Computing (2017) 99:493–506
DOI 10.1007/s00607-017-0553-x

A methodology for measuring structure similarity of fuzzy XML documents

Zhen Zhao[1,2] · **Zongmin Ma**[3]

Received: 25 February 2016 / Accepted: 25 March 2017 / Published online: 10 April 2017

Abstract Document matching has become a crucial task for data integration. A considerable amount of algorithms for comparing XML documents have been proposed in the literature. Yet, the existing approaches fall short in ability to identify structural similarities of fuzzy XML documents. To fill this gap, in this paper, we provide an integrated comparison approach to cope with structural similarities of the fuzzy XML documents. Firstly, we propose a new fuzzy XML document tree model to represent fuzzy XML document. Secondly, we offer element/attribute features similarity measure approach to identify matching nodes. Thirdly, we present an effective algorithm based on the tree edit distance to detect the structural similarities between fuzzy XML document trees represented with the proposed model. Finally, the experimental results demonstrate that our approach can efficiently perform structural similarity measure of the fuzzy XML documents.

Keywords Data integration · Fuzzy XML document · Matching · Structural similarity · Tree edit distance

Mathematics Subject Classification 68Q42(05C05)

✉ Zongmin Ma
zongminma@nuaa.edu.cn

Zhen Zhao
zhaozhen@bhu.edu.cn

[1] School of Computer Science and Engineering, Northeastern University, Shenyang 110819, Liaoning, People's Republic of China

[2] School of Information Science and Technology, Bohai University, Jinzhou 121013, Liaoning, People's Republic of China

[3] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, Jiangsu, People's Republic of China

1 Introduction

In the past few years, with the rapid development of the Internet, the management of Web data becomes ever more significant. XML (Extensible Markup Language) has rapidly emerged as the de-facto standard for representing an ever-increasing amount of data available on the Internet. As a new generation language of the Internet, XML is playing a key role in numerous applications. The application of XML includes data description and storage, information interchange, data integration, and so on [1].

Since XML data sources are independently developed and highly autonomous, the demand for integrating multiple XML documents which come from different application systems becomes ever more urgent. It is necessary to find similar XML documents referring to the same real-world object. Similarity measure of XML documents has also received significant attentions from academic communities due to its significant to many practical applications. At the same time, in many domains, it is sometimes difficult to state all information with crisp data. XML documents with fuzzy information are called fuzzy XML documents. Generally speaking, the increasing applications of fuzzy XML documents motivate us to investigate similarity comparison of heterogeneous fuzzy XML documents. It is necessary to identify the fuzzy XML documents and then consolidate multiple fuzzy XML documents to a single one. Unfortunately, to the best of our knowledge, there are not any reports discussing the similarity measure of the fuzzy XML documents. Aiming at developing an effective comparison approach to efficiently detect fuzzy XML structural similarity, in this paper, we devise an effective and efficient approach to support the similarity measure of the fuzzy XML documents which are gathered from different data sources and may not conform to the same grammar (DTD/XSD).

Treating the fuzzy XML documents as ordered trees, in this paper, we focus on comparing fuzzy XML document tree structures. We take a first step in construction of a new fuzzy XML document tree model, which makes it easier to describe fuzzy data and capture the structural information of fuzzy XML documents. On this basis, we propose an effective algorithm based on the tree edit distance to detect the structural similarity between fuzzy XML document trees. We identify corresponding information (i.e., element/attribute labels) in tree nodes and then decide the similarity of nodes from different fuzzy XML documents. Then the structural similarity relationships between two fuzzy XML documents can be determined from their corresponding nodes. The experimental results validate our approach and show the practicability of our matching algorithm.

The rest of the paper is organized as follows: in Sect. 2, we review the past related works on this subject. After a presentation of preliminaries in Sect. 3, we introduce a novel fuzzy XML document tree model in Sect. 4. The similarity measure approach of the fuzzy XML document is proposed based on this model in Sect. 5. Experimental evaluations are given in Sect. 6, and finally Sect. 7 concludes this paper.

2 Related work

Various methods to estimate structural similarity for XML documents have been proposed in the literature [2–7]. In [2], the authors develop a structural similarity metric for XML documents based on the “XML aware” edit distance between ordered labeled trees. To overcome the limitations of earlier approaches, Dalamagas et al. present a framework for clustering XML documents by structure based on tree edit distance in [3]. In [4], the authors discuss a variety of algorithms utilizing reference sets and bounds providing solutions to the approximate XML join problem between a pair of XML sources. In [5], entity matching framework considers support for blocking and the combination of different match algorithms. In [6], the authors introduce an approach of entity identification in XML documents based on approximate joins. In a previous work [7], the authors propose a benchmark for duplicate detection to XML. Note that most similarity measure algorithms of XML documents mentioned above make use of techniques for finding the edit distance between tree structures [2,3,6].

