Discovery of Frequent Tag Tree Patterns in Semistructured Web Documents

Tetsuhiro Miyahara[1], Yusuke Suzuki[2], Takayoshi Shoudai[2],
Tomoyuki Uchida[1], Kenichi Takahashi[1], and Hiroaki Ueda[1]

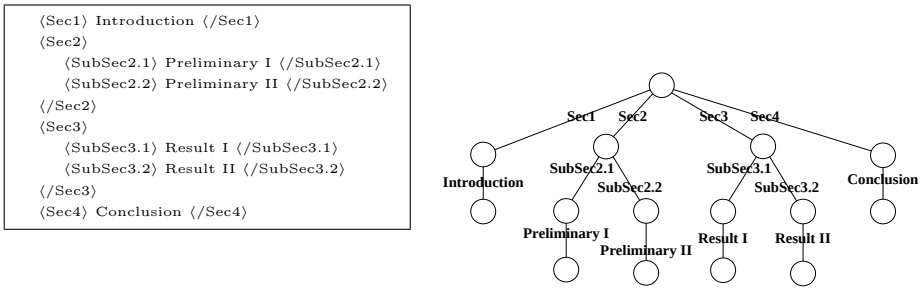
[1] Faculty of Information Sciences,
Hiroshima City University, Hiroshima 731-3194, Japan
{miyahara@its, uchida@cs, takahasi@its, ueda@its}.hiroshima-cu.ac.jp
[2] Department of Informatics, Kyushu University, Kasuga 816-8580, Japan
{y-suzuki,shoudai}@i.kyushu-u.ac.jp

Abstract. Many Web documents such as HTML files and XML files have no rigid structure and are called semistructured data. In general, such semistructured Web documents are represented by rooted trees with ordered children. We propose a new method for discovering frequent tree structured patterns in semistructured Web documents by using a tag tree pattern as a hypothesis. A tag tree pattern is an edge labeled tree with ordered children which has structured variables. An edge label is a tag or a keyword in such Web documents, and a variable can be substituted by an arbitrary tree. So a tag tree pattern is suited for representing tree structured patterns in such Web documents. First we show that it is hard to compute the optimum frequent tag tree pattern. So we present an algorithm for generating all maximally frequent tag tree patterns and give the correctness of it. Finally, we report some experimental results on our algorithm. Although this algorithm is not efficient, experiments show that we can extract characteristic tree structured patterns in those data.

1 Introduction

Background: Due to the rapid growth of Internet usage, Web documents have been rapidly increasing. Then, avoiding inappropriate Internet contents and searching interesting contents for users become more and more important. We need to extract the common characteristics among interesting contents for users. Then, the aim of this paper is to present a data mining technique of extracting meaningful and hidden knowledge from Web documents.

Data mining problems and main results: Web documents such as HTML files and XML files have no rigid structure. Such documents are called semistructured data. Abiteboul et al. [1] presented Object Exchange Model (OEM, for short) for representing semistructured data. Many semistructured data are represented by rooted trees with ordered children, which are called tree structured data. For example, in Fig. 1, the rooted ordered tree T represents the structure which the XML file *xml_sample* has. Then, in this paper, we use tree structured



xml_sample T

Fig. 1. An XML file *xml_sample* and a rooted ordered tree T as its OEM data.

data as OEM data. To formulate a schema on such tree structured data, we define an *ordered tag tree pattern*, or simply a *tag tree pattern*, as a rooted tree pattern with ordered children consisting of tree structures and structured variables. A tree is a rooted tree with ordered children and no variable. A variable can be substituted by an arbitrary tree.

Since a variable can be replaced by an arbitrary tree, overgeneralized patterns explaining given data are meaningless. Then, in order to extract meaningful knowledge from irregular or incomplete tree structured data such as semistructured Web documents, it is necessary to find a tag tree pattern t such that t can explain more data of given tree structured data than a user-specified threshold but any tag tree pattern obtained from t by substituting a variable of t can not. That is, we need to find one of the least generalized tag tree patterns. For example, consider to find one of the least generalized tag tree patterns explaining at least two OEM data in $\{T_1, T_2, T\}$ where T_1 and T_2 are OEM data in Fig. 2 and T in Fig. 1. The tag tree pattern t in Fig. 2 can explain all OEM data in $\{T_1, T_2, T\}$, that is OEM data T_1, T_2 and T are obtained from t by substituting the variable of t with a tree. But t is an overgeneralized pattern and is meaningless. On the other hand, the tag tree pattern t' in Fig. 2 is one of the least generalized tag tree patterns explaining two OEM data T and T_2 but not T_1. For example, T is obtained from t' by substituting the variables x_1, x_2 and x_3 with the trees g_1, g_2 and g_3 in Fig. 2, respectively.

