ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.4, April 2009, pp.910–917
doi: 10.3724/SP.J.1001.2009.03224

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 Nearest Pair 的 XML 关键词检索算法*

吉聪睿, 邓志鸿+, 唐世渭

(北京大学 信息科学技术学院 智能科学系,北京 100871)

An XML Keyword Retrieval Algorithm Based on Nearest Pair

JI Cong-Rui, DENG Zhi-Hong+, TANG Shi-Wei

(Department of Machine Intelligence, School of Electronic Engineering and Computer Science, Peking University, Beijing 100871, China)

+ Corresponding author: E-mail: zhdeng@cis.pku.edu.cn, http://www.cis.pku.edu.cn/faculty/system/dengzhihong/dengzhihong.htm

Ji CR, Deng ZH, Tang SW. An XML keyword retrieval algorithm based on nearest pair. *Journal of Software*, 2009,20(4):910–917. http://www.jos.org.cn/1000-9825/3224.htm

Abstract: As more and more data are expressed and stored in XML format, the study on XML keyword retrieval becomes the focus of IR (information retrieval) and Database. This paper gives and proves some properties of SLCA (smallest lowest common ancestor), which is the key concept of XML keyword retrieval. It also introduces a new XML keyword retrieval algorithm, Nearest Pair, on the basis of the properties above. This algorithm uses the iterative bi-search technology to look for nearest pairs, which can decrease the assistant computation by one order of magnitude. The experimental results show that Nearest Pair outperforms the existing mainstream algorithms in most cases.

Key words: XML; keyword retrieval; SLCA (smallest lowest common ancestor)

摘 要: 随着大量数据以 XML 格式保存,针对 XML 文档的关键词检索技术已经成为信息检索和数据库等相关领域的研究热点.以树的杜威编码为基础,分析并证明了 XML 关键词检索中核心概念 SLCA(smallest lowest common ancestor)的两个重要性质,并在其基础上提出了 Nearest Pair 算法.该算法采用二分迭代查找技术寻找最邻近点,将求解中间结果的次数降低了一个量级.实验结果表明,该算法的性能在绝大多数情况下优于现有主流算法.

关键词: XML;关键词检索;最小公共祖先集合

中图法分类号: TP311 **文献标识码**: A

XML 作为一种新兴的数据格式,因其能够携带元数据信息,能够表达数据之间的结构关系等优点而得到广泛认同和使用.XML 是一种典型的半结构化数据格式:与 HTML 相比,XML 能够表达更多的语义信息.如图 1 所描述的 3 本书的信息,其中每本书的书名、出版社、作者等属性都得到了显式的标注.与关系数据库相比,XML 更加灵活,允许使用不同的标记,如书名既可以用 title 来标注,也可以用 name 来标注.

目前,XML 已经成为数据传输和数据集成的标准格式,广泛应用于电子商务、数字图书馆、内容管理以及中间件等领域.随着 XML 文档数量的增多,对 XML 进行检索的要求也被提了出来.虽然 W3C 陆续颁布了 XML

* Supported by the PKU-FUJISU Yong Scholar Foundation of China (北京大学-富士通青年基金)
Received 2007-08-30; Accepted 2007-11-02

的检索标准 XPath 和 XQuery,但这些标准不仅语法复杂,而且还要求用户必须熟悉底层文档的结构,不利于一般用户的学习和使用.

```
⟨book⟩
  ⟨author⟩
    ⟨No1⟩Ben⟨/No1⟩
    ⟨No2⟩Tom⟨/No2⟩
  ⟨/author⟩
  ⟨title⟩
    …
  ⟨/title⟩
  ⟨publisher⟩
    John Cena Press
  ⟨/publisher⟩
⟨/book⟩
```

```
⟨book⟩
  ⟨author1⟩John⟨/author1⟩
  ⟨author2⟩Ben⟨/author2⟩
  ⟨name⟩
    …
  ⟨/name⟩
  ⟨press⟩
    …
  ⟨/press⟩
⟨/book⟩
```

```
⟨book⟩
  ⟨author⟩
    ⟨No1⟩Peter⟨/No1⟩
    ⟨No2⟩Ben⟨/No2⟩
  ⟨/author⟩
  ⟨title⟩
    John Kennedy
  ⟨/title⟩
  ⟨publisher⟩
    …
  ⟨/publisher⟩
⟨/book⟩
```

