<table>
<thead>
<tr>
<th>分類</th>
<th>内容</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Discovery of Maximally Frequent Tag Tree Patterns in Semistructured Data (New Developments of Theory of Computation and Algorithms)</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Miyahara, Tetsuhiro; Shoudai, Takayoshi; Uchida, Tomoyuki</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (2001), 1205: 71-76</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2001-05</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/41003">http://hdl.handle.net/2433/41003</a></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>
Discovery of Maximally Frequent Tag Tree Patterns in Semistructured Data

Tetsuhiro Miyahara (宮原 哲浩)\(^1\),
Takayoshi Shoudai (正代 隆義)\(^2\), and Tomoyuki Uchida (内田 智之)\(^1\)

\(^1\) Faculty of Information Sciences, Hiroshima City University, Hiroshima 731-3194, Japan (広島市立大学 情報科学部)
\{miyahara@its, uchida@cs\}.hiroshima-cu.ac.jp
\(^2\) Department of Informatics, Kyushu University, Fukuoka 816-8580, Japan (九州大学大学院 システム情報科学研究科)
shoudai@i.kyushu-u.ac.jp

Abstract.
Many documents such as Web documents or XML files do not have rigid structures. Such semistructured documents have been rapidly increasing. We propose a new method for discovering frequent tree structured patterns in semistructured Web documents. We consider the data mining problem of finding a maximally frequent tag tree pattern in semistructured data such as Web documents. A tag tree pattern is an edge labeled tree which has hyperedges as variables. An edge label is a tag or a keyword in Web documents, and a variable can be substituted by any tree. So a tag tree pattern is suited for representing tree structured patterns in Web documents. We give a polynomial time algorithm for finding a maximally frequent tag tree pattern. By using this algorithm, users can get one of knowledge which users want.

1 Introduction

Web data have been rapidly increasing as the Information Technologies develop. Our target for knowledge discovery is the Web data which have tree structures such as documents on World Wide Web or XML/SGML files. Such Web documents are called semistructured data [1]. The purpose of this paper is to discover frequent tree structured patterns which are hidden useful and simple knowledge in real semistructured Web documents.

In this paper, we adopt a variant of the Object Exchange Model (OEM, for short) in [1] for representing semistructured data. As an example, we give an XML file xml_sample and a labeled tree \(O_1\) as its OEM data in Fig. 1. Many real semistructured data have no absolute schema fixed in advance, and their structure may be irregular or incomplete. As knowledge representations for semistructured data, for example, the type of objects [7], tree-expression pattern [9] and regular path expression [2] were proposed. In [4], we presented the concept of term trees as a graph pattern suited for representing tree-like semistructured data. A term tree is a hypergraph whose hyperedges are regarded as variables.

In [6], we gave the knowledge discovery system KD-FGS which receives the graph structured data and produces a hypothesis by using Formal Graph System [8] as a knowledge representation languages. And, in [4], we designed the efficient knowledge discovery system having polynomial time matching algorithms and a polynomial time inductive inference algorithm from tree-like semistructured data. The above two systems find a hypothesis consistent with all input data or a term tree which can explain a minimal language including all input data, respectively. These systems works correctly and effectively for complete data. However, for irregular or incomplete data, the systems may output an obvious or meaningless knowledge. In this paper, in order to obtain knowledge efficiently from irregular or incomplete semistructured data, we give a tag tree pattern which is a special type of a term tree. In Fig. 1, for example, we can obtain OEM
Fig. 1. An XML file `xml.sample` and a labeled tree $o_1$ as its OEM data. A tag tree pattern $p$ which matches OEM data $o_1$ and $o_2$ but does not match OEM data $o_3$.

data $o_1$ and $o_2$ from the tag tree pattern $p$ by replacing variables in $p$ with arbitrary trees but does not obtained OEM data $o_3$ from $p$.

We propose a discovery method for solving two problems for discovering a tag tree pattern as knowledge. We consider the problem to find a maximally tag tree pattern which can explain more data of input Web data than a user-specified threshold. We give a polynomial time algorithm for solving this problem. This algorithm is not necessarily to output a tag tree pattern which a user wants, but we can get efficiently one of frequent tag tree patterns.

