A Framework for Data Structure-Guided Extraction of XML Association Rules ^{*}

Juryon Paik¹, Junghyun Nam², SeungCheol Lee¹, and Ung Mo Kim¹

¹ Dept. of Computer Engineering, Sungkyunkwan University, Republic of Korea
quasa277@gmail.com, {eddie, umkim}@ece.skku.ac.kr

² Dept. of Computer Science, Konkuk University, Republic of Korea
jhnam@kku.ac.kr

Abstract. Because of the widespread interest and use of semi-structured data in XML format, the discovery of useful information from them is currently one of the main research topics on association rule extraction. Several encouraging approaches to developing methods for mining rules from XML data have been proposed. However, efficiency and simplicity are still barriers for further development due to the combinatorial explosion in the number of tree nodes. What is needed is a clear and simple methodology for extracting the knowledge that is hidden in the heterogeneous tree data. In this paper, we show that association rules can be unveiled and provided from any XML documents using a special data structure, called Simple and Effective Lists Structure (SELS), avoiding the computationally intractable problem in the number of nodes. SELS is flexible and powerful enough to represent both simple and complex structured association relationships in XML data.

Keywords: XML mining, Association rules, XML association rule, Tree structured data.

1 Introduction

Data Mining, one of the fast growing computer science fields, is used to extract *interesting* knowledge from rich sources of data stored in data repositories. As one of many ways to represent the knowledge, association rules [1] are effective in describing interesting relations in massive amounts of data.

Currently, XML [11] is being used to encode data virtually all areas of Internet applications. The ability to extract knowledge from them becomes increasingly important. Due to the inherent flexibilities of XML finding knowledge in XML data is more challenging than from traditional well-structured data. Hence, extracting association rules from the semi-structured XML data still remains a major problem, which was first introduced by Braga et al. [3].

^{*} This work was supported in part by the Ubiquitous Autonomic Computing and Network Project, 21st Century Frontier R&D Program and by the university IT Research Center project (ITRC), funded by the Korean Ministry of Information and Communication.

Under the traditional framework for association rule, the basic unit of data to deal with is database record, and the construct unit of a discovered association rule is item having an atomic value [5]. However, since the structure of XML is a tree, it is required to have counterparts of record and item in association relationships. In the light of flexibility and hierarchy of tree, the construct unit of an XML association rule is usually generated by repeated tree joins which are performed by nodes combinations. The combinatorial time for unit generations becomes an inherent bottleneck of mining XML association rules.

In this paper, we define some fundamental concepts applied to the XML association rules. Furthermore, we present a clear and simple methodology for mining association rules from XML trees, which is guided by a special data structure, called Simple and Effective List Structure (SELS). The proposed structure not only reduces significantly the number of combinatorial times for obtaining desirable rules, but also simplifies greatly the extraction processes. To the best of our knowledge, this paper is the first attempt: 1) to define formally association rules from the tree-based data, 2) to provide a data structure model for uncovering XML association rules, and 3) to propose techniques for the data structure-guided extracting of association rules from large XML data.

2 Related Work

Since the problem of extracting association rules was first introduced in [1] upon retail databases, a large amount of work has been done in various directions. The famous Apriori algorithm for extracting association rules was published independently in [2] and in [9]. Then, a number of algorithms for extracting association rules from multivariate data have been proposed [6,8]. Because a huge part of the data available on the Internet is represented in XML, a couple of proposals to exploit XML within data mining have been suggested. Singh et al. [7] proposed to mine association rules that relate structural data values to concepts extracted from unstructured and/or semi-structured data. However, they do not seem suitable for discovering association rules from XML documents. Because their approach is not for the context of native XML documents and it considers just strict parent-child relations. Recently, tools for extracting association rules from XML documents have been proposed in [4,10], but both of them are approaching from the view point of a XML query language. This causes the problem of language-dependent association rules mining.

In this paper, we focus on rules detection from a collection of XML documents currently prevailed over the web. Each of XML documents corresponds to a database record, and possesses a tree structure. Accordingly, we extend the notion of associated item to an XML tree, and build up associations among trees rather than items.