In this paper, we consider three computational problems, **Frequent Tag Tree Pattern of Maximum Tree-size**, **Frequent Tag Tree Pattern of Minimum Variable-size**, and **All Maximally Frequent Tag Tree Patterns** over tag tree patterns. Frequent Tag Tree Pattern of Maximum Tree-size is the problem to find the maximum tag tree pattern t with respect to the number of vertices such that t can explain more data of input data than a user-specified threshold. This problem is based on the idea that the tag tree pattern, which has more vertices than any other tag tree patterns, gives more meaningful knowledge to us. In a similar motivation, we consider the second problem Frequent Tag Tree Pattern of Minimum Variable-size, which is the problem of finding the minimum tag tree pattern t with respect to the number of variables such that t

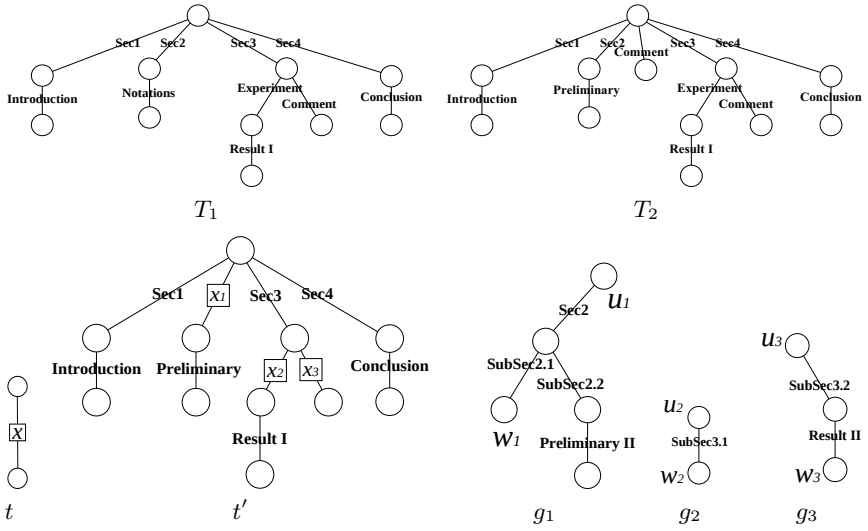


Fig. 2. A tag tree pattern t explains OEM data T_1, T_2 and T in Fig.1. A tag tree pattern t' is one of the least generalized tag tree patterns which explain OEM data T and T_2 but not T_1. A variable is represented by a box with lines to its elements. The label of a box is the variable label of the variable.

can explain more data of input data than a user-specified threshold. Firstly, we show that Frequent Tag Tree Pattern of Maximum Tree-size and Frequent Tag Tree Pattern of Minimum Variable-size are NP-complete. This indicates that it is hard to find the optimum tag tree pattern representing given data. Next, All Maximally Frequent Tag Tree Patterns is the problem to generate all maximally frequent tag tree patterns. This problem is based on the idea that meaningless tag tree patterns are excluded and all possible useful tag tree pattern are not missed. We present a data mining method from semistructured Web documents by giving an algorithm for solving All Maximally Frequent Tag Tree Patterns and show the correctness of our method.

Related works: As knowledge representations for tree structured data, a tree-expression pattern [11] and a regular path expression [4] were proposed. In our previous works [7, 8], we presented the concept of a tag tree pattern with *unordered* children from the view point of the semantics of OEM data. A tag tree pattern is different from such representations in that a tag tree pattern has structured variables which can be substituted by arbitrary trees. Several Data mining methods for discovering characteristic schema from semistructured data were proposed. In [11], Wang and Liu presented the algorithm for finding maximally frequent tree-expression patterns from semistructured data. In [2], Asai et al. presented an efficient algorithm for discovering frequent substructures from a large collection of semistructured data. In [4], Fernandez and Suciu presented the algorithm for finding optimal regular path expressions from semistructured

data. In [7], we proposed a data mining method over an unordered tag tree pattern and reported experimental results of the method. In this work, we focus on the syntactic features of OEM data. In order to apply our method to information extraction [6, 3] from semistructured data, we present a concept of an ordered tag tree pattern.

Organization: This paper is organized as follows. In Section 2, we introduce tag tree patterns as tree structured patterns. Also we define All Maximally Frequent Tag Tree Patterns over tag tree patterns as a data mining problem. In Section 3, we discuss Frequent Tag Tree Pattern of Maximum Tree-size and Frequent Tag Tree Pattern of Minimum Variable-size. In Section 4, we give an algorithm for solving All Maximally Frequent Tag Tree Patterns and show the correctness of our algorithm. In Section 5, we report some experimental results on XML documents.

2 Preliminaries

2.1 Term Trees as Tree Structured Patterns

Let $T = (V_T, E_T)$ be a rooted tree with ordered children (or simply a *tree*) which has a set V_T of vertices and a set E_T of edges. 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 a *term tree*, and elements in V_g, E_g and H_g are called a *vertex*, an *edge* and a *variable*, respectively. We assume that every edges and variables of a term tree are labeled with some words from specified languages. A label of a variable is called a *variable label*. Λ and X denote a set of edge labels and a set of variable labels, respectively, where $\Lambda \cap X = \phi$. For a set S, the number of elements in S is denoted by $|S|$. For a term tree g and its vertices v_1 and v_i, a *path* from v_1 to v_i is a sequence $v_1, v_2, \ldots, v_i$ of distinct vertices of g such that for any j with $1 \leq j < i$, there exists an edge or a variable which consists of v_j and v_{j+1}. If there is an edge or a variable which consists of v and v' such that v lies on the path from the root to v', then v is said to be the *parent* of v' and v' is a *child* of v. 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 $[v, v']$. A term tree g is called *ordered* if every internal vertex u in g has a total ordering on all children of u. The ordering on the children of u is denoted by $<_u^g$. An ordered term tree g is called *regular* if all variables in H_g have mutually distinct variable labels in X.

