使用 JTree 和 XPath 构建动态网页信息抽取系统

JTree and XPath Based Information Extraction on Dynamic Web Pages

董　旻[1,2]　　方　曙[2]　　杨志萍[2]

（1. 中国科学院研究生院　北京　100049；2. 中国科学院国家科学图书馆成都分馆　成都　610041）

摘　要　提出一种利用 JTree 和 XPath 技术实现动态网页信息抽取系统的方法，介绍了系统主要组件的功能和实现方法，并进行实际的抽取试验，论述了此设计的优点。

关键词　动态网页　信息抽取　JTree　XPath　包装器

1　引　言

随着互联网信息的不断增长，人们越来越多地使用搜索引擎来寻找自己需要的信息。研究表明，搜索引擎所能搜索到的信息资源只是能被其建立索引的部分，而很多信息资源不能被搜索引擎索引到[1]。比如专利、文献数据库内的信息，往往需要人们通过查询表单来进行访问。实际上这些不能被搜索引擎索引到的信息内容更加专门化并具有更高的价值。其次，搜索引擎主要是从大量的文档集合中找到与用户需求相关的文档列表；而信息抽取系统则旨在从文本中直接获得用户感兴趣的事实信息。在实际的情报研究中，需要进行统计分析的对象是网页或者文本中特定字段的内容，而不是整个网页或者文本本身。为了能够更有效地支撑情报研究工作，方便地获取和利用这部分信息，需要设计专门的信息抽取工具、包装器（Wrapper）[2]等对这些信息资源进行抽取和整理。本文所要介绍的就是一个实现了信息自动抽取的系统，与其他信息抽取程序比较，这个系统在使用界面上对用户更加直观友好，在抽取功能上实现了动态的包装器自动生成，即对不同的信息资源都能自动生成对应的抽取程序，该系统在中科院西部之光课题“西部地区科研产出能力研究”中得到较为广泛的应用，被应用于各种数据库的特定信息抽取工作上。

通过分析这些不能被搜索引擎索引到的网页信息，发现有以下两方面的特点：a. 动态网页或网页的动态生成。这些网页并不稳定存在于服务器端，只有当用户通过访问接口访问资源数据库时，服务器才动态生成这些网页并产生一个含有用户信息变量在内的 URL 地址指向这些页面。这也是其很难被搜索引擎索引到的原因。b. 网页结构上的一致性。这些由数据库生成的网页在减轻网站的页面维护负担的同时，按网页模版批量生成的页面在结构上是基本一致的。

2　系统模型和相关技术

本文实现的系统是建立在动态网页结构相同的基础上的。对于这些具有相同结构的网页集合，只需取其中一个页面作为样本进行分析，自动生成包装器对象，就可以对整个具有相同结构的网页集合进行信息抽取工作。系统模型如图 1 所示。

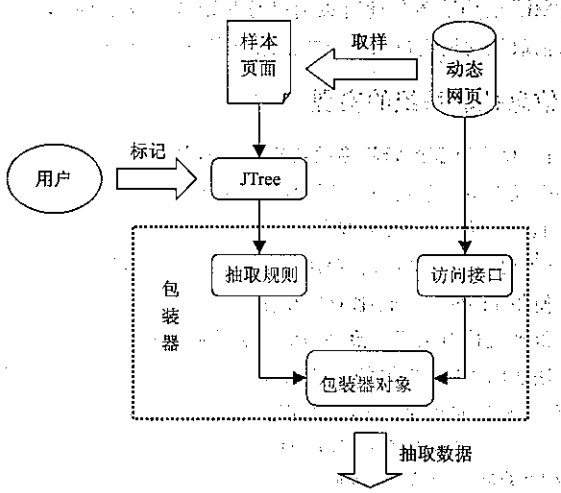

图 1　抽取系统模型

整个系统的工作流程如下：从动态网页集合中任选一个样本页面，由样本页面解析程序对其解析，生成一个树型结构以供用户标记出感兴趣的节点，然后系统读取树上被用户标记过的节点，通过算法自动生成信息抽取规则并封装到包装器对象内，对于该动态网页集合，都可以用这个包装器对象进行自动的信息抽取，输出用户感兴趣的信息域的内容。这个系统模型的实质是描述了一个包装器对象的自动生成过程。

