2F3-03

Discovery of Closed and Maximally Frequent Tag Tree Patterns from Semistructured Documents

Tetsuhiro Miyahara[*1] Takayoshi Shoudai[*2] Kenichi Takahashi[*1] Hiroaki Ueda[*1]

[*1]Faculty of Information Sciences, Hiroshima City University [*2]Department of Informatics, Kyushu University

In order to extract knowledge from semistructured documents such as HTML or XML files, the methods for finding frequent patterns or common characteristics in semistructured documents have been studied extensively. However, the number of frequent patterns grows exponentially with the size of input semistructured data. So discovery of closed frequent patterns in semistructured data has been more and more important, since closed patterns are known to be condensed representations of frequent patterns. We propose new closed frequent tree structured patterns, which are called closed frequent tag tree patterns, and consider discovery of both closed and maximally frequent tag tree patterns from semistructured documents.

1. Introduction

Due to the rapid growth of Internet usage, Web documents such as HTML or XML files have been rapidly increasing. Such Web documents have no rigid structure but have heterogeneous features, and are called semistructured data. In order to extract meaningful and hidden knowledge from such semistructured documents, methods for discovering frequent patterns or common characteristics in semistructured documents have been studied extensively.

However, the number of frequent patterns grows exponentially with the size of input semistructured data. So discovery of closed frequent patterns in semistructured data has been more and more important, since closed patterns are known to be condensed representations of frequent patterns. In this paper, we propose new closed frequent tree structured patterns, which are called closed frequent tag tree patterns, and consider discovery of both closed and maximally frequent tag tree patterns from semistructured documents.

We use rooted trees as representations of semistructured data such as HTML or XML files, according to Object Exchange Model [1]. In this paper, "ordered" means "with ordered children" and "unordered" means "with unordered children". We consider both ordered trees and unordered trees in order to deal with various semistructured data.

To formulate a schema on such tree structured data we have proposed a **tag tree pattern** [5]. A tag tree pattern is an edge labeled tree which has ordered or unordered children and structured variables. An edge label is a tag, a keyword in Web documents, or a wildcard for any string. A variable can match an arbitrary subtree, which represents a field of a semistructured document. In the examples of tree structured data and tag tree patterns in Fig.1, the variables with labels "x_1", "x_2" and "x_3"of the tag tree pattern t_2 match the subtrees g_1,g_2 and g_3, respectively. Thus the tag tree pattern t_2 matches the tree T_1.

Graph or tree-based data mining and discovery of closed frequent patterns have been extensively studied [2, 3, 6, 7, 8]. But almost all previous works on closed patterns deal with itemsets, subgraphs, substructures, subtrees, or subsequences. In order to apply our method to knowledge discovery from heterogeneous semistructured Web documents, our target of discovery is a **closed frequent tag tree patterns**, which is a condensed representation of a common characteristic in semistructured documents. A tag tree pattern is different from other representations of tree structured patterns in that a tag tree pattern has structured variables which can match arbitrary trees and a tag tree pattern represents not a substructure but a whole tree structure.

Let $\mathcal{D}$ be a set of semistructured data (or trees) and σ a threshold, which is called a *minimum support*. The *frequency* of a tag tree pattern π is the ratio of the data which π matches to all data in $\mathcal{D}$. A tag tree pattern π is *frequent* if the frequency of π is greater than or equal to σ. A *refinement* of π is a tag tree pattern which is a specialized pattern of π and obtained by adding a vertex to π or by attaching a new edge label to a variable or wildcard of π. A tag tree pattern π is *closed* if for every refinement π' of π, the frequency of π' is less than the frequency of π. A frequent tag tree pattern π is *maximally frequent* if for every refinement π' of π, the frequency of π' is less than σ.

