Available online at www.sciencedirect.com

ScienceDirect

Fuzzy Sets and Systems ••• (••••) •••–•••

FUZZY sets and systems

www.elsevier.com/locate/fss

Heterogeneous fuzzy XML data integration based on structural and semantic similarities

Zongmin Ma [a,*], Zhen Zhao [b], Li Yan [a]

[a] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu 211106, China
[b] College of Information Science and Technology, Bohai University, Jinzhou, Liaoning 121013, China

Received 23 February 2017; received in revised form 6 April 2018; accepted 30 April 2018

Abstract

Web data integration has become a crucial requirement for Web data management. A considerable number of approaches for integrating Extensible Markup Language (XML) data from heterogeneous data sources have been proposed. Yet these existing approaches are not fit for integration of fuzzy XML data because of their fuzzy characteristics. In this article, we provide a framework to deal with fuzzy XML document integration. Firstly, we propose a new fuzzy XML tree model. Secondly, we present an effective algorithm based on the tree edit distance to identify the structural and semantic similarities between the fuzzy documents represented in the proposed fuzzy XML tree model. Thirdly, we propose an integration strategy that is applied to integrate the fuzzy documents from different data sources. Finally, we conduct experiments to demonstrate that our approach can efficiently integrate fuzzy XML documents.
© 2018 Published by Elsevier B.V.

Keywords: Heterogeneous data; Integration; Fuzzy XML; Structural similarity; Semantic similarity

1. Introduction

With the rapid development of the Internet, Extensible Markup Language (XML) has been developed as the de facto standard for representing the ever-increasing amount of data available on the Web. The main advantage of XML is its openness based on specifications and standards. The flexibility of XML allows documents to be extended to describe content of any size. Many organizations and enterprises have applied XML as a data description and storage model in their Web-based applications. The need for integrating data from different online application systems has rapidly increased with cooperation between institutions [1–4]. XML data integration is a crucial and difficult task because of its complexity and flexibility characteristics. XML data being integrated may reside in distributed and heterogeneous data sources. As a result, data in different sources may have different representations and may refer to the same entity

* Corresponding author at: College of Computer Science & Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China.
E-mail address: zongminma@nauu.edu.cn (Z. Ma).

https://doi.org/10.1016/j.fss.2018.04.018
0165-0114/© 2018 Published by Elsevier B.V.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

in the real world. That is, data from different data sources are potentially conflicting and overlapping. This increases the difficulty of XML data integration. XML data integration does not mean directly putting heterogeneous XML data sources together. To succeed in integrating XML data, it is necessary to identify and remove these conflicts and then eliminate overlapping data in the integrated document.

XML data integration has received significant attention from both the academic community and the industrial community because of its significance to many practical applications. One approach to XML data integration is to build a data warehouse, which is an integrated and unified data repository generated from heterogeneous data sources. Its central goal is to facilitate the process of identifying and reconciling conflicts and overlapping of attribute values extracted across local XML data sources. Since the hierarchical structure of XML documents is heterogeneous, a small difference in the document structures may increase the difficulty of XML document integration.

Current integration approaches for XML data often assume that XML data denote deterministic objects in the real world. In fact, information is often fuzzy in various practical applications. There have been some efforts in fuzzy XML data processing. Several fuzzy XML data models have been proposed to represent fuzzy data [55]. Let us look at an example. Suppose that we do not have complete knowledge of the academic position of a person named "George," and his academic position may be "Associate Professor" with membership degree 1.0 and "Lecturer" with membership degree 0.7. Then we can use the following fuzzy XML fragments:

```
<name>George</name>
<Dist Type = "disjunctive">
  <Val Poss = 1.0>
    <position>Associate Professor</position>
  </Val>
  <Val Poss = 0.7>
    <position>Lecturer</position>
  </Val>
</Dist>
```

In this example, the fuzzy XML document for George represents the fuzzy information on his academic position by using new elements and attributes. With the increase of fuzzy XML representation on the Web, the applications of fuzzy XML integration motivate researchers to investigate and provide various methods for combining heterogeneous fuzzy XML data.

A core problem that needs to be solved in integrating heterogeneous fuzzy XML data is to identify the same or similar elements/attributes and their values. On the basis, the conflicts and overlapping of elements/attributes are to be removed and eliminated, respectively. That is, it is necessary to find conflicts and overlapping of elements/attributes and their value in fuzzy XML data to consolidate multiple fuzzy XML documents into one document. Although entity identification has been extensively investigated in the context of crisp XML data, the proposed approaches cannot be applied to capture the structural and semantic information of fuzzy XML data because of the lack of an effective fuzzy XML structure model.

To integrate fuzzy XML data from heterogeneous data sources, we propose an effective and efficient approach to support fuzzy XML document integration. The main contributions of this work reside in the field of fuzzy XML data management, which are summarized as follows. We take a first step in constructing a new fuzzy XML tree model. This makes it easy to describe fuzzy XML data and capture the structural and semantic information. On this basis, we propose an effective approach to detect the similarity of nodes between fuzzy XML trees. We concentrate on the structural and semantic similarity comparison of heterogeneous fuzzy XML documents that are collected from different data sources. We identify the corresponding information in tree nodes and then decide on the similarity of nodes from different fuzzy XML trees. Relationships between two fuzzy XML documents can be determined from their corresponding nodes. In this way, we can directly detect conflicts and overlapping between fuzzy XML data that are collected from different data sources. The experimental results show that our approach can efficiently perform the integration of fuzzy XML documents.

The rest of the article is organized as follows. After a presentation of related work in Section 2, we propose a fuzzy XML tree model in Section 3. On the basis of the proposed fuzzy XML tree model, we devise an approach to measure

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

the structural and semantic similarity of fuzzy XML data in Section 4. The fuzzy XML data integration strategy is developed in Section 5. Experimental evaluations are presented in Section 6. Finally, Section 7 concludes this article.

2. Related work

2.1. Imperfect XML

Imperfect information extensively exists in many real-world applications. Imperfect properties of information may include *inconsistency*, *imprecision*, *vagueness*, *uncertainty*, and *ambiguity* [18]. Various types of imperfect information have been investigated in the context of XML. A simple model for representing and querying XML with incomplete information is proposed in [5], where a simple framework is devised for acquiring, maintaining, and querying XML documents with incomplete information.

There have been some achievements in the area of probabilistic XML [5–13]. A probabilistic XML model to manage probabilistic data is introduced in [6], in which an implementation of XML query operations is devised and the efficiency of this implementation as well as the probabilistic XML data management is demonstrated by use of a brief case study. In [7], by enumeration of all possible worlds in different subtrees, an uncertain XML data model is applied to store possible appearances of the database instead of one actual appearance. A formal definition of the uncertain XML structure is given, and the semantics behind the data model is discussed. A model for probabilistic semistructured data is proposed in [8], and this has two semantics that are probabilistically coherent. The representation of probabilistic XML models is also investigated in [9–11], in which the abstract framework of representing probabilistic XML is described as a labeled tree with ordinary nodes and distributional nodes. The ordinary nodes represent the actual data, and distributional nodes define the process of generating random XML documents. In [12], uncertain XML data are represented with three probabilistic XML models. The first model assumes an independence among the probabilistic junctions, and the second model encodes probabilistic dependencies. The third model combines these two models and hence is the most general. In [6,13], a concrete model of p-documents, namely, the ProTDB model, is constructed to manage probabilistic XML data.

There have been several efforts in fuzzy XML data representation and processing [14–21]. To model fuzzy data in XML documents, an approach for representing fuzzy data in XML documents by use of both possibility theory and similarity relations is introduced in [14]. It is shown in [14] how to map fuzzy data from a fuzzy relational database into a fuzzy XML document. In [15], an approach for dealing with fuzzy XML data is proposed, in which appropriate document type definitions (DTDs) are defined and a formal syntax for fuzzy data types is introduced. It is shown in [15] how to exchange fuzzy information between fuzzy XML DTDs in various application systems. An XML Schema definition for representing fuzzy data is proposed in [16]. In [17], a fuzzy object-oriented modeling (FOOM) technique schema based on XML is developed to model requirement specifications and incorporate the notion of stereotype to facilitate the modeling of imprecise requirements. To model imprecise requirements with XML, the FOOM schema is formulated on the basis of the key features: fuzzy set, fuzzy attribute, fuzzy rule, and fuzzy association. Fuzziness in XML documents and schemas is investigated in [18,19], and fuzzy XML representation models are developed. Two levels of fuzziness in an XML document are identified in [18], which are membership degrees associated with elements and fuzzy sets related to element values. In [20,21], an extension of XML is made that combines indefiniteness in the values of XML and indefiniteness in the structure of XML into a single fuzzy XML extension. A tool is then developed to define arbitrary membership functions for working with XML, XSD, and DTD documents, and prioritized fuzzy XQuery extension queries. One can refer to [55] for a survey of modeling fuzzy data in XML data model.

