一种快速的 XML 语义检索算法

李新叶,苑津莎
(华北电力大学电子与通信工程系,河北保定 071003)

摘 要: 传统基于关键词的搜索引擎不能充分利用 XML 文档的结构信息,搜索结果往往不精确;而基于结构信息和关键词的 XML 搜索技术又不适用于普通用户.基于关键词的 XML 语义检索克服了以上缺点,但需要提高检索效率.本文深入分析了 XML 文档结构潜藏的语义,提出了新的索引结构及两结点语义相关的判断函数,在此基础上提出了一种快速的 XML 语义检索算法,该算法大大减少了结点对语义相关的判断次数.对实际数据集的测试实验结果显示出新算法的有效性.

关键词: XML 文档;语义检索;索引结构;信息检索
中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112 (2007) 11-2220-06

A Fast Semantic Search Algorithm for XML Data

LI Xin-ye, YUAN Jin-sha
(*Department of Electronic and Communication Engineering, North China Electric Power University, Baoding, Hebei* 071003, *China*)

Abstract: Traditional keyword-based search engine does not consider the additional information provided by the structure of XML documents, it returns imprecise results often; searching according to keywords and structure information of XML documents inputted is not suitable for common users. Semantic search for XML data based on tag—keywords overcomes the limitations above, but its efficiency needs to be improved. This paper analyzes semantic information provided by the structure of XML documents deeply. It puts forward a new index structure for XML data and semantic related decision function between two nodes. Based on this, it proposes a fast semantic search algorithm for XML data. The search algorithm reduces the times to decide semantic correlation greatly. The experimental results with real data sets illustrate the effectiveness of the proposed algorithm.

Key words: XML document; semantic search; index structure; information retrieval

1 引言

XML 由于具有自描述、灵活的数据结构及丰富的数据表示能力等特点,现已被广泛应用到电子商务中数据的表示、数据集成,信息检索等领域,并逐步成为 Internet/intranet 上数据交换的标准.针对 web 上出现的越来越多的 XML 文档,传统的基于关键词的搜索引擎由于未考虑到 XML 文档结构隐藏的语义,搜索结果往往不精确.

近年来,已有许多针对 XML 文档的搜索及索引的研究[1~7],需要用户输入 XML 文档结构信息以获得精确的检索结果,而普通用户往往不知道 XML 文档的结构(路径)信息,因此这些方法不适用于普通用户的检索要求.XML 文档查询语言 XQUERY 也不适用于 XML 搜索引擎,主要原因有[8]:(1)语法复杂;(2)需要知道文档的结构来构造一个查询语句;(3)没有查询结果的排序;(4)查询速度慢.文献[8]提出了一种基于关键词的适用于普通用户检索要求的 XML 语义搜索技术.为检索出符合用户查询请求的语义相关的 XML 文档片段,文献[8]定义了两个 XML 结点相连关系及多个结点间语义相关的概念,搜索引擎首先在索引文件中查找与各关键词匹配的 XML 结点,然后判断这些结点之间是否满足语义相关的条件,如果满足则为检索结果.与传统的基于关键词的检索相比,XML 语义检索提高了检索精度,但由于要对匹配结点进行语义相关的判断,XML 语义检索的效率至关重要.本文研究了 XML 文档结构隐藏的语义,提出了新的索引结构,在此基础上提出了一种快速的语义搜索算法,并结合实例分析比较了本文算法的有效性.

2 相关概念

定义 1 XML 文档树

一个 XML 文档可以转换为一棵带标签的树,XML 元素对应于树的结点,每个结点都有一个标签.结点分

收稿日期:2007-03-12;修回日期:2007-06-20

为内部结点和叶子结点;叶子结点是有值没有子结点的结点,其值为对应元素的内容;内部结点是有子结点的结点.

