九州大学学術情報リポジトリ
Kyushu University Institutional Repository

Extraction of Tag Tree Patterns with Contractible Variables from Irregular Semistructured Data

Miyahara, Tetsuhiro
Faculty of Information Sciences,Hiroshima City University

Suzuki, Yusuke
Department of Informatics, Kyushu University

Shoudai, Takayoshi
Department of Informatics, Kyushu University

Uchida, Tomoyuki
Faculty of Information Sciences,Hiroshima City University

他

https://hdl.handle.net/2324/2966

出版情報：Lecture Notes in Computer Science. 2637, pp.430-436, 2003-05. Springer Berlin
バージョン：
権利関係：The original publication is available at www.springerlink.com

KYUSHU UNIVERSITY

Extraction of Tag Tree Patterns with Contractible Variables from Irregular Semistructured Data

Tetsuhiro Miyahara[1], Yusuke Suzuki[2], Takayoshi Shoudai[2], Tomoyuki Uchida[1], Sachio Hirokawa[3], Kenichi Takahashi[1], and Hiroaki Ueda[1]

[1] Faculty of Information Sciences, Hiroshima City University, Hiroshima 731-3194, Japan
{miyahara,takahasi,ueda}@its.hiroshima-cu.ac.jp.,
uchida@cs.hiroshima-cu.ac.jp
[2] Department of Informatics, Kyushu University, Kasuga 816-8580, Japan
{y-suzuki,shoudai}@i.kyushu-u.ac.jp
[3] Computing and Communications Center, Kyushu University, Fukuoka 812-8581, Japan
hirokawa@cc.kyushu-u.ac.jp

Abstract. Information Extraction from semistructured data becomes more and more important. In order to extract meaningful or interesting contents from semistructured data, we need to extract common structured patterns from semistructured data. Many semistructured data have irregularities such as missing or erroneous data. A tag tree pattern is an edge labeled tree with ordered children which has tree structures of tags and structured variables. An edge label is a tag, a keyword or a wildcard, and a variable can be substituted by an arbitrary tree. Especially, a contractible variable matches any subtree including a singleton vertex. So a tag tree pattern is suited for representing common tree structured patterns in irregular semistructured data. We present a new method for extracting characteristic tag tree patterns from irregular semistructured data by using an algorithm for finding a least generalized tag tree pattern explaining given data. We report some experiments of applying this method to extracting characteristic tag tree patterns from irregular semistructured data.

1 Introduction

In this paper, we present a new method for extracting characteristic tag tree patterns from irregular semistructured data which are considered to be positive data. Web documents such as HTML files and XML files have no rigid structure and are called semistructured data. According to the Object Exchange Model [1], we treat semistructured data as tree structured data. To represent a tree structured pattern common to such tree structured data, we propose a *tag tree pattern*, which is a rooted tree consisting of ordered children, structured variables

K.-Y. Whang, J. Jeon, K. Shim, J. Srivatava (Eds.): PAKDD 2003, LNAI 2637, pp. 430–436, 2003.
© Springer-Verlag Berlin Heidelberg 2003

and edges labeled with tags, keywords or wildcards. A variable can be substituted by an arbitrary tree.

Many semistructured data have irregularities such as missing or erroneous data. In the Object Exchange Model, many essential data are represented as leaves or subtrees. So we introduce a new type of variable, called a *contractible variable*, which is regarded as an anonymous subtree in a tag tree pattern and matches any subtree including a singleton vertex. A usual variable, called an *uncontractible variable*, in a tag tree pattern does not match any singleton vertex.

Since a variable can be replaced by an arbitrary tree, overgeneralized patterns explaining given positive data are meaningless. Then, in order to extract meaningful information from irregular or incomplete tree structured data such as semistructured Web documents, we need to find one of the least generalized tag tree patterns. Consider the examples in Fig. 1. Let T_i' be the corresponding tree which is obtained by retaining the edge labels such as "Sec1" or "SubSec3.1" and ignoring the other edge labels in T_i. The tag tree pattern t_1 explains trees T_1', T_2' and T_3'. That is T_1', T_2' and T_3' are obtained from t_1 by substituting the variables of t_1 with trees. Further, t_1 is a least generalized tag tree pattern. The tag tree pattern t_2 also explains the three trees. But t_2 explains any tree with two or more vertices. So t_2 is overgeneralized and meaningless.

