Schema Mining: Finding Structural Regularity among Semistructured Data

P.A Laur[1], F. Masseglia[1,2], and P. Poncelet[1]

[1] LIRMM UMR CNRS 5506, 161, Rue Ada, 34392 Montpellier Cedex 5, France
Email: {laur,massegli,poncelet}@lirmm.fr
[2] Laboratoire PRiSM, Université de Versailles, 45 Avenue des Etats-Unis, 78035 Versailles Cedex, France

Abstract. Motivated by decision support problems, data mining has been extensively addressed in the few past years. Nevertheless, the proposed approaches mainly concern flat representation of the data and to the best of our knowledge, not much effort has been spent on mining interesting patterns from such structures. In this paper we address the problem of mining structural association of semistructured data, or in other words the discovery of structural regularities among a large database of semistructured objects. This problem is much more complicated than the classical association rule one, since complex structures in the form of a labeled hirearchical objects partially ordered has to be taken into account.

1 Introduction

Motived by decision support, the problem of mining association rules or sequential patterns has recently received a great deal of attention [AIS93,AS94,BMUT97,FPSSU96,SON95,Toi96,SA96,MCP98]. Nevertheless, the proposed approaches mainly concern flat representation of the data and to the best of our knowledge, not much effort has been spent on mining interesting structures from such a data [WL97,WL99]. In fact, this problem is much more complicated than the classical association rule or sequential patterns, since complex structures in the form of a labeled hirearchical objects partially ordered has to be taken into account.

In this paper, we present a new algorithm, called ScM (Schema Mining) for mining structural regularities of semistructured data. The rest of this paper is organized as follows. In section 2, the problem is stated and illustrated. The algorithm ScM is described in section 3. Related work is briefly presented in section 4. Finally section 5 concludes the paper.

2 Problem statement

In this section we give the formal definition of the structural association mining problem. First we formulate the semistructured data which widely

resumes the formal description of the Object Exchange Model (OEM) defined for representing structured data [AQM+97]. Second we look at the structural association mining problem in detail.

The data model that we use is based on the OEM model designed specifically for representing semistructured data. We assume that every object o is a tuple consisting of an *identifier*, a *type* and a *value*. The *identifier* uniquely identifies the object. The type is either *complex* or some identifier denoting an atomic type (like integer, string, gif-image, etc.). When type is complex then the object is called a *complex object* and value is a set (or list) of *identifiers*. Otherwise the object is an *atomic object*, and its value is an atomic value of that type. As we consider set semantics as well as list semantics, we use a circle node to represent an identifier of a set value and a squared node to represented an identifier of a list value.
We can thus view OEM data as a graph where the nodes are the objects and the labels are on the edges. In this paper we assume that there is no cycle in the OEM graph. We also require that: (i) *identifier(o)* and *value(o)* denotes the identifier and value of the object o; (ii) *object(id)* denotes the unique object with an identifier *id*; (iii) Each atomic object has no outgoing edges. (iv) If two edges connect the same pair of nodes in the same direction then they must have different label. We thus assume that we are provided with a labeling function $F_E : E \rightarrow L_E$ where L_E is the domain of edge labels. A single path in the graph is an alternating sequence of objects and labels $< o_1 l_1 o_2 ... o_{k-1} l_{k-1} o_k >$ beginning and ending with objects, in which each label is incident with the two nodes immediately preceding and following it. The number of labels from the source object to the target node in a path, k, is the length of the path. As we consider nested structures, we can consider that the length is similar to the nested level of the structure. Let P_k the set of all paths p where the length of p is k. We now consider multiple path defined as follows: a *multiple path* (or *path* for short) is a set of single paths such as the source object is the same in all the single paths. The length of the path is the maximal length of all single paths. As we are only interested in structural regularity, in the following we do not consider atomic values anymore and we use symbol $\perp$ in order to denote an atomic value in the graph.

A path p_m is a *sub-path* of a path p_n if p_m is included in p_n. We define the inclusion in the following way: A path $p_a =< o_{a_1} l_{a_1} o_{a_2} ... o_{a_{k-1}} l_{a_{k-1}} o_{a_n} >$ is included in another path $p_b =< o_{b_1} l_{b_1} o_{b_2} ... o_{b_{k-1}} l_{b_{k-1}} o_{b_m} >$ if and only if there exists integers $i_1 < i_2 < ... < i_n$ such that $o_{a_1} = o_{b_{i_1}}$, $o_{a_2} = o_{b_{i_2}}, ..., o_{a_n} = o_{b_{i_n}}$.