2.2. Entity identification of XML

A core task in XML data integration is to find all entities that refer to the same real-world object in XML documents. Some methods for entity identification in XML documents have been proposed [22–32]. In [22], to resolve structural conflicts in integrating XML schemas, a data model called *ORA-SS* is applied to capture implicit semantics. The framework for entity matching in [23] perform a given matching task by using or not using training data. On the basis of an extended adjacency matrix, a method for computing the structure and semantic similarity of XML documents is proposed in [24]. In [25], on the basis of an "XML aware" edit distance between ordered labeled trees, a structural similarity metric for XML documents is developed. On the basis of an expressive XML global schema,

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

a framework for XML data integration systems that defines an identification function that aims at globally identifying nodes from different sources is proposed in [26]. In [27], on the basis of the leaf nodes of each pair of subtrees from XML document trees, a leaf clustering algorithm is proposed to identify similarity. An approach to entity identification in XML documents is introduced in [28] that is based on approximate joins. In [29], two strategies are proposed to resolve underlying conflicts in XML data/schema integration, which are conflict eliminating and conflict preserving, respectively. The XML matching approach in [30] introduces a template called *XML Matcher Template*, which describes the main components of an XML matcher. Besides, some approaches (e.g., [31,32]) try to integrate semantic and syntactic assessment into the edit distance computation. This enables one to analyze the structure of XML documents more precisely and obtain more realistic results.

Most of the existing approaches for identifying entities in an XML document apply the techniques for finding the edit distance between tree structures. In addition, a suitable data model must be established to better identify or match entities in XML documents.

2.3. Data integration of XML

Some methods for XML integration have been proposed [26,33–35]. On the basis of the use of Skolem functions and XPath, a proposal for identifying XML instances and solving attribute value conflicts between XML documents is proposed in [33]. A data model is developed in [34] to facilitate the resolution of attribute value conflicts by explicitly representing the conflicts in the integrated schema. To integrate XML documents created by distinct sources in a highly dynamic environment, the approach in [35] integrates similarity information from the content of elements with information from the structure of documents.

There have been few efforts in uncertain XML data integration. In [7], an integration approach is introduced in the context of the uncertain model that is used to capture the remaining integration possibilities. On the basis of the notion of a probabilistic XML tree, an approach is presented for determining a logical semantics for queries on probabilistic XML data in [36]. Inconsistent data in probabilistic XML data integration are reconciled in [37]. In [38], an approach to represent and integrate uncertain XML data is proposed with the evidence theory, where uncertainty is due to conflict values for elements that represent the same real world. The problem of integrating several Web data sources under uncertainty and dependencies is explored in [39]. A basic framework for detecting and reconciling conflicts and overlaps in fuzzy XML documents collected from different data sources is developed in [40].

The flexibility of XML provides opportunities to alleviate some difficulties in representing fuzzy data. Yet this flexibility also introduces new challenges in integrating multiple fuzzy XML data sources.

3. Fuzzy XML tree model

3.1. Fuzzy XML documents

To represent fuzzy data on the Web, a fuzzy XML model is introduced in [18,41]. In this representation model, the fuzziness of XML documents occurs at two levels: the first is the fuzziness in elements, and we use membership degrees associated with such elements; the second is the fuzziness in attribute values of elements, and we use fuzzy sets to represent such values. A fuzzy set that represents a fuzzy element value has two types of interpretations: *disjunctive* semantics and *conjunctive* semantics.

Using the fuzzy XML document model proposed in [18], we give two fragments of fuzzy XML documents shown in Figs. 1 and 2, respectively. One can refer to [18,41] for more details. In the fuzzy XML document, the *Type* attribute node as a child of element *Dist* is applied to indicate the interpretation type of a fuzzy set, having values of *disjunctive* or *conjunctive*. The *Poss* attribute node as a child of element *Val* indicates the membership degree of a given element in the fuzzy XML document, having values of [0, 1]. In addition, each *Dist* element node has an element node as its parent node and one or more *Val* element nodes as its child nodes.

Let us look at the fuzzy XML document shown in Fig. 1. Fuzzy construct *Poss* is adopted together with fuzzy construct *Val* to specify the membership degree of a given element in the fuzzy XML document. This membership degree is represented with the pair <*Val Poss*> and </*Val*>. Lines 7–9 in Fig. 1, for example, indicate the membership degree that George is a lecturer is 0.7. Fuzzy construct *Dist* is used to represent a fuzzy set as an element value. A *Dist* element may have multiple *Val* elements as its children, and each *Val* has an associated *Poss* as its child.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

```
1.  <Teaching>
2.      <Teacher Tid = "007102">
3.        <Dist Type = "disjunctive">
4.          <Val Poss = 1.0>
5.            <Position>Associate Professor</Position>
6.          </Val>
7.          <Val Poss = 0.7>
8.            <Position>Lecturer</Position>
9.          </Val>
10.       </Dist>
11.       <Name>George</Name>
12.     </Teacher>
13.     <Student Sid = "20130111">
14.       <Dist Type = "conjunctive">
15.         <Val Poss = 0.6>
16.           <Email>Mary@hotmail.com</Email>
17.         </Val>
18.         <Val Poss = 1.0>
19.           <Email>Mary@gmail.com</Email>
20.         </Val>
21.       </Dist>
22.       <Name>Mary</Name>
23.     </Student>
24. </Teaching>
```

Fig. 1. First fragment of fuzzy XML document.

Lines 3–10 in Fig. 1, for example, indicate the value of George's academic position is a fuzzy set, in which George is an associate professor or a lecturer with membership degrees 1.0 and 0.7, respectively. Clearly George cannot be both a lecturer and an associate professor. So, in line 3 in Fig. 1, <Dist Type = "disjunctive"> is applied to explicitly indicate its disjunctive interpretation of the fuzzy set. Lines 14–21 in Fig. 1 indicate the value of Mary's e-mail address is a fuzzy set, where the e-mail address of Mary is Mary@hotmail.com or Mary@gmail.com with membership degrees 0.6 and 1.0, respectively. It is possible that Mary may have one e-mail address or two e-mail addresses. So in line 14 in Fig. 1, <Dist Type = "conjunctive"> is applied to explicitly indicate its conjunctive interpretation of the fuzzy set. In this case, the membership degrees can be understood as certainty degrees.

3.2. Fuzzy XML document tree models

With use of the Document Object Model (DOM) [42], an XML document can be represented as a rooted ordered labeled tree. The nodes of the DOM tree represent XML elements, and are labeled with the corresponding element label names. The nodes are ordered following their orders of appearance in the document. Attributes usually appear as the children of their encompassing element nodes, which are sorted by attribute names.

Fuzzy XML documents are clearly different from crisp XML documents because fuzzy XML documents contain new attribute and element types, which are the two attributes *Poss* and *Type* and the two elements *Val* and *Dist*. So a fuzzy XML document cannot be transformed to a DOM tree directly. To capture the structural similarity of fuzzy XML documents, we need to establish a suitable model. Here we propose a *f*uzzy *X*ML document *t*ree *m*odel (FXTM) to represent fuzzy XML documents.

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

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

```
1.  <Teaching>
2.      <Teacher Tid = "007102">
3.        <Dist Type = "disjunctive">
4.          <Val Poss = 1.0>
5.            <Position>Associate Professor</Position>
6.          </Val>
7.          <Val Poss = 0.6>
8.            < Position>Professor</Position>
9.          </Val>
10.       </Dist>
11.       <Name>George</Name>
12.       <Office>B208</Office>
13.     </Teacher>
14.     <Student Sid = "20130425">
15.       <Dist Type = "conjunctive">
16.         <Val Poss = 0.7>
17.           <Email>John@hotmail.com</Email>
18.         </Val>
19.         <Val Poss = 1.0>
20.           <Email>John@gmail.com</Email>
21.         </Val>
22.       </Dist>
23.     <Name>John</Name>
24.   </Student>
25. </Teaching>
```

Fig. 2. Second fragment of fuzzy XML document.

