文章编号：1000－8152(2005)03－0429－05

Web数据中频繁模式树的挖掘

王自强，冯博琴
（西安交通大学 计算机科学系，陕西 西安 710049）

摘要：为了高效地从半结构化WEB数据中挖掘频繁模式树，提出了把半结构化数据表示为标记、有序树，并基于最右路径扩展技术在有序树中发现所有频繁模式树的算法.其基本思想是，首先从只有一个节点的模式树开始，而新增节点只能通过添加到最右路径上来生成新的模式树，另外，还通过维护最右叶子出现次数列表来实现支持度的逐步计算.理论分析和试验结果表明该算法是可行的，并且具有计算性能线性于最大频繁模式总和的优点.

关键词：数据挖掘；Web数据；频繁模式树；有序树
中图分类号：TP311　　**文献标识码**：A

Mining frequent pattern tree in Web data

WANG Zi-qiang, FENG Bo-qin
(Department of Computer Science, Xi'an Jiaotong University, Xi'an Shaanxi 710049, China)

Abstract: To efficiently mine all frequent pattern trees from the semi-structured web data, the semi-structured data were modeled as labeled-ordered tree and an algorithm for mining all frequent pattern trees in an ordered data tree was proposed. This algorithm used rightmost path expansion technique, which started with pattern trees with only one node and nodes were added only to the rightmost path to generate new pattern trees. Furthermore, this algorithm maintained only the occurrences of the rightmost leaves to efficiently implement incremental computation of support. The theoretical analysis and experimental results show that this algorithm scales linearly in the total size of maximal tree pattern and works efficiently in practice.

Key words: data mining; Web data; frequent pattern tree; ordered tree

1 引言(Introduction)

随着互联网等信息技术的飞速发展，大量的半结构化Web数据[1,2]（像HTML, XML等）分布到Internet和Intranet上，如何高效地从这些Web数据中高效地提取有用的信息，尤其是提取有关规则和模式，日益受到人们的重视.近年来，有关学者提出了多种半结构化数据挖掘算法[3~6]，这些算法要么在发现频繁模式时存在着冗余（重复）或者遗漏信息，要么因为时间和空间复杂度太高而无法应用.本文提出了把半结构化数据表示为标记有序树，来研究在给定的半结构数据集合中发现大于一定最小支持度的树形频繁模式，并给出了具有较好时间和空间复杂度的频繁模式树挖掘算法.

2 问题定义(Problem definition)

下面用标记有序树[7]给出半结构化Web数据及及其频繁模式树的定义.对于集合 A，设 $|A|$ 表示 A 的基数（大小），设 $L = \{l_0, l_1, \cdots, l_n\}$ 表示对应于半结构化数据中属性或者用来标记文本的有限字母表.

定义 1　建立在 L 上的标记有序树，简称有序树，是一个六元组 $OT = \{V, E, B, L, M, r\}$.其中 V 是一个有限的节点集合，$E = V \times V$ 表示(parent, child)$\in E$ 满足的双亲－孩子关系.$B \subseteq V \times V$ 表示满足(elder, yonger)$\in L$（可能间接）的兄弟关系.表示把标记 $M(v)$ 赋给节点 $v \in V$ 的标记函数.

定义 2　设 OT 表示一个有序树，树的大小 $|OT|$ 定义为树中的节点数目.对于 OT 中的路径 $P = (x_0, \cdots, x_{n-1})(n \geqslant 1)$，定义路径 P 的长度为 $|P| = n$.对于任意节点 $v \in V$，v 的深度表示为 $d(v)$，其定义是从根节点 r 到 v 的路径长度.有序树 OT 的最长路径表示为从根节点到某个叶子节点的最长路径长度.当有序树 OT 为空树时，表示为 $\perp$.

例如，图1给出了在字母表 $L = \{A, B\}$ 上的有序树 D 和 T，其中圆圈中的字母表示其节点，圆圈旁

收稿日期：2003－09－26；收修改稿日期：2004－06－07.
基金项目：国家"八六三"高技术研究发展计划基金资助项目(2003AA1Z2610).