In this paper, we consider the following data mining problems. **CFOTTP** (resp. **CFUTTP**) is a problem to generate all Closed Frequent Ordered (resp. Unordered) Tag Tree Patterns with frequency above a user-specified threshold from a given set of ordered (resp. unordered) semistructured data. Consider the examples in Fig. 1. For a set of semistructured data $\{T_1, T_2, T_3\}$, the tag tree pattern t_2 is a closed $\frac{2}{3}$-frequent ordered tag tree pattern. In fact, t_2 explains T_1 and T_3, but t_2 does not explain T_2. The tag tree pattern t_1 also explains T_1 and T_3. But t_1 explains any tree with two or more vertices and t_1 is overgeneralized and meaningless.

In our previous work [5], we proposed methods for generating all maximally frequent *ordered* or *unordered* tag tree patterns. Chi et al. [3] gave an algorithm for enumerating closed and maximally frequent subtrees in a set of unordered trees. Their pattern and data trees are different from our pattern and data trees in this work.

2. Preliminaries

Definition 1 (Ordered term trees and unordered term trees) Let $T = (V_T, E_T)$ be a rooted tree with or-

Contact: Email:miyahara@its.hiroshima-cu.ac.jp

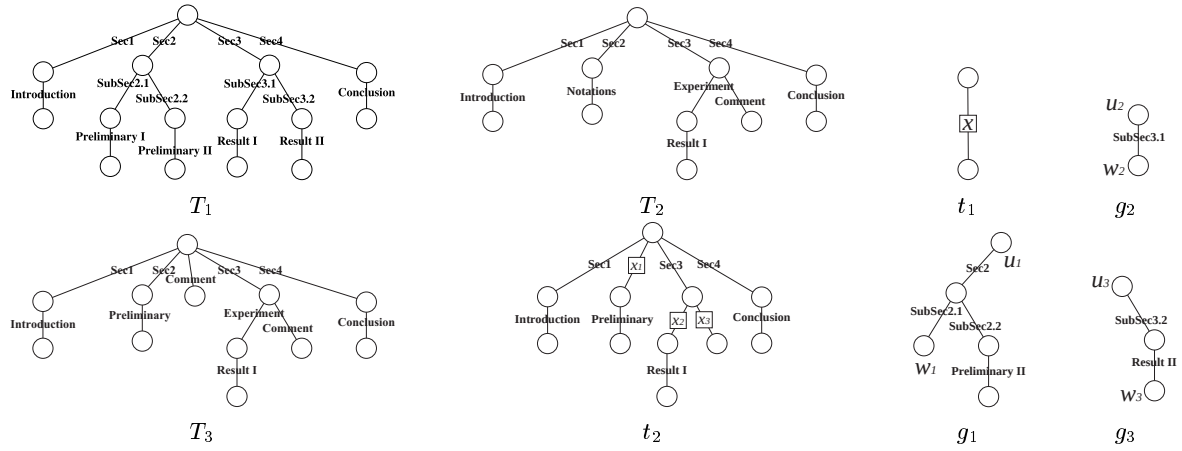


Figure 1: Tag tree patterns t_1, t_2, and trees T_1, T_2, T_3, g_1, g_2, g_2. A variable is represented by a single lined box with lines to its elements. The label inside a box is the variable label of the variable.

dered children or unordered children, which has a set V_T of vertices and a set E_T of edges. We call a rooted tree with ordered (resp. unordered) children an **ordered tree** (resp. an **unordered tree**). Let E_g and H_g be a partition of E_T, i.e., $E_g \cup H_g = E_T$ and $E_g \cap H_g = \emptyset$. And let $V_g = V_T$. A triplet $g = (V_g, E_g, H_g)$ is called an **ordered term tree** if T is an ordered tree, and called an **unordered term tree** if T is an unordered tree. We call an element in V_g, E_g and H_g a *vertex*, an *edge* and a *variable*, respectively.

Below we say a **term tree** or a **tag tree pattern** if we do not have to distinguish between "ordered" and "unordered" ones. We assume that all variable labels in a term tree are different. Λ and X denote a set of edge labels and a set of variable labels, respectively, where $\Lambda \cap X = \phi$. We use a notation $[v, v']$ to represent a variable $\{v, v'\} \in H_g$ such that v is the parent of v'. Then we call v the *parent port* of $[v, v']$ and v' the *child port* of $[v, v']$.

