Sequential Pattern Mining for Structure-Based XML Document Classification

Calin Garboni[1,2], Florent Masseglia[2], and Brigitte Trousse[2,⋆]

[1] West University of Timisoara, Romania
[2] INRIA Sophia Antipolis, AxIS Research Team 2004,
route des Lucioles, BP93,
F-06902 Sophia Antiplis Cedex, France
Surname.Name@inria.fr
http://www-sop.inria.fr/axis/

Abstract. This article presents an original supervised classification technique for XML documents which is based on structure only. Each XML document is viewed as an ordered labeled tree, represented by his tags only. Our method has three steps. After a cleaning step, we characterize each predefined cluster in terms of frequent structural subsequences. Then we classify the XML documents based on the mined patterns of each cluster.

1 Introduction

This work is in the context of the Document Mining track[1] of the Inex[2] Initiative. The objective is to bridge the gap between Machine Learning and Information Retrieval. The Document Mining challenge focused on classification and clustering of XML documents using only their structure or using both their structure and their content. In our work, we use only the structure information of the XML documents. Our goal is to show the relevance of using only the structure information in order to detect different "structural families of documents". Each XML document has a single label corresponding to the structural source of the document. Our work consists of characterising each predefined cluster in terms of frequent "structural" patterns and then classifying ordered labeled trees.

Section 2 introduces the used basic theoretical concepts of sequential pattern mining. Section 3 describes how to characterize each cluster of XML documents in terms of frequent structural subsequences. After presenting some related works, we present our approach in Section 4. Section 5 describes some experiments and results. Finally some conclusions and perspectives are presented.

⋆ The authors want to thank Sergiu Chelcea for his useful support in the experiments.

[1] URL: http://xmlmining.lip6.fr/Home

[2] Inex: INitiative for the Evaluation of XML Retrieval URL:http://inex.is.informatik.uni-duisburg.de/2005/

N. Fuhr et al. (Eds.): INEX 2005, LNCS 3977, pp. 458–468, 2006.

2 Mining Sequential Patterns

In this section we define the sequential pattern mining problem in large databases and give an illustration. The sequential pattern mining definitions are those given by [1] and [10].

In [1], the association rules mining problem is defined as follows:

Definition 1. Let $I = \{i_1, i_2, ..., i_m\}$, be a set of m literals (*items*). Let $D = \{t_1, t_2, ...t_n\}$, be a set of n transactions ; Associated with each transaction is a unique identifier called its TID and an *itemset* I. I is a *k-itemset* where k is the number of items in I. We say that a transaction T *contains* X, a set of some items in I, if $X \subseteq T$. The *support* of an itemset I is the fraction of transactions in D containing I: $supp(I) = \|\{t \in D \mid I \subseteq t\}\|/\|\{t \in D\}\|$. An *association rule* is an implication of the form $I_1 \Rightarrow I_2$, where I_1, $I_2 \subset I$ and $I_1 \cap I_2 = \emptyset$. The rule $I_1 \Rightarrow I_2$ holds in the transaction set D with confidence c if $c\%$ of transactions in D that contain I_1 also contain I_2. The rule $r : \ I_1 \Rightarrow I_2$ has *support* s in the transaction set D if $s\%$ of transactions in D contain $I_1 \cup I_2$ (i.e. $supp(r) = supp(I_1 \cup I_2)$).

Given two parameters specified by the user, *minsupp* and *minconfidence*, the problem of association rule mining in a database D aims at providing the set of frequent itemsets in D, i.e. all the itemsets having support greater or equal to *minsupp*. Association rules with confidence greater than *minconfidence* are thus generated.

As this definition does not take time into consideration, the sequential patterns are defined in [10]:

Definition 2. A *sequence* is an ordered list of itemsets denoted by $< s_1 s_2 \ldots s_n >$ where s_j is an itemset. The *data-sequence* of a customer c is the sequence in D corresponding to customer c. A sequence $< a_1 a_2 \ldots a_n >$ is a *subsequence* of another sequence $< b_1 b_2 \ldots b_m >$ if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \ldots, a_n \subseteq b_{i_n}$.

Example 1. Let C be a client and S=< (3) (4 5) (8) >, be that client's purchases. S means that "C bought item 3, then he or she bought 4 and 5 at the same moment (i.e. in the same transaction) and finally bought item 8".