观察图 1 的 XML 文档树,一个结点代表现实世界中的一个实体,具有相同标签的两个不同结点对应于具有相同类型的两个不同实体.而层次相同的结点 5(标签 book)和 11(标签为 journal),以它们为根结点的子树结构是相同的,我们认为结点 5 和 11 对应于类型相似的两个不同实体.如果结点 na 是 n 的祖先结点,则 n 属于 na 所表示的实体.如果结点 n 和 n'有不同的祖先结点 na 和 na',而且 na 和 na'具有相同的标签,则 n 和 n'在语义上是不相关的,因为它们属于具有相同类型的两个不同实体.下面的定义给出了这种语义关系,其中定义 3,4,5 来自文献[8].

定义 2　语义清晰的 XML 文档

语义清晰的 XML 文档指在一棵 XML 文档树中,任何一个结点的标签都不与其祖先结点的标签重名,标签相同的不同结点代表的实体类型相同,如果它们有不同的祖先结点,则层次相同的祖先结点为同类型的或相似类型的.

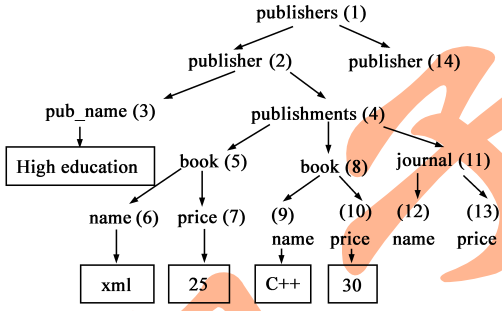

图 1　XML 文档树

定义 3　两个结点相连关系及语义相关概念

令 T 为一棵 XML 文档树,n 和 $n0$ 为 T 中的两个结点.n 和 $n0$ 的最短无向路径由 n 和 $n0$ 的最低层(离根结点最远的)公共祖先分别到 n 和 $n0$ 的路径组成.把这两条路径组成的树表示为关系树 $T_{|n,n0}$.当 n 和 $n0$ 满足以下两条件之一时,称 n 和 $n0$ 是相连的:

(1) $T_{|n,n0}$不包含两个具有相同标签的不同结点;

(2) $T_{|n,n0}$中唯一具有相同标签的两个不同结点为 n 和 $n0$ 本身.

用 R 表示这种相连关系.显然,R 是自反的、对称的二元关系.如果 n 和 $n0$ 是 R 相关的,则它们在语义上是相关的.

图 1 中,结点 6 和 10 的关系树 $T_{|6,10}$包括结点 6,5,4,8 和结点 10,其中有两个不同结点(5 和 8)有相同的标签 book,因此结点 6 和 10 是不相连的.而结点 6 和 7 是相连的.这直观地反映了书名为 xml 的书的价格(price)是 25 而不是 30.考虑结点 5 和 8,根据定义 3,它们是相连的,反映了结点 5 和 8 代表的两本书是由同一出版社出版.由定义 3 也可以得出结点 6 和 13 是相连的,但实际上结点 6 和 13 属于不同的实体,不应满足相连关系.本文对定义 3 加以扩展,将定义 3 中的相同标签改为相同标签或相似类型.

定义 4　语义相关的一组结点

令 T 为一棵树,R 为 T 中结点对的相连关系,则 R 中的结点对是语义相关的.称一组结点 N 为全相关的(all-pairs R-related),如果对 N 中的任一结点对$(n1,n2)$,都有$(n1,n2)\in R$;称一组结点 N 为星型相关的(star R-related),如果对所有结点 $n\in N$,有一个结点 $n^*\in N$ 使结点对$(n^*,n)\in R$.称结点 n^* 为星结点(star center).

如图 1 中,$N=\{3,6,9\}$为星型相关的,因为有一个星结点 3 满足$(3,6)\in R$ 和$(3,9)\in R$.这表明 xml 和 c ++ 这两本书都是由名为 high education 的出版社出版的.但 $N=\{3,6,9\}$ 不是全相关的,因为(6,9)不相连.

定义 5　语义相关的查询结果

设用户查询表达式形为 $Q(S)$,$S=t_1,\cdots,t_m$ 为查询项序列,其形式为 $l:k$,l:或:k,其中 l 为标签,k 为关键词.如果结点 n 的标签为 l 且其后裔结点的值包含关键词 k,称结点 n 满足查询项 $l:k$;如果结点 n 的标签为 l,称结点 n 满足查询项 l:;如果结点 n 为叶子结点且其值包含关键词 k,称结点 n 满足查询项:k.