假设待抽取的资源集合是 ε，抽取得到的信息集合是 i，那么包装器就是一个从 ε 到 i 的映射或者函数。对于不同的 ε 集合，通过包装器构造函数 wrapper(p) 可以得到针对该集合的包装器对象 w，其中 p 是集合 ε 的一个子集或者元素，也就是本文中所说的样本页面。包装器对象 w 就是用来处理资源集合 ε 并得到抽取信息 i。包装器的核心内容就是如何面对不同的待抽取资源集合 ε，生成相应的包装器对象 w。在系统中，通过对 ε 集合进行取样 p，生成 JTree 提供给用户进行标记，针对用户的

作者简介：董　旻，男，1981 年生，研究生，研究方向为知识管理与情报计量；方　曙，男，1957 年生，馆长，研究员，研究方向为情报计量学、知识管理、战略情报；杨志萍，女，1967 年生，馆员，研究方向为信息咨询、科技情报。

标记信息通过算法分析自动生成包装器对象用于对整个 ε 集合进行信息抽取。

从待抽取资源中取样，是为了分析样本页面中人们所需要的信息内容在页面中的位置分布以生成对应的包装器对象。而实际上样本页面的 HTML 源代码是用于客户端浏览器解释执行的使用目的，因此人们所关心的信息内容往往和大量无关代码及标签混杂在一起。对此，这里使用了一个树型结构来重新组织并展示样本页面的结构特征，方便用户将待抽取的信息内容标记出来。系统采用 JTree 来实现这个树型结构，JTree 是 Java 语言中的一个 GUI 组件，使用 JTree 来表示样本页面的结构，一方面使用户可以直接在样本页面上标记需要的信息内容；另一方面由于树型结构本身的特点，可以比较方便地由用户标记的信息生成信息抽取规则并封装成包装器对象。在本系统中，使用 XPath 表达式的形式来描述信息抽取规则。对于 XML 文档，XSL(可扩展样式表语言，Extensible Stylesheet Language)使用 XPath[3] 来标识 XML 文档中的元素位置。因此对整个动态网页信息集合，包装器对象首先将其转换成 XML 或 XHTML 的结构，然后就可以通过分析取样所生成的 XPath 形式的抽取规则把用户感兴趣的信息内容一一定位并提取出来。

3 信息抽取系统的实现

3.1 样本页面解析程序的实现 样本解析程序读取样本页面并生成 JTree，这是一个遍历样本页面中 HTML 标签的过程。由于 HTML 结构上的不规范，为了方便地遍历 HTML 中的标签，首先要对样本页面向 XML 格式或 XHTML 格式进行转化。系统使用 JTidy 工具包把 HTML 源文件转换生成 XML 文档对象并得到文档对象根节点 docRoot，然后使用一个深度优先的遍历算法访问 XML 文档的子节点并将其加载到 JTree 的树根节点下，逐渐生成整棵 JTree。实现生成 JTree 的算法如下。

```
TreeNode makeTree(DocNode docRoot) {
  /* 由当前文档节点生成树节点 */
TreeNode treeNode=new TreeNode(docRoot);
  if 当前文档节点有子节点
    n=子节点数目;
  for i=0 to n//遍历文档子节点
    if 子节点 docNode[i]是叶子节点
    将该节点加到当前树节点 treeNode 下;
    else 子节点 docNode[i]有下级节点
      递归调用本方法 makeTree(docNode[i]);
      将递归返回的树节点加到当前树节点 treeNode 下;
if 当前文档节点没有子节点
    return 当前树节点;
}
```

以上是一个递归的方法，读取的参数 docRoot 即为样本页面 XML 化后的文档根节点，返回的参数 treeNode 是生成的 JTree 的当前根节点，JTree 的所有子节点都已经在上述方法的递归遍历中加载到返回的树节点下。最后只需要调用 JTree 的默认构造方法加载这个节点就生成了整棵 JTree，具体实例如图 2 所示。