Definition 3. The *support* for a sequence s, also called $supp(s)$, is defined as the fraction of total data-sequences that contain s. If $supp(s) \geq minsupp$, with a minimum support value $minsupp$ given by the user, s is considered as a *frequent* sequential pattern.

Example 2. Let us consider the data given in table 1. This can be the result of a pre-processing step performed on raw data from a shop meaning that at time $d1$ (for instance) customer 1 bought item 10. The goal is thus, according to definition 3 and by means of a data mining step, to find the sequential patterns in the file that can be considered as frequent. On of the resulting sequences may be, for instance, `< ( 10 ) ( 30 ) ( 20 ) ( 30 ) >` (with the file illustrated in figure 1 and a minimum support given by the user: 100%).

Table 1. File obtained after a pre-processing step

Client	d1	d2	d3	d4	d5
1	10	30	40	20	30
2	10	30	20	60	30
3	10	70	30	20	30

3 Characterizing a Collection of XML Documents by Frequent Structural Subsequences

In this section, we introduce the principles of structure discovery from a set of XML documents. The idea is similar to the method developed in [5]. Our goal is to extract a schema that will be representative of the whole set of documents. In this context, "representative" will be interpreted as "frequent". In fact, we consider that a frequent sub-tree in a collection of XML documents may be considered as a interesting knowledge regarding this collection. This sub-tree can be exploited as a further DTD (see 4) or may stand for a characteristic of the collection (this is the main idea of our approach and will be detailed in section 5).

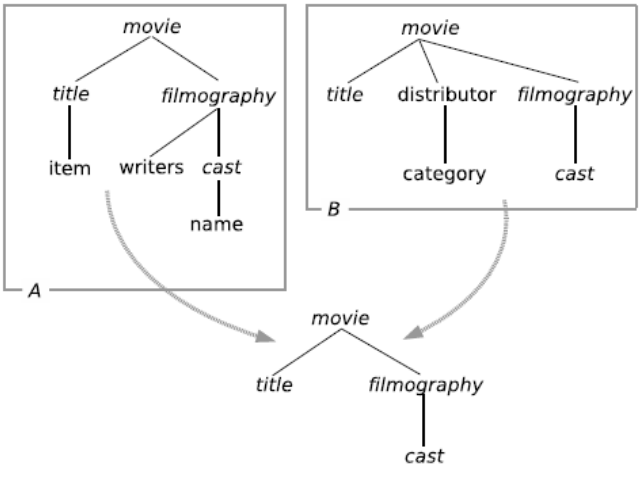

Fig. 1. A frequent sub-tree in a collection of XML documents

Figure 1 gives an illustration of frequent sub-tree mining in a collection of XML documents. Let us consider the documents given in the upper part of figure 1 (respectively labeled "A" and "B"). A frequent sub-tree mining approach is intended to find the sub-tree(s) common to at least $x\%$ (the minimum support) documents of the collection. With the documents labeled "A" and "B" and a minimum support of 100%, the extracted frequent sub-tree is described in the lower part of figure 1. Actually, the tree described by a root node containing

"movie" and having two children ("title" and filmography", which is followed by "cast) is embedded in both "A" and "B". Furthermore, there is no larger frequent sub-tree in this collection.

Our method will rely on structure discovery based on sequential pattern mining. For this purpose, we will use a technique intended to transform any XML tree into a sequence. This technique is described below.

In order to perform such a transformation, the nodes of the XML tree first have to be mapped into identifiers. Then each identifier is associated with its depth in the tree. Finally a depth-first exploration of the tree will give the corresponding sequence. We call this last step the "reduction". The transformation is illustrated by figure 2. The original XML document structure (upper left) is first mapped into a new labelled tree. For instance, the node "filmography" becomes "C_2" wich corresponds to the identifier of "filmography" (C) associated with its depth (2) in the tree. The next step (reduction) aims at writing the corresponding sequence after a depth first navigation in the tree.

