基于知识模式的文档描述构建方法*

苏 健，钟晴江
(浙江大学 城市学院 计算机应用技术研究所，浙江 杭州 310015)

摘 要：鉴于传统文档分析方法不能有效获取弱结构文档的知识描述，提出基于知识模式的文档描述构建方法。该方法综合考虑知识的行文模式与上下文结构特征，从而能够比传统方法更为有效地获取弱结构文档的知识描述。

关键词：知识管理；知识模式；弱结构文档
中图法分类号：TP18 文献标识码：A 文章编号：1001-3695(2006)01-0052-02

A Knowledge Pattern-based Method for Document Description Extraction

SU Jian, ZHONG Qing-jiang
(Institute of Computer Application, City College, Zhejiang University, Hangzhou Zhejiang 310015, China)

Abstract: Traditional methods for document analysis can not extract the description of weekly structured documents effectively and efficiently, a knowledge pattern-based method for document description construction is proposed. This method takes into account both pattern and context feature of the knowledge pattern, and is more effective than traditional ones to acquire the description of weekly structured document.

Key words: Knowledge Management; Knowledge Pattern; Weekly Structured Document

为促进内涵知识的共享，知识管理技术已经成为研究热点[1~6]。知识管理的基本方法是建立所谓的组织记忆(Organizational Memory)[2]，对企事业关注的信息源中内涵的知识作语义清晰的说明并提供高效的存取结构，以便促进对于这些知识的俘获、集成、共享、重用和维护。由于大量可供共享的知识并非以传统的人工智能知识表示方式清晰表述，而是隐含在弱结构的文档(Web 页面、电子文档、E-mail)中[3]，因此，建立弱结构文档的知识描述是进行有效知识管理的关键。传统文档分析方法一般只对文档的关键词进行简单的统计和分析，而不能对相对复杂的行文模式进行分析，因而不能有效地建立文档知识描述。为此，提出基于知识模式的文档描述构建方法。该方法以本体论中定义的知识模式为基础，对文档内容进行模式匹配，并综合考虑模式的上下文结构模式特征，最后建立文档的描述。

1 模式及其定义

模式可分为简单模式与复杂模式，可通过模式定义方法加以定义。

1.1 模式定义元符号

模式定义中将使用以下元符号(字符)，这些符号具有特殊的含义。

|表示“或者”关系。比如 a|b|c 是满足 a 或 b 或 c 的字符序列。

? 表示出现一次或者零次。比如 a? 是满足 a 或者空的字符序列。

* 表示出现零次或者多次。比如 a^* 是满足零个或者多个 a 组合的字符序列。

+ 表示出现一次或者多次。比如 a+ 是满足多个 a 组合的字符序列，且不为空。

% 表示右匹配。比如% a 表示满足任何右侧出现 a 的字符序列。

( ) 表示组合，左括号与右括号必须成对出现。比如 $(abc)^*$ 表示满足 abc, abcabc, abcabcabc 等的字符序列。

< > 表示其他定义的模式名称(包括预定义的名称)，< 与 > 必须成对出现。比如 <space> 表示预定义的空格字符串。

{ } 表示需要返回的字符串，{ 与} 必须成对出现。

[ ] 表示括号中的某一字符。比如[abc] 表示 a, b 或 c. [与] 必须成对出现，对于[与] 之间的编码连续的字符，可通过“-”加以简写。比如，[a - f] 表示 a, b, c, d, e, f 中的某一字符。

/ 转义字符。上面用到的元字符都是模式定义中的保留字符，如果模式定义中需要用到这些字符，那么就用“/”前缀表示。比如 / <abc 表示满足“<abc”的字符序列。

1.2 简单模式与复杂模式

模式可以分为简单模式和复杂模式。模式定义是可嵌套的，即模式定义中可以包含其他预定义的模式定义。

(1) 简单模式。令 a, b, c 为长度为 1 的字符，且 a, b, c 不为元字符。那么如下模式为简单模式：

简单字符 a, b, c;

元字符描述的简单字符 a^*, a+, a?, %a;

简单字符连接串 abc, acb, bac;

收稿日期：2005-01-24；修返日期：2005-03-10
基金项目：国家自然科学基金资助项目(60473097)