This paper is organized as follows. In Section 2, we give the notion of a term tree as a tree structured pattern for semistructured data. And a tag tree pattern is defined for OEM Web Data. Then, we formally define a data mining problem, Maximally Frequent Tag Tree Pattern Problem. In Section 3 we give an algorithm for solving this problem.

2 Preliminaries

2.1 Term Trees as Tree Structured Patterns

Let $T = (V_T, E_T)$ be a rooted tree (or simply tree) with an edge labeling. A variable in $V_T$ is a list $[u, u']$ of two distinct vertices $u$ and $u'$ in $V_T$. A label of a variable is called a variable label. $\Lambda$ 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 $|S|$.

Definition 1. A triplet $g = (V_g, E_g, H_g)$ is called a rooted term tree (or simply a term tree) if $H_g$ is a finite set of variables such that for any $[u, u'] \in H_g$, $[u', u]$ is not in $H_g$, and the graph $(V_g, E_g \cup E'_g)$ is a tree where $E'_g = \{(u, v) \mid [u, v] \in H_g\}$. A term tree $g$ is called regular if all variables in $H_g$ have mutually distinct variable labels in $X$. In particular, a term tree with no variable is called a ground term tree and considered to be a standard tree.

For a rooted term tree $f$, a path from $v_1$ to $v_i$ is a sequence $v_1, v_2, \ldots, v_i$ of distinct vertices such that for $1 \leq j < i$, there exists an edge 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 $r_1$ to $v$, then $v'$ is said to be the parent of $v$ and $v$ is a child of $v'$.

Let $f = (V_f, E_f, H_f)$ and $g = (V_g, E_g, H_g)$ be regular term trees. We say that $f$ and $g$ are isomorphic, denoted by $f \equiv g$, if there is a bijection $\varphi$ from $V_f$ to $V_g$ such that (i) the root of $f$ is mapped to the root of $g$ by $\varphi$, (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, and (iii) $\{u, v\} \in H_f$ or $[v, u] \in H_f$ if and only if $\varphi(u), \varphi(v) \in H_g$ or $[\varphi(v), \varphi(u)] \in H_g$. Two isomorphic regular term trees are considered to be identical.

Let $f$ and $g$ be term trees with at least two vertices. Let $\sigma = [u, u']$ be a list of two distinct vertices in $g$. 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 $e_1 = [v_1, v'_1], \ldots, e_m =$
Let \( g_1, \ldots, g_m \) be \( m \) copies of \( g \) and \( u_i, u'_i \) the vertices of \( g_i \) corresponding to \( u, u' \) of \( g \). For each variable \( e_i = [v_i, v'_i] \) from \( Hf \) and by identifying the vertices \( v_i, v'_i \) with the vertices \( u_i, u'_i \) of \( g_i \). Let the root of the resulting term tree be the root of \( f \). A substitution \( \theta \) is a finite collection of bindings \( \{x_1 := [g_1, \sigma_1], \ldots, x_n := [g_n, \sigma_n]\} \), where \( x_i \)'s are mutually distinct variable labels in \( X \). The term tree \( f\theta \), called the instance of \( f \) by \( \theta \), is obtained by applying the all bindings \( x_i := [g_i, \sigma_i] \) on \( f \) simultaneously. For term trees \( f \) and \( g \), if there exists a substitution \( \theta \) such that \( f \equiv g\theta \), we write \( f \preceq g \). Especially we write \( f \not\preceq g \) if \( f \preceq g \) and \( g \not\preceq f \). In Fig. 2, as examples, we give the term tree \( t \), two ground term trees \( t_1 \) and \( t_2 \), and an instance \( t\theta \) which is obtained by applying a substitution \( \theta = \{x := [t_1, [v_1, v_2]], y := [t_2, [u_1, u_2]]\} \) to the term graph \( t \).