For an ordered term tree g, all children of every internal vertex u in g have a total ordering on all children of u. The ordering on the children of u of an ordered term tree g is denoted by $<_u^g$. Let $f = (V_f, E_f, H_f)$ and $g = (V_g, E_g, H_g)$ be two ordered (resp. unordered) term trees. We say that f and g are *isomorphic*, if there is a bijection φ from V_f to V_g which satisfies the following conditions (1)–(4) (resp. (1)–(3)): (1) The root of f is mapped to the root of g by φ. (2) $\{u, v\} \in E_f$ if and only if $\{\varphi(u), \varphi(v)\} \in E_g$ and the two edges have the same edge label. (3) $[u, v] \in H_f$ if and only if $[\varphi(u), \varphi(v)] \in H_g$. (4) If f and g are ordered term trees, for any internal vertex u in f which has more than one child, and for any two children u' and u'' of u, $u' <_u^f u''$ if and only if $\varphi(u') <_{\varphi(u)}^g \varphi(u'')$.

Let g be a term tree which has at least two vertices and x be a variable label in X. Let $\sigma = [u, u']$ be a list of two vertices in g where u is the root of g and u' is a leaf of g. The form $x := [g, \sigma]$ is called a *binding* for x. Let f and g be two ordered (resp. unordered) term trees. A new ordered (resp. unordered) term tree $f\{x := [g, \sigma]\}$ is obtained by applying the binding $x := [g, \sigma]$ to f in the following way. Let $e = [v, v']$ be a variable in f with the variable label x. Let g' be one copy of g and w, w' the vertices of g' corresponding to u, u' of g, respectively. For the variable $e = [v, v']$, we attach g' to f by removing the variable e from H_f and by identifying the vertices v, v' with the vertices w, w' of g', respectively. A *substitution* θ is a finite collection of bindings $\{x_1 := [g_1, \sigma_1], \cdots, x_n := [g_n, \sigma_n]\}$, where x_i's are mutually distinct variable labels in X and g_i's are term trees. The term tree $f\theta$, called the *instance* of f by θ, is obtained by applying the all bindings $x_i := [g_i, \sigma_i]$ on f simultaneously. We define the root of the resulting term tree $f\theta$ as the root of f. Further we have to give a new total ordering $<_v^{f\theta}$ on every vertex v of $f\theta$. These orderings are defined in a natural way [4].

Example. Let t_2 be a term tree described in Fig. 1. Let $\theta = \{x_1 := [g_1, [u_1, w_1]], x_2 := [g_2, [u_2, w_2]], x_3 := [g_3, [u_3, w_3]]\}$ be a substitution, where g_1, g_2 and g_3 are trees in Fig. 1. Then the instance $t_2\theta$ of the term tree t_2 by θ is the tree T_1.

Definition 2 Let Λ_{Tag} and Λ_{KW} be two languages which consist of infinitely or finitely many words where $\Lambda_{Tag} \cap \Lambda_{KW} = \emptyset$. Let $\Lambda = \Lambda_{Tag} \cup \Lambda_{KW}$. We call a word in Λ_{Tag} a **tag** and a word in Λ_{KW} a **keyword**. An **ordered (resp. unordered) tag tree pattern** is an ordered (resp. unordered) term tree such that each edge label on it is any of a tag, a keyword, and a special symbol "?". Let $\Lambda_?$ be a subset of Λ. The symbol "?" is a wildcard which matches any word in $\Lambda_?$. A tag tree pattern with no variable is called a **ground tag tree pattern**.