Previous similarity measurement methods [2,3] often assume that data in XML documents denote deterministic objects in the real-world. In fact, this assumption is often invalid since they may involve in fuzzy information. At the same time, some data are inherently vague rather than definite since their values are subjective in real-world applications [8]. There have been some efforts in the uncertain XML data processing. A simple model for representing and querying XML with incomplete information are proposed in [9], and a straightforward probabilistic XML model to manage probabilistic data is introduced in [10]. Fuzzy sets have been widely used in the quantification of fuzzy data since Zadeh proposed the theory of fuzzy sets [11]. Fuzzy data are modeled in the XML document in [12] and an approach which deals with fuzzy information based on the XML data are proposed in [13].

Besides,the authors introduce XML validation approach which exploits the concept of tree edit distance to effectively compare XML documents and grammar structures in [14]. Some approaches integrate semantic and syntactic assessment to the edit distance computation process and enable one to analyze the structure of fuzzy XML documents more precisely and get more realistic results [15–17]. However, The establishment, maintenance, and use of large scale domain lexicon/thesaurus require additional expenses, and it is laborious, time consuming. At the same time, the result of semantic similarity calculation is influenced greatly by the quality of lexicon [15,18].

3 Preliminaries

3.1 Fuzzy XML document

In order to represent uncertain information naturally in fuzzy XML data, a representative model of the fuzzy XML document based on “membership degree and possibility distributions” is developed in [19]. In this representation model, an element may be associated with a membership degree. The membership degree associated with an element means the possibility of being its parent’s child element. Also the attribute values of elements may be represented by possibility distributions. It can be seen that there are two semantics for a fuzzy attribute value represented by a possibility distribution: conjunctive and disjunctive. Here we do not present the detailed definitions of fuzzy XML representation model in this paper and only give an example fragment of fuzzy XML document shown in Listing 1. One can refer to [19] for more details.

Listing 1 A fragment of fuzzy XML document

```
<College>
 <Teacher Tid = "007100">
  <Dist Type = "disjunctive">
   <Val Poss = 0.8>
     <Position>associate professor</Position>
   </Val >
   <Val Poss = 0.6>
     <Position>professor</Position>
   </Val >
  </Dist>
  <Name>Tom</Name>
 </Teacher>
 <Teacher Tid = "007101">
  <Dist Type = "disjunctive">
   <Val Poss = 0.8>
     <Position>associate professor</Position>
   </Val >
   <Val Poss = 0.7>
     <Position>professor</Position>
   </Val >
  </Dist>
  <Name>Jack</Name>
 </Teacher>
</College>
```

3.2 Fuzzy XML document tree

The basic data structure of the XML document is the XML document tree. Following the Document Object Model (DOM) [20], XML document which represents hierarchically structured information can be represented as a rooted ordered labeled tree. A fuzzy XML document can be directly transformed into a fuzzy XML document tree according to the DOM. But this fuzzy XML document tree is clearly different from the crisp XML document tree because this fuzzy XML document tree contains new attribute and element types, which are attribute *Poss* and *Type*, in addition to element *Val* and *Dist*. After a fuzzy XML document was transformed into a DOM model, we can identify four kinds of *Val* element nodes as follows:

– They only have *Poss* nodes as child node (type-1).
– They only have leaf nodes as their child nodes except the *Poss* nodes (type-2).
– They only have nonleaf nodes as their child nodes except the *Poss* nodes (type-3).
– They have leaf nodes as well as nonleaf nodes as their child nodes except the *Poss* nodes (type-4).

4 A simplified tree model for fuzzy XML document

4.1 A novel fuzzy XML document tree model

Hereunder, we present a **N**ovel **F**uzzy **X**ML document **T**ree **M**odel (NFXTM for short) by modifying the structure of the original fuzzy XML tree model.

Definition 1 (*NFXTM*) Let NFXTM be an ordered tree T (N, E), where N and E are the sets of nodes and edges of T, respectively. In NFXTM, a 5-tuple (*NodeLabel*, *NodeDepth*, *NodeFuzzy*, *NodeType*, *NodePoss*) is used to represent the nodes of T for the fuzzy XML documents. Here

- *NodeLabel* is the label name of the node.
- *NodeDepth* is the nesting depth of the node in the document. The depth of the root node is defined to be 1.
- *NodeFuzzy* is used to indicate if the node is a fuzzy or crisp one. If the value of *NodeFuzzy* is 1, the node is a fuzzy one. If the value of *NodeFuzzy* is 0, the node is a crisp one. In particular, the value of *NodeFuzzy* of the node "Dist" or "Val" is 0.
- *NodeType* denotes the type of possibility distribution, disjunctive or conjunctive distributions. For a crisp node, the value of *NodeType* is equal to Null.
- *NodePoss* is the memberships of the node or the possibility of attribute values of elements. For a crisp node, the value of *NodePoss* is equal to Null.

