

文章编号:1001-9081(2007)05-1116-03

网络化制造资源垂直搜索引擎的研究与应用

张 建^{1,2}, 程 锦³

(1. 武汉理工大学 经济学院, 湖北 武汉 430070; 2. 贵州省农业科学院, 贵州 贵阳 550006;

3. 贵州大学 CAD/CIMS 工程技术中心, 贵州 贵阳 550003)

(jagua.c@gmail.com)

摘 要:着重研究了网络化制造资源垂直搜索系统的主题爬虫和中文分词技术。通过在主题爬虫中增加评价网页模块, 优先爬行与主题相似度高的网页中的链接, 提高了爬虫的工作效率。在对中文分词词典进行分层存储的基础上, 通过一种改进的简洁的中文分词词典匹配算法, 有效地改善了分词的速度与精度, 并缩减了索引库, 增强了用户的响应。

关键词:网络化制造; 制造资源; 垂直搜索引擎; 页面解析

中图分类号: TP393.09 **文献标识码:** A

Research and application of vertical search engine in networked manufacturing resource

ZHANG Jian^{1,2}, CHENG Jin³

(1. Department of Economy, Wuhan University of Science and Technology, Wuhan Hubei 430070, China;

2. Guizhou Agriculture Science Institute, Guiyang Guizhou 550006, China;

3. Institute of CAD/CIMS, Guizhou University, Guiyang Guizhou 550003, China)

Abstract: This paper put emphasis on the technologies of the system, including the topic crawler and the Chinese word segmentation. To improve the efficiency of the crawler, a model of page evaluation was added into the crawler module; therefore the urls in a page with a high similarity of the topic will be first crawled. Besides, an improved word matching algorithm was proposed to enhance the speed and precision of word segmentation.

Key words: networked manufacturing; manufacturing resource; vertical search engine; html parser

0 引言

对制造资源有效而准确的检索是企业信息化的基础,也是实现网络化制造的必备条件之一。网络化制造的核心是利用网络,跨越不同企业之间存在的空间差距,使制造空间充分外延,实现企业间的资源共享、优势互补、优化组合配置,从而缩短产品的研制周期和费用,提高整个产业链和制造群体的竞争力^[1,2]。但是目前的通用搜索引擎提供的检索难以达到这个要求。

而针对某一特定领域、人群或需求设计的垂直搜索引擎能很好地满足这一需求。它是一种专业搜索引擎,只搜索网络中特定的主题信息,并对信息进行聚合、索引等相关处理,从而提高了用户检索的准确率和精确率。

但是目前在网络环境下如何进行制造资源的智能化搜索和管理还较少。国内的研究主要有:蔡铭、林兰芬等^[3]设计了一个原型系统 Swirrm,以网络化制造本体作为语义和推理支撑,提出多层次信息智能检索模型,完成用户透明化的智能检索;张博锋、周传飞等^[4]提出了制造资源搜索引擎(Manufacturing Resource Search Engine, MRSE),是基于构件的软件开发方法来实现;张英杰等^[5]进行了网络化制造资源智能管理系统的研究等等。

在对网络制造资源垂直搜索原型系统进行设计的基础上,提出系统体系结构,研究主题爬虫的设计、页面解析、中文分词、制造资源索引与检索技术等。论文最后对搜索测试用例进行了设计,并对本系统与 Nutch 测试结果进行了比较分析。

1 系统设计

1.1 框架设计

在全球化制造环境下,制造资源管理系统通过虚拟企业的伙伴选择,制造需求与制造能力和资源之间的匹配选择,实现分散化制造资源的快速配置与应用^[6]。从而达到网络化制造的目的。

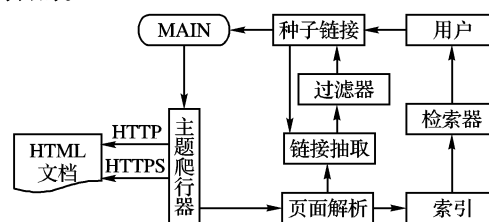


图1 垂直搜索引擎系统总体架构

目前,企业在伙伴选择上通常采用关键词匹配搜索,如Yahoo、Baidu等。但是,在这种方式下搜索结果有很大的干扰性。用户所需要的信息可能淹没在返回的海量检索结果中,让用户难以发现。针对这种情况,本文设计的面向网络制造资源的垂直搜索引擎能很好地解决这个问题。因本系统只收录机械行业的网页,并按相似度给网页排序,使检索返回的结果范围大大缩小。这样有效地提高了系统的查准率,减少了用户的检索时间。

系统主要分为两个模块,爬行模块和索引模块。其中爬行模块包括种子链接准备和获取、相似度分析等。索引模块

收稿日期:2006-11-27;修订日期:2007-02-16 基金项目:国家自然科学基金资助项目(5047185)