For an edge $\{v, v'\}$ of a tag tree pattern and an edge $\{u, u'\}$ of a tree, we say that $\{v, v'\}$ *matches* $\{u, u'\}$ if the following conditions (1)-(3) hold: (1) If the edge label of $\{v, v'\}$ is a tag, then the edge label of $\{u, u'\}$ is the same tag or another tag which is considered to be identical with the tag on $\{v, v'\}$. (2) If the edge label of $\{v, v'\}$ is a keyword, then the edge label of $\{u, u'\}$ is a keyword and the label of $\{v, v'\}$ appears as a substring in the edge label of $\{u, u'\}$. (3) If the edge label of $\{v, v'\}$ is "?", then the edge label of $\{u, u'\}$ is in $\Lambda_?$. A ground ordered (resp. unordered)

tag tree pattern $\pi = (V_\pi, E_\pi, \emptyset)$ *matches* an ordered (resp. unordered) tree $T = (V_T, E_T)$ if there exists a bijection φ from V_π to V_T which satisfies the following conditions (1)–(4) (resp. (1)–(3)): (1) The root of π is mapped to the root of T by φ. (2) $\{v, v'\} \in E_\pi$ if and only if $\{\varphi(v), \varphi(v')\} \in E_T$. (3) For all $\{v, v'\} \in E_\pi$, $\{v, v'\}$ matches $\{\varphi(v), \varphi(v')\}$. (4) If π and T are ordered, for any internal vertex u in π which has more than one child, and for any two children u' and u'' of u, $u' <_u^\pi u''$ if and only if $\varphi(u') <_{\varphi(u)}^T \varphi(u'')$. A tag tree pattern π **matches** a tree T if there exists a substitution θ such that $\pi\theta$ is a ground tag tree pattern and $\pi\theta$ matches T.

$\mathcal{OT}_\Lambda$ (resp. $\mathcal{UT}_\Lambda$) denotes the set of all ordered (resp. unordered) trees whose edge labels are in Λ. $\mathcal{OTTP}_\Lambda$ (resp. $\mathcal{UTTP}_\Lambda$) denotes the set of all ordered (resp. unordered) tag tree patterns whose tags and keywords are in Λ. For π in $\mathcal{OTTP}_\Lambda$ (resp. $\mathcal{UTTP}_\Lambda$), the *language* $L_\Lambda(\pi)$ is defined as $\{$a tree T in $\mathcal{OT}_\Lambda \mid \pi$ matches $T\}$ (resp. $\{$a tree T in $\mathcal{UT}_\Lambda \mid \pi$ matches $T\}$).

3. Closed Frequent Ordered Tag Tree Patterns

A *set of ordered semistructured data* $\mathcal{D} = \{T_1, T_2, \ldots, T_m\}$ is a subset of $\mathcal{OT}_\Lambda$. Let $\Lambda_\mathcal{D}$ be the set of all edge labels of trees in $\mathcal{D}$. The *matching count* of an ordered tag tree pattern π w.r.t. $\mathcal{D}$, denoted by $match_\mathcal{D}(\pi)$, is the number of ordered trees $T_i \in \mathcal{D}$ $(1 \le i \le m)$ such that π matches T_i.

Definition 3 The **frequency** of an ordered tag tree pattern π w.r.t. $\mathcal{D}$ is defined by $supp_\mathcal{D}(\pi) = match_\mathcal{D}(\pi)/m$. Let σ be a real number where $0 < \sigma \le 1$. An ordered tag tree pattern π is **σ-frequent** w.r.t. $\mathcal{D}$ if $supp_\mathcal{D}(\pi) \ge \sigma$.

We assume that $\Lambda_\mathcal{D} \subseteq \Lambda_? \subseteq \Lambda$ where $\Lambda = \Lambda_{Tag} \cup \Lambda_{KW}$ and $\Lambda_{Tag} \cap \Lambda_{KW} = \emptyset$. Let Tag and KW be finite subsets of Λ_{Tag} and Λ_{KW}, respectively, and $\Lambda' = Tag \cup KW \cup \{?\}$. We denote by $\mathcal{OTTP}_\Lambda(\Lambda')$ the set of all ordered tag tree patterns $\pi \in \mathcal{OTTP}_\Lambda$ such that all edge labels of π are in Λ'. For an edge e of a tag tree pattern π and an edge label $\lambda \in \Lambda$, when e is labeled with λ, e is said to be a *λ-labeled edge*, and in particular, when e is labeled with "?", e is said to be a *wildcard edge*.