NFXTM is a rooted ordered tree, in which the nodes represent the fuzzy XML elements/attributes. Element nodes are ordered following their order of appearance in the fuzzy XML document. Attribute nodes appear as children of their encompassing element nodes, sorted left-to-right by attribute name. Suppose that there is a global order defined over all nodes in the fuzzy XML document. The position of a node is consistent with the global order.

Similarity measure is a procedure of evaluating document trees similarity. In order to reduce the complexity of the similarity evaluation, Element/attribute values can be disregarded from the crisp XML document tree. But, in this paper, we cannot simply disregard all element/attribute values from the fuzzy XML document tree. We need to keep the fuzzy values of nodes for computing of the nodes similarity (cf. Sect. 5.1) when the NFXTM tree is constructed. The fuzzy values of nodes can be included into 5-tuple of sibling (element/attribute) nodes in NFXTM.

4.2 NFXTM model construction methods

Compared with the crisp XML document tree model, it can be seen that the fuzzy XML document tree model mentioned in Sect. 3.2 contain four new nodes: *Type*, *Val*, *Poss* and *Dist*. The values of these nodes include fuzzy information. We need to transform fuzzy values under *Poss* nodes into 5-tuple of sibling (element/attribute) nodes as mentioned in Sect. 4.1, and then *Poss* nodes can be deleted in processing of transformation. Of course, *Type* node will not appear in the comparison of structural similarity. Similarly, if the fuzzy XML document tree structure is not influenced, we can disregard the *Val* and *Dist* nodes. That is to say, the operation of deleted the *Val* and *Dist* nodes should not affect the depth of other nodes. Based on the discussion above, we need to carry on processing to these four types of nodes according to the following rules so as to obtain NFXTM model.

- *Rule 1* For the *Type* node, we copy *Type* node value to *NodeType* of its siblings, and delete *Type* node and its sub-tree.

Algorithm 1 MCA

```
Input: A //a fuzzy XML document tree A
Output: MCA (A) //a NFXTM model of fuzzy XML document tree A
Begin
1:  for each tree node tnode in A
2:  inherit NodeFuzzy , NodeType, NodePoss value from its parent node
3:  if (tnode label is identified as a “Type”)
4:      process Type node by applying Rule 1
5:  if (tnode label is identified as a “Val”)
6:      process Val node by applying Rule 2
7:  if (tnode label is identified as a “Poss”)
8:      process Poss node by applying Rule 3
9:  if (tnode label is identified as a “Dist”)
10:     process Dist node by applying Rule 4
11: end for
12: return A
```

– *Rule 2* For the *Val* node, if it is the *Val* node of type-1 (cf. Sect. 3.2), we delete the *Val* and its sub-tree. If it is the *Val* node of type-2, type-3, type-4, we don’t do any treatment.
– *Rule 3* For the *Poss* node, we will copy *Poss* node value to *NodePoss* of its siblings, and delete *Poss* node and its sub-tree.
– *Rule 4* For the *Dist* node, if *Dist* node became a leaf node after the above processing, we delete *Dist* node and its sub-tree.

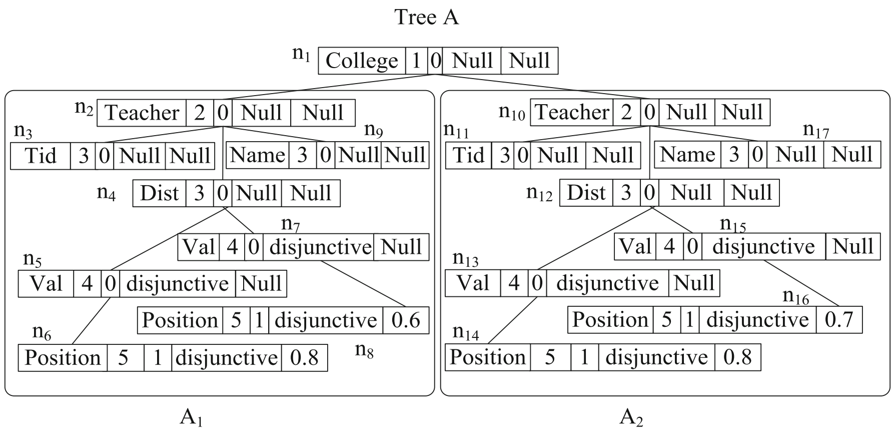
According to these rules, we propose a Model Construction Algorithm (MCA for short) based on the preceding analysis. To sum up, we can delete all *Type*, *Poss* nodes (Algorithm 1, lines 3–4, 7–8), and a part of the *Val* and *Dist* nodes (lines 5–6, 9–11). The rest of the *Dist* and *Val* node can’t be ignored. Although *Dist* and *Val* node are related to fuzzy data, but *Dist* and *Val* nodes are no different with crisp nodes and just indicate that node label name is “*Dist*” and “*Val*”. We can view them as crisp nodes in the structural similarity comparison. So far, fuzzy information is embedded into element/attribute nodes as mentioned in Sect. 4.1. And then, a NFXTM tree shown in the Fig. 1 can be constructed corresponding to the Listing 1. Our method seems to be quite simple, but it is a more detailed description of each fuzzy node. Without loss of generality, when we use “node” we mean “element node” or “attribute node”.