系统读取样本页面然后生成 JTree 提供给用户，由用户直接在 JTree 上标记感兴趣的信息内容(树的节点)。如图 2 所示，这是一份由专利信息样本页面生成的 JTree，用户标记了专利名称、公开号、公开日这 3 个节点作为待抽取的对象，下一步，系统将根据用户的标记自动生成针对这些标记对象的抽取规则。

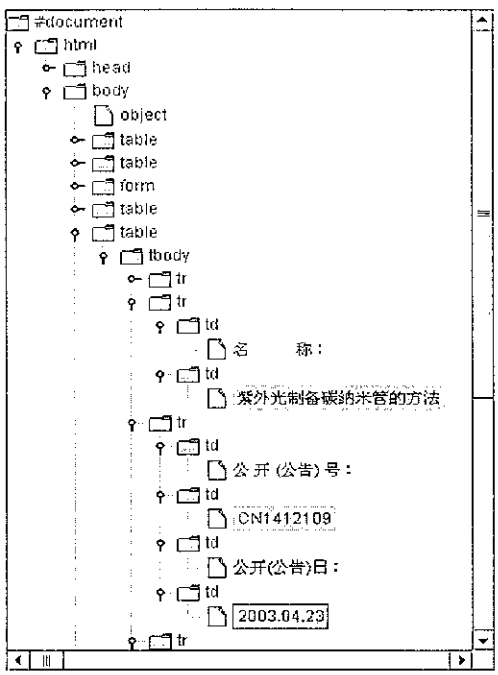

图 2 由样本页面生成的 JTree 具体实例

3.2 抽取规则的生成算法 生成抽取规则的算法是包装器的核心，这里使用的是 XPath 表达式形式的抽取规则。系统通过用户标记节点的信息自动生成抽取规则并封装到包装器中，如此动态地构造包装器对象可以满足不同抽取信息来源的需要。

系统使用 JTree 的 getSelectionPaths()方法获得用户标记节点的路径。例如图 2 中被标记的“紫外光制备碳纳米管的方法”节点，它在 JTree 中的路径输出是[#document, html, body, table, tbody, tr, td]，直接转换成 XPath 路径就是/html/body/table/tbody/tr/td。但这个 XPath 路径并不能保证抽取程序准确定位到用户指定的节点。如前面得到的 XPath 路径/html/body/table/tbody/tr/td，并不唯一表示“紫外光制备碳纳米管的方法”节点的位置，它也同样表示图 2 中“名称：”节点等，在这里对这些由标记节点直接得到的 XPath 路径为基本路径。为了能够取得用户标记节点的准确 XPath 路径，系统在此使用一个筛选算法排除基本路径下所有不符合的节点，算法如下。

```
List parse() {
    rule=抽取规则集合;
    /* 选出所有符合基本 XPath 路径的节点 */
    nodeList=符合基本 XPath 路径的节点列表;
    n=符合基本 XPath 路径的节点数;
    m=用户标记的节点数;
    for i=0 to n//遍历验证每一个节点
      valueN[i]=节点 node[i]的值;
      for j=0 to m//遍历用户标记节点的值
      if valueN[i]与用户标记节点的值 valueT[j]相等
        xPath=节点 node[i]的准确 XPath 路径
```

```
        rule.add(xPath);//添加这条规则
        end if
        end for
      end for
      return rule;//返回抽取规则集合
}
```

算法的基本思路是根据 JTree 上用户标记得到的 XPath 基本路径枚举出所有符合的节点，然后将这些节点的值与用户标记节点的值比对，当相等时说明这个节点就是用户标记的节点，使用该节点对象的 getUniquePath()方法就得到准确的 XPath 路径。

假设 JTree 上用户标记的节点数目是 m 个，而样本页面上一共有节点 n 个，这个筛选算法的时间复杂度是 $O(m*n)$。考虑到无论专利信息还是文献信息，页面上的相关信息条目一般不会超过 20 项，实际上大部分都在 10 项左右，以该算法的效率生成抽取规则是可行的。实际试验中在 m 接近 10、n 在 20～50 的范围内的情况下生成抽取规则所花费的时间可以忽略不计。

