Fuzzy Data Mining for the Semantic Web: Building XML Mediator Schemas

Anne Laurent, Pascal Poncelet, Maguelonne Teisseire

Fuzzy Data Mining for the Semantic Web: Building XML Mediator Schemas

A. Laurent [a] P. Poncelet [b] M. Teisseire [a]

[a] *LIRMM - CNRS UMR 5506*
161, rue Ada
34392 Montpellier Cedex 5 - FRANCE
{*laurent,teisseire*}*@lirmm.fr*

[b] *LGI2P*
EMA - Site EERIE - Parc scientifique G. Besse
30035 Nîmes Cedex 1 - FRANCE
pascal.poncelet@ema.fr

Abstract

As highlighted by the World Wide Web Consortium, XML has been proposed to deal with huge volumes of electronic documents and is playing an increasing important role in the exchange of a wide variety of data on the Web. However, when dealing with such large and heterogeneous data sources, it is necessary to have an idea on the way these data sources are structured. This information is indeed essential in order to build mediator schemas. These mediator schemas are required to query data in a uniform way. Moreover, this information is interesting since it provides users with a semantic structure of the data they can query. Recently schema mining approaches have been proposed to extract in an efficient way the commonly occurring schemas from a collection. Nevertheless, according to the semantic point of view, such approaches suffer from different drawbacks. In this work, we propose thus a fuzzy approach, showing why and how fuzziness is useful in order to extract frequent approximate schemas.

Key words: Fuzzy Logic, Semi-Structured Data, Mediator Schemas, XML, Semantic Structures, Frequent Patterns

1 Introduction

Large amounts of data are available in XML format, and even if large volumes of legacy data are still marked up in HTML, efficient approaches have been

proposed to transform HTML documents into XML documents [12]. These XML documents, stored in many sources distributed over the Web, contain useful information. For instance, many scientists have to manage and share large amounts of data (*e.g.* biologists have to deal with huge amounts of complex data). In this framework, XML is of great interest in order to represent this complicated knowledge.

Recently, sharing data has become very popular. For instance, Peer-based Data Management [23,14] has become a great challenge for scientific data sharing, as Peer-to-Peer (P2P) systems are becoming more and more popular. Nevertheless the information stored in heterogeneous collections of XML documents has to be integrated in order to be processed or to be queried in a uniform way. In all fields like biology and medicine, there will never be a centralized database, which would be impossible to build. Scientists need thus to be provided with tools in order to reach the data they require from multiple databases. Database concepts can help to solve these problems. One result of the research led in data integration and in database federations is mediator-based information systems [13]. For instance, Xyleme is a huge warehouse integrating XML data on the Web [30]. This integration can be performed by providing mediator schemas. A *mediator schema* can be considered as a shared structure through which queries can be defined [15], as shown on Figure 1. However, no complete automatic tool is available to extract semantic knowledge from these large amounts of distributed and heterogeneous data. It is indeed still hardly possible to automatically build mediator schemas, which are thus still hand-crafted.

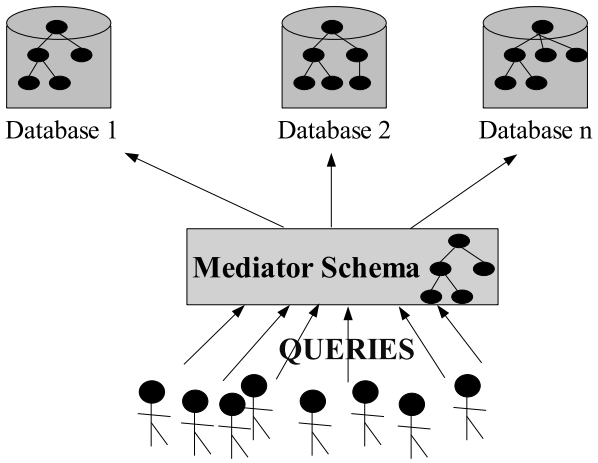

Fig. 1. Querying Data Distributed over the Web

In the past few years, data mining has been extensively studied in the framework of the semantic web. In this paper, we consider the problem of mining XML mediator schemas from a database perspective. By considering that a collection of XML documents could be modelled by labelled ordered trees [1,6]

(*e.g.* Figure 2 illustrates such a transformation [19]), we focus on schema mining which provides tools to mine frequent sub-schemas from large databases.