边的数字表示其标记,树中的节点标记按前序遍历顺序依次编号.

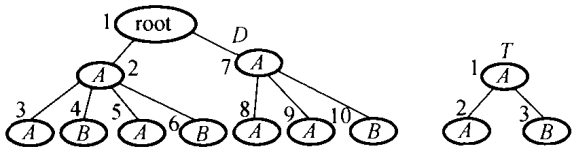

图 1 定义在字母表 $L = \{A, B\}$ 上的有序树

Fig. 1 Order tree definition on $L = \{A, B\}$

定义 3 半结构化数据集是一个三元组 $DB = (T, \mathrm{DOC}, \varphi)$,其中:$T$ 是建立在 L 上的有序树,并假设 T 的根节点用一个不属于字母表 L 的特殊符号 root 表示. $\mathrm{DOC} = \{d_1, \cdots, d_m\}$ 表示文档名;$\varphi: V_T - \{\mathrm{root}\} \to \mathrm{DOC}$ 表示文档命名函数,其定义为:设 T_i 表示其根节点是 root 第 i 个孩子的子树,则 φ 是满足 $\varphi(v) = d_i$ 的函数,其中 $v \in V_{T_i}$.

定义 4 正则范式.具有大小 $n \geqslant 1$ 的有序树 T 称为正则范式,需满足的条件为:1) T 的节点集合为 $V_T = \{1, \cdots, n\}$; 2) V_T 中的所有元素以前序遍历 T 的顺序编号.

引理 1 设 T 是一个具有 n 个节点的有序树,如果 T 是正则范式,则根节点 root = 1,并且 T 中最右边的叶子 $v_{n-1} = n$.

引理 2 对于任何有序树 T,如果 T 是正则范式,则它的双亲 $\mathrm{par}(T)$ 也是正则范式.

定义 5 模式树.具有大小 k 的子结构模式树(文本树),简称 k-模式,是定义在字母表 L 上的正则范式.对于任意 $k \geqslant 1$,所有的 k-模式用 Γ_k 表示,并且所有的模式集用 Γ 表示,即 $\Gamma = \bigcup_k \Gamma_k$.

为了形式化定义频繁模式树的挖掘算法,首先定义数据集中的匹配函数和模式出现次数,然后在模式出现次数的基础上定义频繁模式树.

定义 6 模式匹配函数.设 T 和 D 是两个有序树,分别称为模式树和数据树.如果从 T 的节点到 D 的节点之间的匹配函数 $\pi: V_T \to V_D$ 满足如下条件(说明:其中节点 $v, v_1, v_2 \in V_T$): 1)π 是一对一映射.也即是,如果 $v_1 \neq v_2$,则 $\pi(v_1) \neq \pi(v_2)$. 2)π 保持双亲-孩子关系.也即是,如果 $(v_1, v_2) \in E_T$,则 $(\pi(v_1), \pi(v_2)) \in E_D$. 3)$\pi$ 保持兄弟关系.也即是,如果 $(v_1, v_2) \in B_T$,则 $(\pi(v_1), \pi(v_2)) \in B_D$. 4)$\pi$ 保持每个节点的标记.也即是,$L_T(v) = L_D(\pi(v))$.

从定义 6 可知,如果模式树 T 出现在数据树 D 中,需满足的条件是,从 T 到 D 存在某个映射函数.下面我们在匹配函数的基础上定义模式树出现的次数.

定义 7 模式出现次数.设 k 是一个正整数,在定义 3 和定义 6 的基础上定义模式树 T 的出现次数:1) T 的全部出现次数 $|\mathrm{Toc}(T)|$,其中 $\mathrm{Toc}(T)$ 是一个 k-元组 $(\pi(1), \cdots, \pi(k)) \in (V_D)^k$. 2) T 的根节点出现次数 $|\mathrm{Roc}(T)|$,其中 $\mathrm{Roc}(T) = \pi(1) \in V_D$. 3) T 的最右出现次数 $|\mathrm{RMoc}(T)|$,其中

$$\mathrm{RMoc}(T) = \{\pi(k) \mid \pi: V_T \to V_D \text{ 是匹配函数}\}.$$

根据定义 7 易得如下结论:

引理 3 $|\mathrm{Toc}(T)| \geqslant |\mathrm{RMoc}(T)| \geqslant |\mathrm{Roc}(T)|$.