3 Preliminaries

3.1 Association Rules for Traditional Data

Association rule is one of data mining technique to discover interesting rules or relationships among attributes in databases. It was first introduced by Agrawal et al. [1] to analyze customer habits in market basket transactions. It is an implication of the form $X \Rightarrow Y$, where the rule *body* X and *head* Y are subsets of the set $\mathcal{I}$ of *items* ($\mathcal{I} = \{I_1, I_2 \dots I_n\}$) within a set of *transactions* $\mathcal{D}$ and $X \cap Y = \phi$. A rule $X \Rightarrow Y$ states that the transactions T ($T \in \mathcal{D}$) that contain the items in X ($X \subset T$) are likely to contain also the items in Y ($Y \subset T$). Association rules are characterized by two measures: the *support*, which measures the percentage of transactions in $\mathcal{D}$ that contain both items X and Y ($X \cup Y$); the *confidence*, which measures the percentage of transactions in $\mathcal{D}$ containing the items X that also contain the items Y . More formally, given the function $\text{freq}(X, \mathcal{D})$, which denotes the percentage of transactions in $\mathcal{D}$ containing X , we define: $\text{sup}(X \Rightarrow Y) = \text{freq}(X \cup Y, \mathcal{D})$, $\text{conf}(X \Rightarrow Y) = \frac{\text{freq}(X \cup Y, \mathcal{D})}{\text{freq}(X, \mathcal{D})}$. The problem of mining association rules from a set of transactions $\mathcal{D}$ consists of generating all the association rules that have support and confidence greater than two user-defined thresholds: minimum support (*min_sup*) and minimum confidence (*min_conf*).

3.2 Association Rules for Tree-Structured XML Data

XML represents data as trees, and makes no requirement that the trees be balanced. Indeed, XML is remarkably free-form, with the only requirements being that 1) the root is the unique node denoting a whole document, 2) the other internal nodes are labeled by tags, and 3) the leaves are labeled by contents or attributes of tags. Thus, XML tree is often called rooted labeled tree. For brevity, in the remaining of this paper, we call a rooted labeled tree as simply a tree. Let $T = (N, E)$ be a labeled rooted tree where N is a set of labeled nodes and E is a set of edges.

Definition 1. Given a tree $T = (N, E)$, we say that a tree $F = (N_F, E_F)$ is a *fragment* of T , denoted as $F \preceq T$, if and only if (i) $N_F \subseteq N$, (ii) for all edges $(u, v) \in E_F$, u is an ancestor of v in T , and (iii) the label of any node $v \in N_F$ is preserved in T .

Intuitively, as a fragment in this paper, any fragment F must not break the ancestor-descendant relationship among the nodes in a tree T .

Definition 2. The basic construct unit of XML association rule is referred to as *tree-structured item*, shortly *titem*. Any fragment is eligible for a titem because the whole XML document consists of several fragments and the structure of fragment is tree.

Assume that $\mathcal{D} = \{T_1, T_2 \dots T_n\}$ is a collection of XML documents and $|\mathcal{D}|$ means the number of documents in $\mathcal{D}$. Let $\mathcal{F} = \{F_i, i > 0 \mid F_i \preceq T_j, \text{ for } 0 < j \leq n\}$ be a total set of fragments and $\mathcal{I} = \{I_1, I_2 \dots I_m\}$ be a set of titems, where $\mathcal{I} \subseteq \mathcal{F}$.

Fig. 1. Extraction of association rules from heterogeneous database $\mathcal{D}$

Definition 3. An XML association rule is an implication of the form $X \Rightarrow Y$, which satisfies following two conditions: (i) $X \in \mathcal{I}, Y \in \mathcal{I}$, (ii) $(X \not\subseteq Y) \wedge (Y \not\subseteq X)$.

Different from traditional association rules where associated items are usually denoted using simple tuple-based data, the items in XML association rules have hierarchical tree structures (titem), as indicated by the first clause of the definition. The second clause of the definition requires that titems are independent each other in an XML association rule.

Lemma 1. If one of X or Y ($X \Rightarrow Y$) is assigned the entire tree T ($T \in \mathcal{D}$) as a titem, the other should be assigned among titems $\mathcal{T}'$ which is a subset of fragment $\mathcal{F}'$, where $\mathcal{F}' = \mathcal{F} - \{F_i, i > 0 \mid F_i \preceq T\}$.

Proof. Let assume the body X is a tree T_2 in $\mathcal{D}$ and the head Y is F_{T_2} , one of fragments of T_2 . The association rule $T_2 \Rightarrow F_{T_2}$, however, is against Definition 3. Since F_{T_2} is a fragment of T_2 , it is included in T_2 . Hence, they are not independent each other, which is the second condition of Definition 1. Therefore, any fragment of a tree T in $\mathcal{D}$ cannot be X or Y , if one of X or Y is a tree T .

