一种基于XML结构索引的模式匹配改进算法

李思莉[1],张海清[2]
(1.成都理工大学工程技术学院 电子信息与计算机工程系,四川 乐山 614007;
2.成都信息工程学院,四川 成都 610225)

摘 要:XML已经成为Internet上数据交换和数据集成的事实标准。随着XML的广泛应用,XML文档数量不断增多。如何高效地查询XML数据变得越来越重要。针对目前分支查询中普遍采用的基于堆栈的查询处理算法所存在的问题,提出了一种基于XML结构索引的模式匹配改进算法,通过选择合适的标签编码方式,利用XML结构索引,快速判断出元素之间的相互关系,防止大量不必要节点放入堆栈,从而提高查询处理效率。实验结果证明,文中改进的模式匹配算法TwigModify相比TwigStack以及TwigINLAB在查询处理的性能上有所提高。

关键词:XML;结构索引;模式匹配;算法;TwigModify
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2012)05-0045-03

An Effective Pattern Match Algorithm Based on Structural Index of XML

LI Si-li[1],ZHANG Hai-qing[2]
(1. Department of Electronic Information and Computer Engineering, The Engineering Technical College of Chengdu University of Technology, Leshan 614007, China;
2. Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: XML has become the fact standard for data exchange and data integration on the Internet. With the wide range of applications of XML, the XML document is growing. How to query XML data efficiently becomes increasingly important. For the existence of a stack-based query processing algorithm commonly used in the branch query, proposed an improved algorithm based on XML structure index-based model. By selecting the appropriate label encoding using XML structure index to quickly determine mutual relations between the elements, prevent a large number of unnecessary nodes into the stack, thereby improving query processing efficiency. Experimental results show that the improved pattern matching algorithm TwigModify improves performance of query processing compared with TwigStack and TwigINLAB.

Key words: XML; structural index; pattern match; algorithm; TwigModify

0 引 言

XML已经成为Internet上数据交换和数据集成的事实标准。随着XML的广泛应用,XML文档数量的增加,XML数据的高效查询显得尤为重要。XML数据查询分为两种,全文查询和结构化查询。全文查询也称为基于关键字的查询,其查询技术类似于信息检索技术中的内容检索。W3C工作草案[1]对此进行了具体的研究和说明。结构化查询是根据用户提供的查询表达式,检索树模式[2],寻找匹配结点的过程。文中主要的工作重点集中在对XML数据结构化查询的处理。同时,为了降低XPath,XQuery查询的时间开销,需要对XML文档中的路径信息建立索引,此类索引称为结构索引。近年来,研究者提出了多种结构索引。Fabric[3], A(k)-Index[4], F&B-Index[5]等都是其中具有代表性的结构索引建立方式。它们都有各自的优缺点,文献[6]对此做了一些比较说明。文献[7,8]中提出先对XML文档进行编码,然后采用结构化联接的方式进行查询。这类方法虽然可以找出树模式在XML文档中的匹配,其查询能力也优于结构化索引方法,但是这类方法需要进行大量的联接操作,降低了查询效率。

文中结合这两类查询处理的优点,提出了一种基于XML结构索引的模式匹配改进算法,该算法是在TwigINLAB的基础上作了一些改进,通过选择合适的标签编码方式,利用XML结构索引,快速判断出元素

收稿日期:2011-09-16;**修回日期**:2011-12-20
基金项目:成都理工大学科研发展基金(C122010001)
作者简介:李思莉(1974-),女,讲师,CCF会员,研究方向为XML数据管理、网络安全。

之间相互关系,达到防止大量不必要结点放入堆栈的目的,从而提高查询处理效率。

1 相关工作

基于遍历操作的查询和基于联接操作的查询是通常用于查询处理的两种方法。基于遍历操作的查询在工作负载比较大的情况下,很难满足处理要求。因此,为了提高查询效率,目前出现了很多为XML文档建立索引结构的技术[3~5],但大部分技术却存在标签编码过长以及索引表过大的情况。基于联接操作的查询涉及到分解、匹配以及合并三个阶段。Stack-Tree[7]是其中一种具有代表性的方法。它先将分支查询进行分解,成为一组的二元关系,再分别对它们进行联接组合,最后进行模式树匹配处理。此种方法联接代价较大,并且会产生一些无用的中间结果。TwigStack算法[9]则是将整棵模式树进行匹配处理,而不用保存无用的中间结果,对于只有祖先后代边的模式树匹配它是最优的。文献[10]提出了在进行匹配之前,先预处理用户查询模式。然而,这些方法依然会产生大量无用的中间结果,这必然带来随着工作负载增加,查询效率降低的问题。