(a) "*NodeLabel*" is the label name of an element/attribute node.

(b) "*NodeDepth*" is the nesting depth of an element/attribute node in the XML document.

(c) "*NodeFuzzy*" is used to indicate if a node is a fuzzy or a crisp one. If the value of "*NodeFuzzy*" is 1, the corresponding node is a fuzzy one. If the value of "*NodeFuzzy*" is 0, the corresponding node is a crisp one. The value of "*NodeFuzzy*" for a "*Dist*" or "*Val*" node is 0.

(d) "*NodeDistType*" denotes the interpretation type of fuzzy set (disjunctive or conjunctive semantics). For a crisp node, the default value of "*NodeDistType*" is Null.

(e) "*NodePoss*" is the membership degree of a node or an element value. For a crisp node, the default value of "*NodePoss*" is Null.

An FXTM tree represents a set of possible real-world entities. Entity identification of fuzzy XML documents means a similarity evaluation of their corresponding FXTM trees. To reduce unnecessary computation in the similarity evaluation, the values of elements/attributes are generally disregarded for the structural similarity measure of XML documents. But, in this article, we keep the values of elements/attributes both for data integration and for node similarity calculation when an FXTM tree is constructed from a fuzzy XML document. Note that in a fuzzy XML document, the sibling nodes of *Poss* and *Type* nodes may be element nodes or attribute nodes. So we need to store fuzzy values of *Poss* and *Type* nodes into "*NodeDistType*" and "*NodePoss*" of sibling nodes of *Poss* and *Type* nodes. Fuzzy XML documents contain four kinds of new elements/attributes (*Type*, *Val*, *Poss*, and *Dist*), and their values contain fuzzy information. We need to transform fuzzy values into the corresponding nodes. We deal with these four types of nodes according to the following steps, and obtain the FXTM when a fuzzy XML document is traversed.

Step 1: We inherit *NodeFuzzy*, *NodeType*, and *NodePoss* values from the parent.

Step 2: For the *Type* node, we copy the value of *Type* node to *NodeType* of siblings of *Type* node and delete *Type* node and its subtree.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

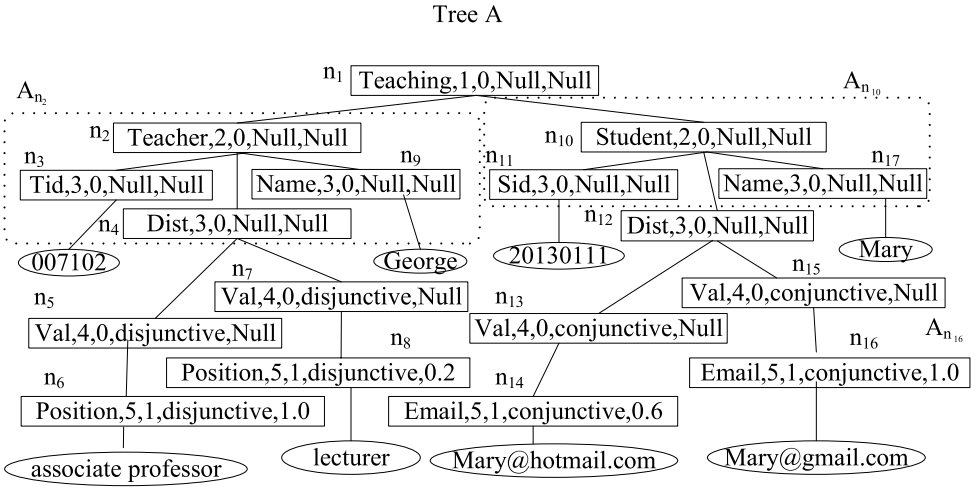

Fig. 3. A fuzzy XML document tree model (FXTM) tree corresponding to the fuzzy XML document in Fig. 1.

Step 3: For the *Val* node, if it has only *Poss* nodes as a child node, we delete the *Val* and its subtree.
Step 4: For the *Poss* node, we copy the value of *Poss* node to *NodePoss* of siblings of *Poss* node and delete *Poss* node and its subtree.
Step 5: For the *Dist* node, if it becomes a leaf node after the above processing, we delete *Dist* node and its subtree.

It can be seen that each node carries the corresponding fuzzy information itself into the FXTM. An FXTM is a rooted ordered tree, in which the nodes represent the elements/attributes of fuzzy XML document and the leaf nodes are values of attributes. The nodes of an FXTM are ordered following their order of appearing in the corresponding fuzzy XML document. The position of a node in the FXTM is consistent with the global order defined over all elements/attributes in the fuzzy XML document. In the FXTM, each node has a unique parent except for the root node, and the edges between nodes represent the containment relationship between these nodes. The descendants of a node include the children of this node as well as the descendants of these children. In the following, we use "node" to represent "element node" or "attribute node."

To sum up, we delete all *Type* and *Poss* attribute nodes in the fuzzy XML trees. Note that the *Dist* and *Val* element nodes cannot be ignored, otherwise the tree structure and depth will be changed, and this will affect the results of tree similarity comparison. The two fuzzy XML documents shown in Figs. 1 and 2 are converted into the two FXTM trees shown in Figs. 3 and 4, respectively. For simplicity, only data values are described as the leaf nodes in Figs. 3 and 4. In addition, two-layer subtrees A_{n_2} and B_{n_2} are marked with a dashed frame in Figs. 3 and 4, respectively.

4. Similarity measures

Similarity measure of fuzzy XML documents is applied to detect possible conflicts and overlaps of fuzzy XML documents. To integrate fuzzy XML documents, we need to find the matching nodes from the fuzzy XML document trees being integrated. Then we use a tree edit distance algorithm to calculate the similarity of the subtree structures according to the matching nodes. For this purpose, we need to identify the node similarity before comparing the subtree structure similarity.

4.1. Node semantic similarity measures

The nodes of fuzzy XML document trees are fundamental items for the similarity measures. So calculating node similarity is one of the key steps in the similarity measures. For two nodes N_1 and N_2 that originate from different fuzzy XML document trees, we use $Sim_{Node}(N_1, N_2)$, which takes a value of [0, 1] to represent their similarity degree. The greater the similarity degree is, the more similar two nodes are. To assess the similarity between a node pair from heterogeneous fuzzy XML documents, the features of nodes as well as their relationships must be fully exploited in

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

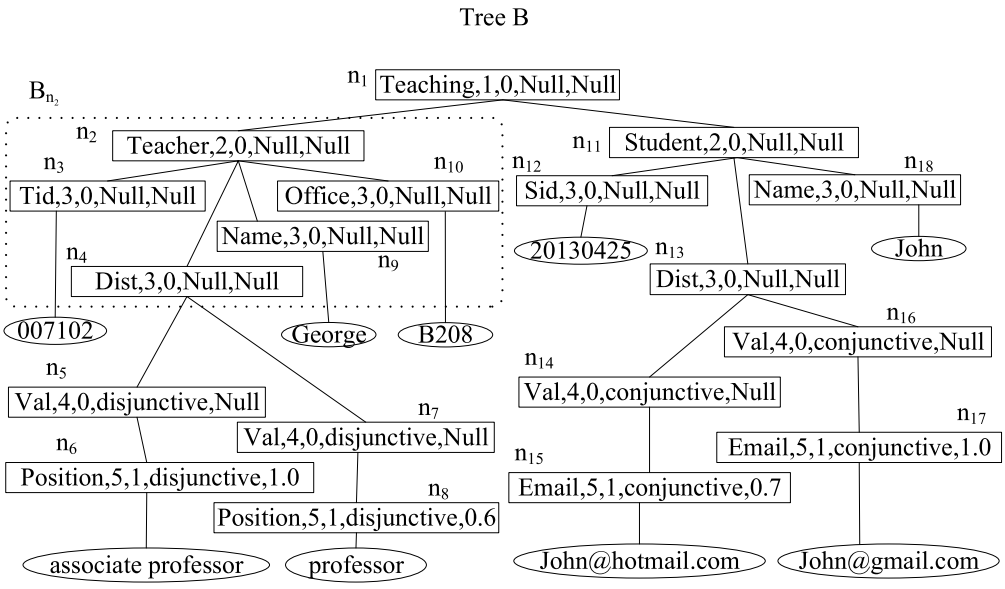

Fig. 4. A fuzzy XML document tree model (FXTM) tree corresponding to the fuzzy XML document in Fig. 2.

the similarity measure. As mentioned in Section 3.2, the features of a node are described with a 5-tuple. Among these features, *NodeLabel*, *NodeType*, and *NodePoss* are the most commonly used features for assessing the similarity of fuzzy XML elements and attributes. Consequently, we present some of the commonly used similarity measures of features according to different features.