满足 Q 中不同查询项的各个结点之间必须是语义相关的.称结点序列 $N=n_1,\cdots,n_m$ 为 Q 的星型相关结果(star R-answer)或全相关结果(all-pairs R-answer)如果 N 中的结点为星型相关(star R-related)或全相关(all-pairs R-related).

如果结点序列 $N=n_1,\cdots,n_m$ 为星型相关结果(star R-answer)或全相关结果(all-pairs R-answer)则称其为 Q 的一个语义相关结果.因为星型相关比全相关条件宽松,因此可以检索到更多的结果.

3　新的索引结构

本文提出了一种新的索引结构,主要包括倒排元素标签索引(ETI),倒排元素值索引(ECI)及结点层次-路径索引(NLPI),如图 2、3、4 所示.其中,docID 是文档号,用来唯一标识 XML 文档; nodeID 是某一 XML 文档树中的结点标识,是按照深度优先遍历顺序对 XML 文档树遍历得到的序号.(docID,nodeID)可以唯一标识一个元素结点或属性结点.W 为关键词 k 在某个叶子结点 n 中的权值,$W(k,n)$定义如下:

$$W(k,n)=occ(k,n)/\max(occ(ki,n)) \qquad (1)$$

式(1)中 $occ(k,n)$为关键词 k 在元素结点 n 的出现次数.

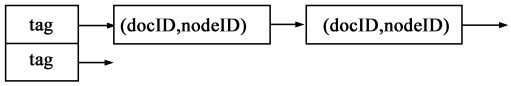

图 2 倒排元素标签索引 (ETI)

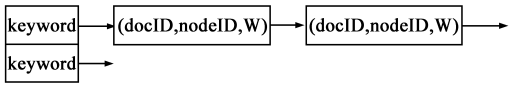

图 3 倒排元素值索引 (ECI)

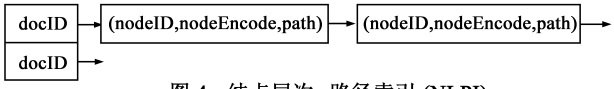

图 4 结点层次 - 路径索引 (NLPI)

nodeEncode 为结点编码,由该结点父结点的编码和其在兄弟结点中的索引号连接而成,中间用“.”分割.如图 5 所示.Path 为结点的路径,如图 5 中结点“1.1.1”的路径为“animals/animal/name”.为节省存储空间,结点层次-路径索引表中的 path 用路径标识 pathID 代替,再增加一个路径索引 PATH(pathID,path).

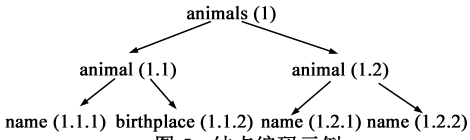

图 5 结点编码示例

ETI 和 ECI 用来查询与关键词相匹配的结点,NLPI 提供了结点的层次结构和路径信息,利用这些信息可以方便快捷地判断两个结点的相连关系.例如,比较图 5 的两个结点“1.1.1” 和“1.1.2”,找到其相同的前缀字符串“1.1.”,“1.1.”有两个点符号,然后对“1.1.1”结点的路径“animals/animal/name”和“1.1.2”结点的路径“animals/animal/birthplace”从第 2 个“/”开始分别截取子串,得到字符串”/name” 和“/ birthplace”.“/name”和 “/birthplace”没有类似“/…./”的相同子串,则说明结点“1.1.1”和“1.1.2”满足相连关系.不同于文献[8],本文判断两个结点的相连关系仅仅通过比较两个结点的编码和路径,此外,本文的索引结构中并没有存储结点对相连关系的索引表,大大减少了内存的开销.

4 XML 语义搜索和快速语义搜索算法

4.1 XML 语义搜索的预处理

(1)查询推荐

对于形为:k 的查询项,利用查询请求推荐技术进行扩展,将符合该查询项的叶子结点标签列出,供用户选择确认.最终查询请求 $Q(t_1,t_2,\cdots t_m)$中的查询项都为标签-关键词或标签的形式.