Fig.1 Examples of XML documents

图 1 XML 文档示例

从 2003 年起,XML 检索领域的研究重心逐渐转移到了 IR 方向,研究人员希望借鉴 HTML 搜索引擎的成功经验,利用关键词查询界面的简单、易用来吸引用户[1−6].与 HTML 检索相比,XML 检索存在以下几个重要的不同:

(1) 最小元素不同:HTML 检索以文档为最小元素,系统的返回结果也是一系列符合要求的文档;而 XML 的检索和返回结果都是以节点为最小元素的,如图 1 中的⟨title⟩等标记.

(2) 元素之间的关系不同:在 XML 中,节点间存在包含和被包含的关系,这是 HTML 检索中所没有的.在 XML 中,如果一个节点包含某个关键词,则认为该节点的所有祖先都包含这个关键词.如图 1 所示的第 1 本书中,因为节点⟨No1⟩包含“Ben”,所以节点⟨author⟩也包含“Ben”.

节点间的包含关系提出了 XML 检索中的两个最重要的问题.首先是索引的冗余问题:在建立倒排索引时,除了要记录那些直接包含关键词的节点外,还要记录这些节点的所有祖先节点;其次是如何回答多关键词查询的问题:返回结果不仅要包含所有的关键词,还要尽可能地贴近文档的语义,也就是说,返回结果要尽可能地剔除冗余的祖先节点.

本文的主要内容包括以下几个部分:

- 讨论目前对 XML 进行编码的主流方法:Dewey ID 编码方法和前后序周游编码方法.前者的功能较全面,而后者在空间和时间上的表现更为出色.
- 分析 SLCA(smallest lowest common ancestor)的性质.目前,主流 XML 关键词检索算法都是通过计算节点集合的 SLCA 来回答多关键词查询的.本文在此基础上提出了一种新的 XML 关键词检索算法,该算法在性能上优于目前现有的主流算法.
- 分析比较本文提出的算法和目前主流 XML 查询算法的性能,指出它们之间的相互关系和优缺点,以及在不同的关键词密度分布情况下算法的优劣.

本文第 1 节讨论 XML 的文档模型和编码方法.第 2 节分析 SLCA 的性质,提出新的 XML 关键词检索算法——Nearest Pair.第 3 节分析实验结果.第 4 节总结全文.

1 XML 文档模型

将 XML 的文本节点看作节点,节点间的关系看作边,由于 XML 节点之间也存在着跨节点或者跨文档的情况,类似 HTML 的超链接(如 XLink 等),所以,每个 XML 文档都可以被看作是一个图.将 XML 作为图来建模的算法有文献[7,8]等等,但目前的主流方法是把每个 XML 文档看作是一棵独立的树.这是因为同一文档内部间的节点关系要更加复杂,也更加重要.另外,树模型的相关算法效率也要远远高于图模型的相关算法效率.

节点之间的包含关系在表达语义信息的同时,也给 XML 关键词检索系统带来了难题.

首先是索引的冗余问题.如果一个节点包含某个关键词,那么该节点的所有祖先都包含这个关键词.如图 1 所示中的第 1 本书,关键词 Ben 的索引中,除了要记录直接包含它的 No1 节点外,还要记录 No1 的所有祖先节点.文档的层次越深,索引的冗余问题就越严重.

其次是多关键词检索问题.为了更好地反映文档的语义,多关键词查询的返回结果要满足两个条件:一是包含所有的关键词,二是该节点的任意子节点不满足前面所述的条件.如图 1 所示中的第 1 本书,检索 Ben 和 Tom 的返回结果应该是 author,而不是 book.因为 author 与关键词的距离更近、更能确切地表示出 Ben 和 Tom 的语义.