5 Similarity measures

5.1 Node similarity measures

Given a fuzzy XML document tree, a node is a fundamental data item for the similarity measures. We use Sim_{Node} (N_1, N_2) to represent the similarity degree of two nodes N_1 and N_2.

To accurately assess the similarity of document node-pairs, a similarity measure should exploit the features of nodes. The characteristics associated with each node in a fuzzy XML document tree are called the node features. Each feature has an associated value. *NodeLabel*, *NodeType*, *NodePoss* are the most commonly used features of nodes.



Tree A
n1 College 1 0 Null Null
n2 Teacher 2 0 Null Null
n3 Tid 3 0 Null Null
n9 Name 3 0 Null Null
n4 Dist 3 0 Null Null
n7 Val 4 0 disjunctive Null
n5 Val 4 0 disjunctive Null
Position 5 1 disjunctive 0.6
n6 Position 5 1 disjunctive 0.8
n8
n10 Teacher 2 0 Null Null
n11 Tid 3 0 Null Null
n17 Name 3 0 Null Null
n12 Dist 3 0 Null Null
n15 Val 4 0 disjunctive Null
n13 Val 4 0 disjunctive Null
n16 Position 5 1 disjunctive 0.7
n14 Position 5 1 disjunctive 0.8
A1
A2

Fig. 1 A NFXTM tree corresponding to the fuzzy XML document in Listing 1

The more similar those features are, the more similar two nodes are. Consequently, these features are commonly used for assessing the similarity of document node-pairs. Depending on the different exploited feature, we present some of the commonly used similarity measures.

The node label name (*NodeLabel*) is considered an important source of information for node matching. Node label names can be syntactically similar (“Stu”, “Student”) or semantically similar (“Worker”, “Staff”). In this paper, we only consider syntactically similarity measures to compute the similarity degree of node label names. According to our understanding, syntactic measures take the advantage of the representation of label names as strings to assess the similarity of two nodes. Various algorithms for a similarity measure between two strings have been proposed in the literature, including Jaro Similarity [21], Lin’s similarity measure [22], Levenshtein distance [23], N-gram distance [24]. Most of them make use of techniques for finding the edit distance between strings. The EditDistance measure has the advantages of simple calculation and easy to understand [21–23]. So it has been applied to syntactic similarity measure of element label/name and achieved the anticipated effects [16,17]. Here we adopt Lin’s similarity measure method in [22] based on the edit distance. The Lin’s similarity measure of two strings is given by the minimum number of operations needed to transform one string into the other. To compute the similarity degree between the values of *NodeLabel* L_1 and L_2, the following formula is used:

$$Sim_{Label}(L_1, L_2) = \frac{1}{1 + editDistance(L_1, L_2)} \tag{1}$$

Here editDistance (L_1, L_2) is the minimum number of single character insertion, deletion, and substitution operations that are needed to transform L_1 to L_2. Each edit operation is assigned a unit cost.

Sometimes it is insufficient that the node label name is considered an only necessary feature for determining the node similarity. Thus, it is necessary that fuzzy information sources are used to compute the node similarity and some false positive matches will be eliminated. The node values of *NodeType* and *NodePoss* are fuzzy information source that makes a contribution in determining the node similarity. These two kinds of similarity are called fuzzy information similarity.

Note that the *NodeType* value of a node in a fuzzy XML document tree is of either disjunctive or conjunctive. So we distinguish the *NodeType* similarity of them, i.e. the possibility that two disjunctive (or conjunctive) nodes from two different fuzzy XML document trees are similar is higher than that one disjunctive (conjunctive) node and one conjunctive (disjunctive) are similar. *NodeType* similarity is represented as Sim_{Type} (T_1, T_2). For the nodes having the same *NodeType* value, the *NodeType* similarity is set to be 1, while the *NodeType* similarity of the nodes having different *NodeType* value is set to be 0.5. It can be shown in following formula.

$$Sim_{Type}(T_1, T_2) = \begin{cases} 1 & if \quad T_1 = T_2 \\ 0.5 & if \quad T_1 \neq T_2 \end{cases} \tag{2}$$

Furthermore, the *NodePoss* value of nodes in a fuzzy XML document tree varies between 0 and 1. The similarity degree of two nodes with small difference value of *NodePoss* is higher than the similarity degree of two nodes with big difference value of *NodePoss* . *NodePoss* similarity is represented as Sim_{Poss} (P_1, P_2). We proposed a formula to measure the *NodePoss* similarity between nodes from two different fuzzy XML document trees.