(2)查询结果范围的确定

设$(n_1,n_2,n_3,\cdots,n_k)$为一个语义相关的结点集,其对应的查询结果范围为以这些结点的最低层公共祖先结点为根结点的子树所代表的 XML 片段.

(3)多个结点星型相关语义的近似

给定一组结点,在该结点集中层次最高(离根结点近)的结点为一个星结点时,称该结点集满足限定的星型相关.满足限定星型相关的结点集也是语义相关的.

设检索请求 $Q(t_1,t_2,\cdots,t_m)$的检索结果用 $Answer(Q)$表示.用 $Answer^A(Q)$表示满足全相关的检索结果,$Answer^S(Q)$表示满足星型相关的检索结果,$Answer^{RS}(Q)$表示满足限定星型相关的检索结果,有:

$$Answer^A(Q)\subseteq Answer^{RS}(Q)\subseteq Answer^S(Q) \quad (2)$$

对查询项 $t_1,t_2,\cdots,t_m$ 分别在索引表中查找第一个匹配结点得到 $n_1,n_2,\cdots,n_m$,如果在结点 $n_1,n_2,\cdots,n_m$ 中层次最高的结点为 n_i,则根据 XML 文档子结构的重复性可以认为与 t_i 匹配的结点集层次最高,这个结点集就作为可能的星结点集.

4.2 XML 语义搜索的理论基础

设查询请求为 $Q(t_1,t_2,\cdots,t_m)$,与 t_i 查询项匹配的结点集用$N(t_i)$表示,与 t_j 查询项匹配的结点集用 $N(t_j)$表示.

命题 1 如果 $n1^*,n2^*\in N(t_i)$,而且 $n1^*,n2^*$是兄弟关系,则在判断结点相连关系时 $n1^*$ 与 $n2^*$ 具有相同的性质,即如果$(n1^*,n)\in R$,则有$(n2^*,n)\in R$,如$(n1^*,n)\notin R$,则有$(n2^*,n)\notin R$,其中 $n\in N(t_j)$.

证明:(1)如果$(n1^*,n)\in R$,则$(n2^*,n)\in R$

当 $n1^*,n2^*$ 是兄弟关系,$(n1^*,n)\in R$ 时,$n1^*$,$n2^*$和 n 在文档树中的位置有两种情况,分别如图 6 和图 7 所示.如果属于图 6 情况,则显然有$(n2^*,n)\in R$.对于图 7 情况,由于 $n1^*,n2^*$是标签相同的兄弟结点,则组成 $T_{|n1^*,n}$和 $T_{|n2^*,n}$的结点集的标签完全相同,因为$(n1^*,n)\in R$,所以有$(n2^*,n)\in R$.

证明:(2)如$(n1^*,n)\notin R$,则$(n2^*,n)\notin R$

当 $n1^*,n2^*$ 是兄弟关系,$(n1^*,n)\notin R$ 时,$n1^*$,$n2^*$和 n 在文档树中的位置必然如图 7 所示.由于 $n1^*,n2^*$ 是标签相同的兄弟结点,组成 $T_{|n1^*,n}$ 和 $T_{|n2^*,n}$的结点集的标签完全相同,因为$(n1^*,n)\notin R$,所以有$(n2^*,n)\notin R$.

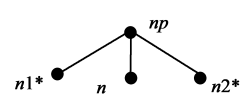

图 6 $n1^*,n2^*$和 n 互为兄弟结点情况

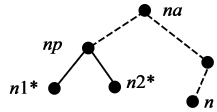

图 7 $n1^*,n2^*$和 n 所处位置的一般情况

命题 2 设 $N(t_i)$为匹配结点集中层次最高的,即 $N(t_i)$为可能的星结点集.已知结点 $n1^*,n2^*\in N(t_i)$,且 $n1^*,n2^*$不为兄弟关系;结点 $n\in N(t_j)$,R 为结点相连关系.如果$(n1^*,n)\in R$,则 $n2^*$ 和 n 的相连关系

或者不需要判断或者有$(n2^*,n)\notin R$.