Definition 1. In this paper, we treat only regular ordered term trees, and then we call a regular ordered term tree a **term tree** simply. In particular, an ordered term tree with no variable is called a **ground term tree** and considered to be a tree with ordered children. Let Λ be a set of edge labels. $\mathcal{OT}_\Lambda$ denotes the set of all ground term trees whose edge labels are in Λ. $\mathcal{OTT}_\Lambda$ denotes the set of all term trees whose edge labels are in Λ.

Let $f = (V_f, E_f, H_f)$ and $g = (V_g, E_g, H_g)$ be term trees. We say that f and g are *isomorphic*, denoted by $f \equiv g$, if there is a bijection φ from V_f to V_g such that (i) the root of f is mapped to the root of g by φ, (ii) $\{u, v\} \in E_f$ if and only if $\{\varphi(u), \varphi(v)\} \in E_g$ and the two edges have the same edge label, (iii) $[u, v] \in H_f$ if and only if $[\varphi(u), \varphi(v)] \in H_g$, and (iv) 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 f and g be term trees with at least two vertices. Let $\sigma = [u, u']$ be a list of two distinct 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. A new 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.

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. Further we define a new total ordering $<_v^{f\theta}$ on every vertex v of $f\theta$ in a natural way. Suppose that v has more than one child and let u' and u'' be two children of v of $f\theta$. If v is the parent port of variables $[v, v_1], \ldots, [v, v_k]$ of f with $v_1 <_v^f \cdots <_v^f v_k$, we have the following four cases. Let g_i be a term tree which is substituted for $[v, v_i]$ for $i = 1, \ldots, k$. *Case 1*: If $u', u'' \in V_f$ and $u' <_v^f u''$, then $u' <_v^{f\theta} u''$. *Case 2*: If $u', u'' \in V_{g_i}$ and $u' <_v^{g_i} u''$ for some i, then $u' <_v^{f\theta} u''$. *Case 3*: If $u' \in V_{g_i}$, $u'' \in V_f$, and $v_i <_v^f u''$ (resp. $u'' <_v^f v_i$), then $u' <_v^{f\theta} u''$ (resp. $u'' <_v^{f\theta} u'$). *Case 4*: If $u' \in V_{g_i}$, $u'' \in V_{g_j}$ $(i \neq j)$, and $v_i <_v^f v_j$, then $u' <_v^{f\theta} u''$. If v is not a parent port of any variable, then $u', u'' \in V_f$, therefore we have $u' <_v^{f\theta} u''$ if $u' <_v^f u''$. Lastly we define the root of the resulting term tree $f\theta$ as the root of f.

Example 1. Let t and t' be two term trees described in Fig. 2. 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. 2. Then the instance $t'\theta$ of the term tree t' by θ is the tree T in Fig. 1.

Definition 2. Let Λ be a set of edge labels. The *term tree language* $L_\Lambda(t)$ of a term tree t is $\{s \in \mathcal{OT}_\Lambda \mid s \equiv t\theta \text{ for a substitution } \theta\}$. The class $\mathcal{OTTL}_\Lambda$ of all term tree languages is $\{L_\Lambda(t) \mid t \in \mathcal{OTT}_\Lambda\}$.

2.2 Tag Tree Patterns and Data Mining Problems

Definition 3. Let Λ_{Tag} and Λ_{KW} be two languages which consist of infinitely or finitely many words where $\Lambda_{Tag} \cap \Lambda_{KW} = \emptyset$. We call words in Λ_{Tag} and Λ_{KW} a **tag** and a **keyword**, respectively. A **tag tree pattern** is a term tree such that each edge label on it is any of a tag, a keyword, and a special symbol "?". 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 a tag which is considered to be identical under an equality relation on tags. (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 we don't care the edge label of $\{u, u'\}$.

A ground tag tree pattern $\pi = (V_\pi, E_\pi, \emptyset)$ *matches* a tree $T = (V_T, E_T)$ if there exists a bijection φ from V_π to V_T such that (i) the root of π is mapped to the root of T by φ, (ii) $\{v, v'\} \in E_\pi$ if and only if $\{\varphi(v), \varphi(v')\} \in E_T$, (iii) for all $\{v, v'\} \in E_\pi$, $\{v, v'\}$ matches $\{\varphi(v), \varphi(v')\}$, and (iv) for any two vertices $v', v'' \in V_\pi$, v' is a younger sibling of v'' if and only if $\varphi(v')$ is a younger sibling of $\varphi(v'')$. 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. Then *language* $L_\Lambda(\pi)$, which is the descriptive power of a tag tree pattern π, is defined as $L_\Lambda(\pi) = \{\text{a tree } T \text{ in } \mathcal{OT}_\Lambda \mid \pi \text{ matches } T\}$ where $\Lambda = \Lambda_{Tag} \cup \Lambda_{KW}$.