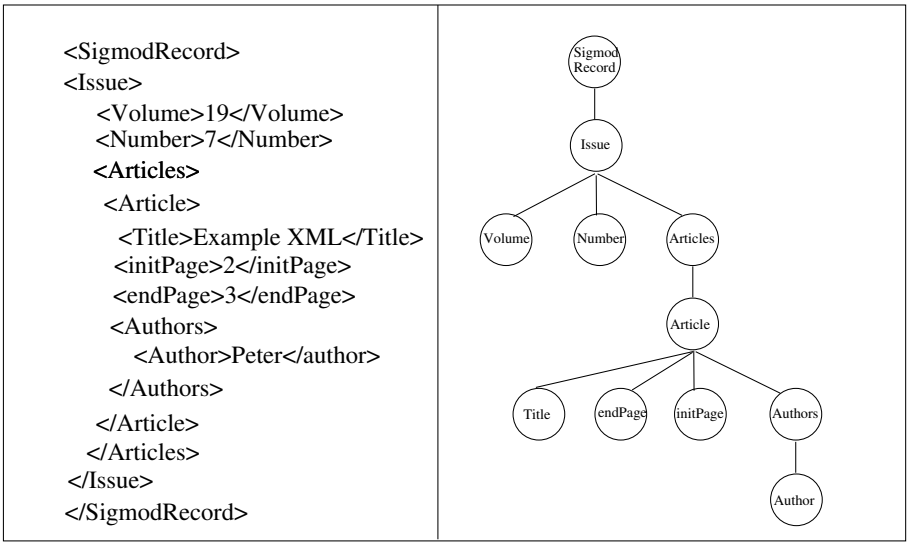

Fig. 2. An XML document modelled by a labelled ordered tree

These frequent sub-schemas are then merged in order to build mediator schemas. We show that schema mining approaches have some drawbacks and that a fuzzy approach is very useful to mine approximate schemas. Moreover, frequent mediator schemas mined from our approach provide an interesting source of information about the data available on the web since these schemas can be seen as semantic structures describing the information. Our approach is based on the definition of the tree inclusion. We define several ways to introduce fuzzy logic in this problem. The main idea is to propose a definition of soft inclusion, meaning that a tree is no more *included or not* in another one, but *gradually included* within it. A degree of inclusion is defined, depending on the way the fuzzy inclusion is considered. Finally, we introduce fuzzy frequent patterns which aim at representing the strength of the links from a frequent tree.

The sequel of the paper is organized as follows. Section 2 goes deeper into presenting the problems of schema mining. In Section 3 we propose our motivations for introducing fuzzy logic in order to provide approximate mediator schemas. Section 4 addresses the Fuzzy Tree Inclusion while in Section 5 we propose to extend the knowledge on the mined semantic structures by providing fuzzy links within the frequent trees. Section 6 concludes the paper.

2 Schema Mining Principles

This section is devoted to stating the schema mining principles. As proposed approaches are mainly extensions of level-wise approach for candidate and frequent generation, we first propose an overview of the association rules problem. Second we consider the schema mining problem. We broadly summarize the problem statement and give an overview of the core of this process.

2.1 A Brief Overview of the Association Rules Problem

The problem of mining association rules, firstly introduced in 1993 in [2], has received a great deal of attention. In brief, the problem is the following.

Let $I = \{x_1, \ldots, x_n\}$ be a set of distinct literals called items. A set $X \subseteq I$ with $k = |X|$ is called a *k-itemset* or simply an *itemset.* Let a database D be a multi-set of subsets of I. Each $T \in D$ is called a transaction. We say that a transaction $T \in D$ supports an itemset $X \subseteq I$ if $X \subseteq T$ holds. An association rule is an expression $X \Rightarrow Y$, where X, Y are itemsets and $X \cap Y = \emptyset$ holds. The fraction of transactions T supporting an itemset X with respect to database D is called the *support* of X, $supp(X) = |\{t \in D | X \subseteq T\}|/|D|$. The support of a rule $X \Rightarrow Y$ is defined as $supp(X \Rightarrow Y) = supp(X \cup Y)$. The confidence of the rule is defined as $conf(X \Rightarrow Y) = supp(X \cup Y)/supp(X)$. As the number of rules grows exponentially with $|I|$, the rule sets are typically restricted by minimal thresholds for the measures of support and confidence, minSupp and minConf respectively.

According to this restriction, the problem can thus be decomposed into two subproblems:

- Find all itemsets having support greater than or equal to minSupp where the support for an itemset is defined as the number of transactions that support the itemset. Such itemsets are called frequent itemsets.
- Use the frequent itemsets to generate the set of rules according to minConf.

As solving the second subproblem is straightforward we will now have a look on how to obtain efficiently itemsets. In [3], the Apriori algorithm was defined to mine such associations. It makes multiple scans over the data (*level-wise* approach). In the first scan, the support of individual items is counted in order to determine which items are frequent. These frequent items are thus called frequent 1-itemsets. From them, new 2-candidates considered as potentially frequent itemsets are generated. A new scan is performed on the database in order to verify the support of each 2-candidate. The process goes on until no new frequent itemset is found.

Firstly introduced by [27,28], schema mining has attracted a lot of attention [5,26,33,37]. The problem is much more complicated than the classical association rule one since complex structures in the form of labelled hierarchical partially ordered data have to be considered.