目前,索引的冗余问题已经得到了很好的解决.因为通过合理的编码方法,系统可以在常数时间内判断一个节点是否包含另一个节点.目前的主要编码方法有前后序周游编码和 Dewey ID 编码两种.

在前后序周游编码方法中,每个节点的 ID 由两个数字组成,分别是该节点在前序周游序列和后序周游序列中的位置.如图 2 中节点 *C*1 的编码为[3,4],而 *A*1 的编码为[4,1].因为[3,4]“包含”[4,1](3<4 而 4>1),所以 *C*1 包含 *A*1,是 *A*1 的祖先节点.

使用前后序周游的编码方法能够判断两个节点之间是否存在包含关系,但在 XML 检索中,更多的需求是找出某个节点的所有祖先节点,在这种情况下,前后序周游编码就无能为力了.

Dewey ID 是一种经典的编码方法,在 XML 中,使用 Dewey ID 对节点进行编码,可以得知每个节点的所有祖先信息.如图 2 中,*A*3 的编码为⟨1.1.2.2⟩,表示它有 3 个祖先,编码分别为⟨1⟩,⟨1.1⟩和⟨1.1.2⟩.除此之外,使用 Dewey ID 编码可以在 $O(d)$时间得知两个节点的关系以及它们的公共祖先,这里,d 为文档树的深度.如在图 1 所示的第 1 本书中,Ben 对应的节点为⟨1.1.1⟩,Tom 对应的节点为⟨1.1.2⟩,两个 Dewey ID 的公共部分即为它们的公共祖先节点,分别为⟨1⟩和⟨1.1⟩.通过比较公共祖先和节点自身的编码,可以得知这两个节点是兄弟节点.上述这两个性质在处理 XML 多关键词查询中起着极为重要的作用.

从直观上讲,Dewey ID 编码需要更多的存储空间,特别是当文档层次很深的时候.在实际应用中,为了压缩空间,往往直接用 bit 来表示 Dewey ID.如图 2 中的文档,每一层分别有 1,2,5,7 个节点,所以每一层所分到的位数分别为 1,2,3,3.这样,A3 的⟨1.1.2.2⟩可以表示为⟨1,01,010,010⟩,即整数 338.但是如果文档的每一层都有很多节点,并且分布非常不均匀,则压缩的效果就不是很理想了.在实验中,千万量级的节点数目就需要使用 64 位整数进行存储.

虽然 Dewey ID 编码的时空效率并不是非常理想,但其强大的功能是其他编码方法所不能比拟的.目前的主流 XML 检索算法均使用 Dewey ID 编码.

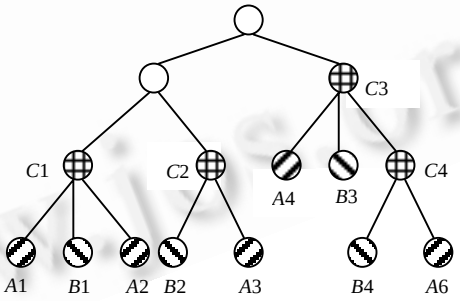

Fig.2 An XML document tree

图 2 一棵 XML 文档树

2 SLCA 的性质

目前,XML 检索系统所面临的最大问题是如何处理多关键词的查询,而这个问题的关键则在于如何获得那

些离关键词最近的节点.如果采用类似 HTML 的方法,在倒排表中存储包含关键词的所有节点(包括所有祖先节点),一方面会造成索引的冗余,不仅浪费空间,而且在求解索引交集时也要付出更多的代价;另一方面,得到交集后,还要花费额外的代价去除冗余的祖先节点.

按照 XML 多关键词检索的定义,结果中应该只包含这样的节点:包括所有的关键词,并且其任意子节点不满足前面所述的条件,这恰好符合 SLCA 的定义.

为了论述方便,这里先简单介绍几个名词和符号的含义.