下面分两种情况证明.

(1)$n1^*$为 n 的祖先

证明:因为 $n1^*,n2^*$标签相同,且 $n1^*$为 n 的祖先,则在语义清晰的 XML 文档中 $n2^*$与 $n1,n$ 均不能为祖先后代关系,$n2^*$与 $n1,n$ 在文档树的位置关系如图 8 所示.由于 $T_{\backslash n2^*,n}$中有两个标签相同的不同结点 $n1^*$和 $n2^*$,所以$(n2^*,n)\notin R$.

图 8　$n1^*$和 n 为祖先后代关系的情况

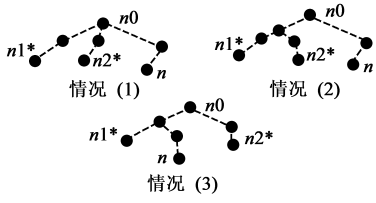

图 9　$n1^*$, $n2^*$和 n 的 3 种位置情况

(2)$(n1^*,n)\in R$,$n1^*$不为 n 的祖先

证明:因为$(n1^*,n)\in R$,$n1^*$不为 n 的祖先,$n1^*$,$n2^*$不为兄弟关系,$n1^*$,$n2^*$标签相同,则在语义清晰的 XML 文档中 $n2^*$与 $n1^*,n$ 均不能为祖先后代关系,那么 $n2^*$与 $n1^*,n$ 在文档树中的位置关系如图 9 所示.对于图 9 中的(1)和(2)两种情况,由于$(n1^*,n)\in R,n1^*,n$ 的最低层公共祖先为 $n0$,如果$(n2^*,n)\in R$,$n2^*$与 n 的最低层公共祖先也为 $n0$,这时不判断 $n2^*$与 n 的相连关系不会影响最终结果.因此当判断出$(n1^*,n)\in R$时,不需要再判断 $n2^*$与 n 的相连关系.对于图 9 中的情况(3),根据语义清晰的 XML 文档的定义,结点 $n1^*$和 $n2^*$的处于相同层次的祖先结点为同类型的或相似类型的.在 $T_{\backslash n2^*,n}$中必有两个标签相同或类型相似的不同结点,所以$(n2^*,n)\notin R$.

综上所述,如果$(n1^*,n)\in R$,则 $n2^*$和 n 的相连关系或者不需要判断或者有$(n2^*,n)\notin R$.

4.3　XML 快速语义搜索算法

XML语义搜索包括两部分,一是对各查询项搜索匹配结点,二是对各匹配结点进行语义相关的判断.本文基于 4.2 节的概念和理论,结合对深度优先遍历得到的结点序号按照升序排序的处理,对各匹配结点进行语义相关的判断过程可以进行如下简化:

(1)只需对属于相同文档号的结点进行相连关系判断;

(2)星结点集 $N(t_i)$中两个或多个星结点为兄弟关系时,只需对其中一个进行与 $N(t_j)$中结点相连关系的判断;

(3)对于星结点 $n1^*\in N(t_i)$,由结点集 $N(t_j)$的当前位置结点开始,查找与 $n1^*$满足相连关系的结点.如果找到一个或若干个 n 与 $n1^*$满足相连关系后下一个结点 n'与 $n1^*$不相连时就不再继续找.然后由 $N(t_i)$中下一个星结点 $n2^*$开始,与 $N(t_j)$中当前位置开始的结点进行相连关系判断.如图 10,对 $N(t_i)$中的结点“1.3”,在 $N(t_j)$中找到结点“1.3.3”与其相连后,指向 $N(t_j)$的指针下移指向结点“1.4.3”,经判断结点“1.4.3”与结点“1.3”不相连,此时指向 $N(t_j)$的指针不再继续下移.指向 $N(t_i)$的指针移到结点“1.6”处,与 $N(t_j)$中当前结点“1.4.3”进行比较.图 10 中实线箭头表示所指向的两结点满足相连关系,虚线箭头表示从箭头指向的位置开始搜索.