The problem could be formulated as follows. A rooted labelled tree $T = (V, E)$ is a direct, acyclic, connected graph with $V = \{0, 1, \ldots, n\}$ as the set of vertices (nodes), $E = \{(x, y)|x, y \in V\}$ as the set of edges. We assume that there is a special vertex $r \in V$ designated as a root and for all $x \in V$, there is a unique path from r to x. Then if $x, y \in V$ and if there is a path from x to y then x is called an *ancestor* of y (*i.e.* y is a *descendant* of x). If the length of the path from two vertices x, y is reduced to one, then the ancestor relationship is considered as a *parent* relationship. For an internal node $x \in V$, we assume that its children $x_1, x_2, \ldots, x_n$ $(n \geq 0)$ are ordered from left to right (*i.e.* there is a sibling relationship between children).

Let denote $supp(S)$ the number of occurrences of the subtree in a tree T. The support of a subtree in a database D is defined as the number of trees in D that contain at least one occurrence of S. A subtree is *frequent* if its support is greater than or equal to a user-specified minimum support (minSupp) value. The schema mining problem is thus reduced to mine all subtrees in D where the support value holds.

The core of the process for mining frequent subtrees is usually quite similar to the association rule approach and is briefly described in algorithm 1. The Gen_Cand function builds candidates. If $k = 1$ then all nodes are considered as potential subtree candidates. Otherwise, candidates of size k (*i.e.* having k nodes) are built by considering frequent subtrees of size $k - 1$. At each step a scan is performed over the database in order to verify if the minSupp constraint holds. The process goes on until no new frequent subtree is found. Refer to [5,38] for details.

3 Why considering Fuzzy Approaches

From the problem statement presented so far, one problem remains about the subtree inclusion. In [16], the author defines ten different tree inclusion relations and in [24], different definitions are proposed for strict, exact, ordered and weak tree inclusion. Let us have a closer look on these inclusion differences.

- The *strict* constraint addresses the labels of trees. When two nodes of dif-

Algorithm 1: Mining Frequent SubTrees

Data: Tree Database D
Result: Frequent Subtrees $\mathcal{F}$
$\mathcal{F} \leftarrow \emptyset$;
$k \leftarrow 1$;
repeat
  $\mathcal{C}_k \leftarrow Gen\_Cand(k)$;
  foreach $c \in \mathcal{C}_k$ **do**
    $c.cpt \leftarrow 0$;
    foreach $T \in D$ **do**
      if *c is a subtree of* T **then**
        $c.cpt$++;
    /* *minSupp stands for a user-specified minimum support value* */
    if $c.cpt \geq minSupp$ **then**
      $\mathcal{F} \leftarrow \mathcal{F} \cup \{c\}$;
  k++;
until $\mathcal{F}$ *does not grow any more*;

ferent trees have similar labels then the inclusion is called strict.

- When the parent relationship is preserved, the inclusion is *exact* and the subtree is called *induced*. Otherwise, *i.e.* when the ancestor relationship is preserved, the inclusion is not exact and the subtree is called *embedded*.
- If the children order is preserved the inclusion is *ordered*.
- When two labels of a tree correspond to a unique label into the other tree, the inclusion is called *weak*.

Figure 3 gives examples of a tree S included in a tree T.

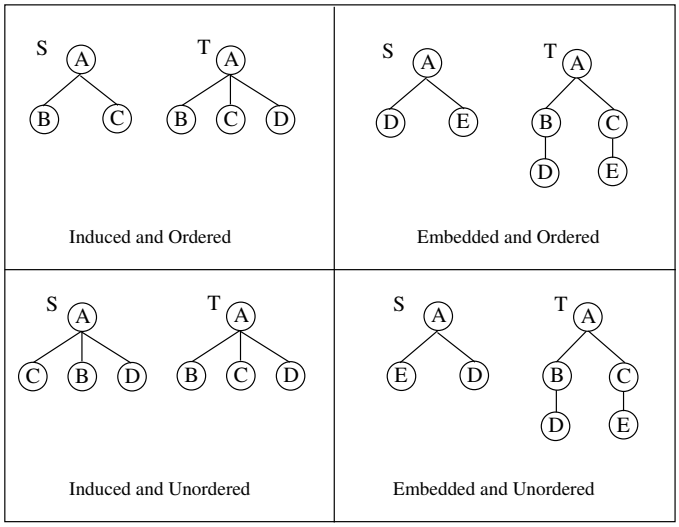

Fig. 3. Inclusion examples

According to the previous inclusion definitions, lot of algorithms were proposed for mining subtrees. Generally they consider strict and not weak inclusions. For example, in [27,28], the authors propose a new approach for discovering frequently occurring subtrees in XML documents. This algorithm is an extension of the level-wise Apriori approach where induced subtrees are considered. The FreqT approach proposed in [5,7] also considers induced and ordered trees and an Apriori-like algorithm is proposed. In [36], TreeMiner was defined to mine labelled, embedded, and ordered subtrees. This work was extended in order to build a structural classifier for XML data [38]. In [37], the author introduces Sleuth for mining frequent, unordered and embedded subtrees. The TreeFinder algorithm [25] also considers embedded trees and its extension Dryade [26] considers closed frequent subtrees, *i.e.* trees not any more included in another one. Other approaches such as gSpan [32] and closeGraph [33] generalize the problem by mining subgraphs.