A tag tree pattern is different from other representations of tree structured patterns such as in [2,12] in that a tag tree pattern has structured variables which can be substituted by arbitrary trees. Recently, Information Extraction has been extensively studied [3,5]. But most studies are for free-text documents. Information Extraction or wrapper extraction from high-level data such as semistructured data or tables is a hot topic in the field of Web learning or Web mining [4,11]. Our extraction method is an application of the methods for extracting characteristic patterns from tree structured data, which are studied extensively [2,6,7,12].

2 Tag Tree Patterns and Extraction Method

2.1 Ordered Term Trees

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 edge and variable of a term tree is 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 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

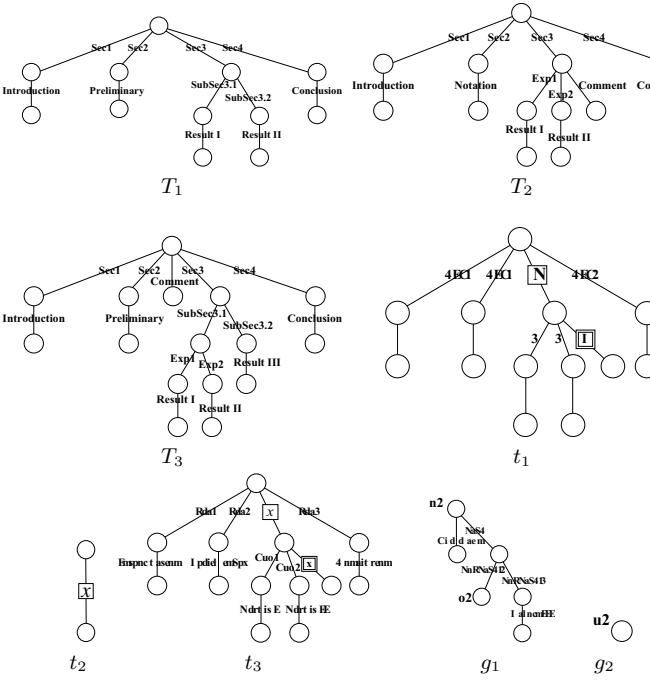


Fig. 1. Tag tree patterns t_1, t_2 and t_3 and trees T_1, T_2 and T_3. An uncontractible (resp. contractible) variable is represented by a single (resp. double) lined box with lines to its elements. The symbol "?" is a wildcard of an edge label. The label inside a box is the variable label of the variable.

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 $[v, v']$.

Let X^c be a distinguished subset of X. We call variable labels in X^c *contractible variable labels*. A contractible variable label can be attached to a variable whose child port is a leaf. We call a variable with a contractible variable label a **contractible variable**, which is allowed to substitute a tree with a singleton vertex, as stated later. We call a variable which is not a contractible variable an uncontractible variable. For a variable $[v, v']$, when we pay attention to the kind of the variable, we denote by $[v, v']^c$ and $[v, v']^u$ a contractible variable and an uncontractible variable, respectively.

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.

An ordered term tree with no variable is called a **ground ordered term tree**, which is a standard ordered tree. $\mathcal{OT}_\Lambda$ denotes the set of all ground ordered term trees whose edge labels are in Λ. $\mathcal{OTT}^c_\Lambda$ denotes the set of all ordered term

trees which have contractible or uncontractible variables, and edge labels in Λ. In this paper, we treat only regular ordered term trees with contractible or uncontractible variables. Therefore we call them **term trees** simply.

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*, if there is a bijection φ from V_f to V_g such that the following conditions (i)-(iv) hold: (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 $[\varphi(u), \varphi(v)] \in H_g$. In particular, $[u, v]^c \in H_f$ if and only if $[\varphi(u), \varphi(v)]^c \in H_g$. (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 $\sigma = [u, u']$ 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. If x is a contractible variable label in X^c, g may be a tree with a singleton vertex u and thus $\sigma = [u, u]$. It is the only case that a tree with a singleton vertex is allowed for a binding. 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. If g is a tree with a singleton vertex, i.e., $u = u'$, then v becomes identical to v' after applying the binding. A *substitution* θ 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 θ, is obtained by applying the all bindings $x_i := [g_i, \sigma_i]$ on f simultaneously.