Data Mining Setting. A *set of semistructured data* $\mathcal{D} = \{T_1, T_2, \ldots, T_m\}$ is a set of trees. The *matching count* of a given tag tree pattern π w.r.t. $\mathcal{D}$, denoted by $match_\mathcal{D}(\pi)$, is the number of trees $T_i \in \mathcal{D}$ $(1 \leq i \leq m)$ such that π matches T_i. Then the *frequency* of 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 $0 \leq \sigma \leq 1$. A tag tree pattern π is **σ-frequent** w.r.t. $\mathcal{D}$ if $supp_\mathcal{D}(\pi) \geq \sigma$. We denote by $\Pi(\Lambda')$ the set of all tag tree patterns π such that all edge labels of π are in $\Lambda' \subseteq \Lambda = \Lambda_{Tag} \cup \Lambda_{KW}$. Let Tag be a finite subset of Λ_{Tag} and KW a finite subset of Λ_{KW}. A tag tree pattern $\pi \in \Pi(Tag \cup KW \cup \{?\})$ is **maximally σ-frequent** w.r.t. $\mathcal{D}$ if (1) π is σ-frequent, and (2) if $L_\Lambda(\pi') \subsetneq L_\Lambda(\pi)$ then π' is not σ-frequent for any tag tree pattern $\pi' \in \Pi(Tag \cup KW \cup \{?\})$.

All Maximally Frequent Tag Tree Patterns
Input: A set of semistructured data $\mathcal{D}$, a threshold $0 \leq \sigma \leq 1$, and finite sets of edge labels Tag and KW.
Problem: Generate all maximally σ-frequent tag tree patterns w.r.t. $\mathcal{D}$ in $\Pi(Tag \cup KW \cup \{?\})$.

Example 2. As examples, we give three OEM data T_1 and T_2 in Fig. 2 and T in Fig. 1 and a maximally $\frac{2}{3}$-frequent tag tree pattern t' in $\Pi(\{\langle\text{Sec1}\rangle, \langle\text{Sec2}\rangle, \langle\text{Sec3}\rangle, \langle\text{Sec4}\rangle\}, \{\text{Introduction, Preliminary,Result I, Conclusion}\}, \text{"?"})$. The tag tree pattern t' in Fig. 2 matches T and T_2, but t' does not match T_1.

3 Hardness Results of Finding the Optimum Frequent Tag Tree Pattern

In this section, we discuss two problems of computing an expressive σ-frequent tag tree pattern. First we show that it is hard to compute the frequent tag tree pattern of maximum tree-size w.r.t. a set of semistructured data. The formal definition of the problem is as follows.

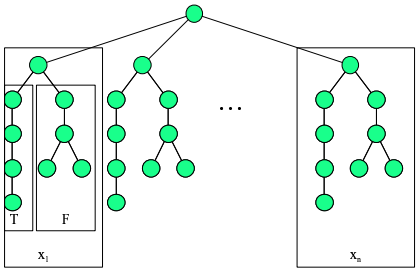


Fig. 3. Tree P_0

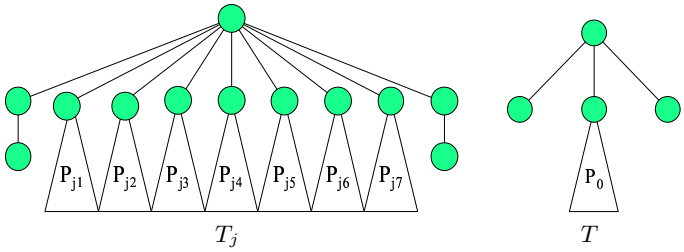


Fig. 4. Tree T_j and T

Frequent Tag Tree Pattern of Maximum Tree-size
Instance: A set of semistructured data $\mathcal{D} = \{T_1, T_2, \ldots, T_m\}$, a real number σ $(0 \leq \sigma \leq 1)$ and a positive integer K.
Question: Is there a σ-frequent tag tree pattern $\pi = (V, E, H)$ w.r.t. $\mathcal{D}$ with $|V| \geq K$?

Theorem 1. Frequent Tag Tree Pattern of Maximum Tree-size *is NP-complete.*

Proof. Membership in NP is obvious. We transform 3-SAT to this problem. Let $U = \{x_1, \ldots, x_n\}$ be a set of variables and $C = \{c_1, \ldots, c_m\}$ a collection of clauses over U with $|c_j| = 3$ for any j $(1 \leq j \leq m)$. For a tree T and a vertex u of T, we denote the subtree consisting of u and the descendants of u by $T[u]$. Let P_0 be the tree which is described in Fig. 3. The root of P_0 has n children. Let $v_1, v_2, \ldots, v_n$ be the n children. For each i $(1 \leq i \leq n)$, $P_0[v_i]$ corresponds to the truth assignment to x_i.

We construct trees $T_1, \ldots, T_m$ from the tree P_0 and $c_1, \ldots, c_m$ in the following way. T_j $(1 \leq j \leq n)$ is described in Fig. 4. The root of T_j has 9 children. Let $v_{j0}, v_{j1}, \ldots, v_{j8}$ be the 9 children. The inner 7 subtrees $T_j[v_{j1}], \ldots, T_j[v_{j7}]$ correspond to the truth assignments that satisfy c_j. Each $T_j[v_{ji}]$ $(1 \leq i \leq 7)$ is constructed as follows. Let $c_j = \{\ell_{j1}, \ell_{j2}, \ell_{j3}\}$ where $\ell_{jk} = x_{n_{jk}}$ or $\bar{x_{n_{jk}}}$ $(1 \leq k \leq 3, 1 \leq n_{jk} \leq n)$. The 7 truth assignments to $(x_{n_{j1}}, x_{n_{j2}}, x_{n_{j3}})$ make c_j *true*.