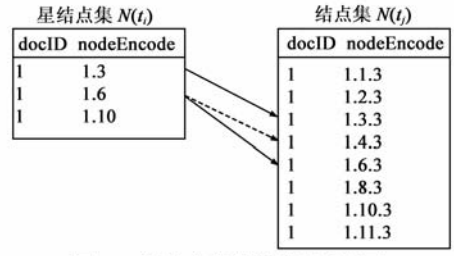

图 10　语义相关判断过程示意图

下面是本文提出的快速语义搜索算法:

```
输入:Q(c1,..ck),ETI,ECI,NLPI/ 与 c1 匹配的结点为星结点
输出:answer   //语义相关结果
算法:
answer = Selected node from ETI, ECI, NLPI that satisfying c1 order by docID, nodeID;
For each term ci in Q(c2,..ck){
  Group = Selected nodes from ETI, ECI, NLPI that satisfying ci order by docID, nodeID;
  pos = 0; Candanswer = Null; find = false;
  for each element ej in answer {
    nj = (ej)*;//answer 中第 j 个元组的星结点
    if(match = false) and (the end of GROUP)
      Set the current position of GROUP to pos;
    match = false;
    while (not to the end of GROUP) {
      n = GROUP(pos)   //当前位置的结点
      if(intercon(nj,n) = true) {
        e = ej join n; Add e into Candanswer;
        find = true; match = true;
        break;
        }
        pos + +;
      }
      If (find = true) answer = Candanswer;
      ELSE return NULL;
  }
  return answer;
```

图 11　快速语义搜索算法

函数 intercon 通过比较两个结点的编码和路径完

成两个结点相连关系的判断,见第 3 节介绍.

由此看出,本文算法在判断多个结点间的语义相关时避免了大量的结点对相连关系的判断,大大提高了检索速度.详见实验分析结果.

5 实验分析

本文列出了 3 种方法下的实验结果进行比较.第一种方法是用文献[8]的索引及星型相关结果查找算法,简称文献[8]方法;第二种是用文献[8]的星型相关结果查找算法,本文的索引及相连函数,简称普通检索算法;第三种是用本文的索引、相连函数及快速限定星型相关结果查找算法,简称快速检索算法.实验环境:800MHZ Pentium Ⅲ计算机,128M 内存,Microsoft Windows2000 Server 操作系统,索引表用 SQL SERVER 2000 关系数据库存储,用 Java 语言开发.数据集来自 ACM SIGMOD 主页,其中一个是 64KB 的 XML 文档(222.xml),另一个是大小约为 500KB 的 SigmodRecord.xml[9].表 1 列举了 8 种不同类型的查询请求.表 2 列出 3 种方法下对 222.xml 文档的检索结果及花费时间,表 3 列出 3 种方法下对 SigmodRecord.xml 文档的检索结果及花费时间.

表 1 8 种查询请求

Q	查询项组成
Q1	Title:database;author:
Q2	Title: ;author:wei sun
Q3	Title: ;author:
Q4	Title:database;author:ben kao
Q5	:database; :peter muth
Q6	Title: ;author:weining zhang
Q7	Title:database;author:hanan boral
Q8	:database; :hanan boral

表 2 对 222.XML 文档的检索结果比较

Q	结果数	检索时间(秒)		
		文献[8]方法	普通检索算法	快速检索算法
Q1	49	740	2.62	**2.23**
Q2	1	13.71	1.20	**0.83**
Q3	239	3104.7	5.1	**2.37**
Q4	1	3.25	0.81	**0.35**
Q5	1	6.31	0.73	**0.31**

表 3 对 SigmodRecord.xml 文档的检索结果比较

Q	结果数	检索时间(秒)		
		文献[8]方法	普通检索算法	快速检索算法
Q1	734		120.43	**43.04**
Q6	1	578.54	10.90	**1.74**
Q3	4196		544.93	**40.84**
Q7	1	99.10	10.25	**9.7**
Q8	1	96.97	2.04	**0.32**