Definition 4. Given $\mathcal{D}$, the *support* and *confidence* of an XML association rule $X \Rightarrow Y$ are defined as: $\text{sup}(X \Rightarrow Y) = \text{freq}(X \cup Y, \mathcal{D}) = \frac{|\mathcal{D}_{X \cup Y}|}{|\mathcal{D}|}$, $\text{conf}(X \Rightarrow Y) = \frac{\text{freq}(X \cup Y, \mathcal{D})}{\text{freq}(X, \mathcal{D})} = \frac{|\mathcal{D}_{X \cup Y}|}{|\mathcal{D}_X|}$, where $\mathcal{D}_{X \cup Y} = \{T_i \mid I_j \in (X \cup Y) \text{ and } I_j \preceq T_i, T_i \in \mathcal{D}\}$, and $\mathcal{D}_X = \{T_i \mid I_j \in X, \text{ and } I_j \preceq T_i, T_i \in \mathcal{D}\}$.

Let us consider three XML trees in Fig. 1(a), where each different information is provided from various data sources, e.g., ordering information (Order) and the detailed information of a product (Details). Consider the extraction of frequently occurring association rules in $\mathcal{D}$ that if a customer purchases HP Desktop, usually s/he is likely to buy HP Printer depicted in Fig. 1(b). The support of an association rule $\text{freq}(X \cup Y, \mathcal{D})$ is 0.66 because both titems X and Y appear together in T_1 and T_3 out of three, i.e., in 66% of the cases. Likewise, $\text{freq}(X, \mathcal{D})$ is 1 since the titem X is included in all of three trees, i.e., in 100% of the cases. Therefore, we can compute the confidence of XML association rule as: $\frac{\text{freq}(X \cup Y, \mathcal{D})}{\text{freq}(X, \mathcal{D})} = \frac{0.66}{1}$, which returns 0.66; it means HP Desktop is usually purchased at the rate of 66% in computer products together with HP Printer.

4 Extracting Association Rules Using SELS

4.1 Design and Construction of SELS

The flexibility of tree data comes from its hierarchical structure. The hierarchy, however, causes technical problems to exploit or extract some valuable partial data from the entire dataset. After some careful examination, we believe that the bottleneck of hierarchy is at fragments generation and item tests. If one can only generate fragments which guarantee being proper items, the rule mining performance can be substantially improved. Simple and Effective List Structure (SELS) is devised to enable avoiding useless fragments generation, reduce computational complexity and enhance fast extraction of desired fragments. It is a set of lists which includes all information of characteristics of trees, such as label, node id, tree id, and parent/ancestor relationships among nodes.

Definition 5. Given a tree database $\mathcal{D}$, node ids and tree ids are members of a single list under each unique node label. The list is operationally divided into two parts; one is for identifying list itself between a number of lists, and the other is for storing all relevant frequency information in a tree database. The former deals with a label of node and node ids being assigned the label. The latter is mostly about frequency of a label; tree id is kept, but there is a special filed for hierarchy of nodes assigned the label in trees; parent node ids are stored. Because complete single list is composed of finally node ids and tree ids, the list is called, *Node and Tree List (NaTL)*. In addition, the leading part for distinguishing each NaTL is named *head of NaTL*, shortly ntl_{head} and the trailing part for counting frequency of each label is *body of NaTL*, ntl_{body} .

Every ntl_{head} consists of three fields in which a label, node ids assigned the label, and pointer to its corresponding ntl_{body} are included. As described in Definition 5, each ntl_{head} is unique due to the label in it thus, the label is called *key* of SELS for database $\mathcal{D}$. The corresponding ntl_{body} is a linkedlist of several elements. A single element is composed of one pointer field for next element and two id fields; one is for tree which contains node(s) assigned the label in ntl_{head} and the other is for parent nodes of the node(s) in the tree. The number of elements in a ntl_{body} means a total number of trees in $\mathcal{D}$ that assign the label to their node(s). In other words, the count of elements gives the information of the frequency of a label with respect to the database $\mathcal{D}$, and represents size of NaTL denoted as $|ntl|$. The size plays a key role to filter out some undesirable NaTLs from SELS.