- SLCA:SLCA 是 Smallest Lowest Common Ancestor 的英文简称,中文翻译为最小公共祖先集合,SLCA 是针对集合与集合之间而言的.$SLCA(A,B)$表示 A,B 两个集合的最小公共祖先集合.如图 2 所示,$SLCA(A,B)=\{C1,C2,C4\}$.
- LCA:LCA 是 Lowest Common Ancestor 的简写,针对节点和节点而言.如 $LCA(A1,B1)=C1$,$LCA(A4,B3)=C3$.
- SLCA 具有结合律,即 $SLCA(A,B,C)=SLCA(SLCA(A,B),C)$,所以,求解两个集合间 SLCA 的算法也可以求解任意多集合间的 SLCA.
- Dewey ID 在 SLCA 相关定理的证明中有重要的作用,下文将把 $DeweyID(a)<DeweyID(b)$简记为 $a<b$,其中,$DeweyID(a)$表示节点 a 的 Dewey ID.

求解两个集合间 SLCA 的朴素算法可以分为两个步骤:首先求解每个元素对之间的 LCA,然后再在所有的 LCA 中去掉祖先节点.如在上例中,$C3$ 虽然是 $A4$ 和 $B3$ 的 LCA,但是 $C3$ 的子节点 $C4$ 也是 LCA,所以 $C3$ 不会出现在最后的结果中,即 $C3$ 不是 SLCA 节点.利用 Dewey ID 编码可以在 $O(d)$时间内获得两个节点的 LCA,按照朴素算法,如果两个集合的大小分别为 m 和 n,算法的代价为 $O(dmn)$.

SLCA 具有如下性质:

引理 1. 设 $c=LCA(a,b)$且 $c\in SLCA(A,B)$.设在 A 中另有节点 a'与 b 的距离更近,则 $c=LCA(a',b)$.这里的距离指的是两个节点编码间的差值.

证明:设 $DeweyID(a)=c_1c_2\ldots c_ka_{k+1}\ldots a_l$,$DeweyID(b)=c_1c_2\ldots c_kb_{k+1}\ldots b_m$,其中,$DeweyID(c)=c_1c_2\ldots c_k$,则 $a_{k+1}\neq b_{k+1}$.设 $DeweyID(a')=a'_1a'_2\ldots a'_ka'_{k+1}\ldots a'_n$,则:

必有 $a'_i=c_i(1\leq i\leq k)$,否则与 a'与 b 的距离更近相矛盾;

必有 $a_{k+1}\neq b_{k+1}\neq a'_{k+1}$,否则,如果 $b_{k+1}=a'_{k+1}$,则 $DeweyID(LCA(a',b))=c_1c_2\ldots c_kb_{k+1}$,这样,$LCA(a',b)$就是 c 的子节点,与 c 是 SLCA 节点相矛盾.

所以,$DeweyID(LCA(a',b))=c_1c_2\ldots c_k$,即 $c=LCA(a',b)$. □

由引理 1 可得如下定理:

定理 1. 设 $SLCA(A,B)=C$,则 C 中的任意节点 c 满足如下性质:$c=LCA(a,b)$,其中,a,b 是集合 A 和 B 中两个彼此距离最近的节点.

如图 2 中,$C1\in SLCA(A,B)$,而 $C1=LCA(A1,B1)$,其中,$B1$ 是集合 B 中离 $A1$ 最近的节点,而 $A1$ 也是集合 A 中离 $B1$ 最近的节点.

因为 $c\in SLCA(A,B)$,根据引理 1,必然能够找到彼此距离最近的 a 和 b,使得 $c=LCA(a,b)$. □

定理 1 的逆定理为:如果 a,b 是集合 A 和 B 中两个距离最近的节点,则 $LCA(a,b)\in SLCA(A,B)$.如果该定理成立,则集合 A,B 的 SLCA 就等于两个集合中所有最近点对的 LCA 的并集.但此逆定理并不成立,如图 2 所示中的 $A4$ 和 $B3$ 虽是彼此最近的节点,但 $LCA(A4,B3)=C3$ 并不是 SLCA 节点.

即使是这样,我们依然可以用求解点对 LCA 的方法得到集合的 SLCA,只不过要遵循一些特定的步骤.

引理 2. 设 $a<a',b<b'$,如果 $LCA(a,b)>LCA(a',b')$,则 $LCA(a,b)$是 $LCA(a',b')$的子节点.