$$Sim_{Poss}(P_1, P_2) = 1 - |P_1 - P_2| \tag{3}$$

It is clear that each individual node similarity measure exploits a specific feature of the node, e.g. the label name measure only makes use of the node label name. But assessing the similarity of nodes using an individual measure is not sufficient. To get node similarity value of node-pairs, we should combine their feature similarity values resulted from different features similarity measures. Following mentioned above, the node similarity of two nodes $\mathrm{N}_1$and N_2, Sim_{Node} (N_1, N_2), is evaluated as the weighted average of their features similarity (*Label* similarity (Sim_{Label}, cf. Form. 1), *Type* similarity (Sim_{Type}, cf. Form. 2), *Poss* similarity (Sim_{Poss}, cf. Form. 3)) scores:

$$\begin{aligned} Sim_{Node}(N_1, N_2) = {} & w_L \times Sim_{Label}(L_1, L_2) \\ & + w_T \times Sim_{Type}(T_1, T_2) + w_P \times Sim_{Poss}(P_1, P_2) \end{aligned} \tag{4}$$

Here w_L, w_T and w_P are provided as input, and $\mathrm{w}_L + \mathrm{w}_T + \mathrm{w}_P = 1$. The user can thus assign which features similarity value is more important to the node similarity by varying parameter w_L, w_T, w_P. For $\mathrm{w}_L = 1$, $\mathrm{w}_T = \mathrm{w}_P = 0$, we only consider label similarity and ignore fuzzy information similarity.

Generally speaking, in order to effectively combine feature similarity, the weighting parameter w_L, w_T, w_P come down to an optimization problem such as weighting parameter should be chosen to maximize the overall node similarity function (cf. Form. 4). This can be addressed using a number of known techniques that apply machine learning in order to identify the best weights for a given problem [25]. We do not further address the weighting parameter here since it is out of the scope of this paper. As for whether nodes are matching or not, we can solve this problem via setting a threshold. If the node similarity degree is greater than the given threshold (cf. Sect. 5.2), we say they are matching.

5.2 Structural similarity measures of fuzzy XML documents

Two fuzzy XML documents, which are collected from different data sources and do not conform to the same grammar (DTD/XSD), can have very different sizes on account of optional and repeating elements. Edit distance measure will necessarily find a distance between such a pair of documents, and will recognize similarity of these documents. In this section, we introduce an edit distance measure with regard to the structural similarity. First we present the tree edit distance definition.

Definition 2 (*Tree edit distance*) The edit distance between two trees A and B is defined as the minimum cost of all edit scripts that transforms A to B, TED (A, B) = Min{Cost_{ES}}.

An edit script ES between two trees A and B is a sequence of edit operations turning A into B. The cost of ES is the sum of the costs of the operations in ES. An optimal edit script between A and B is an edit script between A and B of minimum cost. This minimum cost is called the tree edit distance. In order to capture the structural similarities of fuzzy XML document trees, we need to firstly introduce the notion of matching nodes between two document trees.

Definition 3 (*Matching nodes between document trees*) Given two document trees A = (a_1, …, a_m) and B = (b_1, …, b_n) and a threshold θ, we define the matching nodes between A and B, as the set of pairs of matching nodes N from of A and B, N = (a_i, b_j) A×B, a_i and b_j include in A and B, respectively, with the same depth and relative order (in preorder traversal), having Sim_{Node} (a_i, b_j) $> \theta$.

Following Definition 3, the problem of computing the edit distance between two trees A and B is equivalent to find the minimal cost of edit operations that can transform A to B, in a roundabout way, identifying the maximum number of matching nodes in A and B. In other words, the more number of matching nodes, the lesser number of edit operations, the lesser the edit distance. Associated with each of these edit operations is a nonnegative cost. Our algorithms work with general costs, but in this paper we restrict our presentation and experimentation to constant (unit) costs. In the following, we assume the general case where atomic insertion/deletion operations are of unit costs is equal to 1.

Now we use $\text{Cost}_{DelTree}(A_i)$ to represent the costs of deleting sub-tree A_i and $\text{Cost}_{InsTree}$ (B_j) to represent the costs of inserting sub-tree B_j when comparing two fuzzy XML document trees. These costs of tree insertion and deletion operations are obtained by traversing of sub-tree and are exploited with an adaptation of tree edit distance algorithm (given in Algorithm 2), which provides a similarity measure for fuzzy XML documents. Following the Nierman and Jagadish's main edit distance algorithm [2] and Tekli and Chbeir's improved edit distance algorithm [15], we introduce our Modified Tree Edit Distance (MTED for short) algorithm. Algorithm MTED