2 基本概念

2.1 数据模型与模式树

XML文档使用树型数据模型。模式树用来表示对XML文档的查询,如在Xpath子集{/,//,[ ]}中的查询可以表示为一棵模式树,查询表达式中的每个标签可以表示为模式树中的一个结点,表达式中的轴表示为模式树中的对应边,如图1所示。查询处理就是给定查询模式Q和XML数据D,在模式树中找出与之匹配的XML数据。

2.2 标签编码方式

为了能够快速判定XML结构树中两个结点之间的关系,需要对树中XML结点编码。文中采用的编码方式为<self-level;parent>[11],图1中对XML树进行了编码。

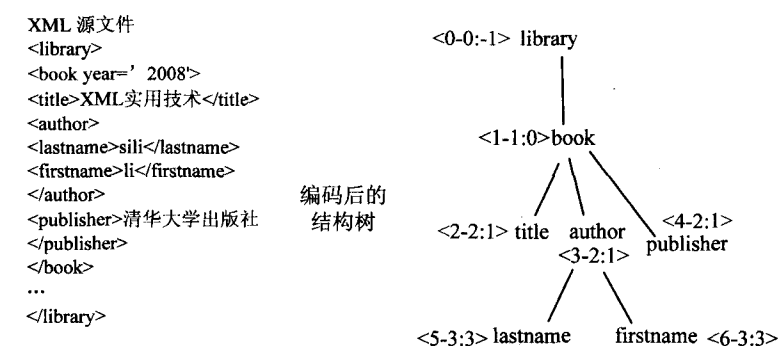

图1 编码后的XML结构树

3 创建结构索引

一个XML结点的路径是从根开始到该结点所经过结点的标签序列$l_1l_2l_3\cdots l_k$,文中采用了文献[11]中创建结构索引的方法,通过元素的编码建立索引,用来辅助跳过不可能发生连接的结点,从而避免对这些结点的处理。表1是在预处理XML文档得到的数据流上建立起来的结构索引表。

表1 部分索引表

Self	parent
0	-1
1	0
2	1
3	1
4	1
5	3
6	3

4 模式匹配

当进行模式树查询时,首先在结构索引上进行预匹配处理,再根据预匹配的结果对XML数据进行化简后的模式树匹配,从而得到最终结果。文中在文献[11]中的TwigINLAB算法的基础上,提出一种改进的模式树匹配算法,此种算法减少无关结点进入堆栈,减少重复查找的次数。

4.1 相关定义

匹配扩展:如果以结点q为根的子查询(子树)与子查询所有查询结点的游标元素匹配。则称结点q有一个匹配扩展。

最近匹配扩展:在某个查询执行点,模式树Q有多个查询结点q_i有匹配扩展,当q_x的游标元素值self小于其他任意查询结点q_i的self,则称查询结点q_x有最近匹配扩展。

4.2 匹配算法

先将预匹配结果组织起来,保证每个索引结点的扩展匹配集只出现一次,数据流中又保存了索引结点间的结构关系。每个XML元素不需要预先进行排序,只需处理一遍,而且也充分利用了索引匹配结果中的结构信息,对两段不可能满足结构关系的XML序列不会进行连接。如Q1 = book[title = 'XML实用技术' AND @year = '2008'],Q2 = book[title = 'XML实用技术'] | | author[lastname = 'sili' and firstname = 'li']。其模式树如图2所示。对Q1的模式树,只有父-子边。对Q2的模式树,book元素与author元素之间使用了祖先-后代边。根据文献

[11]的分解算法,对Q1不会出现重复匹配的问题,且所有产生的中间结果最终都是匹配结果的一部分。但对Q2,使用原来的GetNext方法,会把查询转化为模式树中根到所有叶子结点的单支查询的连接运算,并为模式树的每一个结点设置一个栈,如图3所示。