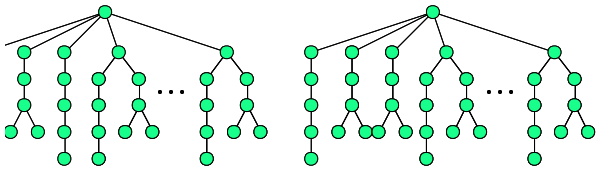


Fig. 5. Two truth assignments for $(x_1, x_2, x_3) = (true, false, true), (true, false, false)$

For the ith truth assignment $(1 \leq i \leq 7)$ and all $1 \leq n_{j1}, n_{j2}, n_{j3} \leq n$, P_{ji} is obtained from P_0 by removing the right (resp. left) subtree rooted at $v_{n_{jk}}$ of P_0 if $x_{n_{jk}}$ is *true* (resp. *false*). This resulting tree P_{ji} becomes $T_j[v_{ij}]$. For example, the left tree of Fig. 5 represents a truth assignment $(x_1, x_2, x_3) = (true, false, true)$.

Lastly let T be the special tree (Fig. 4) which is constructed from P_0. Let $S = \{T_1, \ldots, T_m, T\}$, $\sigma = 1$, and $K = 5n + 4$. Then we can show the following two facts.

1. Let π be a σ-frequent tag tree pattern w.r.t. $\mathcal{D}$. Then the root of π has just three children and the second child of the three children has just n children.
2. Let $G_1, G_2, G_3, g_1, g_2, g_3$ be trees and tag tree patterns described in Fig. 6, respectively. Then g_1 is σ-frequent w.r.t. $\{G_1, G_2, G_3\}$, g_2 is σ-frequent w.r.t. $\{G_1, G_3\}$, and g_3 is σ-frequent w.r.t. $\{G_2, G_3\}$.

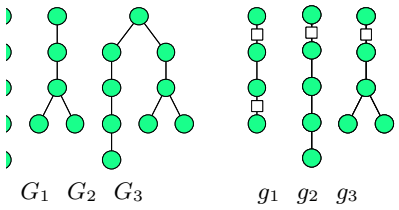


Fig. 6. Trees G_1, G_2, G_3 and tag tree patterns g_1, g_2, g_3

From these two facts, if 3-SAT has a truth assignment which satisfies all clauses in C, there is a σ-frequent tag tree pattern $\pi = (V, E, H)$ w.r.t. $\mathcal{D}$ with $|V| = 5n + 4 = K$ (Fig. 7). Conversely, if there is a σ-frequent tag tree pattern $\pi = (V, E, H)$ w.r.t. $\mathcal{D}$ with $|V| = 5n + 4$, the numbers of the children of the vertices of depth 5 show one of the truth assignment which satisfies C. □

Second we show that it is hard to compute the frequent tag tree pattern of minimum variable-size w.r.t. a set of semistructured data. The formal definition of the problem is as follows.

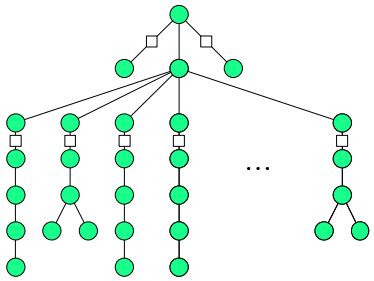


Fig. 7. A tag tree pattern π such that π is $\sigma(=1)$-frequent w.r.t. $\mathcal{D}$

Frequent Tag Tree Pattern of Minimum Variable-size
Instance: A set of semistructured data $\mathcal{D} = \{T_1, T_2, \ldots, T_m\}$, a real number σ $(0 \leq \sigma \leq 1)$ and a positive integer K.
Question: Is there a σ-frequent tag tree pattern $\pi = (V, E, H)$ w.r.t. $\mathcal{D}$ with $|H| \leq K$?

Theorem 2. Frequent Tag Tree Pattern of Minimum Variable-size *is NP-complete.*

Proof. (Sketch) Membership in NP is obvious. The reduction is the same as the one in Theorem 1 but $K = n + 2$. □

4 Generating All Maximally Frequent Tag Tree Patterns

4.1 Algorithm for Generating All Maximally Frequent Tag Tree Patterns

In this section, we present an algorithm for solving All Maximally Frequent Tag Tree Patterns. That is, we give an algorithm which generates all maximally σ-frequent tag tree patterns. The algorithm uses a polynomial time matching algorithm for term trees [10] to compute the frequency of a tag tree pattern and a method for generating all rooted trees with ordered children [9].

Algorithm for solving All Maximally Frequent Tag Tree Patterns
Input: A set of semistructured data $\mathcal{D}$, a threshold $0 \leq \sigma \leq 1$, and finite sets of edge labels Tag and KW.
Output: All maximally σ-frequent tag tree patterns w.r.t. $\mathcal{D}$ in $\Pi(Tag \cup KW \cup \{?\})$.