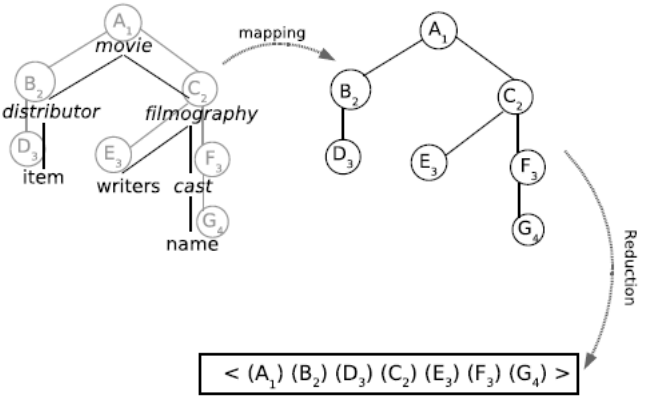

Fig. 2. Transformation of an XML tree into a sequence

Once the whole set of sequences (corresponding to the XML documents of a collection) is obtained, a traditional sequential pattern extraction algorithm is able to extract the frequent sequences. Those sequences, once mapped back into trees, will give the frequent sub-trees embedded in the collection.

4 Related Work

We present here previous works related to our approach. Basically, the studied topics are the following: structure inference from semi-structured data, frequent tree mining, DTD mining and clustering of XML documents.

4.1 Schema Mining

Methods inferring structure from similar semi-structured documents are described in [5]. Efficient approaches for mining regularities are proposed. To

improve the candidate generation, some pruning strategies are described in [12]. In [8], the authors propose a method to extract a "reasonably small approximation" for typing a large and irregular data collection.

4.2 Mining DTDs from a Collection of XML Documents

Related to the previously presented topic, mining DTDs is based on the fact that the semi-structured documents are in the XML format. Here is the formal description of the problem:

Let e be an element that appears in the XML documents ($< e > < /e >$) Given a set I of N input sequences nested within element e, compute a concise and precise DTD for e such that every sequence in I is in conformity with the DTD.

Let us cite some systems trying to solve this problem, such as the IBM Alphaworks DDbE tool. XTRACT [4] is a system that automatically extracts DTDs from XML documents. It consists in Generalization / Factorization / MDL modules for inferring DTDs.

4.3 Frequent Tree Mining

Frequent tree mining refers to an important class of data mining tasks, namely patterns extraction. Many algorithms for finding tree-like patterns are developed. Basically, they adopt a straight-forward generate-and-test strategy [7]. TreeFinder [11] does clustering by counting co-occurrences of labelled pairs (in an Apriori manner). It computes the maximal trees. TreeMiner [13] is based on mining frequent subtrees using the "scope-list" data structure.

4.4 Clustering of XML Documents

Grouping XML documents in different classes with non supervised classification method is generally realized in the following manner:

Firstly the XML files are encoded such that each document is represented by an individual. Then, a distance measure is defined in order to group the similar individuals. The main differences between the XML documents clustering algorithms consist in choosing the distance between any pair of clusters (including single document cluster).

In [9], the authors partition the XML collection into smaller classes in order to infer, for each one a DTD. To compare the XML documents, they assign a different cost based on the tree editing operators. In [2] the authors use almost the same tree editing distance and the structural summaries in order to perform the XML clustering. S-GRACE algorithm [6] considers the distance based on s-graph. XRep [3] is designed on three stages: tree matching, merging of trees and pruning of the merge tree.

5 Our XML Document Supervised Classification Method

In this section, we present our supervised method for classifying the XML documents based on their structure. First of all we consider that a set of clusters is