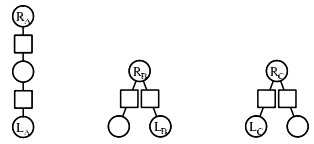
Definition 4 An ordered tag tree pattern π in $\mathcal{OTTP}_\Lambda(\Lambda')$ is **maximally σ-frequent** w.r.t. $\mathcal{D}$ [5] if (1) π is σ-frequent w.r.t. $\mathcal{D}$, and (2) for any $\pi' \in \mathcal{OTTP}_\Lambda(\Lambda')$, if $L_\Lambda(\pi') \subsetneq L_\Lambda(\pi)$ then π' is not σ-frequent. In particular, we say that an ordered tag tree pattern π in $\mathcal{OTTP}_\Lambda(\Lambda')$ is **maximal** w.r.t. $\mathcal{D}$ if π is maximally 1-frequent w.r.t. $\mathcal{D}$.

For $\mathcal{D}' \subseteq \mathcal{D}$ and $\pi \in \mathcal{OTTP}_\Lambda(\Lambda')$, we define the following two functions $itm_{\Lambda'}$: $2^\mathcal{D} \to 2^{\mathcal{OTTP}_\Lambda(\Lambda')}$ and $tid_\mathcal{D}$: $\mathcal{OTTP}_\Lambda(\Lambda') \to 2^\mathcal{D}$.

$$\begin{aligned} itm_{\Lambda'}(\mathcal{D}') &= \{\pi \in \mathcal{OTTP}_\Lambda(\Lambda') \mid \\ &\qquad \pi \text{ is maximal w.r.t. } \mathcal{D}'\}, \\ tid_\mathcal{D}(\pi) &= \{T \in \mathcal{D} \mid \pi \text{ matches } T\}. \end{aligned}$$

Definition 5 An ordered tag tree pattern $\pi \in \mathcal{OTTP}_\Lambda(\Lambda')$ is **closed** w.r.t. $\mathcal{D}$ if $\pi \in itm_{\Lambda'}(tid_\mathcal{D}(\pi))$.

Next we define the following five refinement operators for $\mathcal{OTTP}_\Lambda$. A refinement operator is a substitution which contains only one binding of a tree of size at most 3. Let $\theta_X(x) := \{x := [T_X, [R_X, L_X]]\}$, where $X \in \{A, B, C\}$ and T_A, T_B, and T_C are the following trees.



T_A T_B T_C

Variable-extension refinement operators for a variable label x:

(1) $\theta_A(x)$, (2) $\theta_B(x)$, (3) $\theta_C(x)$.

Wildcard-replacing refinement operator for a variable h:

(4) Replace the variable h with a wildcard edge.

Edge-labeling refinement operator for a wildcard edge e and $\lambda \in \Lambda'$:

(5) Replace the wildcard edge e with a λ-labeled edge.

For any $\pi \in \mathcal{OTTP}_\Lambda(\Lambda')$, the set $R_{\Lambda'}(\pi)$ is defined as the set of all ordered tag tree patterns which are obtained from π by applying one of the above 5 refinement operators only once to either a variable or a wildcard edge of π. An ordered tag tree pattern in $R_{\Lambda'}(\pi)$ is called a *one-step refinement* of π. For $i = 1, 2, \ldots$, let $R_{\Lambda'}^{(0)}(\pi) = \{\pi\}$ and $R_{\Lambda'}^{(i)}(\pi) = \bigcup_{\pi' \in R_{\Lambda'}^{(i-1)}(\pi)} R_{\Lambda'}(\pi')$. Then we define the reflexive and transitive closure of $R_{\Lambda'}(\pi)$ as $R_{\Lambda'}^*(\pi) = \bigcup_{i=0}^\infty R_{\Lambda'}^{(i)}(\pi)$.