证明:设 $c=LCA(a,b)$,$c'=LCA(a',b')$,$DeweyID(c)=c_1c_2\ldots c_m$,$DeweyID(c')=c'_1c'_2\ldots c'_n$,因为 c 是 a 的祖先节点且 $a<a'$,所以有 $c_i\leq c'_i$ $(1\leq i\leq\min\{m,n\})$,而现有 $c>c'$,所以有 $c_i=c'_i$ $(1\leq i\leq\min\{m,n\})$,且有 $m>n$.

所以,c 是 c'的子节点. □

引理 2 的意思是,如果集合 A 和 B 中的节点是按照 Dewey ID 的升序依次求解 LCA 的话,那么,这些 LCA 节点也应该是按照 Dewey ID 升序排列的,如果出现乱序,那么后生成的 LCA 必然是其前一个 LCA 的祖先节点.

引理 3. 设 $a<a',b<b'$,如果 $LCA(a,b)<LCA(a',b')$,则 $LCA(a',b')$和在其后生成的所有 LCA 节点中都没有 $LCA(a,b)$的子节点.

证明:设 c 和 c'的 Dewey ID 如引理 2 的证明中所示,因为 $c<c'$,则必然存在 $c_i< c_i'$ $(1\le i\le \min\{m,n\})$,所以 c'不是 c 的子节点.根据引理 2,在 c'后面生成的 LCA 节点 c'':

c''是 c'的祖先节点,不可能是 c 的子节点.

$c'<c''$,根据前面所述,有 $c_i< c_i' < c_i''$,从而 c''也不可能是 c 的子节点. □

引理 4. 设 $a<a',b<b'$,如果 $c=LCA(a,b)$是 $c'=LCA(a',b')$的子节点,则所有在 c 和 c'之间生成的 LCA 节点要么是 c 的祖先节点,要么是 c'的子孙节点.

证明:设 c 和 c'之间的节点为 c'':

$c''<c$,根据引理 2,c''是 c 的祖先.

$c''>c'$,根据引理 2,c'是 c''的祖先,即 c''是 c'的子孙.

$c<c''<c'$,根据 Dewey ID 的性质,c''要么是 c'的子孙,要么是 c 的祖先. □

由引理 2~引理 4,可得以下定理:

定理 2. 集合 A 和 B 中的节点按照 Dewey ID 的升序或降序排列,按照此顺序依次求解最近点对的 LCA.现有节点 a 和 b 是集合 A 和 B 中最邻近的两个节点,如果 $c=LCA(a,b)$不是 SLCA 节点,那么 c 的子节点 c'必定在 c 的前一步或后一步得到.

引理 2 讨论了“逆序”时,所生成的 LCA 节点的父子关系,引理 4 是对引理 2 的扩充论述,引理 3 补充了“顺序”时所生成的 LCA 节点的父子关系.

定理 2 的证明类似引理 4,这里不再赘述.定理 2 的意思是,如果按照节点的顺序依次生成 LCA 的话,那么,所有存在父子关系的节点必然都是“扎堆”出现的.如图 2 中的 $C3$ 和 $C4$.

事实上,引理 2~引理 4 的结果对于任意点对都成立,也即不要求 a 和 b 是彼此最邻近的节点.但根据定理 1,如果 a 和 b 不是彼此最邻近的点对,则 $LCA(a,b)$不属于 $SLCA(A,B)$的概率将大大提高.

根据定理 1 和定理 2,我们可以得到求解 SLCA 的算法 Nearest Pair:

(1) 依次找出集合 A 和 B 中的最邻近点对,求得它们的 LCA,设为 c.

(2) 将 c 和上一 LCA 结果 c'进行比较,如果 c 不是 c'的子节点,则 $c'\in SLCA(A,B)$.

算法实现如图 3 中左图所示.

函数 *getSLCA* 的主要代价是求解节点间的 LCA,以及本次结果和上次结果的比较,对于指定的 XML 数据来说,这两种运算的代价都是固定的.所以算法的性能取决于函数 *getNearestPair* 的代价.

