第32卷 第20期 计 算 机 工 程 2006年10月
Vol.32 № 20 Computer Engineering October 2006
·软件技术与数据库· 文章编号：1000—3428(2006)20—0043—02 文献标识码：A 中图分类号：TP311

基于 XML 的 Web 数据挖掘关键技术的研究

崔建群[1,2]，何炎祥[2]，郑世珏[1]，吴黎兵[2]
(1. 华中师范大学网络与通信研究所，武汉 430079；2. 武汉大学计算机学院，武汉 430072)

摘 要：由于存在着大量的在线信息，WWW 成为数据挖掘的热点。该文介绍了 Web 网页的数据挖掘技术，提出一种基于 XML 的 Web 数据挖掘模型，阐述将半结构化 HTML 文档转换成良构的 XML 文档的原因，并给出基于 HTML Tide 库的转换代码，介绍了利用 XML 技术从 Web 网页析取数据的关键技术，包括 XHTML、XSLT 和 XQuery 等，对 Web 数据挖掘的其他方面如数据检验和集成作了一定的探讨。
关键词：Web 数据挖掘；XML 模型；关键技术

Research on Key Technologies of Web Mining Based on XML

CUI Jianqun[1,2], HE Yanxiang[2], ZHENG Shijue[1], WU Libing[2]
(1. Institute of Network & Communication Technology, Huazhong Normal University, Wuhan 430079;
2. School of Computer, Wuhan University, Wuhan 430072)

【Abstract】With the huge amount of information available online, the World Wide Web is a fertile area for data mining research. This paper addresses the issues related to data extraction from Web pages, and strongly suggests an XML-based approach for solving it. This paper describes the motivations behind converting semi-structured HTML documents into well-formed XML and presents a portion of conversion source codes that is developed based on HTML Tidy library, illustrates how to extract desired information from Web pages with XML technologies, including XHTML, XSLT and XQuery. It also discusses other aspects in the Web mining project such as data check and data integration.
【Key words】Web data mining; XML-based model; Key technologies

随着Web信息技术的迅速发展，用户可以越来越方便快捷地获得各种信息，但同时也面临着如何从大量Web信息中得到相关和有用的信息问题。虽然通过使用Google、百度、Lycos等搜索引擎，可以大大减少无用信息的干扰，但是这些搜索引擎搜索的结果有时也不完整或不相关，很难完全满足用户的需求[1]。Web数据挖掘技术则可以解决过量信息的问题，为用户提供更精确、更相关的数据。

目前Web上的大量信息都以超文本标记语言HTML文档的形式出现，用户通过浏览器浏览这些HTML文档来获得信息。HTML文档可能手工编写或利用HTML工具进行编写，由于HTML文档的目的并不是为了自动析取而是为了将信息内容表达出来，因此很多Web上的HTML文档格式是不规范的，从这种不规范文档中析取数据比从结构化文档中析取数据要困难得多[2,3]。由于HTML文档具有以上缺陷，本文介绍的Web数据挖掘技术，采用以下方法来实现数据析取：首先将HTML文档转换成XML格式，利用XML格式规范的优点，再从XML文档中更加有效地分析和处理数据。

1 基于 XML 的 Web 数据挖掘模型

本文所要研究的重点在于如何从Web网页上析取结构化数据，为了达到此目的，通过融合网络爬行器技术和基于 XML 的数据析取技术，设计出一种 Web 数据挖掘的框架，如图 1 所示。

根据以上框架，将 Web 数据挖掘分为以下几个步骤：

(1)利用爬行器从互联网上获得目标 Web 网页，这些网页可能是 HTML 文档或 XML 文档；

(2)将 HTML 格式的 Web 页面转换成良构的 XML 格式；

(3)析取器对 XML 格式的文档进行数据析取；

从析取器中得到的 XML 文档送往数据检验器和数据集成器进行检验和集成；

(4)通过 Java 语言中的 JDBC 将提取出的 XML 数据写入关系数据库，以备其他应用程序调用。

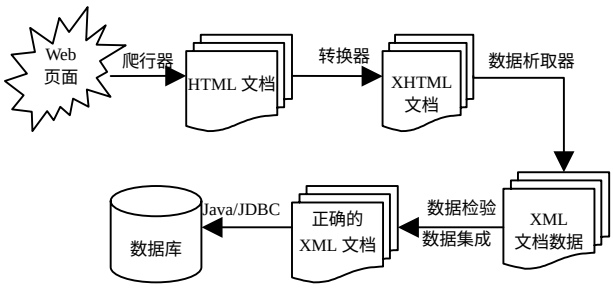

图 1 基于 XML 的 Web 数据挖掘处理模型

2 模型实现的关键技术

如何实现上述模型是本文重点讨论的内容，以下将着重讨论实现基于 XML 的 Web 数据挖掘模型的关键技术。由于已经有很多高效、可靠的网络爬行工具可用，在此对如何利用爬行器获取 Web 页面不作讨论。

2.1 将 Web 文档转换成良构的 XML 格式

转换过程必须保证生成的文档是良构的，不存在任何二义性或错误，将获取的 Web 页面转换成 XML 格式不但可以满足上述要求，而且还可以通过已有的 XML 工具对转换后的 XML 文档作进一步的处理。

基金项目：软件工程国家重点实验室开放基金资助项目
作者简介：崔建群(1974－)，女，博士生，主研方向：网格资源管理，性能服务质量；何炎祥，教授、博导；郑世珏、吴黎兵，博士、副教授
收稿日期：2006-03-30 **E-mail：**jqcui@126.com