For example, let t_3 be a term tree described in Fig. 1 and $\theta = \{x := [g_1, [u_1, v_1]], y := [g_2, [u_2, u_2]]\}$ be a substitution, where g_1 and g_2 are trees in Fig. 1. Then the instance $t_3\theta$ of the term tree t_3 by θ is isomorphic to the tree T_3 in Fig. 1.

2.2 Tag Tree Patterns

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 "?", which is a wildcard of an edge label. 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 (i)-(iii) hold: (i) 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. (ii) If the edge label of $\{v, v'\}$ is a keyword, then the edge label of $\{u, u'\}$ is the same keyword. (iii) 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 the following conditions (i)-(iv) hold: (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 $\{\varphi(v), \varphi(v')\}$. (iv) 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. Then the *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 $\Lambda = \Lambda_{Tag} \cup \Lambda_{KW}$.

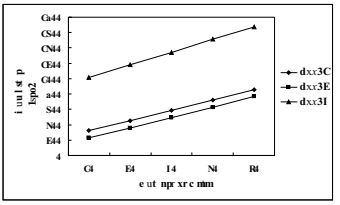
2.3 Extraction of Least Generalized Tag Tree Patterns

We propose a new method for extracting characteristic tag tree patterns from irregular semistructured data which are considered to be positive tree structured data. A tag tree pattern π is a **least generalized tag tree pattern** explaining a given set S of trees which are considered to be positive data, if (i) $S \subseteq L_\Lambda(\pi)$ (π explains S) and (ii) there is no tag tree pattern π' satisfying that $S \subseteq L_\Lambda(\pi') \subsetneq L_\Lambda(\pi)$. The problem for finding a least generalized tag tree pattern for a given set of trees is discussed as the minimal language problem (MINL for short) in the field of computational learning theory [8,10].

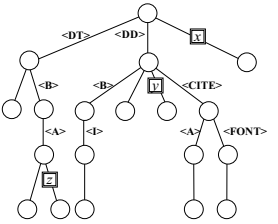
Our extraction method finds a least generalized tag tree pattern explaining a given set of trees S, by using a polynomial time algorithm for solving the MINL problem [10]. First, the algorithm finds a least generalized tag tree pattern t which consists of only uncontractible variables and explains S. Secondly, it finds a tag tree pattern t' which is obtained from t, by replacing a variable in t with an edge labeled with an edge label or a wildcard, or a contractible variable, if the obtained tag tree pattern t' explains S. The algorithm then repeatedly applies the replacing to the tag tree pattern until no more replacing is applicable. Finally it outputs the resulting tag tree pattern. The extraction method uses a polynomial time matching algorithm for deciding whether or not a given tag tree pattern matches a given tree for hypothesis checking [10]. The matching algorithm is an extension of the polynomial time matching algorithms [8,9].

3 Implementation and Experimental Results

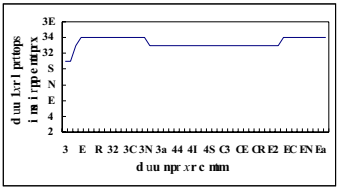
We have implemented the extraction method in Section 2.3, which finds a least general tag tree pattern explaining the given semistructured data. The implementation is by GCL2.2 and on a Sun workstation Ultra-10 clock 333MHz. In Fig. 2 we report some experiments on sample files of semistructured data. In these experiments, an input tree represents a subtree of a parsed tree of an HTML/XML file. The tree structure and HTML/XML 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



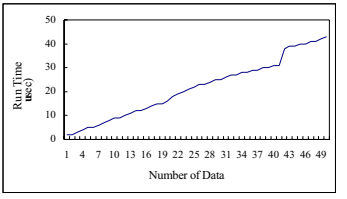
Experiments 1-3



Experiment 4



Experiments 5(a)



Experiment 5(b)

Fig. 2. Experimental results for extracting least generalized tag tree patterns from semistructured data.

tree are converted to the same dummy keyword, in order to pay attention to structures of tags in a parsed tree.