可以使用逐个扫描的方法得到集合 A 和 B 的所有最邻近点对,该方法要求集合 A 和 B 都是按升序排列的,其过程与归并排序类似,如图 4 中左图所示.

二分法是在有序集合中查找元素的最佳方法.同样可以使用二分法查找有序集合中的最邻近点对.首先在 B 中找到 $A[i]$的最邻近元素,设为 $B[j]$,然后再在 A 中查找 $B[j]$的最邻近元素,设为 $A[i']$.如果 $i=i'$,则 $A[i]$和 $B[j]$就是最邻近点对,否则继续查找 $A[i']$在 B 中的最邻近元素 $B[j']$,再看 j 和 j'是否相等,如图 4 中右图所示.

函数 *getNearestPair*1 和 *getNearestPair*2 分别对应上述的扫描法和二分查找法,代码如图 3 中中图和图 3 中右图所示.由于 *getNearestPair*2 的具体实现比较复杂,所以这里只写出其流程的大概.在我们提出的算法 Nearest Pair 中将使用 *getNearestPair*2,即通过迭代二分查找法来寻找集合间的所有最邻近点对.

函数 *getNearestPair*1 需要依次扫描 A 和 B 中的所有元素,代价为 $O(m+n)$,其中,m 和 n 分别为集合 A 和 B 的大小.而在比较理想的情况下,函数 *getNearest*2 能够迅速定位到最邻近点对的可能位置,从而避免了依次扫描的代价.此外,集合的待查部分也将随着算法的运行而迅速减小,如在图 3 中,当 A 中的 15 定位到 B 中的 17 后,B 中的 17 在 A 中查找最邻近元素时,将不用关心 A 中 15 以前的部分.

```
function getSLCA(set A,set B)
{
   c_last=NULL;
   while (there exists nearest pair)
   {
      (a,b)=getNearestPair(A,B);
      c=lca(a,b);
      if (c is not the sub of c_last)
         add c_last to result;
      c_last=c;
   }
}
```

```
function getNearest1(set A,set B)
{
   if (A[pA]<B[pB])
   {
      while (A[pA]<B[pB])
         ++pA;
      dist1=dist(A[pA],B[pB]);
      dist2=dist(A[pA−1],B[pB]);
      if (dist1<dist2)
         return (A[pA],B[pB]);
      else
         return (A[pA−1],B[pB]);
   }
   else
      …;
}
```

```
function getNearest2(set A,set B)
{
   j=find(B,A[i]);
   i′=find(A,B[j]);
   if (i==i′)
      return (A[i],B[j]);
   else
   {
      find nearest B[j′] of A[i′];
      compare j and j′;
      …;
   }
}
```

Fig.3 Pseudo code of algorithm Nearest Pair

图 3 算法 Nearest Pair 的伪码

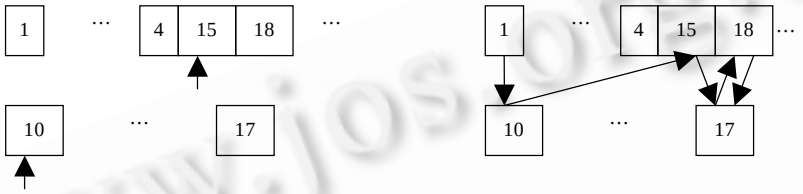

Fig.4 Example of function *getNearest*

图 4 函数 *getNearest* 的运行实例

3 实验结果与分析

栈算法和 Eager Search 算法是目前 XML 关键词检索领域中应用最为广泛的两种算法.

栈算法最早由文献[9]提出,因其需要使用一个栈来保存中间结果而得名.栈算法是第 1 个使用 Dewey ID 对 XML 节点进行编码的算法,算法的复杂度为线性.但由于使用 Dewey ID 可以直接计算节点间的公共祖先(即 LCA),所以栈算法的压栈和弹出操作在某种程度上是不必要的.此外,栈算法无法预测节点集合间的分界,需要计算每一个节点和中间结果的公共祖先,浪费了大量时间.