本文采用一个 HTML-XML 转换器对 HTML 文档的不规范格式进行“修补”，将原始的 HTML 页面转换成 XHTML(Extensible HTML)或 XML 文档。该过滤器采用 HTML Tidy 和 HTML2XML 开发包来实现这一功能。

HTML-XML 转换器首先清除 HTML 文档中格式不规范或错误的地方，然后将其转换成 XHTML 文档。XHTML 是一系列当前和将来的文档类型和程序块，它由 HTML 4 再生和扩展而来。XHTML 系列文档基于 XML，最终被设计用来与基于 XML 的用户代理程序一起工作。XHTML 1.0 是 XHTML 家族的第 1 个文档，它是将 3 种 HTML 4 文档类型应用到 XML 1.0 之后重新形成的。XHTML 被设计为一种语言，它的内容既符合 XML，并且如果依照一些简单的指导方针，也能被 HTML4 用户代理程序识别。以下是用纯 Java 实现的 HTML-XML 转换器的部分源 代码：

```
package    org.w3c.tidy;
public class    HtmlToXml
{ public static void main(String[] argv)
    { …
        String file;
        InputStream in;
        String prog = "Tidy";
        Node    document;
        Out out = new OutImpl();      /* normal output stream */
        int argc = argv.length + 1;
        int argIndex = 0;
        Tidy tidy;
        Configuration configuration;
        String arg;
        tidy = new Tidy();
        configuration = tidy.getConfiguration();
        /* read command line */
        while (argc > 0)
        {
            if (argc > 1 && argv[argIndex].startsWith("-"))
            {
                arg = argv[argIndex].substring(1);
                if (arg.length() > 0 && arg.charAt(0) == '-')
                    arg = arg.substring(1);
                if (arg.equals("asxml") ||
arg.equals("asxhtml"))
                    configuration.xHTML = true;
                --argc;
                ++argIndex;
                continue;                    }
            configuration.adjust();      /*  ensure  config  is
self-consistent */
                …
        /* Internal routine that actually does the parsing.    */
        /* The caller can pass either an InputStream or file name.    */
                document = tidy.parse(null, file, System.out);
                totalwarnings += tidy.parseWarnings;
                totalerrors += tidy.parseErrors;
…
        }   }
```

2.2 数据析取

在得到转换后的规范良构文档后，采用扩展样式表转换语言 XSLT 来实现数据析取。XSLT 是把 XML 文档转化为另一 XML 文档的 XML 转换语言，即将源文档的所有数据或者部分数据(利用 XPath 进行选择)生成另外的 XML 文档或者其他可直接显示或打印的文件格式。

XSLT 是 XSL(eXtensible Stylesheet Language)规范的一部分，XSL 可以选择和过滤 XML 中的数据。XSL 不但包括 XSLT，还包括一个专门用于指定格式的 XML 词汇表，它通过 XSLT 来确定 XML 的文档样式，并描述如何利用格式词汇表将文档转换为另一 XML 文档。

XSLT 描述了从源树到目的树的转换规则，转换通过关联模式和模板来完成。一个模式是一个 XPath 表达式，可以将其视为与 XML 源树的部分相匹配的正则表达式，与字符串的匹配部分相对。模式与源树中的元素进行匹配。成功匹配后，模板成为创建结果树部分的例示。在构建结果树时，可以对源树中的元素进行筛选和重新排序，还可以添加任意结构。

以下是一个从麦当劳网页中析取出早餐菜单，同时过滤掉其他无用信息的 XSLT 例子。XSL 处理器从 XHTML 树的根部开始进行递归查找“menu”元素，一旦找到该元素，则执行包含在模板中的指令。

```
<?xml version=”1.0” encoding=”ISO-8859-1”?>
<!- - Edited with XML Spy v4.2 - ->
<xsl:stylesheet version=”1.0”
  xmlns:xsl=”http://www.w3.org/1999/XSL/Transform”>
<xsl:output version=”1.0” indent=”yes” encoding=”ISO-8859-1”
  omit-xml-declaration=”no” method=”xml”/>
<xsl:template match=”breakfast”>
  <result>
    <xsl:apply-templates/>
  </result>
</xsl:template>
<xsl:template match=”menu”>
  <food>
   <xsl:for-each select=”./item”>
          <NAME><xsl:value-of select=”./”></NAME>
   </xsl:for-each >
  </food>
</xsl:template>
</xsl:stylesheet>
```

如果要析取大量复杂的数据，采用基于 XML 查询语言 XQuery(XML Query)的析取器来实现。XQuery 是一种简洁而又易于理解的语言，被定义成直接查询 XML 并返回 XML 结果的一种语言；同时它也非常灵活，能查询范围很广的 XML 信息源，包括数据库和文档。以下代码说明如何用 XQuery 语言从书目文档“ http://bstore1.example. com/bib.xml ”中查询 Addison-Wesley 在 1991 年后出版的书的书名和年份。

```
<bib>
 {   for $b in doc("http://bstore1.example.com/bib.xml")/bib/book
  where $b/publisher = "Addison-Wesley" and $b/@year > 1991
  return
    <book year="{ $b/@year }">
     { $b/title }
    </book>   }
</bib>
```

2.3 数据检验和集成

通过上述步骤获得包含指定信息的文档后，还需要对获得的数据进行多层次的检验。首先通过语法检查，对输出的每一个 XML 元素进行值与类型的匹配验证；接着进行语义检验，找出 XML 文档中错误或是无关的值。

(下转第 77 页)