4.1.1. Similarity measure of label name

Label names of nodes are regarded as an important information source for node matching. Label names of nodes can be syntactically similar (e.g., "Stu" and "Student") or semantically similar (e.g., "Worker" and "Employee"). In this article, we consider both syntactic similarity and semantic similarity to calculate the similarity between label names of nodes.

Syntactic similarity measure takes label names of two nodes to assess their similarity. There are some commonly used methods for string matching [43–46]. Referring to the Levenshtein edit distance, in this article, we adopt the Wagner–Fischer similarity measure [43]. The Wagner–Fischer similarity measure between two strings is given by the minimum number of operations that are needed to transform one string into the other. The operations include insertion, deletion, and substitution of a single character. Formally, let L_1 and L_2 be two label names of nodes. The syntactic similarity between L_1 and L_2 can be calculated as follows.

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

The semantic similarity measure is another important aspect in evaluating label name similarity of nodes. It is well known that evaluating the semantic similarity of words/expressions may come down to comparing the underlying concepts in the semantic space. Indeed, several methods have been proposed to determine the semantic similarity between concepts in the context of the semantic network [43,47,48]. In this article, we adopt Lin's semantic similarity measurement method in [43]:

$$Sim_{LabelSem}(L_1, L_2, SN) = \frac{2 \log p(L_0)}{\log p(L_1) + \log p(L_2)}, \tag{2}$$

where L_0 is the most specific common ancestor of L_1 and L_2, $p(L_0)$ denotes the occurrence probability of concept L_0, and SN is a weighted semantic network (i.e., a semantic network augmented with concept frequencies) based on a given lexicon (namely, WordNet [49]).

Finally, the label name similarity between two fuzzy XML nodes is evaluated as the weighted average of their syntactic similarities ($Sim_{LabelSyn}$) and semantic similarities ($Sim_{LabelSem}$):

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

$$Sim_{Label}(L_1, L_2) = Sim_{LabelSyn}(L_1, L_2) + (1 - \gamma)Sim_{LabelSem}(L_1, L_2). \quad (3)$$

Here γ is provided as an input weight. By varying coefficient γ, a user can assign an importance to either syntactic or semantic similarities.

4.1.2. Similarity measure of fuzzy information

It is insufficient to only apply node label names in determining node similarity. It is helpful to use fuzzy information sources in calculating node similarity and pruning some false positive matches. Here values of *NodeType* and *NodePoss* are fuzzy information sources, which can be used to determine node similarity.

Note that in a fuzzy XML document tree, the *NodeType* value of a node is either disjunctive or conjunctive. Two nodes having the same *NodeType* value are similar with a high degree. For the nodes having the same *NodeType* value, the *NodeType* similarity is set to 1, while the *NodeType* similarity of the nodes having different *NodeType* values is set to 0. The *NodeType* similarity is represented as $Sim_{Type}(T_1, T_2)$. Then we have

$$Sim_{Type}(T_1, T_2) = \begin{cases} 1 & \text{if } T_1 = T_2, \\ 0 & \text{if } T_1 \neq T_2. \end{cases} \quad (4)$$

Furthermore, in a fuzzy XML document tree, the *NodePoss* value of a node ranges from 0 to 1. For two nodes from different fuzzy XML document trees, their *NodePoss* similarity increases with decreasing difference between their *NodePoss* values. The *NodePoss* similarity is represented as $Sim_{Poss}(P_1, P_2)$. Then we have

$$Sim_{Poss}(P_1, P_2) = 1 - |P_1 - P_2|. \quad (5)$$

A similarity measure can be obtained through evaluation of node features. Basically, a variety of node features along with a variety of measures to assess the similarity should be considered. An overall node similarity is obtained by combination of the similarity measures of different node features. This results in a challenge: how to combine these measures. Without a proper means for combination, the final node similarity measures may fail to reflect the correct similarity relationships. For two nodes N_1 and N_2, their node similarity, denoted $Sim_{Node}(N_1, N_2)$, is evaluated as the weighted average of their feature similarities. Then we have

$$Sim_{Node}(N_1, N_2) = w_L \times Sim_{Label}(L_1, L_2) + w_T \times Sim_{Type}(T_1, T_2) + w_P \times Sim_{Poss}(P_1, P_2), \quad (6)$$

were w_L, w_T, and w_P are provided as weight coefficients, and $w_L + w_T + w_P = 1$.

Generally, the weighting coefficients should be chosen so as to maximize the overall node similarity. The weighting coefficients are usually provided by a human expert, who, on the basis of the scenario at hand, may emphasize label similarity or fuzzy information similarity of the nodes. In addition, the weighting coefficients may be set by application of some well-known techniques such as machine learning to identify the best weights for a given problem [50]. We do not address this issue further because this is out of the scope of this article. In this article, we use only a small number of node features. To acquire more feature similarities, the proposed approach needs to be extended so that it covers more node features. But this does not mean that more node features taking part in the similarity measure must lead to more accurate node similarity.

4.2. Subtree structural similarity measures

To integrate fuzzy XML documents effectively, we need first to determine integration granularity. If integration processing is conducted at the level of nodes, the integration granularity is too small, and it is very difficult to finish integration. A two-layer subtree corresponds to a minimal but complete object. Its parent/child structure corresponds to an element/subelement, element/attribute, attribute/value, or element/value. Moreover, a two-layer subtree contains its structural and semantic information such as the relationship between parent and child nodes.

In this article, we integrate fuzzy XML documents based on two-layer subtrees, which are the integration units. We propose a similarity measure based on the two-layer subtrees. On the basis of the tree edit distance, in this section, we introduce the approach of subtree structural similarity measure to solve the problem of entity identification. For this purpose, we present several definitions of the tree edit operations as follows.

Definition 2 *(Ordered tree).* An ordered tree is a rooted tree, where each node is ordered. For a tree T, we use $T_{[i]}$ to represent the ith node of T in preorder traversal. $T_{[0]}$ designates the root node of T. Assume that node $T_{[0]}$ has k

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

children and these children are uniquely identified from left to right as $T_{[1]}, T_{[2]}, \ldots, T_{[k]}$. Then $T_{[i]}.Label$, $T_{[i]}.Depth$, $T_{[i]}.Fuzzy$, $T_{[i]}.Type$, and $T_{[i]}.Poss$ denote the relevant information of the ith node of T.

Definition 3 *(Two-layer subtree).* Given a tree T with a nonleaf node p, a subtree rooted at p is a two-layer subtree T_p of T if all the child nodes of p are included in T_p. Note that T_p does not include the posterity nodes of p. The nodes of T_p follow the same parent/child edge relationship and node order as they do in T.

In Fig. 3, for example, subtrees A_{n_2} rooted in node n_2 and $A_{n_{16}}$ rooted in node n_{16}, which are identified by a dashed box, are two-layer subtrees. Here A_{n_2} includes only the child nodes of n_2 (i.e., nodes n_3, n_4, and n_9), excluding the posterity nodes of n_2 (e.g., nodes n_5 and n_7, which are the child nodes of n_4). Subtree $A_{n_{10}}$ rooted in node n_{10}, identified by a dashed box, is not a two-layer subtree because node n_{12} is a child node of n_{10} but is not included in $A_{n_{10}}$.

Definition 4 *(Leaf node subtree).* Given a tree T with a nonleaf node l, suppose that all child nodes of l are leaf nodes. The leaf node subtree T_l is a subtree rooted at l, which includes all the child nodes of l. A leaf node subtree is a special case of a two-layer subtree.

In Fig. 3, for example, subtree $A_{n_{16}}$, identified by a dashed box, is a leaf node subtree. Subtree A_{n_2}, identified by a dashed box, is not a leaf node subtree because nodes n_3, n_4, and n_9, which are the child nodes of node n_2, are not leaf nodes.