These works mainly focus on the database scan and on data representation in order to propose scalable algorithms. Furthermore all the existing approaches consider *crisp* inclusion when mining frequent subtrees, *i.e.* a subtree is included or not into another one, which is too restrictive from a semantic point of view. In order to avoid this crisp inclusion problem, we propose to fuzzify this definition in order to better describe the data available on the Web. It is indeed very interesting to mine approximate schemas in order to have a better idea of the semantic structures we are provided with.

In a general framework, some works have been proposed in order to integrate fuzzy mining methods with the web [4,20]. In the more precise framework of tree and semi-structured data mining, some works have been proposed, especially to deal with approximate tree matching. [11] introduces a method to compare XML documents based on fuzzy bags. In [10,9], XML data are mined to build clusters of XML data, thus defining a distance measure between XML documents. [29] proposes an approximate method for graph schema extraction. Approximation is achieved by summarizing the semi-structured data graph using an incremental clustering method.

However as far as we know, no fuzzy method for mining frequent subtrees from huge databases has been proposed. We propose thus:

- to define the notion of fuzzy tree inclusion within this framework and
- to mine fuzzy frequent patterns.

4 Fuzzy Tree Inclusion

As shown on algorithm 1, a tree candidate has to be fully included within a tree from the database to be considered when computing its support. In this work, we argue that a tree may be considered to a certain extent within this computation even if it not fully included. For this purpose, we define four ways to consider fuzzy inclusion of a tree within another one:

- The first way aims at considering the vertical paths of trees. Contrary to induced one, embedded inclusion allows us to soften the ancestor-descendant relationship. Nevertheless, no methods exist to control this degree of relationship.
- In the second way, we address the horizontal paths of trees. While classical approaches consider that sibling nodes are ordered or not, there is no way to consider the proportion of sibling nodes included.
- The third way generalizes the previous one by considering the proportion of nodes.
- The last one considers similarities between nodes.

These definitions are then integrated within a new algorithm (algorithm 2) which mines frequent subtrees in a fuzzy way. They aim at describing the extent to which a tree is included in another one. While classical approaches deal with crisp inclusion, meaning that a tree is or is not included within another one, we propose to use a degree, defined between 0 and 1. The four ways this degree can be obtained are described below. It corresponds to four ways the function $Fuzzy\_Inclusion\_Degree$ from algorithm 2 may be defined.

In algorithm 2, the degree *cpt* of a candidate is updated after each tree scanning by the $Compute\_new\_cpt$ function. Several possibilities are given when merging these degrees, depending on the way the counting is performed. These possibilities are described in algorithm 3 by considering several counting methods: *sigma*-count, *thresholded*-count, *thresholded sigma*-count. This method is supposed to be known, as well as the associated threshold value ω if necessary.

For all these four ways fuzzy inclusion may be defined. We consider a node by node computation as done in the classical approach. However, this computation does not aim at matching *exactly* one node from the candidate tree to another one from the database tree but rather to compute to which extend a node corresponds to another one. This computation node by node leads to a degree *deg* returned by a $Fuzzy\_Match$ function. This function takes into account the node n being considered, the tree S from which this node is originated, and the target tree T. The tree S has to be considered in order to keep the tree structure, and not only the nodes without any connections. We have thus to deal with a set of fuzzy degrees, ranging between 0 and 1 to be merged

Algorithm 2: Mining Fuzzy Frequent SubTrees

Data: Tree Database D
Result: Frequent Subtrees $\mathcal{F}$
$\mathcal{F} \leftarrow \emptyset$;
$k \leftarrow 1$;
repeat
  $\mathcal{C}_k \leftarrow Gen\_Cand(k)$;
  foreach $c \in \mathcal{C}_k$ **do**
    $c.cpt \leftarrow 0$;
    foreach $T \in D$ **do**
      $c.cpt \leftarrow Compute\_new\_cpt(c.cpt, Fuzzy\_Inclusion\_Degree(c, T))$;
    / minSupp stands for a user-specified minimum support value */*
    if $c.cpt \geq minSupp$ **then**
      $\mathcal{F} \leftarrow \mathcal{F} \cup \{c\}$;
  k++;
until $\mathcal{F}$ *does not grow any more*;
return $\mathcal{F}$;

Algorithm 3: Merging Fuzzy Degrees. $Compute\_new\_cpt$ Function.

Data: od: old_degree, i: degree of fuzzy inclusion to be merged
Result: d: new degree
if *counting_method = sigma-count* **then**
  $d \leftarrow od + i$;
/ ω stands for a user-specified minimum degree */*
if *counting_method = thresholded count with minimum degree* ω **then**
  if $i \geq \omega$ **then**
    $d \leftarrow od + 1$;
  else
    $d \leftarrow od$;
if *counting_method = thresholded sigma-count with minimum degree* ω **then**
  if $i \geq \omega$ **then**
    $d \leftarrow od + i$;
  else
    $d \leftarrow od$;
return d;

in order to compute the $Fuzzy\_Inclusion\_Degree$.