定义 8 模式树的支持度.在数据树 D 中模式树 T 的支持度表示为 T 的最右出现次数与 D 中节点总和的比值,记作 $\sup_D(T) = \dfrac{|\mathrm{RMoc}(T)|}{|D|}$.

定义 9 η-频繁模式.设 $0 < \eta \leqslant 1$ 是一个正数,模式树 T 是 η-频繁模式,需满足的条件为 $\sup_D(T) \geqslant \eta$.

定义 10 频繁模式树挖掘.给定建立在标记 L 上的数据树 D,和正数 $0 < \eta \leqslant 1$(称为最小支持度),发现所有的 η-频繁模式树 T,使得 $\sup_D(T) \geqslant \eta$.

3 频繁模式树挖掘算法(Frequent pattern tree mining algorithm)

下面具体讨论频繁模式树挖掘算法,其基本思想是:利用树的逐步最右扩展技术来枚举产生所有正则无重复有序树,从而发现所有频繁模式树.在描述算法之前,首先论述算法中关于树枚举的关键技术:最右扩展方法和关于最右出现次数的更新计算.

3.1 最右扩展方法(Rightmost expansion method)

为了高效地实现搜索模式树,使搜索算法无重复地枚举所有的模式树.文中采用了最右扩展技术,为了表述方便,把有序树中最右边的叶子(Rightmost Leaf)记做 $\mathrm{RML}(T)$.

有序树最右扩展的基本思想如图 2 所示.当构建模式树时,首先从有单个节点构成的树开始,对于大小 $k \geqslant 2$ 的树,枚举算法通过添加一个节点到大小 $k-1$ 为的树的最右分支来扩展,从而产生一个更

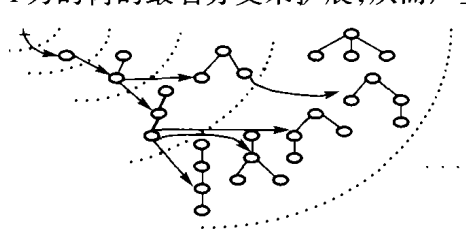

图 2 有序树的最右扩展方法

Fig. 2 Rightmost expansion of order tree

大的具有大小为 k 的有序树.

定义 11　最右分支.对于有序树 T,T 的最右分支是从根节点到 T 的最右叶子的唯一路径.

定义 12　(m,f)-扩展树.设 T 是建立在 L 上的大小为的模式树,$m(0 \leqslant m \leqslant d(T))$ 是一个正整数,x 是 T 的最右边叶子 $\mathrm{RML}(T) = k-1$,并且 $y = \mathrm{par}_T^m(x)$ 是 x 的第 m 个双亲,于是 T 的 (m,f)-扩展是通过添加一个新的具有标记 f 的节点 e 到节点 y,使得新添加的节点 e 成为 y 的最右边孩子而获得的.如图 3 所示.

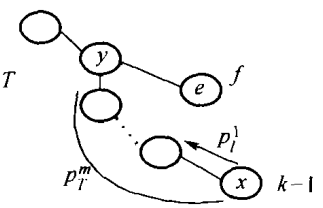

图 3　(m,f)-扩展树

Fig. 3　(m,f) expansion tree

从定义 12 可知,最右扩展实际上是 (m,f)-扩展树的特殊情况,也即是,当路径 m 和和节点 f 确定时的情况.

引理 4　对于正整数 $k \geqslant 2$,如果有序树 T 是 $(k-1)$-模式树,则 T 的最右扩展是 k-模式树.

证　设 T' 是 T 的最右扩展,于是可得 T' 的大小为 $|T'| = k$,由于新添加的节点 $v_k = k$ 是 T 的最右边叶子,因而当以前序遍历 T' 时,v_k 便是最后一个节点.于是可得 T' 是 k-模式树.