从表 2 和表 3 看出,普通检索算法要比文献[8]方法效率高,说明利用本文的索引结构与相连函数实现两结点相连关系的判断要比文献[8]的方法有效,尤其是对于结点总数为 9007(去除了属性结点后)的 XML 文档 SigmodRecord.xml,用文献[8]方法需要 81117042 (9007^2-9007)个空间存储所有结点对的相连关系,即使只建立部分相连关系索引,假设部分相连关系索引占所有结点对相连关系索引的 0.75%,也需要大约 608377 个空间.用哈希函数存储该索引时,由于解决地址冲突带来的时间开销,使文献[8]方法的效率很低.对于查询 Q1 和 Q3,用文献[8]方法所需检索时间很长(以小时计),在表 3 中没有列出.在 3 种方法中快速检索算法是最快的一种,尤其是当检索结果数越大时快速检索算法的优势越明显,因为这时结点相连关系判断时间是检索时间的主要部分,而快速检索算法省却了大量的结点对相连关系的判断.

6 结论

本文提出一种新的 XML 索引结构及基于该索引结构的两结点相连判断方法.对 XML 文档标签及文档结构所隐藏的语义做了进一步的分析,提出了相关定理及其证明,基于所提出的理论基础提出了一种快速的语义检索算法,实验结果表明本文所提出的 XML 索引结构及快速语义检索算法大大提高了检索效率,这在 XML 语义搜索引擎应用中是非常实用的.

参考文献:

[1] Theobald,G Weikum. The index-based XXL search engine for querying XML data with relevance ranking[A]. 8th International Conference on Extending Database Technology (EDBT)[C]. Prague: Springer-Verlag, 2002. 477 – 495.

[2] 吴劲,陈泽琳.基于部分匹配的 XML 文本文档向量检索模型[J].电子学报,2002,30(12A):2169 – 2171.
Wu Jin, Chen Ze-lin. Vector retrieval modeling using partial match pattem for text-rich XML documents[J]. Acta Electronica Sinica, 2002, 30(12A): 2169 – 2171. (in Chinese)

[3] 曲卫民.中文 XML 信息检索系统的研究[D].北京:中科院软件研究所,2004.
Qu Wei-min. Research on chinese XML information retrieval system[D]. BeiJing: Institute of Software, Chinese Academy of Science, 2004(in Chinese)

[4] 王晓燕,王海洋,等.自行调整粒度的 XML 向量空间检索[J].武汉大学学报(理学版),2004,50(5):6O9 – 613.
Wang Xiao-yan, Wang Hai-yang, et al. XML Vector Space Retrieval of automatic granularity[J]. Journal of Wuhan University (Natural Science Edition), 2004, 50(5): 609 – 613. (in Chinese)

[5] 王海波,姜吉发,等.XML 搜索引擎研究[J].计算机应用研究,2001,18(4):68 – 71.
Wang Hai-bo, Jiang Ji-fa, et al. Research of XML search engine [J]. Application Research Of Computers, 2001, 18(4): 68 – 71. (in Chinese)

[6] 孙伟.基于 XML 半结构数据索引的研究[D]. 哈尔滨:哈

尔滨工程大学,2004.
Sun Wei. Research of Semistructured Data Index Technology Based on XML[D]. Harbin: Harbin Engineering University, 2004.(in Chinese)
[7] 郭永民.XML 文档检索技术研究[D].太原:太原理工大学,2003.
Guo Yong-min. The Research of XML Documents Retrieval [D]. TaiYuan: Institutes of Technology of Taiyuan, 2003.(in Chinese)
[8] Sara Cohen, Jonathan Mamou, et al. XSEarch: a semantic search engine for XML[A]. Proceedings of the 29th VLDB Conference[C]. Berlin : Morgan Kaufmann Publishers, 2003. 45 – 56.
[9] ACM SIGMOD. Available Products[DB/OL]. http://www.acm.org/sigmod/record/xml,2006-12-01/2007-3-10.

作者简介:

李新叶　女,1969 年出生于河北平山,华北电力大学电子与通信工程系,博士.研究方向包括智能信息处理、信息检索等.
E-mail:yljh654@sina.com

苑津莎　男,1957 年生于天津,华北电力大学教授,博导.研究方向包括智能信息处理、电磁场、计算机网络等.