In order to provide the user with a large set of solutions, we consider the OWA operators (*Ordered Weighted Aggregator*) [31], as illustrated by algorithm 4.

An OWA operator of dimension n is a mapping

$$F : R^n \rightarrow R$$

that has an associated n vector $W = (w_1, w_2, \ldots, w_n)^T$ such that $w_i \in [0, 1]$ and $\sum_{i=1}^{n} w_i = 1$. We have $F(a_1, a_2, \ldots, a_n) = \sum_{j=1}^{n} w_j \cdot b_j$ where b_j is the j^{th} largest value of the a_i.

In this framework, it is thus possible to compute the final degree in multiple ways. In the most pessimistic way, the *minimum* is computed (logical *and*), representing the lower boundary. In the very optimistic way, the *maximum* is computed (logical *or*), representing the upper boundary. Between these two boundaries, all possibilities are offered by tuning the weights, defining several degrees of andness/orness. The measure of orness is defined as follows:

$$orness(W) = \frac{1}{n-1} \sum_{i=1}^{n} ((n-i) \cdot w_i)$$

The measure of andness is defined as $andness(W) = 1 - orness(W)$.

Therefore, the user can choose whether he wants *all* nodes to be similar, *some of them* (using any desired degree of andness/orness), or *at least one* in a very flexible manner.

Algorithm 4: Computing the Fuzzy Inclusion Degree. $Fuzzy\_Inclusion\_Degree$ Function.

Data: S candidate tree, T target tree
Result: Fuzzy Degree $d \in [0, 1]$
foreach *node* $n \in S$ **do**
  $deg_n \leftarrow Fuzzy\_Match(n, S, T)$;
$d \leftarrow OWA_{n=1}^{|S|} deg_n$;
return d;

Note that all the OWA operators are not associative. This is the reason why the computation of the degree d in algorithm 4 can only be done after the whole scan of the database. In some cases, this degree is computed using an associative operator (*e.g.* minimum), which allows for computing it on the fly. This way, it may be possible to cut off the calculus. For instance, when computing the minimum, the process can be stopped as soon as the degree becomes lower than the minimum threshold, if one threshold is considered in algorithm 3.

Let us consider fuzzy indirect links within trees. For instance, Figure 4 shows a tree S which is embedded within a tree T. When taken into account, embedded trees are counted within the final support whatever the ancestor-descendant relationship may be. There is no consideration of the number of nodes separating the ancestor node from the descendant one. Even if it is of great interest to consider two nodes as being related even if they are not directly linked, we argue that it becomes irrelevant to consider that two nodes are related if they are separated by a high number of nodes.

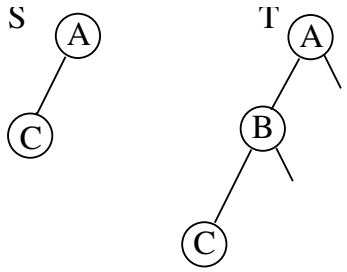

Fig. 4. Fuzzy Vertical Paths

In our approach, we propose thus to give a scope to this ancestor-descendant relationship. This scope is defined considering the number of nodes between ancestor and descendant nodes. Since it does not make sense to consider crisp boundaries, we propose to consider a fuzzy scope for the ancestor-descendant relationship. We consider fuzzy membership functions describing the ancestor-descendant relationship depending on the number of nodes separating the two nodes being considered, as shown on Figure 5 when considering no more than five nodes in a fuzzy way. For this purpose, we use fuzzy quantifiers [34,35]. Fuzzy set theory enables to represent fuzzy quantifiers, such as *at least 2, most*, by membership functions (as illustrated by Figure 5 and Figure 7). In this framework, *absolute* quantifiers like *at least 2* are distinguished from *relative* ones (*most* for instance).

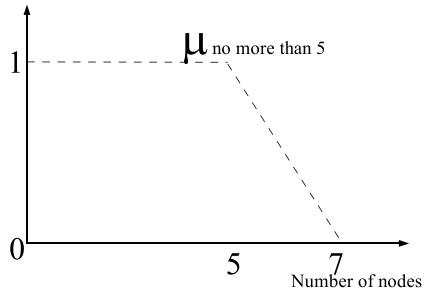

Fig. 5. Fuzzy Ancestor-Descendant Degree. Absolute Fuzzy Quantifier.

The *Fuzzy_Inclusion_Degree* from algorithm 2 is then computed by merging the degrees to which nodes can be matched. This matching degree results from

the extend to which a node may be linked to another one within a user-defined fuzzy ancestor-descendant relationship.

4.2 Fuzzy Horizontal Paths

In this section, we now consider fuzzy level inclusion. When considering ordered trees in the crisp approaches, a subtree S is included within a tree T only if all nodes of S can be mapped to nodes of T in the same order. In our approach, we propose to soften this definition by considering the proportion of nodes being included and well-ordered. For instance, Figure 6 shows an ordered tree S which is not embedded within an ordered tree T in the classical approaches since one of the node is misordered.