引理 5　对于正整数 $k \geqslant 2$,如果有序树 T 是一个 k-模式树,则必存在唯一的 $(k-1)$-模式树 R,使得 T 是 R 的最右扩展.

证　假设 T 是通过在某个 $(k-1)$-模式树 R 上添加一个节点 k 作为 R 的最右边叶子而获得的,于是可得,从 T 中移去节点 k 是创建 T 的前辈的唯一路径,由于 k 作为 T 的最右边叶子的方法是唯一的,因此 T 的前辈 R 也是唯一.

3.2　最右出现次数的更新计算(Rightmost occurrences update calculation)

为了有效地计算每个枚举模式树的支持度,采用的方法是,只计算模式树的最右出现次数 $RMoc(T)$,然后通过最右扩展的方法来逐步更新模式树的最右出现次数 $RMoc(T)$.为了有效地存储从每个模式树 T 到数据树 D 的匹配 $\pi: V_T \to V_D$,这里采取的方法是,仅存储最右边叶子 k 的映射信息 $\pi(k)$,而不是存储匹配函数 $\pi: V_T \to V_D$ 的全部信息 $(\pi(1),\cdots,\pi(k))$.这样就大大地提高了计算的效率,下面的引理就说明了如何逐步地从 T 计算其最右扩展 T' 的出现次数.

引理 6　设 $(k-1)$ 模式树 T 出现在数据树 D 中,并且 $\pi: V_T \to V_D$ 是 T 到 D 的匹配函数,设 T' 是 T 的 (m,f) 扩展,则函数 $\phi: V_{T'} \to V_D$ 是从 T' 到 D 的匹配函数,需满足的条件为:1)ϕ 是 π 的扩展,也即是,对于任何正整数 $i = 1,\cdots,k-1$,有 $\phi(i) = \pi(i)$ 成立.2)$\phi(k)$ 是双亲 $par_D(i)$ 的一个孩子,其中 $i = par_D^p(\pi(k-1))$,并且 $\phi(k)$ 是 i 的右兄弟.3)$L_D(\phi(k)) = f$.

定义 13　设 x 是数据树 D 中的任一节点,对于任何正整数 $p \geqslant 0$,节点 x 的第 p 个顶部记做 $Top^{(p)}(x)$,其定义为:如果 $p = 0$,则是 $Top^{(p)}(x)$ 是 x 的孩子;如果 $p > 0$,则 $Top^{(p)}(x)$ 是 x 的第 $(p-1)$ 个双亲的下一个兄弟.

定义 14　顶部二元关系.对于任何正整数 $m \geqslant 0$ 和 $l \in L$,二元关系 $TOP^{(m,l)} = \{(x,y) \mid x,y \in V_D, (Top^{(m)}(x),y) \in B_D, L_D(y) = l\} \subseteq V_D \times V_D$.

由引理 6 和定义 13,14 可得如下引理:

引理 7　设 $m \geqslant 0$ 和 $l \in L$,并且 T 是 S 的 (m,l) 扩展,则 $RMoc(T) = TOP^{(m,l)}(RMoc(S))$.

3.3　算法描述(Algorithm description)

基于以上关键技术论述的基础上,下面给出本文提出的频繁模式树挖掘算法,然后给出其有关算法分析及实例分析.

算法　挖掘频繁模式树

Algorithm Find_Frequent_Tree

输入:标记集 L,建立在 L 上的数据树 D,最小支持度 $0 < \eta \leqslant 1$.

输出:D 中的所有 η-模式树 T.

步 1　遍历数据树 D,计算所有的 1-模式频繁树 T_1,计算它们的最右出现次数列表 $RMoc$;

步 2　$k = 2$;

步 3　while $\mathrm{RMoc}_{k-1} \neq \Phi$　do

步 4　$(T_k, \mathrm{RMoc}_k) = \mathrm{Extend\_Tree}(T_{k-1}, \mathrm{RMoc}_{k-1})$;

步 5　对于每个模式 $P \in T_k$　do

步 5.1　由 $\mathrm{RMoc}_k(P)$ 计算 $\sup_D(P)$;

步 5.2　如果 $\sup_D(P) \geqslant \eta$,则 $T_k = T_k \cup \{P\}$;

步 6　k ++;

步 7　end while.