Eager Search 算法由文献[10]提出,是第 1 个使用 SLCA 的概念来解决 XML 多关键词查询的算法.Eager Search 深入研究了 SLCA 的性质,并给出了求解单个节点和集合间 SLCA 的算法.但是,Eager Search 并没有提出“最邻近点对”的概念.在关键词节点集合的大小较为悬殊时,Eager Search 具有出色的性能,但是如果所有的关键词节点集合都很大,那么算法的表现就不那么出色了.

本文分别比较了求解两个关键词和 3 个关键词的 SLCA 时,Eager Search 算法和 Nearest Pair 算法的性能以及相应函数的调用次数.本文的实验数据来自 DBLP,使用 Berkeley DB 作为外存索引.379M 的 XML 文件中共有节点 10 925 944 个,Dewey ID 的长度为 41 位.实验用机器为一部配有 1.66G 双核 CPU,1G 内存的笔记本.

实验搭载了 3 种环境,其主要区别在于关键词节点集合的大小,见表 1,实验结果如图 5 所示.

Table 1 Keyword-Based queries used in our experiment

表 1 实验用关键词查询

		Setting 1	Setting 2	Setting 3
Queries with two keywords	The size of node set of keyword 1	1 559 583	1 559 583	118 971
	The size of node set of keyword 2	900 578	109112	109 112
Queries with three keywords	The size of node set of keyword 1	1 559 583	1 559 583	120 478
	The size of node set of keyword 2	900 578	189 971	118 971
	The size of node set of keyword 3	889 651	109 112	109 112

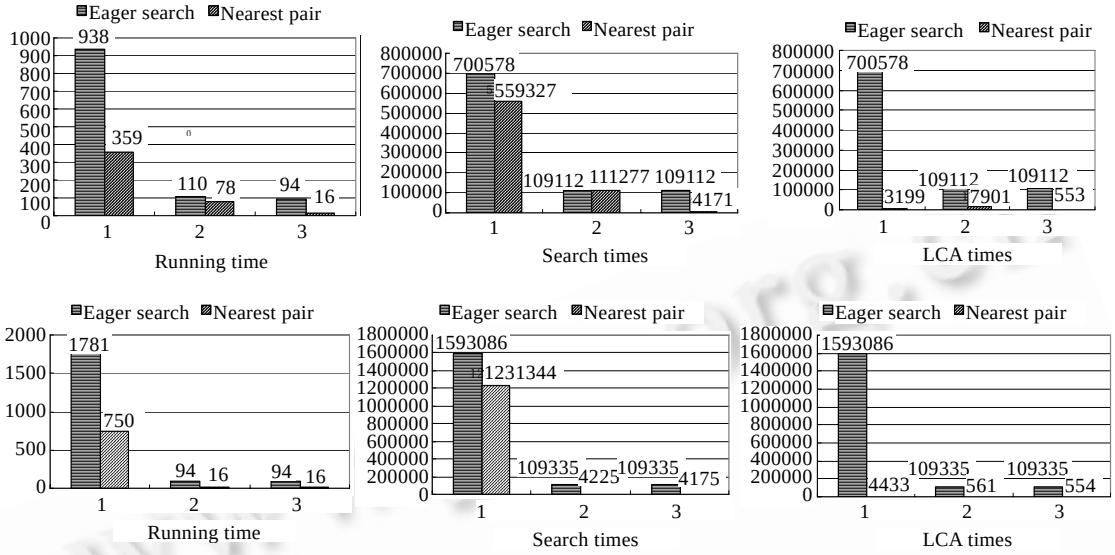

Fig.5 Experimental results

图 5 实验结果

在 3 种情况下,Nearest Pair 的性能都要优于 Eager Search.Nearest Pair 使用“最邻近点对”的概念指导查询,提高了中间结果命中 SLCA 的概率,将求解 LCA 的次数降低了一个数量级.但是在搜索次数方面,Nearest Pair 的优势并不明显,这是因为 Eager Search 只需一次搜索就可以确定其最邻近元素,而 Nearest Pair 最少需要两次搜索才能确定两个元素是彼此最邻近的.如果关键词集合中彼此最邻近的元素是稠密分布的(如关键词集合的大小差异悬殊),Nearest Pair 的优势就被大大削弱了.从总体来看,Nearest Pair 在绝大多数情况下都具有更加出色的性能.