Fig. 6. Fuzzy Horizontal Paths

In our approach, we consider S as being included in T with a certain degree since the other nodes satisfy the inclusion. This degree is reported in the $Fuzzy\_Inclusion\_Degree$ used in the algorithm. It depends on the proportion of nodes that are well-ordered within the nodes of the level. We consider thus a fuzzy membership function defining the degree (ranging from 0 to 1) to which *most* nodes are well-ordered, as illustrated by Figure 7.

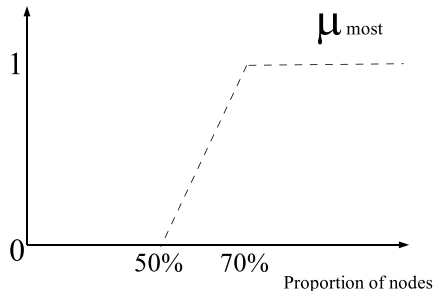

Fig. 7. Relative Fuzzy Quantifier: Proportion of Nodes

4.3 Partial Inclusion

In this section, we consider partial node inclusion. In a general way, *all* the nodes of a subtree S must be included in a tree T if S is included in T. However, we argue that this is too restrictive when mining data from the real

world when imperfections are often present. For instance, Figure 8 shows a tree S having 75% of its nodes included in T.

Fig. 8. Partial Inclusion

In our approach, we propose thus to define partial inclusion by considering the proportion of nodes of S being present in T. This proportion is then related to a fuzzy quantifier which estimates to which extent the node has to be taken into account when computing the function $Fuzzy\_Inclusion\_Degree$.

Note that in the classical case, mining totally included trees allows to cut in the database scan since whenever a node cannot be matched, there is no need looking for the other ones. In our approach, outliers are accepted, which may be considered as a drawback considering scalability. However, it is still possible to cut off the search when the proportion has been overpassed.

4.4 Similar Trees

In this section, we consider fuzzy similarities [8]. These similarities are related to some knowledge about the domain we are dealing with. For instance, when dealing with biological data, it is possible to know to which extent the node labelled by some bacteria is similar to the node labelled by another one. Let us now consider that we are provided with this kind of knowledge on the data. One of drawbacks of schema mining approaches is that the inclusion detection is only performed on nodes having same labels (C.f. Section 3 - strict inclusion). According to the semantic Web point of view, this restriction is usefulness since two different labels could describe similar concepts.

In our approach, we consider fuzzy approaches to overcome this drawback. For instance, Figure 8 shows a tree S that should be matched with T if we know that the concept D is close to B. For this purpose, we consider fuzzy ontologies as done in [21], and more generally fuzzy relations [17] describing to which extent two nodes are similar.

Let us recall that a fuzzy relation R between two reference sets X and Y is defined as a fuzzy subset from $X \times Y$. In particular, when X and Y are finite,

the relation can be described by a matrix $M(R)$ including the values of its membership function.

It is possible to compute the transitive closure of a fuzzy relation. Let G be the graph associated to the fuzzy relation R and let f_R be its membership function. The *transitive closure* of R is the R_T such that $f_{R_T}(x, y) \neq 0$ iff there exists a path in G from the node x to the node y.

Let R_T be the *max-min* transitive closure of R and $R \circ R$ be the max-min composition of R and R. R_T is computed as follows:

(1) $R' = R$
(2) While R' changes
- $R = R'$
- $R' = R \bigcup (R \circ R)$

(3) Stop. $R_T = R'$

In our approach, we consider a fuzzy relation R defined on the universe of node labels N. We have thus $R : N \times N \rightarrow [0, 1]$ describing to which extent a node $n \in N$ is similar to a node $n' \in N$. By computing the transitive closure of R, all nodes can be compared to every other nodes. Symmetric relations are considered since we deal with similarities. This degree of similarity is taken into account in the $Fuzzy\_Match$ function when computing the $Fuzzy\_Inclusion\_Degree$, as illustrated by algorithm 5. In this algorithm, the tree structure preservation is checked. This verification is required in order to prevent the tree structure from being destroyed. This verification must thus take into account the previous nodes that have been matched together.

Algorithm 5: Computing the Fuzzy Node Matching. $Fuzzy\_Match$ Function.

Data: n node, S candidate tree, T target tree
Result: Fuzzy Degree $d \in [0, 1]$
foreach *node* $n' \in T$ **do**
  if *n matches n' and respects the S tree structure* **then**
    $deg_{n'} \leftarrow f_R(n, n')$;
$d \leftarrow OWA_{n'=1}^{|T|} deg_{n'}$;
return d;

Algorithm 5 proposes to compute the final degree as a merged value from all the similarity degrees with nodes from T. In the classical case, the algorithm is more efficient since there is no need to scan all nodes from T. Another possibility is thus to stop the scan of T as soon as a similar node $n' \in T$ is found, leading to algorithm 6. In this case, we consider a user-defined γ threshold.

Algorithm 6: Fuzzy Node Matching with Stop Criterion. $Fuzzy\_Match$ Function.