Proposition 1 *For $\pi \in \mathcal{OTTP}_\Lambda(\Lambda')$, if $\pi' \in R_{\Lambda'}(\pi)$ then $tid_\mathcal{D}(\pi') \subseteq tid_\mathcal{D}(\pi)$.*

Lemma 1 *We assume that $Tag \cup KW \subsetneq \Lambda_? \subsetneq \Lambda$. For $\pi', \pi \in \mathcal{OTTP}_\Lambda(\Lambda')$, $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$ if and only if $\pi' \in R_{\Lambda'}^*(\pi)$.*

Lemma 2 *We assume that $Tag \cup KW \subsetneq \Lambda_? \subsetneq \Lambda$. For any $\pi \in \mathcal{OTTP}_\Lambda(\Lambda')$, π is closed w.r.t. $\mathcal{D}$ if and only if there does not exist $\pi' \in \mathcal{OTTP}_\Lambda(\Lambda')$ such that $\pi' \in R_{\Lambda'}(\pi)$ and $supp_\mathcal{D}(\pi') = supp_\mathcal{D}(\pi)$.*

Next we define formally the problem of generating all closed ordered tag tree patterns.

All Closed Frequent Ordered Tag Tree Patterns (CFOTTP)

Input: A set of ordered semistructured data $\mathcal{D}$, a threshold $0 < \sigma \le 1$, and a finite set of tags Tag and a finite set of keywords KW.

Assumption: $Tag \cup KW \subsetneq \Lambda_? \subsetneq \Lambda$.

Problem: Generate all closed σ-frequent ordered tag tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}_\Lambda(Tag \cup KW \cup \{?\})$.

By using Lemma 2 and the method for generating all maximally frequent ordered tag tree patterns [5], we can give a method for generating both closed and maximally frequent ordered tag tree patterns.

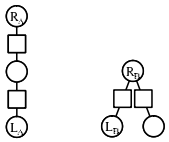
4. Closed Frequent Unordered Tag Tree Patterns

In Section 3., we discussed closed *ordered* tag tree patterns. In a similar way, we can define closed *unordered* tag tree patterns.

A *set of unordered semistructured data* $\mathcal{D} = \{T_1, T_2, \ldots, T_m\}$ is a subset of $\mathcal{UT}_\Lambda$. Let $\Lambda_\mathcal{D}$ be the set of all edge labels of trees in $\mathcal{D}$. We assume that $\Lambda_\mathcal{D} \subsetneqq \Lambda_? \subseteq \Lambda$ where $\Lambda = \Lambda_{Tag} \cup \Lambda_{KW}$ and $\Lambda_{Tag} \cap \Lambda_{KW} = \emptyset$. Let Tag and KW be finite subsets of Λ_{Tag} and Λ_{KW}, respectively, and $\Lambda' = Tag \cup KW \cup \{?\}$. We denote by $\mathcal{UTTP}_\Lambda(\Lambda')$ the set of all unordered tag tree patterns $\pi \in \mathcal{UTTP}_\Lambda$ such that all edge labels of π are in Λ'. In the same way as in Section 3., we can define $itm_{\Lambda'}$ and $tid_\mathcal{D}$.

Definition 6 An unordered tag tree pattern $\pi \in \mathcal{UTTP}_\Lambda(\Lambda')$ is **closed** w.r.t. $\mathcal{D}$ if $\pi \in itm_{\Lambda'}(tid_\mathcal{D}(\pi))$.

We define the following four refinement operators for $\mathcal{UTTP}_\Lambda$. Let $\theta_X(x) := \{x := [T_X, [R_X, L_X]]\}$, where $X \in \{A, B\}$ and T_A and T_B are the following trees.



T_A T_B

Variable-extension refinement operators for a variable label x:

(1) $\theta_A(x)$, (2) $\theta_B(x)$.

Wildcard-replacing refinement operator for a variable h:

(3) Replace the variable h with a wildcard edge.

Edge-labeling refinement operator for a wildcard edge e *and* $\lambda \in \Lambda'$:

(4) Replace the wildcard edge e with a λ-labeled edge.