作者简介:张建(1955-),男,贵州贵阳人,教授,博士,主要研究方向:企业信息化;程锦(1981-),男,江西于都人,硕士研究生,主要研究方向:企业信息化、计算机集成制造。

包括页面解析和中文分词。整个系统体系结构如图1所示。

1.2 主题爬虫设计

主题爬虫是垂直搜索引擎的基础与核心。它主要是在普通爬虫的基础上增加相似度分析模块、种子获取模块、排序模块。主题爬虫的伪代码如下所示:

```
Topic-Spider ( starting_urls ) {
    foreach link ( starting_urls )
        EnQueue( frontier, link );
    while ( isEmptyQueue() ) {
        link := DeQueue_Link ( url_queue );
        doc := fetch( link );
        if ( sim( Q, P ) > = r ) {
            EnQueue( frontier, extract_links( doc ) );
        } else continue;
    }
}
```

EnQueue()是将链接加入到 url_queue; DeQueue_Link()是将链接取出后并为之删除。

1.3 相似度计算

主题爬虫与普通爬虫的本质区别在于前者只保存与主题相关的网页,删除与主题无关的网页。

如何判断网页与主题的相关性?本文采用向量空间模型方法。将文档的词条项提取出来,映射为一个特征向量 $V(d) = (t_1, \omega_1(d); \dots; t_n, \omega_n(d))$, 其中 $t_i (i = 1, 2, \dots, n)$ 为一列互不雷同的词条项, $\omega_i(d)$ 为 t_i 在 d 中的权值。在信息检索中常用的词条权值计算方法为 TF-IDF 函数见公式(1):

$$\omega_i(d) = \frac{tf_i(d) \log\left(\frac{N}{n_i} + 0.1\right)}{\sqrt{\sum_{i=1}^n (tf_i(d))^2 \times \log^2\left(\frac{N}{n_i} + 0.1\right)}} \quad (1)$$

其中 N 为所有文档的数目, n_i 为含有词条 t_i 的文档数目, $tf_i(d)$ 为 t_i 在 d 中出现的频率。

为了增加相似度分析的准确性。在对分档分词之后,根据词条的权值 $\omega_i(d)$ 以及两文档的相似度可用向量的夹角余弦公式来表示,因此可删除频率高但对主题又无影响的字词。两文档 d_i, d_j 的相似度可表示为公式(2):

$$Sim(d_i, d_j) = \cos\theta = \frac{\sum_{k=1}^n \omega_k(d_i) \times \omega_k(d_j)}{\sqrt{\left[\sum_{k=1}^n \omega_k^2(d_i)\right] \left[\sum_{k=1}^n \omega_k^2(d_j)\right]}} \quad (2)$$

指定一个阈值 r , 当 $\cos < d_i, d_j > \geq r$ 时就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,取值将在测试时说明。

1.4 改进的 RMM 算法

汉语自动分词是汉语信息处理的前提。本系统中的页面解析模块和索引模块都需要分词技术。通常分词技术分为两类^[7]: 第一类主要基于字典、词库的匹配和词的频度统计。它主要有三部分组成: 词库(典)、分词算法、词典机制; 第二类是无词典切分, 主要基于句法、语法分析, 并结合语义分析, 通过对上下文内容所提供信息的分析对词进行界定, 这类方法一般不易实现。

常见的切词算法如下:

1) 最大正向匹配法(Maximum Matching method, MM)的基本思想为: 设 D 为词典, MAX 表示 D 中的最大词长, str 为待

切分的字串。MM 法是每次从 str 中取长度为 MAX 的子串与 D 中的词进行匹配。若成功, 则该子串为词, 指针后移 MAX 个汉字后继续匹配, 否则子串逐次减一进行匹配。

2) 逆向最大匹配法(Reverse Maximum Matcing method, RMM)。基本原理与 MM 法相同, 不同的是分词的扫描方向, 它是从右至左取子串进行匹配。

统计结果表明, 单纯使用正向最大匹配的错误率为 $1/169$, 单纯使用逆向最大匹配的错误率为 $1/245$, RMM 法在切分的准确率上比 MM 法有很大提高。本系统选用 RMM 算法。为了进一步提高效率, 笔者设计了一个改进的专业词典存储格式及相应的切词算法。通过该算法, 系统根据词典对文档进行专业词语的切分, 而文档中非专业词语则不进行切分。因此提高了切分速度, 另外也精简了索引库, 增强了对用户的响应。