Data: n node, S candidate tree, T target tree

Result: Fuzzy Degree $d \in [0, 1]$

foreach *node* $n' \in T$ **do**

  /* γ *stands for a user-specified threshold* */

  if *n matches n' and respects the S tree structure and* $f_R(n, n') \geq \gamma$ **then**

    return $f_R(n, n')$;

Fuzzy morphisms may also be considered in order to deal with this problem [22,18].

5 Fuzzy Frequent Subtrees

Classical approaches mine frequent trees that do not provide much information about the occurrences within the database except the fact that these subtrees are embedded in a sufficient number of trees from the database (depending on the user-defined minimum support value). We propose thus to extend the knowledge on the mined semantic structures by providing fuzzy links within the frequent trees. These fuzzy links are more or less thick. They help knowing whether they are *very shared, middle shared* or *a little shared,* as illustrated on Figure 9.

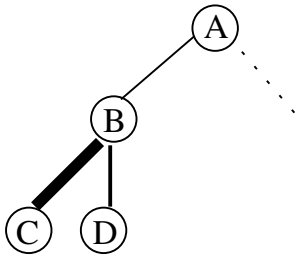

Fig. 9. Fuzzy Tree

In this framework, frequent subtrees are fuzzy acyclic graphs. Each link between two nodes is labelled by a degree ranging from 0 to 1. This degree is obtained by considering the proportion of trees from the database that contain the link. This proportion is then mapped to fuzzy quantifiers such as *very shared*, middle shared or little shared, as shown by Figure 10. It is graphically represented by taking into account the fuzzy quantifier having the greatest membership degree. Each fuzzy quantifier is then represented by a different thickness.

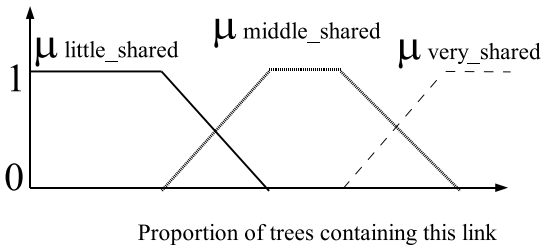

Fig. 10. Fuzzy Quantifiers for Fuzzy Trees

6 Conclusion

Recently mediator-based integration of heterogeneous data sources over the web has shown that it is a very efficient way for dealing with different types of Web documents such as XML ones. One of the main advantages of such approaches is that an heterogeneous collection of XML documents could be integrated into mediator schemas in order to be processed or queried in a uniform way. However, we have seen that no complete automatic tool is available to extract semantic knowledge from these large amounts of distributed and heterogeneous data. It is indeed still hardly possible to automatically build mediator schemas, which are thus still hand-crafted. Even if schema mining approaches exist to extract in an efficient way the commonly occurring schemas from a collection, they suffer from different drawbacks. The main drawback is that they consider *crisp* inclusion when mining frequent subtrees, *i.e.* a subtree is included or not into another one. In the semantic web context this constraint is too restrictive. In this paper we have thus proposed to soften such an inclusion in order to better describe the data available on the Web.

Data mining is of great interest for the semantic Web. In coming years, the Internet is likely to become one of the principal application areas of fuzzy logic. In this framework, mining fuzzy XML mediator schemas is crucial (i) in order to be automatically provided with a mediator schema to query data and (ii) in order to get semantic structures from the huge amounts of distributed and heterogeneous data. The approach we propose here can be used in order to mine frequent web structures from web sites. Moreover, mediator schemas, fuzzy or not, can be used in the framework of fuzzy queries over web data.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web.* Morgan Kaufmann, 2000.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large database. In *Proceedings of the International Conference on Management of Data (ACM SIGMOD 93)*, pages 207–216, 1993.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB 93)*, pages 487–499, 12–15 1994.

[4] D. Arotaritei and S. Mitra. Web mining: a survey in the fuzzy framework. *Fuzzy Sets and Systems*, 148:5–19, 2004.

[5] T. Asai, K. Abe, S. Kawasoe, H. Arimura, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In *Proceedings of the 2nd SIAM International Conference on Data Mining (SDM 02)*, Arlington, VA, USA, April 2002.

[6] T. Asai, H. Arimura, K. Abe, S. Kawasoe, and S. Arikawa. Online algorithms for mining semi structured data stream. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 02)*, pages 27–34, Maebashi, Japan, December 2002.

[7] T. Asai, H. Arimura, T. Uno, and S. Nakano. Discovering frequent substructures in large unordered trees. In *Proceedings of the 6th International Conference on Discovery Science*, pages 47–61, Sapporo, Japan, October 2003.

[8] B. Bouchon, M. Rifqi, and S. Bothorel. Towards general measures of comparison of objects. *Fuzzy Sets and Systems*, 84(2):143–153, 1996.

[9] P. Ceravolo and E. Damiani. Mining class hierarchies from XML data: Representation techniques. In *FUZZY DAYS*, 2004.