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 label 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 tag tree pattern \( \pi = (V_\pi, E_\pi, \emptyset) \) matches a tree \( T = (V_T, E_T) \) if there exists a bijection \( \varphi \) from \( V_\pi \) to \( V_T \) such that (i) the root of \( \pi \) is mapped to the root of \( T \) by \( \varphi \), (ii) \( \{v, v'\} \in E_\pi \) if and only if \( \{\varphi(v), \varphi(v')\} \in E_T \), and (iii) for all \( \{v, v'\} \in E_\pi \), \( \{v, v'\} \) matches \( \{\varphi(v), \varphi(v')\} \). A tag tree pattern \( \pi \) matches a tree \( T \) if there exists a substitution \( \theta \) such that \( \pi\theta \) is a ground tag tree and \( \pi\theta \) matches \( T \). Then language \( L(\pi) \), which is the descriptive power of a tag tree pattern \( \pi \), is defined as \( L(\pi) = \{\text{a tree } T \mid \pi \text{ matches } T\} \).

Data Mining Problems. A set of semistructured data \( D = \{T_1, T_2, \ldots, T_m\} \) is a set of trees. The matching count of a given tag tree pattern \( \pi \) w.r.t. \( D \), denoted by \( \text{match}_D(\pi) \), is the number of trees \( T_i \in D \) (1 \( \leq i \leq m \)) such that \( \pi \) matches \( T_i \). Then the frequency of \( \pi \) w.r.t. \( D \) is defined by \( \text{supp}_D(\pi) = \text{match}_D(\pi)/|D| \). Let \( \sigma \) be a real number where \( 0 \leq \sigma \leq 1 \). A tag tree pattern \( \pi \) is \( \sigma \)-frequent w.r.t. \( D \) if \( \text{supp}_D(\pi) \geq \sigma \). We denote by \( \Pi(L) \) the set of all tag tree patterns such that all edge labels are in \( L \). Let \( Tag \) be a finite subset of \( \Lambda_{Tag} \) and \( KW \) a finite subset of \( \Lambda_{KW} \). A tag tree pattern \( \pi \in \Pi(Tag \cup KW \cup \{?\}) \) is maximally \( \sigma \)-frequent w.r.t. \( D \) if (1)
\(\pi\) is \(\sigma\)-frequent, and (2) if \(L(\pi') \subsetneq L(\pi)\) then \(\pi'\) is not \(\sigma\)-frequent for any tag tree pattern \(\pi' \in \Pi(\text{Tag} \cup \text{KW} \cup \{?\})\). In Fig. 1, as examples, we give a maximally \(\frac{3}{4}\)-frequent tag tree pattern \(p\) in \(\Pi\{\langle\text{Fruits}\rangle, \langle\text{Name}\rangle, \langle\text{Shape}\rangle\} \cup \{\text{melon}\} \cup \{?\}\) with respect to OEM data \(a_1, a_2\) and \(a_3\). The tag tree pattern \(p\) matches \(a_1\) and \(a_2\), but \(p\) does not match \(a_3\).

Maximally Frequent Tag Tree Pattern Problem

**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:** Find a maximally \(\sigma\)-frequent tag tree pattern \(\pi \in \Pi(\text{Tag} \cup \text{KW} \cup \{?\})\) w.r.t. \(\mathcal{D}\).

All Maximally Frequent Tag Tree Patterns Problem

**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 \(\sigma\)-frequent tag tree patterns w.r.t. \(\mathcal{D}\) in \(\Pi(\text{Tag} \cup \text{KW} \cup \{?\})\).

### 3 Polynomial Time Algorithm for Finding a Maximally Frequent Tag Tree Pattern

In order to design an efficient knowledge discovery system from tree structured data, we give a polynomial time algorithm for finding a maximally frequent tag tree pattern.

**Lemma 1** ([3]). *Let \(t\) be a regular term tree and \(T\) a tree. The problem for deciding whether or not \(T \in L(t)\) is computable in polynomial time.*

For a tag tree pattern \(\pi\), we can compute \(\text{match}_{\mathcal{D}}(\pi)\) in polynomial time by a membership algorithm, which is an extension of the membership algorithm used in Lemma 1, for each tree in \(\mathcal{D}\) and \(\pi\).