元字符描述的简单字符连接串 (abc)*, (abc)+, (abc)?。

(2) 复杂模式。它包括通过元字符连接的简单模式,以及复杂模式与复杂模式连接所得的模式。比如

简单模式的连接 (abc)* gh;

“|”连接 (abc)|(bca), (abc)*|(bca)+;

“()”连接 ((abc)|(bca))*。

2　模式分析树及其构造

2.1　模式分析树

对每个模式,建立一个模式分析树。该模式分析树是一个“与或树”。

对于简单模式而言,其模式分析树将是一棵仅有根节点的分析树; 而对于复杂模式,可将模式递归划分为若干棵子树,主要包括两种情况: “与树”和“或树”。

令 A 为一个复杂模式定义。

如果 A::=$A_1A_2\cdots A_n$,那么 A 可以表示为由 n 棵子树 $A_1, A_2, \cdots, A_n$ 组成的“与树”,如图 1 所示。

如果 A::=$A_1|A_2|\cdots|A_n$,那么 A 可以表示为由 n 棵子树 $A_1A_2\cdots A_n$ 组成的“或树”,如图 2 所示。

2.2　模式分析树构造

模式分析树的构造目标是:通过对模式定义的分析,生成对应该模式的模式分析树。

模式分析树的构造方法是:将复杂的模式定义逐层细化,分割为若干个子模式,如果某一子模式还是复杂模式,那么将其继续细分,直到化为简单模式为止。例如,对模式(abc|cd)(ef)* gr 的模式分析树如图 3 所示。

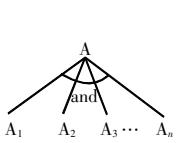

图 1　与树

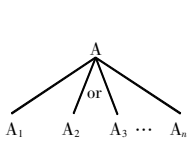

图 2　或树

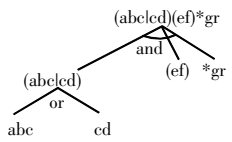

图 3　模式分析树

3　模式匹配

对于某一给定的字符串可通过模式匹配算法来判断该字符串是否满足某模式的定义。模式匹配算法是建立文档描述的基础。

3.1　模式匹配原则

令 A 为模式分析树上的某一非叶子节点,令 S 为被分析的字符串。

(1) 如果 A 包含若干个“与子树”,从左到右依次为 $A_1, A_2, \cdots, A_n$。那么:

如果 S 从左到右依次满足 $A_1, A_2, \cdots, A_n$,那么就称 S 匹配 A。

否则,就称 S 不匹配 A。

(2) 如果 A 包含若干个“或子树”,从左到右依次为 $A_1, A_2, \cdots, A_n$。那么:

如果 S 从左到右依次满足 $A_1, A_2, \cdots, A_n$ 之一,那么就称 S 匹配 A。

如果都不满足,那么就称 S 不匹配 A。

3.2　模式匹配算法

令 P 为模式的根节点, S 为输入字符序列。

步骤 1　初始化　令 A = P。

步骤 2　如果 A 是叶子节点(即子节点数为零),那么对输入字符序列按模式进行匹配,并返回结果(匹配成功或者失败)。

步骤 3　如果 A 不是叶子节点(即子节点数为零),那么令 A 的子树从左到右依次为 $A_1, A_2, \cdots, A_n$,进行以下判断:

(1) 如果 A 的子树是“与树”,那么对 $A_i, i = 1, 2, \cdots, n$,按 i 从小到大顺序,(通过递归调用)分别判断各个子树是否匹配,如果全部都匹配,那么返回匹配成功,否则返回匹配失败。

(2) 如果 A 的子树是“或树”,那么对 $A_i, i = 1, 2, \cdots, n$,按 i 从小到大顺序,(通过递归调用)分别判断各个子树是否匹配,如果某一个子树能够匹配,那么返回匹配成功,否则返回失败。

4　知识模式与文档描述构建方法

知识模式可以表示为二元组: (模式, 上下文结构特征)。其中模式就是前文阐述的文档行文模式。由前文可知,通过模式定义方法,可以定义本体论中的各种类型知识所对应的行文模式,而每个模式将对应一棵模式分析树; 并且可以通过模式匹配方法,来检查文档中是否包含符合某一模式特征的字符串。当然,仅通过模式匹配方法还不能较好地建立文档描述,还需要考虑该模式的上下文结构特征。某一模式匹配之后,它的上下文情况往往说明该模式的重要性与匹配的准确性。因此,模式还需要上下文结构特征加以限定。所以,文档描述构建方法首先根据知识模式中的模式对文档内容进行模式匹配,从而发现满足模式要求的字符串,然后通过知识模式中的上下文结构特征来判断获取的字符串是否具有实际意义或者意义的大小程度。