Example 1 The path $\{category: \perp, name: \perp, address: \{street: \perp, city: \perp, zipcode: \perp\}, price: \perp\}$ is not a sub-path of $\{category: \perp, name: \perp, address: \{street: \perp, city: \perp, zipcode: \perp\}, nearby: \{price: \perp, category: \perp, name: \perp\}, nearby: \{category: \perp, name: \perp, address: \perp, address: \perp, price: \perp, zipcode: \perp\}\}$ since the label *price* is not at the same level in the graph.

Let DB be a set of transactions where each transaction T consists of transaction-id and a multiple path embedded in the OEM graph and involved in the transaction. A support value (*supp(s)*) for a multiple path in the OEM graph gives its number of actual occurrences in DB. In other words, the support of a path is defined as the fraction of total data sequences that contain p. In order to decide whether a path is frequent or not, a minimum support value (*minSupp*) is specified by user, and the multiple path expression is said *frequent* if the condition $supp(s) \geq minSupp$ holds.

Given a database DB of customer transactions the problem of regularity mining, called *Schema mining*, is to find all maximal paths occurring in DB whose support is greater than a specified threshold (minimum support). Each of which represents a *frequent path*. From the problem statement, discovering structural association sequential patterns resembles closely to mining association rules [AIS93,AS94,BMUT97,FPSSU96,SON95,Toi96] or sequential patterns [AS95,SA96]. However, elements of handled association transactions have structures in the form of a labeled hirearchical objects, and a main difference is introduced with partially ordered references. In other word we have to take into account complex structures while in the association rules or sequential patterns elements are atomics, i.e. flat sets or lists of items.

3 ScM algorithm

In this section we introduce the ScM algorithm for mining structural association of semistructured data in a large database. We split the problem into the following phases:

1. **Mapping phase:** The transaction database is sorted with transaction id as a major key and values embedded in a set-of are sorted according to the lexicographic order. In order to efficiently find structural regularity among path expressions, each path expression is mapped in the following way. If the object value is a part of a *set-of* value, then we merge an 'S' to the label (resp. a 'L' for a *list-of* value). Furthermore, in order to take into account the level of the label into the transaction, we append to the label an integer standing for the level of the label in the nested structure. When two labels occur at the same level and if the first one directly follows the second one, they are grouped together in the same set otherwise they form a new set of labels. The ordering of the *list-of* value is taken into account by creating a new set of labels. The "composite" transaction which results from the union of such sets obtained from original transaction describe a *sequence of label* and the ordering of such a sequence may be seen as the way of navigating through the path expression.
2. **Mining Phase:** The General algorithm is used to find the frequent paths in the database.

Example 2 In order to illustrate the mapping phase, let us consider the following transactions $[t_1, \{a :< e : \{f : \perp, b : \perp\}, d :< g : \perp, h : \perp >>, c : \perp\}]$.

According to the previous discussion, the transaction t_1 is mapped into the following: $[t_2, (Sa_1)\ (Le_2)\ (Sf_3Sb_3)\ (Lg_3)(Lh_3)(Ld_2)\ (Sc_1)]$.

Our approach for mining structural regularities fully resumes the fundamental principles of the classical association rule problem [AIS93]. At each step k, the DB is browsed for counting the support of current candidates (procedure VERIFY_CANDIDATE). Then the frequent sequence set L_k can be built. From this set, new candidates are exhibited for being dealt at the next step (procedure CANDIDATE-GENERATION). The algorithm stops when the longest frequent paths, embedded in the DB are discovered thus the candidate generation procedure yields an empty set of new candidates. Due to lack of space we do not provide the full description of the algorithms but rather an overview.
In order to improve the efficiency of the generation as well as the organization of candidates paths we use a prefix tree structure close to the structure used in [MCP98] and improved in [MPC99]. At the k^{th} step, the tree has a depth of k. It captures all the candidate k-path as well as all frequent paths: once candidates are counted and determined frequent they remain in their proper position in the tree and become frequent paths. Any branch, from the root to a leaf stands for a candidate path, and considering a single branch, each node at depth l ($k \geq l$) captures the l^{th} label of the path. Furthermore along with a label, a terminal node provides the support of the path from the root to the considered leaf (included). Sets cutting is captured by using labelled edges. In order to illustrate this difference, let us consider two nodes, one being the child of the other. If the labels emboddied in the node originally occurred in the same set, the edge linking the nodes is labelled with a dashed link otherwise it is labelled with a line.
In order to improve the efficiency of retrievals when candidates paths are compared to data-sequences, we assume that each node is provided with two hashtables. The fist one is used to consider labels occurring in the same set while the other one is used for taking into set cutting.