**Theorem 1.** Maximally Frequent Tag Tree Pattern Problem is computable in polynomial time.

**Proof.** The following strategy consisting of three steps works correctly for finding one of maximally \(\sigma\)-frequent tag tree patterns for \(\mathcal{D}\):

**Step 1.** We find a maximally \(\sigma\)-frequent tag tree pattern with no edge. We start with only one vertex which becomes a root and visit unvisited vertices in breadth-first manner. For each vertex \(v\), the following procedure is repeated until no vertex can be added: If there exists a linear chain consisting of only variables such that the tag tree pattern remains \(\sigma\)-frequent when the end point of the linear chain is attached to the vertex \(v\), then attach the end point of the longest such linear chain to the vertex \(v\).

**Step 2.** Replace each variable with an edge labeled with "?" if the tag tree pattern remains \(\sigma\)-frequent.

**Step 3.** Replace each "?" with labels in Tag and KW if the tag tree pattern remains \(\sigma\)-frequent.

The detail of the algorithm is described in Fig. 3.

Let \(\pi = (V_\pi, E_\pi, H_\pi)\) and \(\pi' = (V'_{\pi'}, E'_{\pi'}, H'_{\pi'})\) be tag tree patterns. We write \(\pi \approx \pi'\) if there exists a bijection \(\varphi : V_\pi \rightarrow V'_{\pi'}\) such that for \(u, v \in V_\pi\), \(\{u, v\} \in E_\pi\) or \(\{u, v\} \in H_\pi\) if and only if \(\varphi(u), \varphi(v)\) \(\in E'_{\pi'}\) or \(\varphi(u), \varphi(v)\) \(\in H'_{\pi'}\). Since \(\Lambda_{\text{Tag}}\) and \(\Lambda_{\text{KW}}\) are infinite, we can show the following two claims.

**Claim.** For any two tag tree patterns \(\pi, \pi'\) with \(\pi \approx \pi'\), \(\pi \preceq \pi'\) if and only if \(L(\pi) \subseteq L(\pi')\).

**Claim.** Let \(\pi = (V_\pi, E_\pi, H_\pi)\) be an output tag tree pattern by procedure MF-TTP when a set of semistructured data \(\mathcal{D}\) is given. If there exists a tag tree pattern \(\pi'\) such that \(\pi'\) is \(\sigma\)-frequent w.r.t. \(\mathcal{D}\) and \(L(\pi') \subseteq L(\pi)\) then \(\pi \approx \pi'\).

Let \(\pi = (V_\pi, E_\pi, H_\pi)\) be an output tag tree pattern by procedure MF-TTP. Suppose that there exists a tag tree pattern \(\pi' = (V'_{\pi'}, E'_{\pi'}, H'_{\pi'})\) such that \(\mathcal{D} \subseteq L(\pi') \subsetneq L(\pi)\). From the above two claims, we obtain \(\pi' \approx \pi\) and then \(\pi' < \pi\). We have two possibilities. The first case we have
Algorithm MF-TTP;
begin  
  // Step 1  
  \(\pi := \text{Basic-Tree}(D)\);
  // Step 2  
  foreach variable \([u, v] \in H_\pi\) do begin  
    let \(\pi'\) be a tag tree pattern which is obtained from \(\pi\) by replacing variable \([u, v]\)  
    with an edge labeled with "?";  
    if \(\pi'\) is \(\sigma\)-frequent w.r.t. \(D\) then \(\pi := \pi'\)  
  end;
  // Step 3  
  foreach edge \([u, v] \in E_\pi\) with an edge label "?" do begin  
    foreach edge label \(w \in \text{Tag} \cup \text{KW}\) do begin  
      let \(\pi'\) be a tag tree pattern which is obtained from \(\pi\) by replacing "?" with \(w\);  
      if \(\pi'\) is \(\sigma\)-frequent w.r.t. \(D\) then begin  
        \(\pi := \pi'\); break end  
    end  
  end  
return \(\pi\) end;