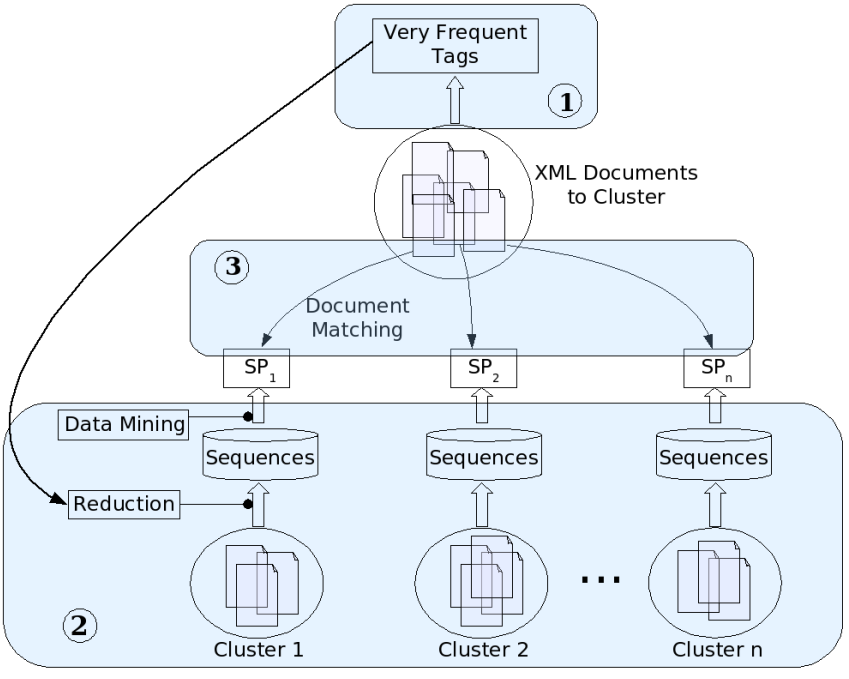

Fig. 3. Overview of the clustering method

provided coming from a previous clustering on a past collection. More formally, let us consider S_1 a first collection of XML documents (the training collection) and $C = \{c_1, c_2, ...c_n\}$ the set of clusters defined for the documents of S_1. Let us now consider S_2 a new collection of XML documents (the test collection). Our goal is to classify the documents of S_2 by taking into account the set of clusters C.

5.1 Overview

To this end, our method will perform as illustrated in figure 3. It is based on the following steps:

1. First of all, we perform a cleaning step: we extract the frequent tags embedded in the collection. This step corresponds to step "1" in figure 3. The main idea is to remove irrelevant tags for clustering operations. A tag which is very frequent in the whole collection may be considered as irrelevant since it will not help in separating a document from another (the tag is not discriminative).
2. Then we perform a data mining step on each cluster from the training collection (namely "C" in the foreword of this section). This step corresponds to step "2" in figure 3. For each cluster, the goal is to transform each XML document into a sequence (according to the techniques

described in section 3). Furthermore, during the mapping operation, the frequent tags extracted from step 1 are removed. Then on each set of sequences corresponding to the original clusters, we perform a data mining step intended to extract the sequential patterns. For each cluster C_i we are thus provided with SP_i the set of frequent sequences that characterizes C_i.

3. Finally the last step of our method relies on a matching between each document of the collection and each cluster which is characterized by a set of frequent structural subsequences (extracted from the second step). The following subsection describes the used matching technique. This last step corresponds to step 3 in figure 3.

5.2 Measuring the Distance Between Documents and Clusters

We tested several measures in order to decide which class each test document belongs to. The two best measures are based on the longest common subsequence. They compute the average matching between the test document and the set of sequential patterns which describes a cluster. More formally, the score for the document D_i in the cluster C_j is defined as follows:

$$score(D_i, C_j) = \frac{\sum_{k=1}^{|C_j|} \frac{|LCS(D_i, sp_{C_j}(k))|}{|sp_{C_j}k|}}{|C_j|}$$

Where $LCS(D_i, sp_{C_j}(k))$ is the longest common subsequence between the document and the k^{th} sequential pattern of C_j.

However, we noticed some particular cases where the scores of a document could be the same for two different clusters. In order to tackle this problem, we provide a modified score measure, which is defined as follows:

$$score2(D_i, C_j) = \frac{\sum_{k=1}^{|C_j|} \frac{i \times |LCS(D_i, sp_{C_j}(k))|}{|sp_{C_j}k|}}{|C_j|}$$

Where i is equal to:

- 0.8 if $sp_{C_j}k$ is a subsequence of D_i (100 % matching) and $sp_{C_j}k$ is "long" ($|sp_{C_j}k| > 0.7 \times (maximum\ length\ of\ a\ sequential\ pattern)$)
- 0.6 if $sp_{C_j}k$ is a subsequence of D_i (100 % matching) and $sp_{C_j}k$ is "short" ($|sp_{C_j}k| < 0.7 \times (maximum\ length\ of\ a\ sequential\ pattern)$)
- 0.4 if $sp_{C_j}k$ only matches D_i at 50 %.
- 0.2 for the remaining cases.

6 Experiments

In order to evaluate and validate our approach, we have exploited the documents provided by one of the INEX'05 tracks on document mining: the MovieDB. The extraction methods are written in C++ on a Pentium PC running a Red-Hat system.