Definition 5 *(Insert node).* Given a node x, $InsNode(x)$ is an operation of node insertion. It is applied to tree T at its ith node, and yields a tree with nodes $T_{[1]}, \ldots, T_{[i]}, x, T_{[i+1]}, \ldots, T_{[m]}$.

Definition 6 *(Delete node).* Let $T_{[i]}$ be a leaf node of tree T. $DelNode(T_{[i]})$ is an operation of node deletion. It is applied to T at its ith node, and yields a tree with child nodes $T_{[1]}, \ldots, T_{[i-1]}, T_{[i+1]}, \ldots, T_{[m]}$.

Definition 7 *(Edit script).* An edit script is a sequence of edit operations $ES = \langle op_1, op_2, \ldots, op_k \rangle$, where op_i $(1 \leq i \leq k)$ is an edit operation. Applying the edit operations in ES to a tree T according to their order of appearance in ES, we obtain a new tree T'. The cost of ES, denoted $Cost_{\mathrm{ES}}$, is defined as a sum of the costs of all operations in ES.

Definition 8 *(Tree edit distance).* The edit distance between two trees A and B, denoted $TED(A, B)$, is defined as the minimum cost of all edit scripts that can transform A to B; that is, $TED(A, B) = Min\{Cost_{\mathrm{ES}}\}$. For a sequence of different edit scripts, $\{Cost_{\mathrm{ES}}\}$ denotes a set of edit costs of all edit scripts transforming A to B.

The tree edit distance, which is widely used to detect the structural similarity of XML documents, is a natural extension of the string edit distance. Here the distance between two trees is the cost of edit operations that transform one tree to another tree. Generally, each of the edit operations is associated with a nonnegative value as its operation cost. The edit distance in Definition 8 works with general costs. In this article, we use a unit cost instead of an actual cost. A unit cost, which corresponds to a node operation (i.e., inserting or deleting a node), is equal to 1. For two-layer subtrees A and B, the problem of evaluating their structural similarity can be defined as a problem of computing their corresponding tree edit distance. To capture the structural similarities of two-layer subtrees, we introduce the notion of matching nodes between two-layer subtrees.

Definition 9 *(Matching nodes).* Given two-layer subtrees $A = (a_1, \ldots, a_m)$ and $B = (b_1, \ldots, b_n)$ as well as a threshold θ, the matching nodes between A and B are defined as a set of pairs $N = \{(a_i, b_j)\}$, where $\{(a_i, b_j)\} \in A \times B$ and $Sim_{Node}(a_i, b_j) > \theta$.

Intuitively speaking, applying the edit distance to whole trees could result in a larger deviation of the similarity measure than applying the edit distance only to the subtrees. To increase the computational accuracy of the similarity measure, we choose two-layer subtrees as the granularity of calculating similarity.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

Definition 10 *(Two-layer subtrees' edit distance, TSED).* Given two-layer subtrees $A = (a_1, \ldots, a_m)$ and $B = (b_1, \ldots, b_n)$, the TSED between A and B is defined as the minimum cost of all edit scripts that transform A to B.

It can be seen from Definitions 9 and 10 that for two subtrees A and B, calculating their edit distance just means finding the minimum number of edit operations that can transform A to B and then identifying the maximum number of matching nodes in A and B. Following Nierman and Jagadish's edit distance algorithm [25] as well as Tekli and Chbeir's improved edit distance algorithm [31], we introduce our TSED algorithm in Algorithm 1. Our algorithm provides a more accurate similarity measure of two-layer subtrees based on the tree edit distance.

Algorithm 1 TSED.

```
Input:  A, B                                  //Two-layer subtrees to be compared
Output: TSED(A, B)                            //Edit distance between A and B
Begin
1    M = Degree(A)                            //The degree of two-layer subtrees of A
2    N = Degree(B)                            //The degree of two-layer subtrees of B
3    Dist[][] = new[0...M][0...N]
4    If (Sim_Node(A[0], B[0]) > θ)            //Node matching
5    {
6      Dist[0][0] = 0;                        //Structurally matching nodes are associated with a cost of 0
7      MatchNodeNumber add 1                  //Counting matching nodes
8    }
9    Else
10   {
11     Dist[0][0] = 1 //Unit costs
12   }
13   For (i = 1 to M) {Dist[i][0] = Dist[i−1][0] + Cost_DelNode(A[i])}     //Cost_DelNode(A[i]) is a cost of 0 or 1
                                              //Total cost of deleting all nodes in the source document tree
14   For (j = 1 to N) {Dist[0][j] = Dist[0][j−1] + Cost_InsNode(B[j])}     //Cost_InsNode(B[j]) is a cost of 0 or 1
                                              //Total cost of inserting all nodes in the destination document tree
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                    {IF (Sim_Node(A[i], B[j]) > θ) {Dist[i−1][j−1], MatchNodeNumber add 1}
21                      else {Dist[i − 1][j − 1] + 1}},
22                    Dist[i−1][j] + Cost_DelNode(A[i])     //Cost_DelNode(A[i]) is a cost of 0 or 1
23                    Dist[i][j−1] + Cost_InsNode(B[j])     //Cost_InsNode(B[j]) is a cost of 0 or 1
24                   }
25       }
26   }
27 Return Dist[M][N]
```

We apply Algorithm 1 to identify the structural similarity of two-layer subtrees. The algorithm starts by counting the total number of matching nodes between A and B (lines 4–7 in Algorithm 1). The algorithm calculates the sum of the costs of deleting every node in the source document tree (line 13 in Algorithm 1) and inserting every node of the destination tree (line 14 in Algorithm 1). Finally, the set of insertion/deletion operations that have the minimum overall cost is identified (lines 15–26 in Algorithm 1).

In short, for source subtree A and destination subtree B, the proposed algorithm recursively goes through them and combines the insertion and deletion operations to identify those of minimum cost. Then we can determine the edit distance that transforms A into B. At the same time, the number of matching nodes between A and B, *MatchNodeNumber*, is identified by the proposed algorithm. Being different from the TSED algorithm, which goes through the two-layer subtrees to identify the minimum edit cost, the algorithms of Nierman and Jagadish [25] and Tekli and Chbeir [31] capture the edit distance of the entire trees. Clearly, the granularity of calculating similarity is different between our algorithm and the algorithms of Nierman and Jagadish [25] and Tekli and Chbeir [31]. In addition, the number of similar nodes is calculated with our algorithm when the edit distance of two-layer subtrees is calculated.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

Now we define an adjusting coefficient α of the tree edit distance and measure how matching nodes affect the edit distance. We calculate α according to the number of matching nodes. To obtain the value of α, we normalize the matching node pair *MatchNodeNumber* values in the interval [0, 1] via the corresponding subtree cardinalities, Max(|A|, |B|). Here |A| and |B| mean the number of nodes in A and B, respectively.

$$\alpha = \frac{1}{1 + \frac{MatchNodeNumber}{\text{Max}(|A|,|B|)}}. \tag{7}$$

- $\alpha = 1$: the maximum value of α is attained when no matching nodes are identified in the given source/destination subtree.
- $\alpha = 1/2$: the minimum value of α is attained when the source subtree is identical to the destination subtree.

We define the minimum value of α to guarantee that the node operations always cost less than the whole subtree operations. That is, inserting or deleting a single node costs less than a subtree. For the subtree with two nodes, its maximum edit cost is 2 and minimum edit cost is 1, which is equivalent to the edit cost of a single node and is half of the maximum edit cost. The edit distance depends on the minimum edit cost. The TSED algorithm goes through the two-layer subtrees to identify the minimum edit cost. The maximum value of α is 1 and the minimum value of α is $1/2$, half its maximum value.

We use $Sim_{FXST}(A, B)$ to represent the similarity degree of two-layer subtrees A and B. Here A and B originate from different fuzzy XML document trees, and FXST means *fuzzy XML subtree*. We use the TSED and the coefficient α to define $Sim_{FXST}(A, B)$ as follows.

$$Sim_{FXST}(A, B) = \frac{|A| + |B| - \text{TSED}(A, B) \times \alpha}{|A| + |B|}. \tag{8}$$

This similarity measure is consistent with the similarity concept in [32] and satisfies the following metric properties:

- $Sim_{FXST}(A, B) \in [0, 1]$.
- $Sim_{FXST}(A, B) = 1$ if A and B are identical.
- $Sim_{FXST}(A, B) = 0$ if A and B have no common characteristics.
- $Sim_{FXST}(A, B) = Sim_{FXST}(B, A)$.
- Similarity increases with the commonality between A and B and decreases with their difference.