使用上述算法可以计算出图 2 中用户标记的 3 个节点的准确 XPath 路径是：/html/body/table[4]/tbody/tr[2]/td[2]

/html/body/table[4]/tbody/tr[3]/td[2]

/html/body/table[4]/tbody/tr[3]/td[4]

方括号内的数字表示该节点是其父节点的第几个子节点，从图 2 中验证可知这个路径是准确并能唯一确定用户所标记的节点位置的。

3.3 **包装器工厂** 包装器工厂(Wrapper Factory)，套用了程序设计领域设计模式(Design Patterns)中的工厂模式的概念。针对不同的待抽取资源，包装器工厂读取样本页面 p，通过构造函数 wrapper(p)构造出不同的包装器对象来进行信息抽取。因此，包装器工厂实际上也就是抽取程序的生成模块。

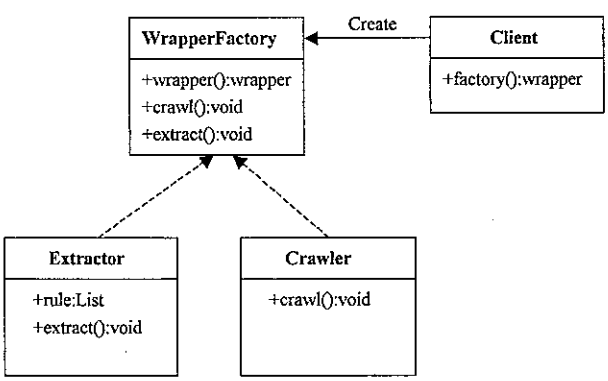

图 3 包装器工厂

图 3 是包装器工厂的类图，工厂根据将样本页面作为输入参数，返回生成的包装器对象。该包装器对象由两个部分组成，一个是访问待抽取资源的接口，一般是个爬虫程序(Crawler)，另一部分就是本文所讨论的包装器的核心、抽取规则及算法。在图 3 中，Extractor 封装了 XPath 形式的抽取规则(rule 对象)，并根据规则使用 extract()方法进行抽取工作，例如对于图 2 中用户标记的 3 个节点进行抽取，实现代码如下。

```
void extract(Document doc){
    name=doc.selectSingleNode(rule.get(1)).getStringValue();
    pubNum=doc.selectSingleNode(rule.get(2)).getStringValue();
    pubDate=doc.selectSingleNode(rule.get(3)).getStringValue();
}
```

doc 参数是待抽取的文档对象，rule 对象是通过解析用户在 JTree 上的标记的节点而得到的 XPath 路径集合。将这个路径应用到待抽取资源集合上，使用 dom4j 工具包实现的 selectSingleNode(XPath Expression)方法取得标记位置上的节点对象，并调用 getStringValue()方法获取节点的值，也就是用户所需要的信息内容。

4 试 验

对 USPTO[4]的专利数据进行抽取试验，取 100 个专利信息页面作为待抽取信息集，任选一个页面作为样本并标记页面上所有与专利有关的信息，这个过程重复 5 次，生成 5 个包装器。对包装器抽取结果进行人工验证，试验结果如表 1 所示。

表 1 包装器抽取结果

取样	标记节点数(个/样本页)	抽取准确率(%)
样本 1	11	93
样本 2	8	83
样本 3	11	96
样本 4	12	95
样本 5	11	89

表中的准确率描述的是抽取出的信息内容是否和标记的内容一致程度。造成不一致的原因是页面结构的差异性。如果某个待抽取信息的位置在某个页面上比其他页面有偏移，那么利用 XPath 定位的抽取规则所能定位到的是原位置上不相符的信息内容。例如标记的节点有专利名称，包装器根据该标记抽取的信息集合应该就是 100 项专利名称。而实际抽取出的专利名称只有 96 项，其余 4 项为空或者是不相关的页面信息。准确率的计算公式如下：

$$\text{抽取准确率} = \left(\sum_{\text{样本集}} \frac{\text{该页面抽取出的正确信息数}}{\text{页面信息内容数量}}\right) \div \text{样本数} \times 100\%$$