例 1　文档描述一般包括如下描述字段: 题目、作者、作者单位、关键词列表、文档领域、文档说明等。这些字段在本体论中包含相对应的模式。题目所对应的模式定义为

<题目>::=(<newline>|<space>)*{TitleString}<space>*<newline>

其中, <space>是预定义的空格字符, {TitleString}是具体的需要返回的标题字符串, <newline>是预定义的换行符。因此,由题目模式可知,文档的题目的前序字符串必须是换行符或者空格的序列,后序字符串必须是长度大于等于零的空格序列,最后以换行符结束。

显然,上述题目模式仅当出现在文档起始处才有具体意义; 如果出现在文档中间,基本上不会是文档的题目。因此,题目知识模式的搜索匹配还需要增加文档起始这一结构特征。

例 2　文档由一系列关键词加以描述,但并不是任意词或者词组都可以成为关键词。所允许的关键词一般由本体论中给出或加以限定。对于某一特定文档,其关键词将按照关键词模式的加权次数进行排序。加权次数不同于普通出现次数,它综合考虑了关键词模式的出现位置。比如科技文档或者正式文件中经常出现如下关键词模式:

关键词: <keywordString>([,;] <keywordString>)*

即“关键词:”前缀的若干个由逗号或者分号分割的关键词序列。显然,这样的关键词具有最高的权值。

(下转第 150 页)

这样一来,只要输入 XML 语言文件,XSLT 模板就可以结合输入的文件,共同生成各种形式的输出文件。支持 XSLT 语言标准的处理器将 XML 文件转换为 XML 文件或者其他文本文件。这里顺带介绍几个开源的 XSLT 处理器,如 SAXON XSLT 处理器、Microsoft XML Parser3,在这里笔者所采用的是 Apache Xalan XSL 处理器[10]。在配置好 Xalan 中 xalan.jar, xmlapis.jar, xercesImpl.jar 这些 jar 包的环境参数 classpath 后,在 DOS 窗口下键入如下的命令就可以运行 XSLT 脚本生成所需要的代码:

Java org.apache.xalan.xslt.Process -IN foo.xml -XSL foo.xsl -OUT foo.Java

该语句的意思是:执行 XSLT 处理器 org.apache.xalan.xslt.Process,读取 man.XML 文件和 Page.XSL 文件,生成 CustomerPage.Java 文件。

6 结束语

本文主要是讨论了代码生成技术在 MDA 中的实现。采用 XSLT 技术的代码生成器,可以方便地将描述业务模型的 UML 转换为 XML 文件,并且能够对这些 XML 文件进行准确的语义检查,同时读取根据业务模型实现实例抽取出的共性的 XSLT 程序模板,通过 XSLT 处理器生成简洁完整的源代码。如果所提供的业务模型信息足够完整,各个业务处理逻辑正确,通过这个代码生成器映射出来的代码可直接编译运行,实现基本的增,删,改等业务操作。当然,这样设计的代码生成器也有它的不足,首先是这样的基于 MDA 的代码生成器,需要有能够完整详细描述系统业务逻辑和数据的元数据模型,这就需要有大量的开发经验和优秀的设计师支持。其次,XSLT 的模板过于冗长,学习起来起点较高,能够熟练撰写 XSLT 模板的开发者较少。但是不可否认,采用 XSLT 技术的代码生成器使得模型到代码的转换变成可能,并且生成的程序不再只是代码框架,而是对应于模型的完整的应用程序代码,这也极大地扩展了代码生成技术的应用领域。

参考文献:

[1] oaquin Miller, Jishnu Mukerji. MDA Guide version 1.0.1 [EB/OL]. http://www.omg.org/docs/omg/03-06-01.pdf, 2003-06-12.

[2] OMG. Model Driven Architecture(MDA) [EB/OL]. Available at Java and XSLT Eric M. Burke OReilly & Associates Inc., 2003.