Let n be the maximum number of vertices over all trees in $\mathcal{D}$. We repeat the following three steps for $k = 2, \ldots, n$. Let Π_k^σ be the set of all σ-frequent tag tree patterns with at most k vertices and no edge. Let $\Pi_k^\sigma(\Lambda')$ be the set of all σ-frequent tag tree patterns with at most k vertices and edge labels in $\Lambda' \subseteq \Lambda$.

1. Generate all tag tree patterns with k vertices and no edge, by using an algorithm for generating all rooted trees with ordered children on k vertices [9]. For each tag tree pattern π with k vertices and no edge, we compute the frequency of π and if the frequency is greater than or equal to σ then we add π to $\mathit{\Pi}_k^\sigma$.
2. For each $\pi \in \mathit{\Pi}_k^\sigma$, we try to substitute variables of π with edges labeled with "?" as many as possible so that all σ-frequent tag tree patterns in $\mathit{\Pi}_k^\sigma(\{?\})$ are generated. This work can be done in a backtracking way. Then for each $\pi \in \mathit{\Pi}_k^\sigma(\{?\})$, we try to replace ?'s with labels in $Tag \cup KW$ as many as possible so that all σ-frequent tag tree patterns in $\mathit{\Pi}_k^\sigma(Tag \cup KW \cup \{?\})$ are generated. This work can be done in a backtracking way.
3. Finally we check by using the maximality test algorithm in Section 4.2 whether or not $\pi \in \mathit{\Pi}_k^\sigma(Tag \cup KW \cup \{?\})$ is maximally σ-frequent.

4.2 Correctness of the Generating Algorithm

In this section, we consider the following problem to complete our generating algorithm in Section 4.1. Let $\mathit{\Lambda}_{Tag}$ and $\mathit{\Lambda}_{KW}$ be infinite or finite languages of tags and keywords, respectively.

MAXIMALITY TEST
Instance: A set of semistructured data $\mathcal{D}$, a threshold $0 \leq \sigma \leq 1$, and two finite sets $Tag \subseteq \mathit{\Lambda}_{Tag}$ and $KW \subseteq \mathit{\Lambda}_{KW}$, and a tag tree pattern $\pi \in \mathit{\Pi}_k^\sigma(Tag \cup KW \cup \{?\})$ satisfying the following conditions:
(i) Any tag tree pattern obtained from π by replacing any variable in π with an edge which has a label in $Tag \cup KW \cup \{?\}$ is not σ-frequent w.r.t. $\mathcal{D}$.
(ii) Any tag tree pattern obtained from π by replacing any edge with a label "?" in π with an edge which has a label in $Tag \cup KW$ is not σ-frequent w.r.t. $\mathcal{D}$.
Question: Decide whether or not π is a maximally σ-frequent tag tree pattern w.r.t. $\mathcal{D}$.

We show an algorithm for solving MAXIMALITY TEST. If the target of our interest is only skeleton of given data, we ignore the edge labels of the data and find a tag tree pattern in $\mathit{\Pi}(\{?\})$. We note that if $|\mathit{\Lambda}| = |\mathit{\Lambda}_{Tag} \cup \mathit{\Lambda}_{KW}| = 1$ the label "?" is meaningless. Thus when $|\mathit{\Lambda}| = |\mathit{\Lambda}_{Tag} \cup \mathit{\Lambda}_{KW}| = 1$ we identify the unique label in $\mathit{\Lambda}$ with "?". Let x_2 be a variable such that the siblings just before and after x_2 are variables and x_2 has only one child which connects to x_2 with a variable (See the right figure of Fig. 8). We call the variable like x_2 a *surrounded* variable. We omit the proof of the next lemma.

Lemma 1. *Let π' be a tag tree pattern which has a surrounded variable x_2. Let π be the tag tree pattern obtained from π' by replacing the variable x_2 with an edge which has a label "?" (Fig. 8). Then $L_\mathit{\Lambda}(\pi') = L_\mathit{\Lambda}(\pi)$.*

Our maximality test algorithm consists of the following steps.

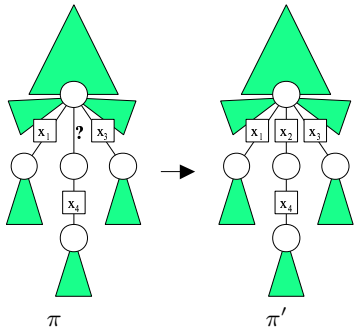
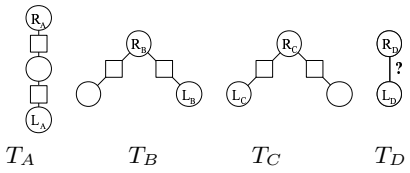


Fig. 8. This center edge of π is replaced with a variable.

1. If π has the substructure like the left figure of Fig. 8, then for all the substructures we replace the center edges with variables (See the right figure of Fig. 8). Let π' be the tag tree pattern after the replacements.