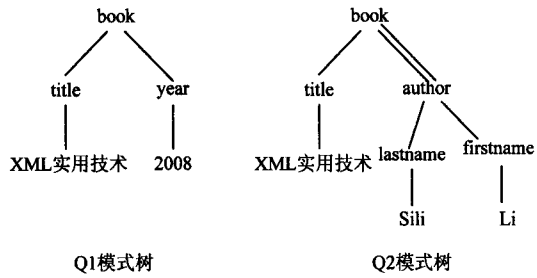

图2　模式树

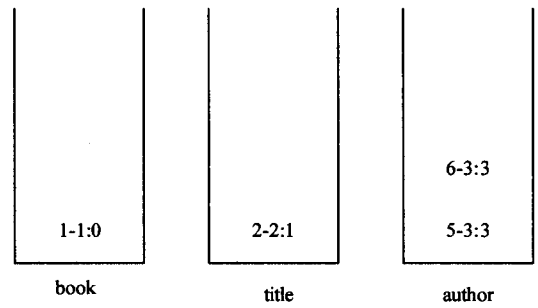

图3　分解算法示意图

文献[11]中GetNext的问题在于对book||author[lastname='SiLi']和book||author[firstname='Li']这两个单支路径查询匹配中都包含的结点author进行了重复匹配,且把与最终结果无用的中间结果放入了堆栈中。文中对模式树查询处理的算法除了对文献[11]中模式匹配算法进行了修改和改进,其余均与文献[11]中提供的算法类似,其改进的算法如下:

```
Procedure GetNext(q)
{  input:current nodes in data stream
output:nodes needed processed
1if (q.isLeaf()) then
2    return q;
3for (qi in q.children) do
4    ni=getNext(qi);
5    if (ni! =qi) then
6    return ni;
7endfor;
8 while (! checkAncestor(q,ni))
9{
10 if (getSelf(q)>getSelf(ni))
Return ni;
11    advance(q);
12 }
13 if (getSelf(q)>getSelf(ni))
14    return ni;
15    return q;
}
```

在getNext()函数中,如果q是一个叶查询结点(调用isLeaf函数判断),则该函数直接返回结果。否则递归向下到最左叶子查询结点。从叶结点开始,GetNext找到以q为根的子查询(子树)中有最近匹配扩展的查询结点。假定q的所有子结点都有它们自己的匹配扩展,为了能返回查询结点q,必须执行advance()保证结点q有一个查询扩展。在5行,如果结点n_i不等于q的孩子结点,则立刻返回n_i,8行跳过所有与查询结果无关的结点。checkAncestor函数检查两个结点是否具有祖先-后代关系。

原来匹配算法中getNext()函数中存在的问题:对于包含分支路径查询的XML查询,根据GetNext方法可知,查询会转化为模式树中根到所有叶子结点的单支查询的连接运算,这样,几个路径匹配都包括的结点就会被重复的匹配,降低了算法运行的效率。

改进后的模式匹配算法充分利用了索引匹配结果的结构信息,通过直接定位查询结点的关系,避免了无用中间结果的产生。

5　实验结果

在实验中使用大小不同的6个XML数据文档,其具体特性见表2。实验中使用了从简单到复杂的三组查询,分别对应以上6个数据文档。查询使用XPath表示模式树,标签后加"*"的结点表示需要返回,查询描述见表3。机器配置CPU为Intel 2.8G,内存1G DDR2,操作系统为Windows XP,编程环境为VC++ 6.0。

表2　XML实验文档的特性

文档大小	XML结点数	不同标签数	最深层数
1.3M	10242	27	6
2.5M	24698	30	17
3.9M	78945	42	57
5.6M	100348	54	60
8.9M	126780	69	68
11.6M	173259	80	75

表3　模式树查询

	查询模式树(标签后加*的结点需要返回)
Q1	person[/birthday]/regions/item/description/text*
Q2	person//address/description//text*
Q3	person[[/birthday]//regions]//text*

测试了文中的算法,并且和文献[2]中的TwigStack以及TwigINLAB算法进行了比较,实验结果证明了改进后的模式匹配算法在一定程度上提高了算法运

(下转第52页)

法开放测试性能比较。

从实验结果可以看出，混合模型分类准确度和分类速度有明显提高，效率优于单一的 BP 神经网络和朴素贝叶斯分类器。

4 结束语

文中分析了数据挖掘系统中 Bayes 分类以及神经网络分类方法，提出了一种（神经网络 + 贝叶斯 + SVM）混合分类方法。该方法利用了朴素贝叶斯分类快速、稳健的特点和神经网络准确率高、非线性特征，并且利用 SVM 处理小样本、非线性及高维模式识别问题的优势进行二次补偿识别。实验结果表明了，这种方法的分类效果以及识别率都比单独的分类器的效果更优，这种混合模型能够很好地避免各分类器的缺点，并综合各个分类器的优势，将其组合使用，分类器的性能有所提高。

参考文献：

[1] Elkan C. Boosting and Naive Bayesian Learning[R]. San Diego: Dept of Computer Science and Engineering, University Calif at San Diego, 1997.