步 8　return $T = T_1 \cup \cdots \cup T_{k-1}$.

函数　计算模式树的扩展

Function Extend_Tree($T_{\mathrm{old}}, \mathrm{RMoc}_{\mathrm{old}}$)

输入:模式树集 T_{old} 及其最右出现次数列表

$\mathrm{RMoc}_{\mathrm{old}}$.

输出:模式树集 T_{old} 的最右扩展树 T_{new} 及其最右出现次数列表 $\mathrm{RMoc}_{\mathrm{new}}$.

步 1 $T_{\mathrm{new}} = \Phi, \mathrm{RMoc}_{\mathrm{new}} = \Phi$;

步 2 for 每个模式树 $P \in T_{\mathrm{old}}$ do

步 3 对于深度 m 小于 T 的最右叶子深度 do

步 3.1 对于相应节点标记 $l \in L$ do

步 3.1.1 计算 P 的 (m,l) 扩展 P';

步 3.1.2 $\mathrm{RMoc}_{\mathrm{new}}(P') =$ Comp_ New_ RMOC($\mathrm{RMoc}_{\mathrm{old}}(P), m, l$);

步 3.1.3 $T_{\mathrm{new}} = T_{\mathrm{new}} \cup \{P'\}$;

步 4 end for;

步 5 return T_{new} 和 $\mathrm{RMoc}_{\mathrm{new}}$.

函数 模式树最右出现次数的计算

Function Comp_ New_ RMOC ($\mathrm{RMoc}_{\mathrm{old}}, m, l$)

输入:旧的最右出现次数列表 $\mathrm{RMoc}_{\mathrm{old}}$,深度 $m \geqslant 0$,节点标记 l.

输出:更新后的最右出现次数列表 $\mathrm{RMoc}_{\mathrm{new}}$

步 1 $\mathrm{RMoc}_{\mathrm{new}} = \Phi$; last = NIL;

步 2 for 每个节点 $v \in \mathrm{RMoc}_{\mathrm{old}}$ do

步 2.1 如果 $m = 0$,则 v' 是 v 的第一个孩子;

步 2.2 if $m \geqslant 1$ then do

步 2.2.1 如果 last = $\mathrm{par}_D^m(v)$, 则跳转到步 2;/ * 重复子树检测 * /

步 2.2.2 如果 last $\neq \mathrm{par}_D^m(v)$, 则 $v' = \mathrm{par}_D^{m-1}(v)$; last = next($v'$);

步 2.3 while $v' \neq NIL$ do

步 2.3.1 如果 $L_D(v') = l$, 则 $\mathrm{RMoc}_{\mathrm{new}} = \mathrm{RMoc}_{\mathrm{new}} \cup \{v'\}$;

步 2.3.2 v' = next(v'); / * 下一个兄弟 * /

步 3 end while.

步 4 end for;

步 5 return $\mathrm{RMoc}_{\mathrm{new}}$.

3.4 算法分析(Algorithm analysis)

下面的引理给出了计算频繁模式树的复杂性分析.首先分析计算模式最右出现次数 Comp_ New_ RMOC 的时间复杂性.由引理 6 可得如下引理:

引理 8 函数 Comp_ New_ RMOC 在给定输入 ($RMoc(T), m, l$) 的条件下,计算 T 的 (m,l) 扩展 S 的最右出现次数列表 $\mathrm{RMoc}(S)$ 的时间复杂度为 $O(kbn)$. 其中 k 表示 T 的大小,b 表示 T 的最大分支数,$n = |\mathrm{RMoc}(T)|$.

引理 9 函数 Extend_ Tree 计算两个数据集 T_{new} 和 $\mathrm{RMoc}_{\mathrm{new}}$ 所花费的时间复杂度为 $O(k^2 lbN)$. 其中 l 表示输入标记 L 的大小,b 是数据树的最大分支数,N 表示输入最右出现次数列表 $\mathrm{RMoc}_{\mathrm{old}}$的大小.