1.4.1 词典存储

本系统采用的词典为机械行业的专业词典。词典采用分层存储^[8]的形式, 一共分为3层, 如图2所示(每一个字母代表一个字)。

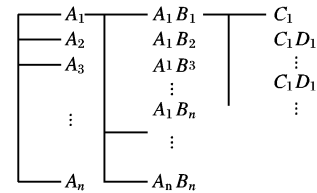


图2 词典分层存储格式

第一层存储所有单字。第二层保存所有的双字词和多字词的前两个字。每一个可成词的单字对应一系列第二层节点, 用来存储所有以该字为词首的双字, 并保存该词下层的最大长度 L 。第三层逐一存储以某一双字为首的所有词, 为了减少存储空间, 只存储除去该双字以外的部分。采用这种存储结构, 数据结构简单易于实现, 减少了指针运算, 有利于提高分词效率。

1.4.2 匹配方法

由于词库中的最大词长通常大于所切分出的词长。为了提高切分的效率, 匹配方法采用逐次减一个字的方法。

假设对一个句子 $C_1 C_2 \dots C_n$ 进行分词处理, 具体算法如下:

- 1) 初始化 $i = 1, j = 0, strl$ 为待切分字符串;
- 2) 从 $strl$ 中取出 $C_{n-L-j-i} \dots C_{n-j}$, 调用 MM 算法判断是否存在;
- 3) 不存在, 则 $i++$ 转 2);
- 4) 存在, 则 $C_{n-L-i} \dots C_{n-j}$ 为词, 保存, $j = i, n = n - L + i$ 转 2);

该算法的主要思想是在网页转换成文本的基础上, 进行逐句切分, 在 RMM 的基础上用 MM 算法来判断该词是否存在, 提高分词的精度。MM 算法详细步骤如下:

- 1) 初始化 $j = 0, i = 1$;
- 2) 从 $strl$ 中取出 $C_{i+j} C_{i+j+1}$, 判断是否存在;
- 3) 不存在, 则 C_{i+j} 为单字词, 返回 False;
- 4) 存在, 则取出该词下层的最大长度 n ;
- 5) 若 $n = 0$ 或 1 , 则 $C_{i+j} C_{i+j+1}$ 为词, 返回 False;
- 6) 若 $n = 2$, 返回 True;
- 7) 否则返回 False。

1.5 索引与检索

页面解析之后, 即可进行索引。本文索引采用第三方开源工具 Lucene。Lucene 是 apache 软件基金会 Jakarta 项目组的一个子项目, 是一个开放源代码的全文检索引擎工具包, 提供了完整的查询引擎和索引引擎。

1.5.1 Lucene 用法

由 Lucene 的类 `IndexWriter` 创建索引时,需要指出待索引文档存放目录和存放索引的目录。另外要注意的是必须替换 Lucene 自身提供的分词算法,以达到系统的分词需求。索引建立之后,由类 `IndexSearcher` 建立检索工具,建立时需设置索引存放路径。然后使用 `QueryParser.parser()` 方法构建一个 `Query` 类型的查询对象,最后将该对象传给 `IndexSearcher` 类的检索接口,完成查询。相关细节不在此累述。目前基于 Lucene 的应用已经比较成熟,有大量文献资料可查,如参考文献[9]等等。

1.5.2 Java 解析 HTML

由于 Lucene 只索引文本文件。所以在对页面进行索引之前,还需对页面进行预处理,去除图片链接、Script 代码,抽取出网页内容。功能上并不复杂,用软件包 `javax.swing.text.html.parser` 和 `javax.swing.text.html` 即可。主要用以下几个类 `HTMLEditorKit.Parser`, `HTMLEditorKit.Parser`, `HTML.Tag`, `HTML.Attribute`。

第一步是实现 `HTMLEditorKit.Parser.getParser()` 方法的 `public` 访问性。代码如下:

```
import javax.swing.text.html.*;
public class ParserGetter extends HTMLEditorKit {
    public HTMLEditorKit.Parser getParser() {
        return super.getParser();
    }
}
```

第二步是先构建 `ParserCallBack` 类的子类,实现解析待处理的文件,再重写 `handleComment`, `handleEndTag`, `handleError`, `handleSimpleTag`, `handleStartTag` 和 `handleText` 方法。当 `HTMLEditorKit.Parser` 对象在解析 HTML 文件时,遇到不同的标签则调用相应的回调方法来处理。

第三步是将文档写入 `Reader` 流中,传给 `HTMLEditorKit.Parser`。首先通过抽象类 `URLConnection` 建立与 URL 指定的数据源的动态链接。再实现 `HTMLEditorKit.Parser` 类和 `HTMLEditorKit.ParserCallBack` 类的实例化,最后使用 `HTMLEditorKit.Parser` 对象的 `parse()` 方法。