Candidate_Generation
The candidate generation builds, step by step, the tree structure. At the beginning of step 2, the tree has a depth of 1. All nodes at depth 1 (frequent labels) are provided with children supposed to capture all frequent labels. This means that for each node, the created children are a copy of its brothers. When the k^{th} step of the general algorithm is performed, the candidate generation operates on the tree of depth k and yields the tree of depth k+1. For each leaf in the tree, we must compute all its possible continuations of a single item. Exactly like at step 2, only frequent items can be valid continuations. Thus only paths captured by nodes at depth 1 are considered. Associated leaves, by construction of the tree, are brothers.

Verify_Candidate
In fact, to discovery candidates paths included in a data sequence, the data-sequence is progressively browsed beginning with its first label, then with the

second, and so on. From the item, a navigation is performed through the tree until a candidate path is fully detected. This is done by comparing successively following items in the data sequence with the children of the current node in the prefix-tree structure. Finally when a leaf is reached, the examined path support the candidate and its counting value must be incremented.

The SCM algorithm is implemented using Gnu C++. Due to lack of space we do not report experiments about our approach but interested reader may refer to [Lau00].

4 Related Work

To the best of our knowledge there is few work on mining such a structural regularity in a large database. Nevertheless, our work is very related to the problem of mining structural association of semistructured data proposed in [WL97,WL99] where a very efficient approach for mining such regularities is provided. The author propose a very efficient approach and they give some pruning strategies in order to improve the candidate generation. Nevertheless our work has some important differences. Unlike their approach we are insterested in all structures embedded in the database while they are interested in mining tree expression which are defined as a path from the root of the OEM graph to the atomic values. According to this definition of the tree expression they cannot find regularities such as $identity$: $\{address$: $< street$: $\perp$, $zipcode$: $\perp >\}$ (Cf. section 2). In fact, when parsing the database in order to find frequent tree, they are only provided with maximal tree and when only a part of the tree is frequent it is not discovered.
Discovering structural information from semistructured data has been studied in some interesting works. In this context, they are some propositions for extracting the typing of semistructured data. For instance in [NAM98] they extract the structure implicit in a semistructured schema. This approach is quite different from our since we address the repetition of a structure in a schema. Nevertheless we assume that our transaction database has been preprocessed using such a technique in order to provide an OEM graph where the raw data has been casted in terms of this structure.

5 Conclusion

In this paper we present the SCM approach for mining structural association of semistructured objects in a large database. Our approach is based on a specific generation of candidates efficiently performed using a new structure as well as a very well adapted mapping of the initial database.

References

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD Conference*, pages 207–216, Washington DC, USA, May 1993.

[AQM+97] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J.L Wiener. The Lorel Query Language for Semi-Structured Data. *International Journal on Digital Libraries*, 1(1):68–88, April 1997.

[AS94] R. Agrawal and R. Srikant. Fast Algorithms for Mining Generalized Association Rules. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)*, Santiago, Chile, September 1994.

[AS95] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, Tapei, Taiwan, March 1995.

[BMUT97] S. Brin, R. Motwani, J.D. Ullman, and S. Tsur. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In *Proceedings of the International Conference on Management of Data (SIGMOD'97)*, pages 255–264, Tucson, Arizona, May 1997.

[FPSSU96] U.M. Fayad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1996.

[Lau00] P.A. Laur. Schema Mining : vers une approche efficace. In *Technical Report (in french), LIRMM, France*, 2000.

[MCP98] F. Masseglia, F. Cathala, and P. Poncelet. The PSP Approach for Mining Sequential Patterns. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98), LNAI, Vol. 1510*, pages 176–184, Nantes, France, September 1998.

[MPC99] F. Masseglia, P. Poncelet, and R. Cicchetti. An Efficient Algorithm for Web Usage Mining. *Networking and Information Systems Journal*, 2(5-6), December 1999.

[NAM98] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting Schema for Semistructured Data. In *Proceedings of the International Conference on Management of Data (SIGMOD'98) - SIGMOD Record 27(2)*, 1998.

[SA96] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96)*, pages 3–17, Avignon, France, September 1996.

[SON95] A. Savasere, E. Omiecinski, and S. Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases. In *Proceedings of the 21 st International Conference on Very Large Databases (VLDB'95)*, pages 432–444, Zurich, Switzerland, September 1995.

[Toi96] H. Toivonen. Sampling Large Databases for Association Rules. In *Proceedings of the 22nd International Conference on Very Large Databases (VLDB'96)*, September 1996.

[WL97] K. Wang and H.Q. Liu. Mining Nested Association Patterns. In *Proceedings of the SIGMOD'97 Workshop on Research Issues on Data Mining and Knowledge Discovery*, May 1997.

[WL99] K. Wang and H. Liu. Discovering Structural Association of Semistructured Data. *IEEE Transactions on Knowledge and Data Engineering*, 1999.