于是由引理 8 和引理 9 可得整个频繁模式树挖掘算法的时间复杂度为如下引理.

引理 10 设 D 是输入数据集,L 为输入标记集,最小支持度为 $0 < \eta \leqslant 1$,于是算法 Find_ Frequent_ Tree 挖掘所有 η - 频繁模式树所花费的时间为 $O(M + k^2 blN)$. 其中 M 表示数据集 D 的大小,N 表示所有频繁模式树的最右出现次数列表 RMoc 的大小总和.

下面的定理说明了算法 Find_ Frequent_ Tree 的计算性能线性于最大频繁模式总和.

定理 1 算法 Find_ Frequent_ Tree 在计算过程中最多能够枚举 $O(klM)$ 个模式,其中 M 表示最大 η 频繁模式的大小总和.

3.5 相关算法比较(Relative algorithm comparison)

为了解决冗余(重复)树的出现,作者采用的是最右扩展技术,其基本思想是:当构建模式树时,首先从由单个节点构成的模式树开始,然后新增加的节点仅仅添加到最右路径上来产生新的模式树.由引理 4 和引理 5 可知,通过递归地调用最右扩展技术构建的模式树是唯一的,从而避免了频繁模式树的冗余(重复)构造,另外,图 2 表示的有序树最右扩展技术形象地说明了模式树的有序构建过程,计算模式树最右出现次数的函数 Comp_ New_ RMOC( ) 具体给出了避免重复子树的算法描述.而作者在构建模式树的过程中采用树枚举方法,保证了生成频繁模式树的无遗漏性.

另外,在时间复杂度和空间度方面,由定理 1 可知,本文提出算法的时间复杂度线性于最大频繁模式树的总和,在空间存储方面,由 3.2 节可知,这里存储的仅仅是模式树中最右边叶子的映射信息,而不是模式树到数据树之间的全部映射信息,因而大大地节省了存储空间.而文献[3]和[4]中的算法时间复杂度和存储空间需求均为指数复杂度.虽然文献[2,5,6]提出的算法具有较小的时间复杂度,但是其不足之处为,在生成频繁模式树的过程中生成了大量冗余(重复)树,从而增加了计算的复杂性.综上所述,本文提出的算法具有较好的时间和空间复杂度.

4 实例分析(Example analysis)

以图 1 中的数据树 D 来说明本文算法的工作过程.假设最小支持度 $\sigma = 0.19$. 图 4 说明了由算法 Find_ Frequent_ Tree 生成频繁模式树的过程,限于篇

幅,我们略去了在树扩展过程中最右出现次数的更新过程表.