Swing 中包含了完整的 HTML 解析器功能,用户可以使用它进行开发,解析 HTML 文档,从而达到灵活适用的目的。

2 测试分析

Mencze^[10]等提出用于评价主题网络爬虫系统的指标同传统信息检索一样主要有两个:(1)爬准率(Precision);(2)爬全率(Recall)。定义分别为: $Precision = \frac{\text{采集的目标网页数}}{\text{总爬行数}}$; $Recall = \frac{\text{采集的目标网页数}}{\text{总的目标页面数}}$ 。

为了评价该系统的性能,将本系统与 Nutch 的搜索结果进行了比较。Nutch 是 Apache Software Foundation 开发的一个开源搜索引擎,用 Java 编写。可以为用户构建专业的搜索引擎。Nutch 可用于本地文件系统的检索、内部互联网的检索和全球互联网的检索。有关 Nutch 的使用可参见官方网站^[11]。

在测试中,所选平台为装有 Windows XP SP2, MS-SQL Server 2000、JDK5.0、IntelliJ IDEA 等。计算机为 P4 CPU 2.66GHz,内存 1GB。Nutch 执行抓取时,线程数设为 500、深度为 3。种子链接两者相同,都由人工选择,尽量包括各种成型方法的中心网页,并指定一些目标页面,取阈值 $r = 0.1$ 。实验结果如图 3、图 4 所示。

通过图 3、图 4 的比较,由于该系统经过页面解析,优先爬行跟主题相似度高的网页中的链接。该系统的爬全率、爬准率明显优于 Nutch。

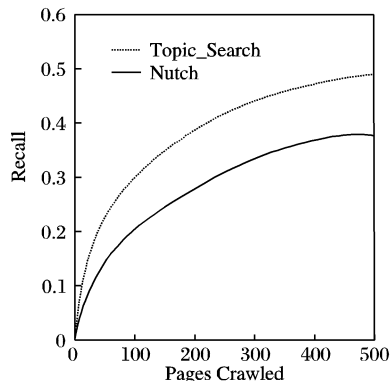


图3 爬全率曲线图

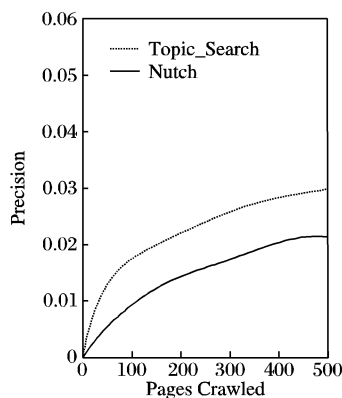


图4 爬准率曲线图

3 结语

本文从理论和实践两个方面分析了面向网络制造资源垂直搜索引擎的可行性,并充分说明一个主题爬虫设计方案的可行性。以本文研究的主题爬虫为基础可以开发主题搜索引擎,结合到具体应用主题爬虫可以在受限领域内进行面向主题的信息有效采集。将来的工作是进一步改善中文分词,增加未登录词识别与歧义处理模块。

参考文献:

- [1] 李少波, 谢庆生. 基于 ASP 的动态联盟制造资源管理框架研究[J]. 中国机械工程, 2005, 16(6): 502-506.
- [2] 谢庆生. 我国制造业 ASP 发展的模式与发展策略[J]. 数字制造科学, 2004 1(1): 66-70.
- [3] 蔡铭, 林兰芬, 董金祥. 制造资源智能检索系统研究与实现[J]. 计算机辅助设计与图形学学报, 2004, 16(4): 543-548.
- [4] 张博峰, 周传飞, 方爱国, 等. 基于构件技术的制造资源搜索引擎研究与实现[J]. 2005, 26(9): 2268-2270.
- [5] 张英杰, 曹岩. 基于网络化的制造资源智能管理系统的研究[J]. 西安交通大学学报, 2004, 38(3): 270-273.
- [6] 何汉武, 郑德涛, 陈新, 等. 面向虚拟企业构造的合作企业搜索方法研究[J]. 计算机集成制造系统, 2005, 11(6): 862-868.
- [7] 孙茂松, 左正平, 黄昌宁. 汉语自动分词词典机制的实验研究[J]. 中文信息学报, 2000, 14(1).
- [8] 吴栋, 滕育平. 中文信息检索引擎中的若干技术[J]. 计算机应用, 2004, 24(7): 128-131.
- [9] 李刚, 宋伟, 邱哲. 征服 Ajax + Lucene 构建搜索引擎[M]. 北京: 人民邮电出版社, 2006. 193-308.
- [10] MENCZER F, PANT G, RUIZ M. Evaluating topic-driven web crawlers[A]. Proceeding ACM SIGIR[C]. 2001.
- [11] <http://lucene.apache.org/nutch/tutorial.html> [EB/OL], 2006-04-10.