$\theta_A(x) = \{x := [T_A, [R_A, L_A]]\}$
$\theta_B(x) = \{x := [T_B, [R_B, L_B]]\}$
$\theta_C(x) = \{x := [T_C, [R_C, L_C]]\}$
$\theta_D(x) = \{x := [T_D, [R_D, L_D]]\}$



2. If there exists a variable x in π' which is not a surrounded variable such that $\pi'\theta_X(x)$ is σ-frequent w.r.t. $\mathcal{D}$ for any $X \in \{A, B, C, D\}$, then π is not maximally σ-frequent w.r.t. $\mathcal{D}$.
3. If there exists a surrounded variable x such that $\pi'\theta_X(x)$ is σ-frequent w.r.t. $\mathcal{D}$ for any $X \in \{A, B, C\}$, then π is not maximally σ-frequent w.r.t. $\mathcal{D}$.
4. If both x_1 and x_2 are surrounded variables (See Fig. 9), we check whether or not $\pi'\theta_D(x_1)\theta_D(x_2)$ is σ-frequent w.r.t. $\mathcal{D}$. If $\pi'\theta_D(x_1)\theta_D(x_2)$ is σ-frequent w.r.t. $\mathcal{D}$, then π is not maximally σ-frequent w.r.t. $\mathcal{D}$.

If π passes all the above tests, π is maximally σ-frequent w.r.t. $\mathcal{D}$.

Lemma 2. *Let $\Lambda = \Lambda_{Tag} \cup \Lambda_{KW}$. Let $\pi = (V_\pi, E_\pi, H_\pi)$ be an input tag tree pattern which is decided to be maximally σ-frequent w.r.t. $\mathcal{D}$ by the above strategy. If there is a tag tree pattern $\pi' = (V_{\pi'}, E_{\pi'}, H_{\pi'})$ with no surrounded variable which is σ-frequent w.r.t. $\mathcal{D}$ and moreover $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$, then $\pi = \pi'$.*

Proof. (Sketch) We can show the following claims.

Claim 1. There exists a bijection $\xi : V_{\pi'} \to V_\pi$ such that for any $u, v \in V_{\pi'}$, $\{u, v\} \in E_{\pi'}$ or $[u, v] \in H_{\pi'}$ if and only if $\{\xi(u), \xi(v)\} \in E_\pi$ or $[\xi(u), \xi(v)] \in H_\pi$.

Claim 2. Let $\xi : V_{\pi'} \to V_\pi$ be a bijection defined in *Claim 1*. For any $u, v \in V_{\pi'}$, if $[u, v] \in H_{\pi'}$ and $\{\xi(u), \xi(v)\} \in E_\pi$ then $L_\Lambda(\pi') \not\subseteq L_\Lambda(\pi)$. Therefore if $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$ then $[u, v] \in H_{\pi'}$ implies $[\xi(u), \xi(v)] \in H_\pi$.

Claim 3. Let $\xi : V_{\pi'} \to V_\pi$ be a bijection defined in *Claim 1*. When $|\Lambda| = |\Lambda_{Tag} \cup \Lambda_{KW}| \geq 2$, for any $u, v \in V_{\pi'}$ with $\{u, v\} \in E_{\pi'}$, if an edge label of $\{u, v\}$ is "?" and $\{\xi(u), \xi(v)\}$ is an edge in E_π with a label in $Tag \cup KW$ then $L_\Lambda(\pi') \not\subseteq L_\Lambda(\pi)$. Therefore if $|\Lambda| \geq 2$ and $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$ then $\{u, v\} \in E_{\pi'}$ with a label "?" implies $\{\xi(u), \xi(v)\} \in E_\pi$ with a label "?" or $[\xi(u), \xi(v)] \in H_\pi$.

From the conditions of an input tag tree pattern π of MAXIMALITY TEST, for any $u, v \in V_{\pi'}$, if $[\xi(u), \xi(v)] \in H_\pi$ then $[u, v] \in H_{\pi'}$, and if $\{\xi(u), \xi(v)\} \in E_\pi$ which has a label "?" then $\{u, v\} \in E_{\pi'}$ and the label of $\{u, v\}$ is "?". Thus we conclude that $\pi = \pi'$. □

We can easily see that the above steps 1–4 run in polynomial time by using a polynomial time matching algorithm for term trees [10].

Theorem 3. MAXIMALITY TEST *is computable in polynomial time.*

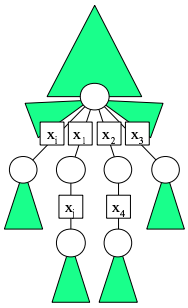


Fig. 9. Both x_1 and x_2 are surrounded variables.

5 Implementation and Experimental Results

We have implemented the algorithm for generating all maximally frequent tag tree patterns in Section 4 on a DELL workstation PowerEdge 6400 with Xeon 700 MHz CPU. We report some experiments on a sample file of semistructured data. The sample file is converted from a sample XML file about garment sales data such as *xml_sample* in Fig. 1. The sample file consists of 172 tree structured data. The maximum number of vertices over all trees in the file is 11, the maximum depth is 2 and the maximum number of children over all vertices is 5. In the experiments described in Fig. 10, we gave the algorithm "<Weeknumber>"