Algorithm 2 MTED

```
Input: A, B //fuzzy XML document tree to be compared
Output: MTED (A, B) //Edit distance between A and B
Begin
1:    M = Degree(A) // The number of first level sub-trees of A
2:    N = Degree(B) // The number of first level sub-trees of B
3:    Dist[ ][ ] = new[0· · ·M][0· · ·N]
4:    If (A[0]. NodeDepth == B[0]. NodeDepth and $\mathrm{Sim}_{Node}$ (A[0], B[0]) $> \theta$) //node matching
5:    {
6:      Dist[0][0] = 0; // matching nodes are associated null costs
7:      MatchNode add 1 // Counting matching nodes
8:    }
9:    Else
10:     {
11:       Dist[0][0] = 1
12:     }
13:     For (i = 1 to M) Dist[i][0] = Dist[i-1][0] + $\mathrm{Cost}_{DelTree}(A_i)$
14:     For (j=1 to N) Dist[0][j]=Dist[0][j-1]+ $\mathrm{Cost}_{InsTree}(B_j)$
15:     For (i=1 to M)
16:     {
17:       For (j=1 to N)
18:         { //identifies the set of insertion/deletion operations having the minimum overall cost
19:         Dist[i][j]=Min{
20:                 Dist[i-1][j-1] + MTED($A_i$, $B_j$),
21:                 Dist[i-1][j] + $\mathrm{Cost}_{DelTree}(A_i)$,
22:                 Dist[i][j-1] + $\mathrm{Cost}_{InsTree}(B_j)$
23:                 }
24:         }
25:     }
26:     Return Dist[M][N]
```

provides an improved and more accurate structural similarity measure of the fuzzy XML document trees. The algorithm starts by counting the total number of matching nodes between A and B (Algorithm 2, lines 4–7), Note that the update operation is specifically disregarded in MTED in order to allow the identification of matching nodes. Then MTED computes the sum of the costs of deleting every node in the source document tree (line 13) and inserts every node of the destination tree (line 14). Consequently, MTED identifies the set of insertion/deletion operations having the minimum overall cost (lines 15–25). In short, algorithm MTED recursively goes through the fuzzy XML document trees being compared, combining tree insertion and tree deletion operations so as to identify those of minimal cost. The node matching is applied to the roots of the fuzzy XML document trees being compared, as well as the roots of each pair of sub-trees considered in the recursive process (line 4–7), whereas tree insertion and tree deletion operations are applied to corresponding first-level sub-trees (lines 13–14, 21–22). Hence, the number of matching nodes between fuzzy document trees is identified by this algorithm. At the same time, we can determine the edit distance of transforming a source tree A into a destination tree B.

We normalize a similarity coefficient α to measure the effect of matching nodes on the edit distance.

$$\alpha = \frac{1}{1 + \frac{MatchNode}{2\times Max(|A|,|B|)}} \tag{5}$$

Tree Edit Distance (cf. Definition 2) is defined as follows.

$$TED(A, B) = MTED(A, B) \times \alpha \tag{6}$$

Following [15], we give the formal definition of similarity of the fuzzy XML documents. This similarity measure is also consistent with the formal definition of similarity in [22].

$$Sim_{FXDoc}(A, B) = \frac{|A| + |B| - TED(A, B)}{|A| + |B|} \tag{7}$$

6 Experimental evaluations

6.1 Evaluation criteria and data sets

In the context of documents matching, let us look at the similarity evaluation criteria adopted in our experimental evaluation. Owing to the proficient use of predecessors, we make use of the precision, recall and F-measure metrics defined in [3], to evaluate the effectiveness (quality) of our approach. The effectiveness of similarity measure is commonly determined with the standard measures precision (P), recall (R) and F-measure with respect to a manually determined "perfect" result. Therefore, as with traditional information retrieval evaluation, high precision and recall, F- measure characterize a good matching method.

To test our method's effectiveness in evaluating fuzzy XML structural similarity, we used the following data sets [26] generated based on real fuzzy XML documents for the experimental evaluation. For this real data sets, we considered the SigmodRecord.xml, DBLP.xml and NASA.xml. Of course, we need to make use of a random data generation method that transformed the data sets based on these documents into a fuzzy data set. That is, we add artificially fuzzy nodes to these documents. After that, they are converted into fuzzy XML document. Due to their relatively large size, we carefully split each fuzzy XML document into several documents (the document size according to the arithmetic progression distribution, from 10 KB to 10 MB). We experimented on a set of 90 documents corresponding to SigmodRecord.xml (30 documents), DBLP.xml (30 documents) and NASA.xml (30 documents).The characteristics of the real data sets are summarized in Table 1.