以上实验是在标记所有节点、抽取所有相关信息内容的条件下进行，而在实际情况中，用户可能只需要页面上几项信息内容，在这种情况下准确率会有进一步的提高。

5 结 语

针对动态网页这种网络资源的信息抽取工作，系统主要有以下两个方面的优势：一是对用户的友好性。抽取信息的使用者，比如情报分析人员，本身往往不懂得设计抽取程序。在需要对某种信息资源进行抽取时，用户首先要与抽取程序设计人员进行沟通，设计人员再根据用户的需求，分析待抽取的资源的结构、设计程序、生成包装器等，然后进行抽取工作把抽取的信息提交给用户或者让用户自己运行抽取程序。而本系统通过一个树型界面直接让用户标记所需要的信息内容，然后自动将标记信息翻译成抽取规则并开始抽取工作，完全不需要程序设计人员的干预。

另一个优点是包装器的动态生成技术。用户所感兴趣的往往不止一种信息资源，比如需要的信息可能分布在多个数据库内。这种情况下，普通的抽取工作需要对每一个 (下转第 78 页)

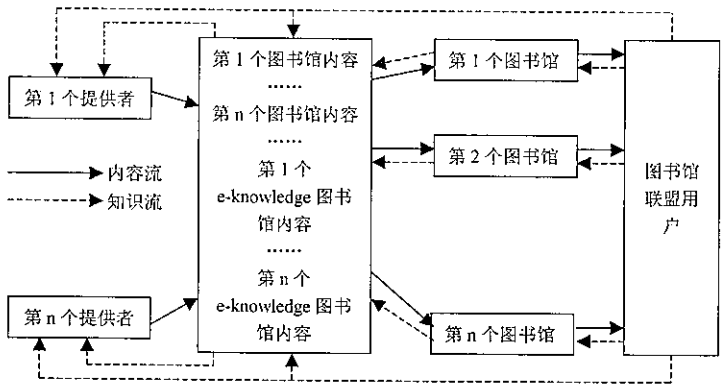

图 1 图书馆联盟知识管理系统模型

上述工具和系统并非全部具有可借鉴价值,但目前可供参考的图书馆知识管理工具和系统具有以下特点:a.多数是部分解决方案。目前,图书馆建设的知识管理系统,如知识库、知识交流工具、知识共享工具等,多是部分解决方案,而非系统化的、全面的知识管理系统解决方案。这些工具只能在某一领域发挥作用,不能全面地管理图书馆内的隐性知识和显性知识。b.系统功能简单。目前可见的图书馆知识管理系统只有三个,这三个知识管理系统的功能相对简单,不能对图书馆全面引入知识管理技术和方法提供系统支持。c.针对某个馆的实际,特定性比较强。大多数图书馆知识管理工具(系统)是针对本馆的实际情况研究和开发的,如澳大利亚防卫图书馆的研究信息管理者系统,针对性非常强,无法移植到其他图书馆。d.缺乏可操作的系统建设方案。图书馆需要一个可操作的系统建设方案,此类通用系统解决方案是图书馆知识管理系统建设的关键。

正如 Charles T. Townley 所说:"图书馆很少运用业务信息创建或应用组织知识。没有为了运用组织知识而建构组织,也没有应用组织知识来改进服务或学术信息的传递[11]。"不仅如此,图书馆学领域或者很少进行知识管理引入到图书馆管理中的理论研究,或者只是停留于简单或重复的概念引用,而没有进行深入的理论研究和实践探讨。从上面的调查可以看出,国内外对此进行的研究并不深入,同时也存在着对知识管理的理解不同。笔者曾对国内外该领域的研究进行了调查[12],发现国内更多的是局限于理论的畅想,而很少有实践或实证研究。通过对国内、外对图书馆知识管理研究的调查,笔者认为需要加强以下两方面工作:其一,加强图书馆知识管理理论研究,研究的基本内容就是要建立科学的图书馆引入知识管理的体系,包括理论体系、技术体系和系统体系,从而指导图书馆管理实践;其二,建设图书馆知识管理系统,利用现代技术为图书馆引入科学管理理念服务,只有将理论与实践结合起来才能发挥理论的指导作用,从而进一步丰富和促进理论的发展。