[10] P. Ceravolo, E. Damiani, and B. Oliboni. Fuzzy techniques for metadata construction. In *Proceedings of the Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2004). Special Session "Fuzzy Logic in the Semantic Web, a New Challenge"*, pages 1019–1026, 2004.

[11] P. Ceravolo, M. C. Nocerino, and M. Viviani. Knowledge extraction from semi-structured data based on fuzzy techniques. In *Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES 04)*, pages 328–334, 2004.

[12] C.Y. Chung, M. Gertz, and N. Sundaresan. Reverse engineering for Web data: From visual to semantic structure. In *Proceedings of the 18th International Conference on Data Engineering (ICDE 02)*, pages 53–63, 2002.

[13] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the world wide web: A survey. *ACM SIGMOD Record*, 27(3):59–74, 1998.

[14] A. Halevy, Z. Ives, P. Mork, and I. Tatarinov. Piazza: data management infrastructure for semantic web applications. In *Proceedings of the 12th International World Wide Web Conference (WWW 03)*, pages 556–567, 2003.

[15] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proceedings of the 19th International Conference on Data Engineering (ICDE 03)*, pages 505–515, 2003.

[16] P. Kilpelinen. *Tree matching problems with applications to structured text databases*. PhD thesis, University of Helsinski, November 1992.

[17] G. J. Klir and T. A. Folger. *Fuzzy Sets, Uncertainty, and Information*. Prentice Hall, 1988.

[18] L. J. Kohout. Defining homomorphisms and other generalized morphisms of fuzzy relations in monoidal fuzzy logics by means of bk-products. In *Proceedings of the 7th Joint Conference on Information Sciences (JCIS 03)(Subsection: 9th International Conference on Fuzzy Theory and Technology)*, page 13, 2003.

[19] H. P. Leung, F. L. Chung, and S. C. Chan. On the use of hierarchical information in sequential mining-based XML document similarity computation. *Knowledge and Information Systems*, 7(4), 2005.

[20] S.K. Pal, V. Talwar, and P. Mitra. Web mining in soft computing framework: Relevance, state of the art and future directions. *IEEE Transactions on Neural Networks*, 13(5):1163–1177, 2002.

[21] D. Parry. A fuzzy ontology for medical document retrieval. In *Proceedings of the 2nd Australasian Workshop on Data Mining and Web Intelligence (DMWI 04)*, pages 121–126. Australian Computer Society, 2004.

[22] A. Perchant and I. Bloch. Fuzzy morphisms between graphs. *Fuzzy Sets and Systems*, 128(2):149 – 168, 2002.

[23] I. Tatarinov, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyska, G. Miklau, and P. Mork. The Piazza peer data management project. *SIGMOD Record*, 32(3):47–52, September 2003.

[24] A. Termier. *Extraction of Frequent Trees in an Heterogeneous Corpus of Semi Structured data: Application to XML documents mining*. PhD thesis, Paris South University, April 2004.

[25] A. Termier, M.C. Rousset, and M. Sebag. TreeFinder: a first step towards XML data mining. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 02)*, pages 450–457, Maebashi City, Japan, 2002.

[26] A. Termier, M.C. Rousset, and M. Sebag. Dryade: A new approach for discovering closed frequent trees in heterogeneous tree databases. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 04)*, pages 543–546, Brighton, UK, 2004.

[27] K. Wang and H. Liu. Discovering typical structures of documents: A road map approach. In *Proocedings of the 21 ACM SIGIR International Conference on Information Retrieval*, pages 146–154, Melbourne, Australia, August 1998.

[28] K. Wang and H. Liu. Discovering structured association of semistructured data. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):353–371, 2000.

[29] Q.Y. Wang, J. X. Yu, and K.F. Wong. Approximate graph schema extraction for semi-structured data. In *Proceedings of the 7th International Conference on Extending Database Technology (EDBT 00)*, pages 302–316, 2000.

[30] L. Xyleme. A dynamic warehouse for XML data of the web. *IEEE Data Engineering Bulletin*, 24(2):40–47, June 2001.

[31] R. Yager. Families of owa operators. *Fuzzy Sets and Systems*, 57(3):125 – 148, 1993.

[32] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 02)*, pages 721–724, Maebashi City, Japan, December 2002.

[33] X. Yan and J. Han. CloseGraph: Mining closed frequent graph patterns. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 03)*, pages 286–295, Washington, DC, USA, August 2003.

[34] M. Ying and B. Bouchon. Quantifiers, modifiers and qualifiers in fuzzy logic. *Journal of Applied Non-classical Logics*, 7(3):335–342, 1997.

[35] L. A. Zadeh. A computational approach to fuzzy quantifiers in natural languages. *Computing and Mathematics with Applications*, 9(1):149–184, 1983.

[36] M. Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 02)*, pages 386–395, Edmonton, Canada, July 2002.

[37] M. Zaki. Efficiently mining frequent embedded unordered trees. *Fundamenta Informaticae*, 65:1–20, 2005.

[38] M.J. Zaki and C. Aggarwal. Xrules: An effective structural classifier for XML data. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD 03)*, pages 316–325, 2003.