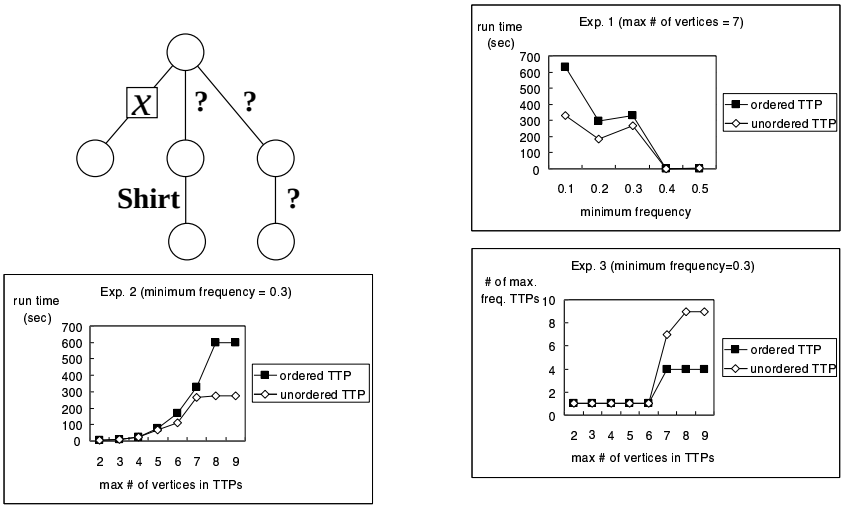


Fig. 10. Experimental results for generating all maximally frequent tag tree patterns. A maximally σ-frequent tag tree pattern obtained in the experiment.

and "<Designnumber>" as tags, and "Summer" and "Shirt" as keywords. The algorithm generated all maximally σ-frequent tag tree patterns w.r.t. the sample file for a specified minimum frequency σ. We can set the maximum number ("max # of vertices in TTPs") of vertices of tag tree patterns in the hypothesis space.

We explain the results of Fig. 10. "TTP" means a tag tree pattern. In order to evaluate the usefulness and performance of our data mining method in this paper, we have two types of experiments. The results of "ordered TTP" are given by the algorithm for generating all maximally frequent "ordered" tag tree patterns in this work. The results of "unordered TTP" are given by the algorithm for generating all maximally frequent "unordered" tag tree patterns in our previous work [7].

Exp.1 shows the consumed run time (sec) by the two algorithms for varied minimum frequencies and the specified max # of vertices=7. Exp.2 gives the consumed run time (sec) by the two algorithms for the specified minimum frequency =0.3 and varied max numbers of vertices of TTP in the hypothesis spaces. Also, Exp.3 shows the numbers of maximally frequent TTPs obtained by the two algorithms for the specified minimum frequency =0.3 and varied max numbers of vertices of TTP in the hypothesis spaces. These experiments show that the method for generating all maximally frequent "ordered" tag tree patterns is more time-consuming than the one for generating all maximally frequent "unordered" tag tree patterns. But it is effective as compared with the size of hypothesis spaces. Also maximally frequent ordered tag tree patterns capture more precisely characteristic structures than maximally frequent unordered tag tree patterns.

6 Conclusions

In this paper, we have studied knowledge discovery from semistructured Web documents such as HTML/XML files. We have proposed a tag tree pattern which is suited for representing tree structured patterns in such semistructured data. We have shown that it is hard to compute the frequent tag tree pattern of maximum tree-size. So we have given an algorithm for generating all maximally frequent tag tree patterns. We can improve this algorithm by using the method in [5].

Acknowledgments. This work is partly supported by Grant-in-Aid for Scientific Research (C) No.13680459 from Japan Society for the Promotion of Science and Grant for Special Academic Research No.1608 from Hiroshima City University.

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) (to appear)*, 2002.
3. C.-H. Chang, S.-C. Lui, and Y.-C. Wu. Applying pattern mining to web information extraction. *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-2001), Springer-Verlag, LNAI 2035*, pages 4–15, 2001.
4. M. Fernandez and D. Suciu. Optimizing regular path expressions using graph schemas. *Proceedings of the 14th International Conference on Data Engineering (ICDE-98), IEEE Computer Society*, pages 14–23, 1998.
5. K. Furukawa, T. Uchida, K. Yamada, T. Miyahara, T. Shoudai, and Y. Nakamura. Extracting characteristic structures among words in semistructured documents. *Proc. PAKDD-2002, Springer-Verlag, LNAI (to appear)*, 2002.
6. N. Kushmerick. Wrapper induction: efficiency and expressiveness. *Artificial Intelligence*, 118:15–68, 2000.
7. T. Miyahara, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Discovery of frequent tree structuted patterns in semistructured web documents. *Proc. PAKDD-2001, Springer-Verlag, LNAI 2035*, pages 47–52, 2001.
8. T. Shoudai, T. Uchida, and T. Miyahara. Polynomial time algorithms for finding unordered tree patterns with internal variables. *Proc. FCT-2001, Springer-Verlag, LNCS 2138*, pages 335–346, 2001.
9. W. Skarbek. Generating ordered trees. *Theoretical Computer Science*, 57:153–159, 1988.
10. Y. Suzuki, T. Shoudai, T. Miyahara, and T. Uchida. Polynomial time inductive inference of ordered tree patterns with internal variables from positive data. *Proc. LA Winter Symposium, Kyoto, Japan*, pages 33–1 – 33–12, 2002.
11. K. Wang and H. Liu. Discovering structural association of semistructured data. *IEEE Trans. Knowledge and Data Engineering*, 12:353–371, 2000.