[3] David S Frankel. 应用 MDA [M]. 鲍志云. 北京: 人民邮电出版社, 2003.

[4] Birgit Demuth Heinrich Hussmann, Sven Obermaier. Experiments with XMI Based Transformations of Software Models[R]. 2003.

[5] OMG. Meta Object Facility(MOF) Specification version 1.4 [EB/OL]. http://www.omg.org/docs/formal/00-04-03.pdf, 2002-04.

[6] Soumen Sarkar. Model Driven Programming Using XSLT[R]. 2002.

[7] OReilly. Java & XSLT[M]. 2003.

[8] The Latest XSLT Specification. 国际互联网联盟的最新 XSLT 规范[EB/OL]. 2004-05.

[9] J Craig Cleaveland. 用 XML 与 Java 创建程序生成器[M]. 胡俊. 美科海电脑技术出版社, 2003.

[10] Apache Xalan XSL 处理器的命令集[EB/OL]. http://xml.apache.org/xalan-j/commandline.html, 2004-05.

作者简介:

陈翔(1979-),男,硕士研究生,主要研究领域为分布式计算、软件工程;王学斌(1978-),男,博士研究生,主要研究领域为分布式计算、软件工程;吴泉源(1942-),男,教授,博士生导师,主要研究方向为智能软件与分布计算。

(上接第 53 页)另外,如果某一关键词出现在文档的各级标题中,那么该关键词的权值将比出现在正文中的关键词权值大,并且权值将按标题的不同层次从高到低依次减少,即出现在一级标题的关键词模式的权值最大,出现在二级标题中的关键词模式的权值相对较小,以此类推,层次越深的标题中出现的关键词模式的权值依次减少,而正文中出现的关键词模式权值最小。

标题模式定义如下:

< 标题 > :: = (< space > | < newline >) * < preTitle > < space > * { < titleString > } < newline > *

其中, < titleString > 是需要返回的具体标题字符串, < preTitle > 是标志标题级别的特征字符串,以下是相关定义:

< preTitle > :: = (第 < titleNumber > (章 | 节 | 点) < dot > ?) |(< titleNumber > < dot > ?) |(< leftBracket > < titleNumber > < rightBracket >)

< titleNumber > :: = < number > | < chineseNumber > | < englishNumber >

< number > :: =([0-9] < dot > ?) | ([1-9] + [0-9] *) | ([0-9] < dot > [0-9] *) | ([1-9] + [0-9] * < dot > [0-9] *)

< englishNumber > :: =([a-zA-Z] * < dot > ?) | ([a-zA-Z] * < dot > [a-zA-Z] *)

< chineseNumber > :: = < CN > |(< CN > 十 < CN >) |(< CN > 百 < CN > 十 < CN >)

< CN > :: =[一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 九 | 零] ;

5 结束语

知识管理的对象大多数是弱结构文档,而建立这些弱结构文档的知识描述是实施有效知识管理的关键。提出基于知识模式的文档描述构建方法。为知识管理提供了一种有效的方法与技术。

参考文献:

[1] teffen Staab, Rudi Studer, Hans-Peter Schnurr, *et al*. Knowledge Processes and Ontologies[J]. IEEE Intelligent Systems, 2001, 16(1):26-34.

[2] Gerhard Fischer, Jonathan Ostwald. Knowledge Management: Problems, Promises, Realities, and Challenges[J]. IEEE Intelligent Systems, 2001, 16(1):60-72.

[3] Dieter Fensel. Ontology-based Knowledge Management[J]. IEEE Computer, 2002, 35(11): 56-63.

[4] Jay Liebowits. Knowledge Management and Its Link to Artificial Intelligence[J]. Expert Systems with Application, 2001, 20:1-6.

[5] Yogesh Malhotra. Expert Systems for Knowledge Management: Crossing the Chasm between Information Processing and Sense Making [J]. Expert Systems with Application, 2001, 20:7-16.

[6] David G Schwartz, Dov Te'eni. Tying Knowledge to Action with kMail[J]. IEEE Intelligent Systems, 2000, 15(3): 33-39.

作者简介:

苏健(1974-),男,讲师,博士,主要从事知识管理、数据挖掘、决策支持等领域的研究;钟晴江(1965-),男,讲师,硕士,主要从事软件工程、系统集成等领域的研究。