参 考 文 献

1 C.T.汤力.知识管理与学术图书馆.国外社会科学,2001;(6)

2 Ronald C. Jantz. Knowledge Management in Academic Libraries: Special Tools and Processes to Support Information Professionals. Reference Services Review,2001;(1)

3 Qihao Miao. From Literature Center to Knowledge Portal: Shanghai Library in Search of Excellence 2.0. Library Review, 2001;(7)

4 中国科学院文献情报中心电子馆务系统. http://eoffice.las.ac.cn/eoffice/login.jsp

5 Sharon Teng, Suliman Hawamdeh. Knowledge Management in Public Libraries. Aslib Proceedings,2002;(3)

6 Iain Brown, Lee Williams. Delivering 'Information Capability: The Application of Knowledge Management in the Defence Library Service. http://www.csu.edu.au/special/online99/proceedings99/brown-williams.htm.2003-09-09

7 North Suburban 图书馆系统. http://www.infotoday.com/it2003/presentations/Stoll-Taylor-Handout.pdf.2003-09-21

8 Keitha Booth. Protecting the Library's Own History: Stepstowards Knowledge Management at the National Library of New Zealand te Puna. Lianza2004 conference

9 Ganesha 数字图书馆. http://gdlhub.indonesiadln.org/doc/brosur-gdl31.pdf.2003-09-09

10 Anthi Katsirikou. Consortia and Knowledge Management. The Functional Context and an Organizational Model. http://www.iatul.org/conference/proceedings/vol12/papers/Katsirikou.pdf.2003-09-09

11 Charles Thomas Townley. Knowledge Management and Academic Libraries. College and Research Libraries, 2001;62(1)

12 Charles T. Townley. Knowledge Management and Academic Libraries. College and Research Libraries,2002;63(1)

(责编:梅王京)

(上接第 75 页)信息源进行分析并设计抽取程序。而本系统通过包装器动态生成技术,充分满足用户各种需求。针对不同的待抽取资源,系统读取样本页面给用户进行简单的标记,然后根据标记信息自动生成包装器程序并完成抽取工作,节省了人力,提高了效率。

在实际的课题应用当中,本系统对于多种文献、专利等类型的网络资源库,都取得了较好的抽取效果,但实际上应用对象范围有限,并且其网页结构比较固定。而面对更多网页结构差异巨大的资源库,如何提高系统抽取准确率,使系统更具有通用性,是下一步研究工作的重点。

参 考 文 献

1 Michael K. Bergman. The Deep Web: Surfacing Hidden Value. http://www.press.umich.edu/jep/07-01/bergman.html (Accessed Sep.10,2006)

2 Nicholas Kushmerick. Wrapper Introduction: Efficiency and Expressiveness. Artificial Intelligence 118(2000)

3 Ling Liu, Calton Pu, Wei Han. An XML-Enabled Data Extraction Toolkit for Web Sources. Information Systems 26 (2001)

4 Patent Full-Text and Full-Page Image Databases. http://www.uspto.gov/patft/index.html (Accessed Jan.29, 2007)

5 张智雄.信息抽取技术及其在数字图书馆中的应用前景分析.现代图书情报技术,2004;(6)

6 Document Object Model (DOM) Specification. http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407 (Accessed Sep.20,2006)

7 Dennis Sosnoski. XML and Java Technologies: Document Models, Part1: Performance. http://www-128.ibm.com/developerworks/xml/library/x-injava/index.html (Accessed Oct.1,2006)

8 Arasu A, Garcia-Molina H. Extracting Structured Data from Web Pages. ACM SIGMOD 03 (2003)

9 Chamberlin D, Rebie J, Florescu D. Quilt: An XML Query Language for Heterogeneous data Sources. Proc International Workshop on the Web and Databases(WebDB'2000), Dallas, Texas, 2000

(责编:梅王京)