4 总结和展望

构建 XML 的搜索引擎是一个很大的课题,目前,其研究焦点仍然是查询算法的设计和改进.这是因为查询算法是 XML 搜索引擎的核心技术,也是起步技术.如果 XML 关键词检索算法能够比较完美地解决树结构特性带来的一系列问题,那么搜索引擎其他方面的研究如结构检索和结果排级等,都可以站在一个比较高的起点上.

虽然通过SLCA回答多关键词查询是目前研究的主要思路,但其缺点也是显而易见的:SLCA并不能代表所有的关键词节点,在语义上,SLCA 不具有完备性.XML 关键词检索算法仍然有许多值得研究和改进的地方.

致谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是北京大学信息科学技术学院智能科学系讨论班上的同学和老师表示感谢.

References:

[1] Agrawal S, Chaudhuri S, Das G. DBXplorer: A system for keyword-based search over relational databases. In: Agrawal R, ed. Proc. of the 18th Int'l Conf. on Data Engineering (ICDE). San Jose: IEEE Computer Society, 2002. 5−16.

[2] Aditya B, ed. BANKS: Browsing and keyword searching in relational databases. In: Papadias D, ed. Proc. of the 28th Int'l Conf. on Very Large Data Bases (VLDB). Hong Kong: Morgan Kaufmann Publishers, 2002. 1083−1086.

[3] Goldman R, Shivakumar N, Venkatasubramanian S, Garcia-Molina H. Proximity search in databases. In: Gupta A, ed. Proc. of the 24th Int'l Conf. on Very Large Data Bases (VLDB). New York: Morgan Kaufmann Publishers, 1998. 26−37.

[4] Hristidis V, Papakonstantinou Y. Discover: Keyword search in relational databases. In: Papadias D, ed. Proc. of the 28th Int'l Conf. on Very Large Data Bases (VLDB). Hong Kong: Morgan Kaufmann Publishers, 2002. 850−861.

[5] Hristidis V, Papakonstantinou Y, Balmin A. Keyword proximity search on XML graphs. In: Dayal U, ed. Proc. of the 19th Int'l Conf. on Data Engineering (ICDE). Bangalore: IEEE Computer Society, 2003. 367−378.

[6] Li Y, Yu C, Jagadish HV. Schema-Free XQuery. In: Nascimento MA, ed. Proc. of he 13th Int'l Conf. on Very Large Data Bases (VLDB). Toronto: Morgan Kaufmann Publishers, 2004. 72−83.

[7] Bhalotia G, Hulgeri A, Nakhe C, Chakrabarti S, Sudarshan S. Keyword searching and browsing in databases using BANKS. In: Agrawal R, ed. Proc. of the 18th Int'l Conf. on Data Engineering (ICDE). San Jose: IEEE Computer Society, 2002. 431−440.

[8] Kacholia V, Pandit S, Chakrabarti S, Sudarshan S, Desai R, Karambelkar H. Bidirectional expansion for keyword search on graph databases. In: Böhm K, ed. Proc. of the 31st Int'l Conf. on Very Large Data Bases (VLDB). Trondheim: ACM Press, 2005. 294−303.

[9] Guo L, Shao F, Botev C, Shanmugasundaram J. XRANK: Ranked keyword search over XML documents. In: Halevy AY, Ives ZG, Doan A, eds. Proc. of the 2003 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD). San Diego: ACM Press, 2003. 16−27.

[10] Xu Y, Papakonstantinou Y. Efficient keyword search for smallest LCAs in XML databases. In: Ozcan F, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD). Baltimore: ACM Press, 2005. 537−538.

吉聪睿(1983－),男,江苏连云港人,硕士生,主要研究领域为信息检索,数据库知识与发现.

邓志鸿(1973－),男,博士,副教授,主要研究领域为信息检索,数据库与知识发现.

唐世渭(1939－),男,教授,博士生导师,CCF 高级会员,主要研究领域为数据库系统.