Procedure Basic-Tree(D);
begin  
  // Each variable is assumed to be labeled with a distinct variable label.  
  \(d := 0; \pi := (\{r\}, \emptyset, \emptyset)\);  
  \(\pi := \text{breadth-expansion}(r, \pi, D)\);  
  max-depth := the depth of \(\pi\); \(d := d + 1\);  
  while \(d \leq \text{max-depth} - 1\) do begin  
    \(v := \text{the leftmost vertex of } \pi\) at depth \(d\);  
    \(\pi := \text{breadth-expansion}(v, \pi, D)\);  
    while there exists a sibling of \(v\) which is not yet visited do begin  
      \(v' := \text{a sibling of } v\) which is not yet visited;  
      \(\pi := \text{breadth-expansion}(v', \pi, D)\);  
    end  
    \(d := d + 1\);  
  end  
return \(\pi\) end;

Procedure breadth-expansion(v, \(\pi\), D);
begin  
\(\pi' := \text{depth-expansion}(v, \pi, D)\);  
while \(\pi \neq \pi'\) do begin  
  \(\pi := \pi'\);  
  \(\pi' := \text{depth-expansion}(v, \pi, D)\);  
end  
return \(\pi\) end;

Procedure depth-expansion(v, \(\pi\), D);
begin  
Let \(\pi\) be \((V_\pi, \emptyset, H_\pi)\);  
Let \(v'\) be a new vertex and [\(v, v'\)] a new variable;  
\(\pi' := (V_\pi \cup \{v'\}, \emptyset, H_\pi \cup \{[v, v']\})\);  
while \(\pi\) is \(\sigma\)-frequent w.r.t. \(D\) do begin  
  \(\pi := \pi'\);  
  \(v := v'\);  
  Let \(v'\) be a new vertex and [\(v, v'\)] a new variable;  
  \(\pi' := (V_\pi \cup \{v'\}, \emptyset, H_\pi \cup \{[v, v']\})\);  
end  
return \(\pi\) end;

Fig. 3. MF-TTP: An algorithm for finding a maximally \(\sigma\)-frequent tag tree pattern.
is that there exists a variable $[u, v] \in H_\pi$ such that $\pi' \preceq \pi \{x := [T_a, [u', v']]\}$ where $x$ is a variable label of $[u, v]$ and $T_a$ is a tree which have only one edge $\{u', v'\}$ labeled with $a \in Tag \cup KW \cup \{\}\}$. The second case is that there exists an edge $\{u, v\} \in E_\pi$ labeled with "?" such that $\pi'$ is a tag tree pattern by replacing the "?" with a label in $Tag \cup KW$. In the both cases, the existence of $\pi'$ leads to a contradiction. Here we show the first case. Let $\pi''$ be a tag tree pattern which is obtained in the procedure $MF-TTP$ just before trying to replace the variable $[u, v]$ with an edge labeled with $a$. Then we have that $\pi''\{x := [T_a, [u', v']]\}$ is not $\sigma$-frequent w.r.t. $D$ and there exists a substitution $\theta$ with $\pi \equiv \pi''\theta$. Since $\pi' \preceq \pi \{x := [T_a, [u', v']]\}$ $\equiv \pi''\theta \{x := [T_a, [u', v']]\}$ $\equiv \pi''\{x := [T_a, [u', v']]\}\theta$, we have $\pi' \preceq \pi''\{x := [T_a, [u', v']]\}$. Since $\pi$ is $\sigma$-frequent w.r.t. $D$, also $\pi''\{x := [T_a, [u', v']]\}$ is $\sigma$-frequent w.r.t. $D$. This is a contradiction.

4 Conclusions

In this paper, we have considered knowledge discovery from semistructured Web documents such as XML files. We have proposed a tag tree pattern which is suited for representing tree structured pattern in such semistructured data. We have given an algorithm for solving Maximally Frequent Tag Tree Pattern Problem. An algorithm for solving All Maximally Frequent Tag Tree Patterns Problem and the experimental results on the algorithm were reported in [5]. Thus, we have shown that a tag tree pattern and the algorithms are useful for knowledge discovery from semistructured Web documents.

References