The coefficient α in $Sim_{FXST}(A, B)$, which is obtained by means of the TSED, can strengthen the effectiveness of the tree edit distance. Here α is calculated according to the number of matching nodes. The smaller the number of matching nodes is, the bigger the values of both the edit distance and α are.

Example 1. Let us look at two-layer subtrees A_{n_2} in Fig. 3 and B_{n_2} in Fig. 4. To calculate the similarity degree of A_{n_2} and B_{n_2}, we start by executing the TSED algorithm and calculate the edit distance and the number of matching nodes of A_{n_2} and B_{n_2}. We have $\text{TSED}(A_{n_2}, B_{n_2}) = 1$.

- $Dist[0][0] = 0$, having $R(A_{n_2})$ and $R(B_{n_2})$ matching.
- $\text{TSED}(A_{n_2}, B_{n_2}) = 1$, cost of inserting node n_{10} into subtree A_{n_2}.
- $MatchNodeNumber = 4$, number of matching nodes of A_{n_2} and B_{n_2}.
- $Sim_{FXST}(A_{n_2}, B_{n_2}) = \frac{4+5-1\times\frac{1}{1+\frac{4}{5}}}{4+5} = 0.9383$.

4.3. Overall complexities

This section provides a complexity analysis for the similarity measure approach presented in Section 4.2, including *time complexity* and *space complexity*. Let A and B be two two-layer subtrees being compared. First, let us look at the time complexity. For the identification of both structural and semantic similarity between A and B, the overall time complexity of our similarity measure approach can be simplified to $O(|A| \times |B| \times |SN| \times \text{Depth}(SN))$. Here |A|

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

and $|B|$ denote the cardinalities of A and B, respectively, $|SN|$ denotes the cardinality of the semantic network that is exploited for semantic similarity assessment, and Depth(SN) denotes the maximum depth of the semantic network. So our approach is similar to the existing XML-based tree edit distance comparison approaches (e.g., [31]).

Now let us look at the space complexity. Our approach requires memory space usage to store the two-layer subtrees being compared as well as the distance matrixes and the semantic vectors being calculated. The space complexity of our similarity measure approach can be simplified to $O(|A| \times |B|)$, which is similar to the existing XML-based tree edit distance comparison approaches (e.g., [31]).

5. Fuzzy XML data integration

In this section, we propose a data integration framework for reconciling fuzzy XML documents in heterogeneous data sources. The fuzzy XML documents are represented with the FXTM in Section 3.2. The basic idea of our integration framework is that we complete the integration of FXTM trees by integrating two-layer subtrees after resolving semantic and structural conflicts. For this purpose, we define some strategies and rules to integrate different types of two-layer subtrees. We use Figs. 3 and 4 to illustrate the rules for the two-layer subtree integration. We present the types of structural and semantic conflicts and our approaches to resolve the conflicts in Section 5.1. Section 5.2 details the materialization algorithm of our integration processing.

5.1. Resolution of semantic and structural conflicts

A core problem in fuzzy XML document integration is, after identification of the same real-world objects from heterogeneous data resources, to resolve possible incompatibilities and ambiguous representations of these objects that exist in different heterogeneous data resources. Generally, similar subtree components describe the same real-world object. As a result, they can be integrated into a single subtree after possible conflicts have been resolved. According to the semantic and structural relationships between subtree components, we identify three types of conflicts as follows:

1. *Element/attribute naming conflicts*. Two kinds of naming conflicts can be identified: *synonyms* and *homonyms*. The synonym is semantically related to data items that are named differently. The homonym is semantically unrelated to data items that are named equivalently. Synonym elements/attributes have different names but the same definitions. Homonym elements/attributes have the same name but different definitions. Homonym naming conflict may be a scale or datatype difference among the terminal elements. Synonym naming conflict can be resolved by use of a substitution group through semantic similarity measurement or through human interaction. Homonym naming conflicts for the elements are overcome through use of a scale adjustment or a union of disjoint datatypes.

We look at an example of an employee's salary. Suppose that the element *salary* is defined as an integer datatype in document A and a decimal datatype in document B. This conflict can be resolved through a scale adjustment. We can assign "decimal" as the datatype of element *salary* in the integrated document. Again, let us look at an example of an employee's ID. It is possible that *id* is defined as a string in A and is defined as an integer in B. This conflict can be resolved by our defining a new string datatype, which is a union of the datatype that is used in the integrated document.

2. *Attribute value conflicts*. For two subtrees that are identified as the same real-world object, it is possible that their properties have conflicting values. Several approaches have been proposed to handle this kind of attribute value conflict [51,52]. Here the attribute values correspond to the leaf nodes of fuzzy XML document trees. In this article, we consider attribute value conflicts without fuzzy value conflicts (fuzzy value conflicts are discussed in list item 3). The basic idea is to keep all conflicting attribute values. Let V_A and V_B be attribute-value pairs of $A(Att, X_A)$ and $B(Att, X_B)$, respectively, where Att is an attribute, and X_A and X_B are the values of Att. Assume $X_A \neq X_B$ and that there is a conflict between attribute values X_A and X_B. We integrate V_A and V_B to form an attribute-value pair with schema (Att, X), where $X = \{X_A, X_B\}$. Considering the position of teacher "George" in Figs. 3 and 4, for example, attribute "position" has the value {*professor*, *lecturer*} in Fig. 3 and the value {*associate professor*, *professor*} in Fig. 4. After an integration operation, the value of attribute "position" is {*professor*, *associate professor*, *lecturer*}.

3. *Fuzzy value conflicts*. In addition to the conflicts in list items 1 and 2, there is a kind of conflict that is connected to fuzzy values. Fuzzy value conflicts, which occur at the level of nodes, include *inconsistent membership degrees* and *inconsistent fuzzy sets*. Here it is assumed that there are no naming and attribute value conflicts because they can be

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

resolved beforehand. The membership degree conflict can be resolved by use of Zadeh's intersection operator [53]. Let us look at the two-layer subtrees A_{n_6} and B_{n_8} in Figs. 3 and 4, respectively. The fuzzy set conflict can be resolved by use of Zadeh's union operator.

5.2. Integration of fuzzy XML documents

For two fuzzy XML documents that are to be integrated, one is regarded as a *local fuzzy XML document* and the other is regarded as a *foreign fuzzy XML document*. We assume that the local fuzzy XML document and the foreign fuzzy XML document have the same schema so that we can concentrate only on data integration in the following.

The foreign fuzzy XML document is generally regarded as a "new" information source to supplement the local fuzzy XML document. The new information in the foreign fuzzy XML document can be directly added to the local fuzzy XML document if the new information refers to "new" real-world objects that do not occur in the local fuzzy XML document. It is possible that the new information in the foreign fuzzy XML document is similar to the information in the local fuzzy XML document. Similar information may refer to the same real-world object, which exists both in the local fuzzy XML document and in the foreign fuzzy XML document. At this point, we need to integrate similar information together after solving possible conflicts arising in the similar information. In the following, we propose two rules to deal with fuzzy XML document integration.

Rule 1. For the two-layer subtree rooted in node n_{b_i} of the local fuzzy XML document and the two-layer subtree rooted in node n_{f_j} of the foreign fuzzy XML document, we say they are similar if their similarity degree $Sim_{FXST}(n_{b_i}, n_{f_j})$ is greater than the given threshold δ. In other words, these two subtrees contain a "conflicting" pair and they refer to the same real-world object.

Given that two similar subtrees refer to the same real-world object, we need keep only one subtree in the integrated fuzzy XML document. If the two-layer subtrees are subtrees with leaf nodes and the values of the leaf nodes are identical, there is a membership degree conflict, which can be resolved by use of the approach given in Section 5.1. If the two-layer subtrees are subtrees with nonleaf nodes and the similarity degree of the two-layer subtrees, which are respectively rooted in child nodes n_{b_i} and n_{f_j}, is greater than the given threshold δ, we need to recursively compare the subtrees until the foreign fuzzy XML document tree is completely traversed. When the two-layer subtrees are subtrees of nonleaf nodes and the similarity degree of the two-layer subtrees, which are respectively rooted in child nodes n_{b_i} and n_{f_j}, is less than δ, the two-layer subtree rooted in n_{b_i} of the local fuzzy XML document and the two-layer subtree rooted in n_{f_j} of the foreign fuzzy XML document only have structural similarity. Then the whole subtree rooted in n_{f_j} should be added to the local fuzzy XML document tree as a sibling subtree of n_{b_i}.