4.2 Method of Refining SELS

Initially constructed SELS is composed of NaTLs of every unique node label. Therefore, it is highly possible that some NaTLs having not frequently occurred node labels are included in the SELS.

Definition 6. Given a *min_sup*, an NaTL is *frequent* if and only if size of its body $|\text{ntl}_{\text{body}}|$ is greater than or equal to $|\mathcal{D}| \times \text{min_sup}$, otherwise *unfrequent*.

After filtering out unfrequent NaTLs, only frequent NaTLs whose body sizes satisfy the threshold are remained in SELS. This firstly filtered SELS is called *shallowly-frequent SELS*, shortly *sfS*. It is similar to having only frequent 1-size subtrees which have only one node labeled by a key of SELS. In traditional generate-and-test approaches, each frequent 1-size subtree is grown by joining of subtrees or enumeration of subtrees. In our approach, however, due to SELS, there is no need to grow subtrees. Instead, we verify each parent node in elements of ntl_{body} if it is unfrequent or not. Because all nodes composing a tree must be frequent in order the tree has to be frequent, as a first step to expand a single frequent node, the frequency of every parent node related to it should be considered. However, it is not guaranteed in *sfS*. This is the reason why the SELS is called shallowly-frequent after filtration of some NaTLs.

To refine *sfS* the unfrequent NaTLs are required. A *candidate_hash_table* is generated containing them, and it is built simultaneously with *sfS*. In the tabel, node ids are keys of table, a labeling function plays a role of hash function, node labels are indexes, and ntl_{body} s of unfrequent $\text{ntl}_{\text{heads}}$ are records of indexes. Because every node is assigned only one label and it is not changed until artificial manipulation is performed, we assume that the labeling function is same as a hashing function. After finishing the refinement, the hash table is removed from the memory. The purpose of second refinement of SELS to deal with every parent node in elements and to make *sfS* contain all frequent nodes. We call this acquired SELS, *deeply-frequent SELS*, abbreviated *dfS*. To obtain *dfS* we proceed as follows: (1) a parent node in an element is verified by the *candidate_hash_table* to detect if the node is assigned by unfrequent label or not. (2) If so, the node is marked ‘replace’ and its record is retrieved to search a node id assigned by frequent node labels. (3) Step (1) and (2) continues until the node id assigned by any frequent node label is found. (4) The original parent node id is replaced by the found node which is actually an ancestor node of the original parent node. (5) Through step (1) to (3), if no any other nodes are frequent, the original parent node id is replaced by NULL. After applying above refining procedure over all parent nodes in elements, *dfS* holds only frequent node ids and associated labels (keys of *dfS*).

Given a *dfS*, both $\text{ntl}_{\text{heads}}$ and ntl_{body} s are associated together depending on ancestor-descendant relationship. Note that every node id is only assigned by one of the keys of *dfS*. At least one big labeled tree ¹ is derived from *dfS*. We call this derived tree as *Minimum Support Satisfying Trees* (MSST). During the construction of MSST, each edge has its own count which represents how much often it is occurred. If the frequency does not satisfy $\sigma \times |\mathcal{D}|$, the corresponding edge is deleted from MSST. After scanning all edges of MSST, tree(s) which is(are) satisfying *min_sup* is(are) derived. The stepwise refinement of $\text{SELS}_{\mathcal{D}}$ and its derived MSST are illustrated in Fig. 2. Due to the lack of space, we only show the prefix of labels in SELS and MSST; only heads of NaTLs and labels in MSST .

¹ To avoid derive graphs we duplicate nodes shared by different parents/ancestors.

Fig. 2. Stepwise Refinement of SELS and Derived Trees

4.3 Correlating Concrete Contents with Structures

Let (h_i, e_{h_i}) and (h_j, e_{h_j}) be two NATLs in MSST. The sizes of two elements, $|e_{h_i}|$ and $|e_{h_j}|$, are greater than $|\mathcal{D}| \times \text{min_sup}$. Let t_i and t_j be the all tree indexes included in e_{h_i} and e_{h_j} , respectively. The sizes of each t_i and t_j are exactly same as the sizes of each e_{h_i} and e_{h_j} . We denote a path between them by $\mathcal{P}_{ij} = (h_i, h_j)$ where h_i is an ancestor of h_j by Definition 1. Let $\mathcal{I} = \{I_1, \dots, I_m\}$ be a set of titems. Assume that $I_1 = \mathcal{P}_{ij} = (h_i, h_j)$ and $I_2 = \mathcal{P}_{pq} = (h_p, h_q)$ be titems. The confidence of $I_1 \Rightarrow I_2$ is computed as:

$$\text{conf}(I_1 \Rightarrow I_2) = \frac{\text{freq}(I_1 \cup I_2, \mathcal{D})}{\text{freq}(I_1, \mathcal{D})} = \frac{|t_i \cap t_j \cap t_p \cap t_q|}{|t_i \cap t_j|} \quad (1)$$

Theorem 1. Given a *min_sup* and MSST, all of XML association rules conforming to **any min_conf** are extracted.

