度，但是会提高数据库的安全性等级，下面给出一个样例：

|  |
| --- |
| //插入语句  String sql="insert into url\_table values(null,? ,? ,?,?,?)";  PreparedStatement pstmt=*conn*.prepareStatement(sql);  //设置参数  **for**(**int** i=1;i<=params.length;i++){  pstmt.setString(i, params[i]);  }  //执行更新操作  pstmt.executeUpdate(); |

同时我们考虑到，数据库更新记录操作不需要和线程同步，可以采用异步更新的策略，这样我们可以节省使用一些锁机制(相当耗费性能)。实可以采用如下策略实施：在数据库存储类中创建一个队列和一个计时器，计时器的功能是每个一段时间，触发执行数据库更新操作，队列的功能是保存有效的URL链接（这些链接均为有效链接，与主题高度相关）。每个工作线程均可以向队列中插入URL链接，另一个线程每隔一段时间被触发，执行插入操作。这样的批量操作能减少数据库更新带来的性能损失。

#### 实验结果与总结

1. 实验运行结果截图：



图1-2 实验界面，可以设置主题，初始种子URL，

同时还要给出关键词集合

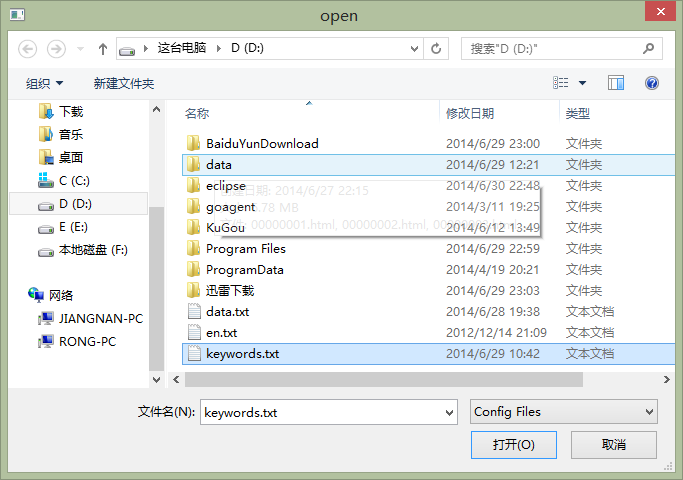


图1-3 选择包含关键词集合文件

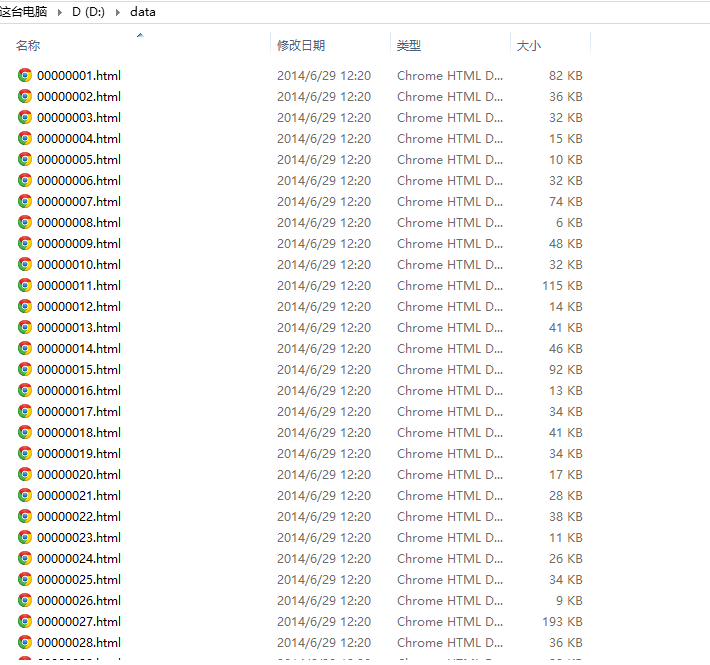


图1-4 程序结束后，在文件夹下保存了

一系列主题高度相关的HTML文件



图1-4 程序结束后，在数据库中保存了

一系列主题高度相关的URL链接记录项

1. 实验结论：

本次实验实践了主题爬虫的设计方案，实现了最基本的逻辑功能，同时还遇到了一些实践环节的编码问题，加深了对java的理解与使用。同时我们还可以对下载下来的网页进行进一步处理，利用成熟的算法提取有效的信息。例如：网页中还有一些doc,pdf文件，也可对其做解析，若是高度相关，我们也需要将其下载下来。同时网页中还有一些不相关的广告，如何辨别并去除这写无效链接将会是一个重要研究方向。

原本计划使用google 的pagerank算法辅助评价网站的重要性，采用的策略是通过URL链接访问google官网，获取返回的pagerank值，由于国内尚未架设google的服务器，导致链接经常超时中断，无法正常使用该服务，再加上时间仓促，这一方面的工作不了了之。

#### 参考文献

1. 赵泉等编著.信息检索.机械工业出版社,2008

2. 李培编著.信息检索与信息融合.人民邮电出版社,2007

3. 王芳,陈海建. Microcomputer Applications.深入解析Web主题爬虫的关键性原理.2011

4. 汪涛,樊孝忠. 计算机应用.主题爬虫的设计与实现.2004