Table 1 Characteristics of the real data sets

Filename	Size	No. of element	No. of attributes	Avg-depth
dblp.xml	23 MB	3,332,130	404,276	2.90228
nasa.xml	3 MB	476,646	56,317	5.58314
SigmodRecord.xml	467 KB	11,526	3737	5.14107

6.2 Experimental results

In the following, we show several evaluation scenarios reporting on the experimental results. The first scenario is to evaluate the quality of real data sets described in the preceding paragraphs. In the first step, the similarity degree between every pair of different fuzzy XML documents is first computed by using our method. All the matching tasks have been performed and the values were derived after a set of experiments. Second Step, we started a series of classification tasks by varying the similarity threshold in the [0, 1] interval. The fuzzy XML documents which similarity degrees are greater than a given threshold will be grouped together. Lastly, according to the clustering results, we compute precision, recall and F-measure for each of the classification sets in the multilevel classification phases, the results of these evaluations are reported in Fig. 2. From Fig. 2, we can see that inconsistent documents are gradually filtered from the classification sets, while varying the classification threshold from 0 to 1. Results show that our algorithm yields optimal classes at a very early stage of the multilevel classification process (with thresholds < 0.5). We also note that higher threshold values (e.g. thresholds > 0.8) are not much improved performance in these sets of tasks due to the fact that it was very hard to find more right matches at such a higher threshold value. In second scenario, we experimented with real data sets (chose ten document, size from 100 KB to 2 MB) for runtime performance analysis. We can know that the complexity of our structural comparison algorithm is polynomial following the size of the fuzzy XML document trees being compared. This polynomial dependency on the size of each tree (document) is experimentally verified, timing results being presented in Fig. 3. The timing experiments were implemented in JDK 1.6, and performed on a system with Intel Core i5 processor, 4 GB RAM and running on Windows 7. The results in Fig. 3 reflect that the time to identify the structural similarity between fuzzy XML document trees grows in a linear dependency on the size of each tree being compared.

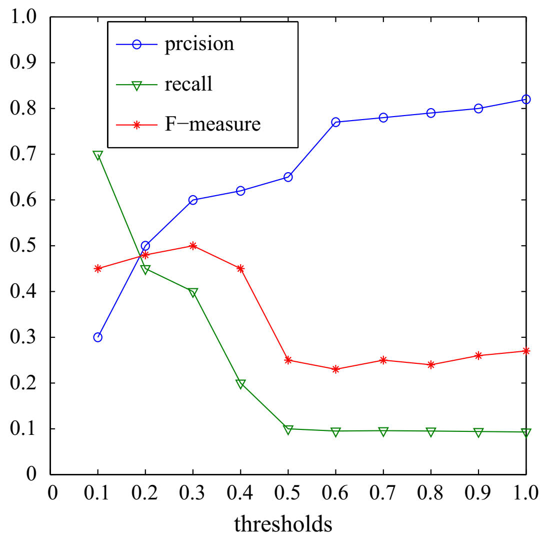


Fig. 2 Matching quality for real-world data sets

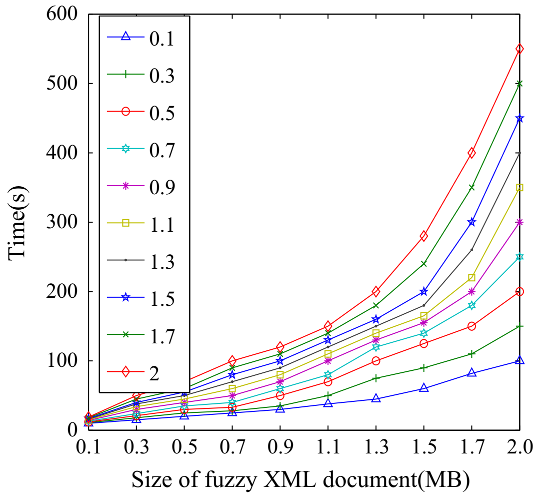


Fig. 3 Timing results to compare similarity for pair-wise fuzzy XML documents

7 Conclusion

In order to deal with the issue of the similarity measure of the fuzzy XML documents effectively, in this paper, we propose a novel fuzzy XML document tree model to capture the structural information of fuzzy XML documents. Based on the exploited information in node, we categorize node similarity computation into label similarity measures and fuzzy information similarity measures. Among them, the former captures the element label names and the latter exploits fuzzy information of the node itself. We compare the similarity of document structures based on the fuzzy XML document tree model by modified tree edit distance algorithm. The experimental results show that our approach can efficiently perform matching on the fuzzy XML documents. Note that this paper concentrate on the structural similarities between fuzzy XML documents without considering their contents (i.e., element/attribute values). In the near future, we will investigate the similarity of fuzzy XML documents, combining the structural similarities and the content similarities together. In addition, the similarity calculation developed in this paper is based only on the syntactic level in calculating similarity of labels. It has been unveiled that the semantic aspect plays an important role in the similarity calculation [16]. Some efforts have proposed the integration of the semantic aspect in the similarity calculation with the help of a lexicon (i.e.,WordNet) [15,17]. We plan to enhance our approach proposed in this paper by using lexicon and then label normalization [27] in future work.

Acknowledgements This work was supported by the *National Natural Science Foundation of China* (61370075 & 61572118) and the *Program for New Century Excellent Talents in University* (NCET- 05-0288).