Proof. Let (h_i, e_{h_i}) , (h_j, e_{h_j}) and (h_k, e_{h_k}) be three NATLs in MSST. Assume that $|t_i \cap t_k| = \beta$, and $|t_i \cap t_j \cap t_k| = \gamma$. Given *min_sup* and *min_conf* are s_m and c_m , respectively, where $0 < s_m, c_m \leq 1$. Let x and y be a tree index in t_j and t_k , respectively, for $x \neq y$.

- **item_a** ($x \in t_i \wedge y \notin t_i$): The h_i and h_j forms $\mathcal{P}_{ij}$. It is a titem I_{ij} . The support of item_a is definitely greater than or equal to s_m due to the characteristics of MSST.
- **item_b** ($y \in t_i \wedge x \notin t_i$): The h_i and h_k forms $\mathcal{P}_{ik}$. It is a titem I_{ik} . The support of item_b is also greater than or equal to s_m .
- **Association Rule** The confidence of an implication of the form $I_{ij}(\text{item}_a) \Rightarrow I_{ik}(\text{item}_b)$ is computed by using the equation (1):

$$\text{conf}(I_{ij} \Rightarrow I_{ik}) = \frac{\gamma}{\beta} = \begin{cases} \text{rules with confidence} \geq c_m & \text{if } \gamma \geq \beta \times c_m, \\ \text{rules with confidence} < c_m & \text{if } \gamma < \beta \times c_m. \end{cases}$$

5 Conclusion

This paper has presented a framework for data structure-guided association rules extraction from XML trees. We are currently finishing touches of practical algorithms for our approach and evaluating the performance results.

References

1. Agrawal, R., Imielinski, T., Swami, A. N.: Mining Association Rules between Sets of Items in Large Databases. Proc. of the ACM SIGMOD Intl. Conf. on Management of Data (1993) 207–216
2. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. Proc. of the 20th Intl. Conf. on VLDB (1994) 478–499
3. Braga, D., Campi, A., Klemettinen, M., Lanzi, P. L.: Mining Association Rules from XML Data. Lecture Notes in Computer Science, Vol. 2454. Springer-Verlag, Berlin Heidelberg New York (2002) 21–30
4. Braga, D., Campi, A., Ceri, S., Klemettinen, M., Lanzi, P. L.: A Tool for Extracing XML Association Rules. Proc. of the 14th IEEE Intl. Conf. on Tools with Artificial Intelligence (2002) 57–64
5. Feng, L., Dillon, T., Weigand, H., Chang, E.: An XML-Enabled Association Rule Framework. Lecture Notes in Computer Science, Vol. 2736. Springer-Verlag, Berlin Heidelberg New York (2003) 88–97
6. Han, J., Fu, Y.: Discovery of Multiple-Level Association Rules from Large Databases. Proc. of the 21st Intl. Conf. on VLDB (1995) 420–431
7. Singh, L., Scheuermann, P., Chen, B.: Generating Association Rules from Semi-Structured Documents Using an Extended Concept Hierarchy. Proc. of the 6th Intl. Conf. on Information and Knowledge Management (1997) 193–200
8. Srikant R., Agrawal, R.: Mining Generalized Association Rules. Proc. of the 21st Intl. Conf. on VLDB (1995) 409–419
9. Toivonen, H.: Sampling Large Databases for Association Rules. Proc. of the 22th Intl. Conf. on VLDB (1996) 43–52
10. Wan, J. W. W., Dobbie, G.: Extracting Association Rules from XML Documents Using XQuery. Proc. of the 5th ACM Intl. Workshop on Web Information and Data Management (2003) 94–97
11. The World Wide Web Consortium (W3C). Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation, <http://www.w3.org/TR/2004/REC-xml-20040204/>, 2004.