[2] 黄友平. 贝叶斯网络研究[D]. 北京：中国科学院计算技术研究所，2005.

[3] 张连文，郭海鹏. 贝叶斯网引论[M]. 北京：科学出版社，2006：186-188.

[4] Witten I H, Frank E. Data Mining Practical Machine[M]. San Francisco: ELSEVIER, 2005.

[5] Burges C J C. A Tutorial on Support Vector Machines for Pattern Recognition[J]. Data Mining and Knowledge Discovery, 1999, 2(2): 121-167.

[6] 汪杭军，张广群，方陆明. 粗糙集属性约简算法的实现与应用[J]. 计算机工程与设计，2007，28(4)：777-779.

[7] 余瑞康，施润身. 聚类思想在贝叶斯算法中的应用[J]. 计算机工程与应用，2006，42(3)：159-160.

[8] 马海峰，孙名松. 基于多层前向神经网络入侵检测系统的研究[J]. 哈尔滨理工大学学报，2004，9(2)：52-55.

[9] 刘道群，孙庆和. 基于遗传神经网络的入侵检测模型[J]. 激光杂志，2005，26(6)：73-74.

[10] Murph P M, Aha D W. UCI repository of machine learning databases[DB/OL]. 2006. http://www.ics.uci.edu/~mlearn/MLReposi-tory.html.

[11] 周金宇，谢里阳. 基于神经网络预拟合的 B 样条曲面反求[J]. 东北大学学报：自然科学版，2003(6)：556-559.

[12] Suzuki J, Fujino A, Isozaki H. Semi-Supervised Structured Output Learning Based on a Hybrid Generative and Discriminative Approach[C]//Proc. of EMNLP-CoNLL. [s. l.]: [s. n.], 2007: 791-800.

[13] Cheng Mingyang, Wu Hung-Wen, Su A W Y. On Non-Uniform Rational B-Spline Surface Neural Networks[J]. Neural Process Lett, 2008, 28(1): 1-15.

（上接第 47 页）

行效率。

为了验证文中的算法从一定程度上提高了查询匹配的效率，进行了不同关系的查询，Q1 只有父-子关系单支查询，Q2 是带有祖先-后代关系的单支查询，Q3 是一种较为复杂的混合关系查询。

6 结束语

实验证明，在 TwigINLAB 基础上修改的模式匹配算法，在不同关系的查询中能够改善执行效率，有一定的优越性。在以后的研究中，将使用局部相似性来改进 TwigModify，避免对路径信息过于细化，即使对结构复杂的 XML 文档也能保持合适的大小，同时能够提高在索引上的查询速度。

参考文献：

[1] W3C. XML Query Language[M/OL]. 2005. http://www.w3.org/XML/XQuery.

[2] Miklau G, Suciu D. Containment and equivalence for an XPath fragment[C]//Proc. of the PODS. [s. l.]: [s. n.], 2002: 65-76.

[3] Cooper B F, Sample N, Franklin M J, et al. A fast index for semistructured data[C]//Proc. of the VLDB. [s. l.]: [s. n.], 2001: 341-350.

[4] Kaushik R, Shenoy P, Bohaaon P, et al. Exploiting local similarity for indexing paths in graph-structured data[C]//Proc. of the ICDE. [s. l.]: [s. n.], 2002: 129-140.

[5] Kaushik R, Bohaaon P, Naughton J F, et al. Covering indexes for branching path queries[C]//Proc. of the SIGMOD. [s. l.]: [s. n.], 2002: 133-144.

[6] 张 博，耿志华，周傲英. 一种支持高效 XML 路径查询的自适应结构索引[J]. 软件学报，2009，20(7)：1812-1824.

[7] Al-Khalifa S. Structural joins: a primitive for efficient XML query pattern maching[C]//ICDE. [s. l.]: [s. n.], 2002.

[8] Jiang H, Wang W, Lu H. Holistic twig joins on indexed XML documents[C]//VLDB. [s. l.]: [s. n.], 2003.

[9] Bruno N, Srivastava D, Koudas N. Holistic Twig Joins: Optimal XML Pattern Matching[C]//Proc. of ACM SIGMOD. [s. l.]: [s. n.], 2002: 310-321.

[10] Polyzotis N, Garofalakis M, Loannidis Y. Approximate XML Query Answers[C]//Proc. of ACM SIGMOD. [s. l.]: [s. n.], 2004: 263-274.

[11] Haw S C, Lee C S. Structural Query Optimization in Native XML Databases: A Hybrid Approach[J]. Journal of Applied Sciences, 2007, 7(20): 2934-2946.