Example 2. Let us look at the two fuzzy XML document trees shown in Figs. 3 and 4. We can calculate the similarity degree between the two-layer subtree rooted in node n_3 of the foreign tree B and the two-layer subtree rooted in node n_3 of the local tree A. Their similarity degree is greater than the given threshold. Here these two two-layer subtrees are a kind of subtree with leaf nodes and they refer to the same real-world object. Their integration results in a two-layer subtree that has membership degree Null, and then the two-layer subtree rooted in node n_3 of B is deleted. In addition, the two-layer subtree rooted in node n_{11} of B is similar to the two-layer subtree rooted in node n_{10} of A. But the child nodes of these two two-layer subtrees are not similar. At this point, we recognize that the two-layer subtree rooted in node n_{11} of B is a "new" one. As a result, the integration is completed by addition of it to A as the sibling of node n_{10} of A, and the two-layer subtree rooted in node n_{11} of B is finally deleted. The result of integrating the two fuzzy XML document trees A and B in Figs. 3 and 4 is shown in Fig. 5.

Rule 2. For the two-layer subtree rooted in node n_{b_i} of the local fuzzy XML document and the two-layer subtree rooted in node n_{f_j} of the foreign fuzzy XML document, we say they are not similar if their similarity degree $Sim_{FXST}(n_{b_i}, n_{f_j})$ is less than the given threshold δ (i.e., these two subtrees refer to different real-world objects).

The two-layer subtree rooted in node n_{b_i} of the local fuzzy XML document and the two-layer subtree rooted in node n_{f_j} of the foreign fuzzy XML document do not refer to the same real-world object no matter whether they are

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

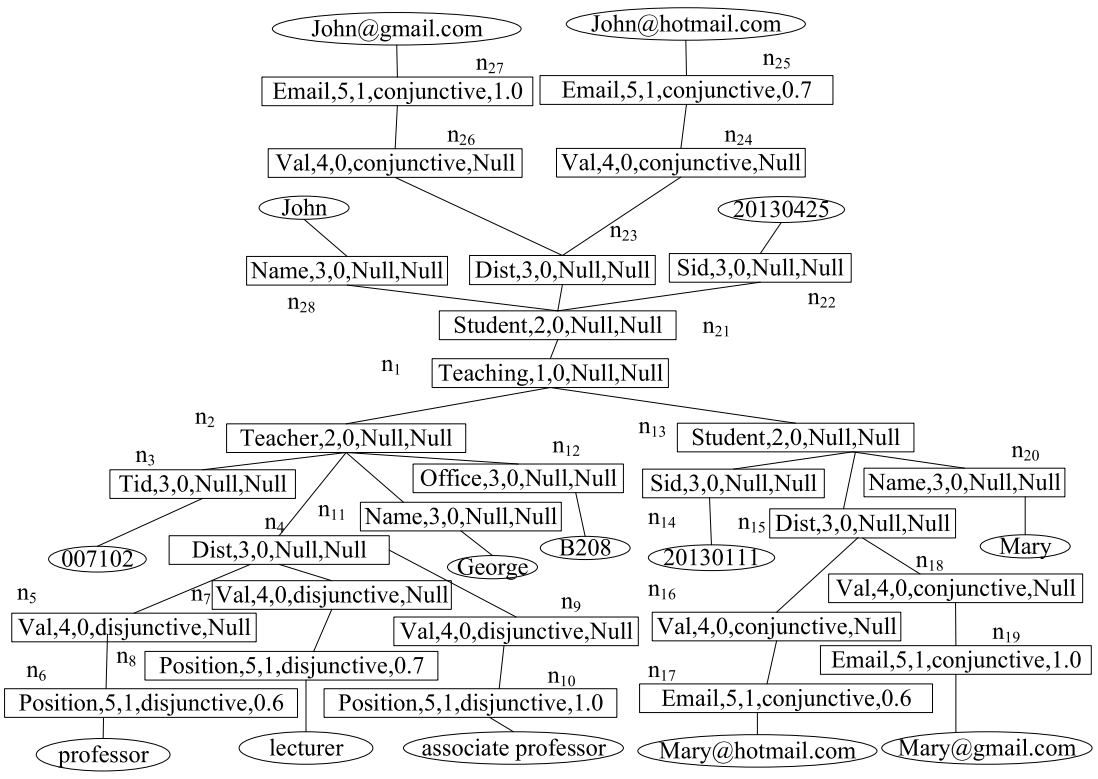

Fig. 5. A fuzzy XML document tree model (FXTM) tree corresponding to the integrated fuzzy XML document.

subtrees with leaf nodes or nonleaf nodes. As a result, the whole subtree rooted in n_{f_j} should be added to the local fuzzy XML document tree as a sibling subtree of node n_{b_i}, and then the two-layer subtree rooted in node n_{f_j} of the foreign fuzzy XML document should be deleted.

Example 3. Let us look at the two fuzzy XML document trees shown in Figs. 3 and 4. We can calculate the similarity degree between the two-layer subtree rooted in node n_{10} of the foreign tree B and the two-layer subtree rooted in nodes n_3, n_4, and n_9 of the local tree A. Their similarity degree is less than the given threshold. Then we recognize that the two-layer subtree rooted in node n_{10} of B is a "new" one. As a result, the integration is completed by addition of it to A as the sibling of node n_9 of A.

The data integration algorithm IFXD based on the rules mentioned above takes two parameters FD_b and FD_f. The returned integration result is an integrated fuzzy XML document FD_t.

Integration of two fuzzy XML documents involves integrating all two-layer subtrees of both fuzzy XML document trees. When the foreign fuzzy XML document tree is traversed, according to the integration rules and the integration algorithm in Algorithm 2, a two-layer subtree of the foreign fuzzy XML document can be added to the local fuzzy XML document as a new two-layer subtree because there is not a similar two-layer subtree that can be found in the local fuzzy XML document. The integration rules can guarantee that all new information in the foreign fuzzy XML document can be added to the local fuzzy XML document without loss of information.

In general, an integrating sequence may produce multiple possible results for all subtrees referring to either the same or different real-world objects. Also, different integrating sequences may lead to different integration results. For the integration of teachers' position information in Figs. 3 and 4, for example, we can obtain different integration results, no matter which one is selected as the foreign tree. Note that both results are correct. By application of Algorithm 2, the integrated FXTM tree from the fuzzy XML documents in Figs. 1 and 2 is presented in Fig. 5.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

Algorithm 2 IFXD.

```
Input: FD_b, FD_f                                   //Fuzzy XML documents FD_b and FD_f that are to be integrated
Output: FD_t                                        //Integrated Fuzzy XML documents FD_t
Begin
1   FT_b = FXTM(FD_b)                               //FXTM FT_b from FD_b
2   FT_f = FXTM(FD_f)                               //FXTM FT_f from FD_f
3   For (each n_{b_i} in FT_b)                      //Two-layer subtree rooted in node n_{b_i}
4   {For (each n_{f_j} in FT_f)                     //Two-layer subtree rooted in node n_{f_j}
5     { do while (!Empty(FT_f))
6       {If (Sim_FXST(n_{b_i}, n_{f_j}) > δ)                    //Two-layer subtree similarity degree computation
7           If (Isleafnodesubtree (n_{b_i}, n_{f_j}))           //Subtree rooted in n_{b_i}, n_{f_j} is leaf node subtree
8               If (Sim_Node(n_{b_i} → child, n_{f_j} → child) > θ)  //Value equality
9                   {n_{b_i} → child.Poss = min(n_{b_i} → child.Poss, n_{f_j} → child.Poss)
10                   deltreesubtree(n_{f_j});}                  //Deleting subtree rooted in n_{f_j} of the foreign tree
11              Else                                            //Inserting leaf nodes
12                  IFXD(n_{b_i}, n_{f_j} → sibling)            //Integrate sibling subtree of n_{f_j}, n_{f_j} → sibling
13          Else
14              If (Sim_FXST(n_{b_i} → child, n_{f_j} → child) > δ & !Isleafnodesubtree (n_{b_i} → child, n_{f_j} → child))
15                  IFXD(n_{b_i} → child, n_{f_j} → child)       //Integrate child subtree of n_{b_i}, n_{f_j}
16              Else                                            //n_{f_j} added as a sibling of n_{b_i}
17                  {InsertSub-tree(n_{b_i} → common parent, n_{f_j}); deltreesubtree(n_{f_j});}
18        Else
19              If (n_{b_i} exist sibling)
20                IFXD(n_{b_i} → sibling, n_{f_j})              //Integrate sibling subtree of n_{b_i}, n_{b_i} → sibling
21              Else
22                {InsertSub-tree(n_{b_i} → parent, n_{f_j}); deltreesubtree(n_{f_j});}   //n_{f_j} added as a sibling of n_{b_i}
23      }
24    }
25  }
26  FD_t = Transform (FT_b) //FXTM tree is converted to fuzzy XML document
27 Return FD_t
```

5.3. Overall complexities

The complexity of the integration algorithm presented in Section 5.2 includes time complexity and space complexity. First, let us look at the time complexity of our integration algorithm. Suppose that we integrate two fuzzy XML document trees A and B. Then the time complexity of our integration algorithm is simplified to $O(|A|^2 \times |B|^2)$, in which $|A|$ and $|B|$ denote the cardinalities of A and B, respectively.