6.1 Data

The MovieDB corpus is a collection of heterogeneous XML documents describing movies. It was built using the IMDB database. It contains 9643 XML documents. There are 11 predefined structure categories which correspond to transformations of the original data structure. Each cluster is characterized in terms of frequent sequential patterns (cf. the step 2). For example, we obtain for the following sequential patterns for cluster 1 and cluster 2:

- Cluster 1:
  - < (0_movie) (1_title) (1_url) (1_Country_of_Production) (2_item) (2_item) (1_filmography) (3_name) >
  - < (0_movie) (1_title) (1_url) (1_Country_of_Production) (2_item) (2_item) (2_item) (1_filmography) >
  - < (0_movie) (1_title) (1_url) (1_Filmed_In) (2_item) (2_item) (1_filmography) >
  - ...

Results for the first score (C.f. 5.2):

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
C1	593										
C2		481	4								
C3		41	659								
C4				171	1						
C5				49	387						
C6						199		10			21
C7							256		6		
C8								768			
C9									333		
C10						1		1		384	
C11						426		8			12
Recall	**1.0000**	**0.9215**	**0.9940**	**0.7773**	**0.9974**	**0.3179**	**1.0000**	**0.9759**	**0.9823**	**1.0000**	**0.3636**

Results for the second score (C.f. 5.2):

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
C1	593										
C2		474	11								
C3		29	671								
C4				171	1						
C5				36	400						
C6						206					24
C7							262				
C8								764			4
C9							1		332		
C10						1				383	2
C11						428					18
Recall	1.0000	0.9423	0.9839	0.8261	0.9975	0.3244	0.9962	1.0000	1.0000	1.0000	0.3750

Fig. 4. Experiments on MovieDB Collection m-db-s-0-test

– Cluster 2:
  - < (0_CL) (2_DC) (2_DC) (2_DC) (2_DC) >
  - < (0_CL) (1_BQ) (2_AX) (3_EH) (3_DT) (3_EH) >
  - < (0_CL) (1_BQ) (3_EH) (3_EH) (3_EH) (3_DT) >
  - < (1_EJ) >
  - ...

MovieDB has been preprocessed using a Porter stemmer. There are four collections, structure only based, with the different degrees of difficulty and overlapping of the classes (files m_db_s 0 to 3). A training collection was provided. The size of this collection was equal to the size of the document collection to be classified. For the structure only track, we use only the tree structure, without any attributes or content of the XML documents.

Results for the first score (C.f. 5.2):

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
C1	596										
C2		469	15								
C3		567	132								
C4				163	9						
C5				418	18						
C6						18		1		94	116
C7							255		6		
C8								346		421	
C9									332		
C10						2				378	6
C11								4		179	264
Recall	1.0000	0.4527	0.8980	0.2806	0.6667	0.9000	1.0000	0.9858	0.9822	0.3526	0.6839

Results for the second score (C.f. 5.2):

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
C1	596										
C2		437	47								
C3		10	689								
C4				171	1						
C5				36	400						
C6						7				218	4
C7							4		257		
C8								15		752	
C9									332		
C10						3				379	4
C11										447	
Recall	1.0000	0.9776	0.9361	0.8261	0.9975	0.7000	1.0000	1.0000	0.5637	0.2110	0.0000

Fig. 5. Experiments on MovieDB Collection m-db-s-0-test

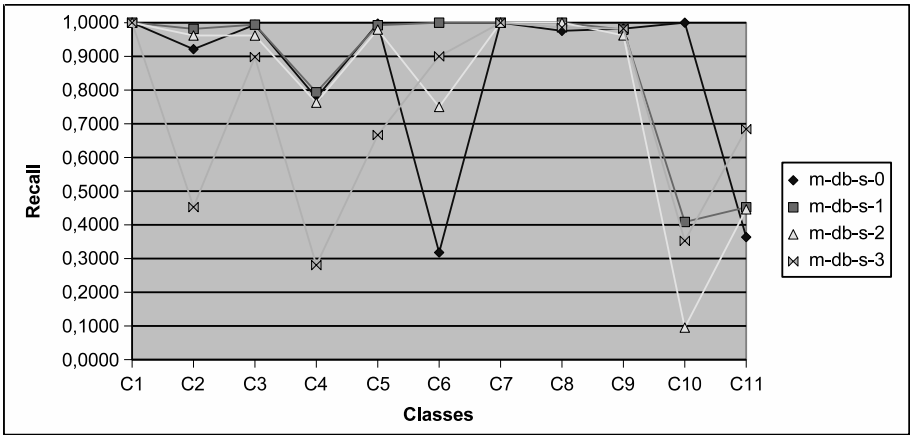

Fig. 6. Experiments on MovieDB Collection

6.2 Results

In this section we will give some results based on our approach. Below is given a part of the table built for the XML documents of the test collection (cf. step 3.).