In Exp. 1 to 3 , we made samples of artificial HTML files in order to evaluate our method. The input file for Exp. 1 consists of trees with about 40 vertices. The input file for Exp. 2 consists of 90 % of trees with about 40 vertices and 10 % of trees with about 20 vertices. The input file for Exp. 3 consists of 90 % of trees with about 40 vertices and 10 % of trees with about 70 vertices. The graphs for Exp. 1 to 3 show the running time of the method with varying the number of data for the three experiments. The numbers of vertices of the obtained tag tree patterns of the method are almost same for the three experiments. This shows that the method has robustness for irregularities of sample semistructured data.

In Exp. 4, the sample HTML file is a result of a search engine of a web site with a local search function (http://www.ael.org). The sample file consists of 10 trees with about 18 vertices. A tree in the sample file is a record of bibliographic data. The obtained tag tree pattern in Fig. 2 (Exp. 4) is a least generalized tree pattern explaining the sample file. An edge with no edge label represents an edge with the dummy keyword. So the obtained tag tree pattern is considered to be a wrapper for such tree structured data. In Exp. 5, the sample XML file is from the DBLP bibliographic database (http://dblp.uni-trier.de/xml/dblp.xml). The sample file consists of 50 trees with about 18 vertices. The graphs in Exp. 5(a) and 5(b) show the numbers of vertices of the obtained tag tree pattern and the running time of the method with varying the number of data. The obtained tag tree patterns from the files with 42 or more number of data are the same. This shows that small number of data are sufficient for the method to extract a characteristic tag tree pattern from such bibliographic data.

4 Conclusions

In this paper, we have studied Information Extraction from semistructured data. We have proposed a new method for extracting characteristic tag tree patterns from irregular semistructured data. Also, we have reported some experiments of applying this method to extracting least generalized tag tree patterns from HTML/XML files. 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.2101 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)*, pages 158–174, 2002.
3. C.-H. Chang, S.-C. Lui, and Y.-C. Wu. Applying pattern mining to web information extraction. *Proc. PAKDD-2001, Springer-Verlag, LNAI 2035*, pages 4–15, 2001.
4. W.W. Cohen, H. Mathew, and S.J. Lee. A flexible learning system for wrapping tables and lists in HTML documents. *Proc. WWW 2002*, pages 1–21, 2002.
5. N. Kushmerick. Wrapper induction: efficiency and expressiveness. *Artificial Intelligence*, 118:15–68, 2000.
6. T. Miyahara, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Discovery of frequent tree structured patterns in semistructured web documents. *Proc. PAKDD-2001, Springer-Verlag, LNAI 2035*, pages 47–52, 2001.
7. 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.
8. Y. Suzuki, R. Akanuma, T. Shoudai, T. Miyahara, and T. Uchida. Polynomial time inductive inference of ordered tree patterns with internal structured variables from positive data. *Proc. COLT-2002, Springer-Verlag, LNAI 2375*, pages 169–184, 2002.
9. Y. Suzuki, T. Shoudai, T. Miyahara, and T. Uchida. A polynomial time matching algorithm of structured ordered tree patterns for data mining from semistructured data. *Proc. ILP-2002, Springer-Verlag, LNAI (to appear)*, 2003.
10. Y. Suzuki, T. Shoudai, T. Miyahara, T. Uchida, and S. Hirokawa. Polynomial time inductive inference of ordered term trees with contractible variables from positive data. *Proc. LA Winter Symposium, Kyoto, Japan*, pages 13–1 – 13–11, 2003.
11. T. Taguchi, K. Koga, and S. Hirokawa. Integration of search sites of the World Wide Web. *Proc. of CUM, Vol.2*, pages 25–32, 2000.
12. K. Wang and H. Liu. Discovering structural association of semistructured data. *IEEE Trans. Knowledge and Data Engineering*, 12:353–371, 2000.