Lemma 3 *We assume that* $Tag \cup KW \subsetneqq \Lambda_? \subsetneqq \Lambda$. *For* $\pi', \pi \in \mathcal{UTTP}_\Lambda(\Lambda')$, $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$ *if and only if* $\pi' \in R^*_{\Lambda'}(\pi)$.

Lemma 4 *We assume that* $Tag \cup KW \subsetneqq \Lambda_? \subsetneqq \Lambda$. *For any* $\pi \in \mathcal{UTTP}_\Lambda(\Lambda')$, π *is closed w.r.t.* $\mathcal{D}$ *if and only if there does not exist* $\pi' \in \mathcal{UTTP}_\Lambda(\Lambda')$ *such that* $\pi' \in R_{\Lambda'}(\pi)$ *and* $supp_\mathcal{D}(\pi') = supp_\mathcal{D}(\pi)$.

Next we define formally the problem of finding all closed unordered tag tree patterns.

All Closed Frequent Unordered Tag Tree Patterns (CFUTTP)

Input: A set of unordered semistructured data $\mathcal{D}$, a threshold $0 < \sigma \leq 1$, and a finite set of tags Tag and a finite set of keywords KW. b

Assumption: $Tag \cup KW \subsetneqq \Lambda_? \subsetneqq \Lambda$.

Problem: Generate all closed σ-frequent unordered tag tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{UTTP}_\Lambda(Tag \cup KW \cup \{?\})$.

By using Lemma 4 and the method for generating all maximally frequent unordered tag tree patterns [5], we can give a method for generating both closed and maximally frequent unordered tag tree patterns.

5. Experimental Results

We have implemented the method in Section 3., which generates all closed σ-frequent ordered tag tree patterns and all maximally σ-frequent ordered tag tree patterns from semistructured data. The implementation is by GCL2.2 and on a Sun workstation Blade1000 with clock 900MHz. We report some experimental results. In these experiments, an input tree represents a subtree of a parsed tree of an HTML file. The tree structure and HTML tags in a parsed tree are preserved in the corresponding input tree. Attributes and their values are ignored. No equality relation on tags is assumed. All string data in a parsed tree are converted to the same dummy keyword, in order to pay attention to structures of tags in a parsed tree. In these experiments, the sample HTML file is a result of a search engine of a web site with a local search function (http://www.ael.org).

These experiments show that the number of frequent tag tree patterns grows exponentially with the size of input semistructured data. In the case of tag tree patterns, the constraint of closed frequentness is strong. So the number of closed frequent tag tree patterns is slightly larger than that of maximally frequent tag tree patterns. Then, we can conclude that closed tag tree patterns are useful and condensed representations of frequent tree structured patterns in semistructured documents.

6. Conclusion

In this paper, we have proposed new closed frequent tree structured patterns, which are called closed frequent tag tree patterns, and have considered discovery of both closed and maximally frequent tag tree patterns from semistructured documents.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.

[2] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. *Proc. 2nd SIAM Int. Conf. Data Mining (SDM-2002)*, pages 158–174, 2002.

[3] Y. Chi, Y. Yang, Y. Xia, and R. Muntz. CMTreeMiner: Mining both closed and maximal frequent subtrees. *Proc. PAKDD-2004, Springer-Verlag, LNAI 3056*, pages 63–73, 2004.

[4] T. Miyahara, Y. Suzuki, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Discovery of frequent tag tree patterns in semistructured web documents. *Proc. PAKDD-2002, Springer-Verlag, LNAI 2336*, pages 341–355, 2002.

[5] T. Miyahara, Y. Suzuki, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Discovery of maximally frequent tag tree patterns with contractible variables from semistructured documents. *Proc. PAKDD-2004, Springer-Verlag, LNAI 3056*, pages 133–134, 2004.

[6] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. *Proc. ICDT-1999*, pages 398–416, 1999.

[7] T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. *Proc. DS-2004, Springer-Verlag, LNAI 3245*, pages 16–31, 2004.

[8] T. Washio and H. Motoda. State of the art of graph-based data mining. *SIGKDD Explorations*, 5:59–68, 2003.