In figure 4 the classification result is provided for the test file 0 (`m-db-s-0-test`) with both score functions defined in section 5.2. For each predefined class, the recall is calculated. The scores are quite good, except the 6th and the 11th classes due to the strong similarity of their extracted sequential patterns. Indeed, the candidates are allocated to the most general class, i.e. the class which contains most of the sequential patterns describing another class. Moreover when two scores are equal for a same document, whe arbitrary chose the first class. In figure 5 we report the classification result for the test file 3 (`m-db-s-3-test`) with both score functions. We can observe that the difference between the functions is clear when the level of noise in the files increases. This can be observed for clusters C2, C4 and C5 for instance, where the second score performs better than the first one.

In Figure 6 the degradation of the results according to the degradation of the four data collections is presented.

Based on the synthesis made by the organisers during the Inex'05 workshop related to the Document Mining track our result is situated in the top 3 ranking (our recall being quite good, between 0.8 and 0.95).

7 Conclusion

In this article we introduced a new supervised classification method for XML documents which is based on a linearization of the structural information of XML documents and on a characterization of each cluster in terms of frequent

sequential patterns. Experiments on the MovieDB collection validated the efficiency of our approach.

As perspective we plan to improve our method for better taking into account certain types of XML documents clusters characterised today by a very similar set of frequent sequential patterns. Various ways of measuring the distance between a document and each cluster will be studied.

References

1. 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.
2. Theodore Dalamagas, T. Cheng, K. Winkel, and T. Sellis. Clustering xml documents using structural summarie. In Proc. of ClustWeb - International Workshop on Clustering Information over the Web in conjunction with EDBT 04, Crete, Greece, 2004.
3. F. De Francesca, G. Gordano, R. Ortale, and A. Tagarelli. Distance-based clustering of xml documents. pages 75–78. ECML/PKDD'03 workshop proceedings, September 2003.
4. Minos Garofalakis, Aristides Gionis, Rajeev Rastogi, S. Seshadri, and Kyuseok Shim. XTRACT: a system for extracting document type descriptors from XML documents. pages 165–176, 2000.
5. P.A. Laur, F. Masseglia, and P. Poncelet. Schema mining: Finding structural regularity among semi structured data. Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'2000), Poster session, Lecture Notes in Artificial Intelligence, Springer Verlag, September 2000.
6. Wang Lian, David Wai-Lok Cheung, Nikos Mamoulis, and Siu-Ming Yiu. An efficient and scalable algorithm for clustering xml documents by structure. *IEEE Trans. Knowl. Data Eng*, 16(1), January 2004.
7. Tetsuhiro Miyahara, Takayoshi Shoudai, Tomoyuki Uchida, Kenichi Takahashi, and Hiroaki Ueda. Discovery of frequent tree structured patterns in semistructured web documents. In *PAKDD '01: Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 47–52, London, UK, 2001. Springer-Verlag.
8. Svetlozar Nestorov, Serge Abiteboul, and Rajeev Motwani. Extracting schema from semistructured data. pages 295–306, 1998.
9. Andrew Nierman and H. V. Jagadish. Evaluating structural similarity in XML documents. In *Proceedings of the Fifth International Workshop on the Web and Databases (WebDB 2002)*, Madison, Wisconsin, USA, June 2002.
10. 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.
11. A. Termier, M.-C. Rousset, and M. Se'bag. Treefinder: a first step towards xml data mining. In *In International Conference on Data Mining (ICDM 2002), Maebashi City, Japan, 2002*, 2002.
12. Ke Wang and Huiqing Liu. Discovering structural association of semistructured data. *Knowledge and Data Engineering*, 12(2):353–371, 2000.
13. M. Zaki. Efficiently mining frequent trees in a forest, July 2002. In KDD'02.