References

1. Thomo A, Venkatesh S (2008) Rewriting of visibly pushdown languages for xml data integration. In: Proceedings of the 17th ACM conference on information and knowledge management. ACM, Napa Valley, pp 521–530

2. Nierman A, Jagadish HV (2002) Evaluating structural similarity in XML documents. In: Proceedings of ACM SIGMOD WebDB, vol 2. ACM, Madison, pp 61–66
3. Dalamagas T, Cheng T, Winkel KJ et al (2006) A methodology for clustering XML documents by structure. Inf Syst 31(3):187–228. doi:10.1016/j.is.2004.11.009
4. Guha S, Jagadish HV, Koudas N, Srivastava D, Yu T (2006) Integrating XML data sources using approximate joins. ACM Trans Database Syst 31(1):161–207
5. Köpcke H, Rahm E (2010) Frameworks for entity matching: a comparison. Data Knowl Eng 69(2):197–210. doi:10.1016/j.datak.2009.10.003
6. Ribeiro L, Hӓrder T (2006) Entity identification in XML documents. In: 18th GI-workshop on the foundations of databases, pp 130–134
7. Weis M, Naumann F, Brosy F (2006) A duplicate detection benchmark for XML (and relational) data. In: SIGMOD 2006 workshop on information quality for information systems. Chicago
8. Oliboni B, Pozzani G (2008) Representing fuzzy information by using XML schema. In: Proceedings of the 19th international conference on database and expert systems application. Turin, pp 683-687. doi:10.1109/DEXA.2008.44
9. Abiteboul S, Segoufin L, Vianu V (2006) Representing and querying XML with incomplete information. ACM Trans Database Syst 31(1):208–254
10. Nierman A, Jagadish HV (2002) ProTDB: probabilistic data in XML. In: Proceedings of the 28th international conference on vary large data bases. Hong Kong, VLDB Endowment, pp 646–657. doi:10.1016/B978-155860869-6/50063-9
11. Negoita C, Zadeh L, Zimmermann H (1978) Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets Syst 1:3–28
12. Gaurav A, Alhajj R (2006) Incorporating fuzziness in XML and mapping fuzzy relational data into fuzzy XML. In: Proceedings of the 2006 ACM symposium on applied computing. ACM, Dijon, pp 456–460. doi:10.1145/1141277.1141386
13. Turowski K, Weng U (2002) Representing and processing fuzzy information-an XML-based approach. Knowl Based Syst 15(1):67–75. doi:10.1016/S0950-7051(01)00122-8
14. Tekli J, Chbeir R, Traina AJ, Traina C, Fileto R (2015) Approximate XML structure validation based on document- grammar tree similarity. Inf Sci 295:258–302
15. Tekli J, Chbeir R (2012) A novel XML document structure comparison framework based-on sub-tree commonalities and label semantics. Web Semant 11:14–40. doi:10.1016/j.websem.2011.10.002
16. Algergawy A, Nayak R, Saake G (2010) Element similarity measures in XML schema matching. Inf Sci 180(24):4975–4998. doi:10.1016/j.ins.2010.08.022
17. Wojnar A, Mlýnková I, Dokulil J (2010) Structural and semantic aspects of similarity of document type definitions and XML schemas. Inf Sci 180(10):1817–1836
18. Sabbah T, Selamat A, Ashraf M, Herawan T (2014) Effect of thesaurus size on schema matching quality. Knowl Based Syst 71:211–226. doi:10.1016/j.knosys.2014.08.002
19. Ma ZM, Yan L (2007) Fuzzy XML data modeling with the UML and relational data models. Data Knowl Eng 63(3):972–996. doi:10.1016/j.datak.2007.06.003
20. Nicol G, Wood L, Champion M et al (2001) Document object model (DOM) level 3 core specification. W3C Work Draft 13:1–146
21. Cohen W W, Ravikumar P, Fienberg S E (2003) A comparison of string distance metrics for name-matching tasks. In: Kdd workshop on data cleaning and object consolidation, vol 3. Washington, pp 73–78
22. Lin D (1998) An information-theoretic definition of similarity. In: Proceedings of the international conference on machine learning. Madison, pp 296–304
23. Levenshtein VI (1966) Binary codes capable of correcting deletions. Insertions Revers Sov Phys Doklady 6:707–710
24. Navarro G (2001) A guided tour to approximate string matching. ACM Comput Surv 33(1):31–88
25. Marie A, Gal A (2008) Boosting schema matchers. In: Proceedings of the OTM 2008 confederated inter. Conferences. Springer, Monterrey, pp 283–300
26. XML Data Repository. http://www.cs.washington.edu/research/xmldatasets/
27. Sorrentino S, Bergamaschi S, Gawinecki M, Po L (2010) Schema label normalization for improving schema matching. Data Knowl Eng 69(12):1254–1273. doi:10.1016/j.datak.2010.10.004