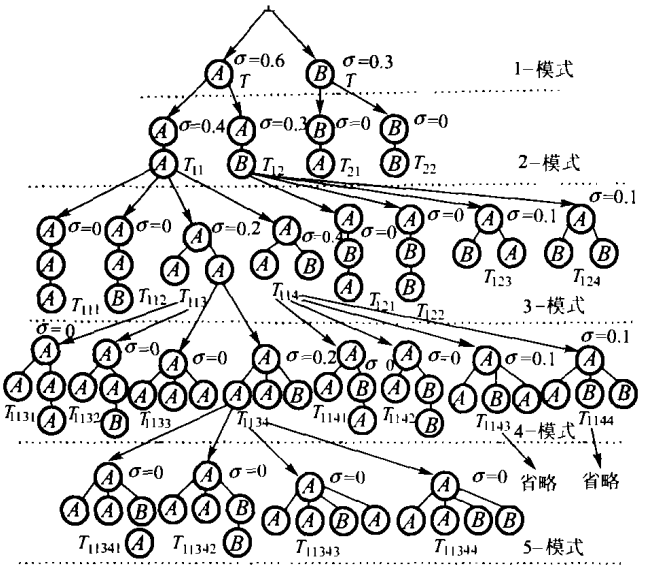

图 4　模式树的扩展

Fig. 4　Pattern tree expansion

首先算法 Find_Frequent_Tree 通过遍历数据树 D 获得频繁 1-模式的集合 Γ_1 和它们的相应最右出现次数 RMoc(Γ_1),然后调用过程 Extend_Tree (Γ_1, RMoc(Γ_1)) 来产生候选树 Γ_2 和它们的相应最右出现次数列表 RMoc(Γ_2). 在图 4 中,我们看到 $\Gamma_2 = \{T_{11}, T_{12}, T_{21}, T_{22}\}$,并且它们是通过在其前辈 T_1 和 T_2 中添加新的节点 A 或 B 而获得的. 同时由图 4 可知, Γ_2 中的两个模式 T_{11} 和 T_{12} 的支持度大于 0.19,因而 T_{11} 和 T_{12} 是所要的频繁模式树,而模式 T_{21} 和 T_{22} 的支持度等于 0,而小于 0.19,故从模式树中去掉,通过重复这一过程,算法中止在阶段 5 而获得全部支持度大于 0.19 的频繁模式树.

$$T = P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5 = \{T_1, T_2\} \cup \{T_{11}, T_{12}\} \cup \{T_{113}, T_{114}\} \cup \{T_{1134} \cup \Phi = \{T_1, T_2, T_{11}, T_{12}, T_{113}, T_{114}, T_{1134}\}.$$

5　结论(Conclusions)

为了高效地从中发现频繁模式,文中采用了把半结构化数据表示为标记、有序树,提出了基于最右扩展路径技术的频繁模式树算法,并对其进行了算法分析,说明本文提出的算法具有较好的时间和空间复杂度,最后用实例说明了算法的可行性.

参考文献(References):

[1] ABITEBOUL S, BUNEMAN P, SUCIU D. *Data on The Web: From Relations to Semi-Structured Data and XML* [M]. San Diego: Morgan Kaufmann Press, 1999: 58 - 86.

[2] ASAI T, ABE K, KAWASO E, et al. Efficient substructure discovery from large semi-structured data [C]// GROSSMAN R L, HAN J W, KUMAR V, et al. *Proc of the Second SIAM Int Conf on Data Mining* (*SDM'02*). Baltimore: SIAM Press, 2002: 142 - 160.

[3] WANG J T, CHIRN G W, MARR T G, et al. Combinatorial pattern discovery for scientific data: some preliminary results [C]// SNODGRASS R T, WINSLETT M. *Proc of Association for Computing Machinery Special Interest Group on Management of Data* (*ACM SIGMOD'94*). Pittsburgh PA: ACM Press, 1994: 115 - 125.

[4] ARIMURA H, WATAKI R, FUJINO R. An efficient algorithm for text data mining with optimal string patterns[C]// RICHTER M M, SMITH C H, WIEHAGEN R, et al. *Proc of the 9th Int Conf on Algorithmic Learning Theory* (*ALT' 98*), *Lecture Notes in Computer Science*. New York: Springer, 1998: 247 - 261.

[5] WANG K, LIU H. Schema discovery for semi-structured data [C]// ZAKI M J, HO C T. *Proc of Knowledge Discovery and Data Mining' 99* (*KDD' 99*), *Lecture Notes in Computer Science*. New York: Springer, 1999: 271 - 283.

[6] ZAKI M. Efficiently mining frequent trees in a Forest [C]// ZAIANE O R, SIMOFF S J, DJERABA C. *Proc of Knowledge Discovery and Data Mining' 02* (*KDD' 02*), *Lecture Notes in Computer Science*. New York: Springer, 2002: 114 - 124.

[7] AHO A V, HOPCROFT J E, ULLMAN J D. *Data Structures and Algorithms* [M]. Boston: Addison-Wesley Press, 1983: 18 - 83.

作者简介:

王自强　(1973—),男,西安交通大学电信学院博士研究生,主要研究方向为数据挖掘、智能网络,E-mail: wzqagent@xinhuanet.com;

冯博琴　(1942—),男,西安交通大学教授,博士生导师,主要研究方向为智能网络.