Now let us look at the space complexity of our integration algorithm. Our approach requires memory space to store the fuzzy XML document trees being integrated as well as the distance matrixes and the semantic vectors that are calculated. The space complexity of our integration algorithm is simplified to $O(|A| \times |B|)$.

6. Experimental evaluations

To evaluate the effectiveness and efficiency of our approach for heterogeneous fuzzy XML data integration and further investigate the impact of different parameters on them, we conducted some experiments with data sets and report the experimental results. We first present several evaluation criteria and data sets we applied in our experiments. Then we present our experimental results. Our experiments were implemented in JDK 1.6 and performed on a system with an Intel Core i5 processor, with 4 GB RAM, and running on Windows 7.

6.1. Evaluation criteria

Basically, we evaluate our integration approach from two aspects: the first one is that of output quality and the second one is that of execution performance. Output quality is applied to measure how well the result data after integration with our integration approach can represent the source data. For this purpose, we compare the experimental

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

results against the real results derived manually. Execution performance is applied to measure how fast our integration approach runs. We assess execution time and scale-up performance.

Concerning integration quality, we consider three performances: *Correctness*, *Completeness*, and *F-measure*. Following the work of Dalamagas et al. [54], we introduce notation that is applied in computing the effectiveness measure as follows:

- A is the number of fuzzy XML documents/elements that are correctly integrated by the integration system.
- B is the number of fuzzy XML documents/elements that are incorrectly integrated by the integration system.
- C is the number of fuzzy XML documents/elements that should be integrated but are not actually integrated by the integration system.

With A, B, and C, now we can define *Correctness*, *Completeness* and *F-measure*.

Correctness is applied to determine the correctness degree of the integration results. Formally, it is defined as the number of the correctly integrated documents/elements divided by the total number of the integrated documents/elements:

$$Correctness = \frac{A}{A+B}. \tag{9}$$

Completeness is applied to assess the completeness degree of the integration results. Formally, it is defined as the number of the correctly integrated documents/elements divided by the number of fuzzy XML documents/elements that should have been integrated.

$$Completeness = \frac{A}{A+C}. \tag{10}$$

Neither *Correctness* nor *Completeness* alone can accurately assess the output quality. So, we use the notion of *F-measure* to assess the output quality. *F-measure* is the weighted harmonic mean of correctness and completeness:

$$F\text{-}measure = \frac{2 \times Correctness \times Completeness}{Correctness + Completeness}. \tag{11}$$

All three measures presented above have a maximum of 1.0 and a minimum of 0.0. In this article, the three measures mentioned above are actually average correctness, average completeness, and average F-measure, which refer to the average values of measures over all integration tasks, respectively. The effectiveness of data integration is commonly determined by the standard measures of correctness, completeness, and F-measure. Generally speaking, high correctness, completeness, and F-measure mean a good data integration approach.

6.2. Data sets

We apply several data sets[1] based on synthetic and real XML documents for experimental evaluations.

6.2.1. Synthetic data sets

Our synthetic data sets originate from two different domains "Auction" and "University," denoted D_1 and D_2, respectively. Each domain has its corresponding DTDs. Before our experiments are performed over D_1 and D_2, we first add fuzzy nodes to these DTDs manually. We adopt the XML documents generator[2] to generate the corresponding fuzzy XML documents. Each DTD is applied to generate 50 corresponding documents.

The characteristics of D_1 and D_2 are summarized in Table 1.

6.2.2. Real data sets

We take NASA.xml and DBLP.xml as our real data sets, denoted D_3 and D_4, respectively. For our experiments on fuzzy XML data integration, we use a random data generation method to transform the data in D_3 and D_4 into fuzzy

[1] All data sets are collected from http://www.cs.washington.edu/research/xmldatasets/.

[2] http://www.stylusstudio.com/xml-generator.html.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

Table 1
Characteristics of the synthetic data sets.

Domain	DTD	Number of documents	Number of elements	Number of fuzzy elements	Average depth	Data set
Auction	ebay.dtd	50	156	15	3.75641	D_1
	ubid.dtd	50	342	30	3.76608	
	yahoo.dtd	50	342	30	3.76608	
University	reed.dtd	50	10546	1000	3.19979	D_2
	uwm.dtd	50	66729	5000	3.95243	
	wsu.dtd	50	74557	10000	3.15787	

Table 2
Characteristics of the real data sets.

File name	Number of documents	Number of elements	Number of fuzzy elements	Average depth	Data set
NASA.xml	150	476646	5000	5.58314	D_3
DBLP.xml	150	3332130	10000	2.90228	D_4

data. That is, we artificially add fuzzy information to D_3 and D_4, and then they are converted into fuzzy XML data sets. Moreover, to examine the impact of document sizes, we randomly split each fuzzy XML file into 150 documents. Split documents conform to the same schema as the original fuzzy XML file. Their sizes are determined from 100 KB to 10 MB according to arithmetic progression distributions.

The characteristics of D_3 and D_4 are summarized in Table 2.

6.3. Experimental results

We wish to find an optimal parameter configuration in our experiments. We observe how parameters θ (see Definition 9) and δ (see Rule 1 and Algorithm 2) impact on the measure of the integrated data quality (e.g., *Correctness*, *Completeness*, and *F-measure*). In this scenario, the synthetic data set D_1 is used as the experimental data set. In our experiments, we choose the fuzzy XML documents whose pairwise similarity degrees are greater than the given threshold θ and then integrate them together.

Given n fuzzy XML documents, we construct a fully connected graph G with n vertices and $[n \times (n-1)]/2$ weighted edges. The weight of an edge corresponds to the similarity degree between the connected vertices. According to Tekli and Chbeir's edit distance algorithm [31] as well as our TSED algorithm, the similarity degrees of the connected fuzzy XML documents can be obtained by traversal of these fuzzy XML document trees. Consequently, similar fuzzy XML documents are identified by removal of the edges with weights that are less than θ. Such similar fuzzy XML documents can be integrated. First, each pair of similar fuzzy XML documents is integrated over the synthetic data set D_1 by use of our approach. We perform a series of integration tasks by varying the threshold in the interval $[0, 1]$. Second, for the integration results with different thresholds, we apply the same query posed to them and calculate their correctness, completeness, and F-measure as shown in Figs. 6 and 7. It is shown in Figs. 6 and 7 that inconsistent documents are gradually integrated on the same data sets when the thresholds vary from 0 to 1. Clearly our approach yields different integration qualities with different threshold values of θ and δ. The main reason is that the numbers of identified nodes/subtrees, which refer to the same real-world objects, change when the thresholds change. Basically, it is very hard to find right matches for a higher threshold (e.g., threshold 0.9 is set). After synthetically considering correctness, completeness, and F-measure, we set $\theta = 0.5$ and $\delta = 0.7$ to obtain an overall optimization in integration quality.

To test the effectiveness of our approach, we evaluate the qualities of data integration over different data sets. Here, for a given data set, its integration quality is obtained by our averaging the separate qualities of data integration with different thresholds. It is shown in Fig. 8 that the integration qualities over D_3 and D_4 are better than the integration qualities over D_1 and D_2. This is mainly because the fuzzy XML documents in the real data sets are more homogeneous than the fuzzy XML documents in the synthetic data sets. On the other hand, data integration

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

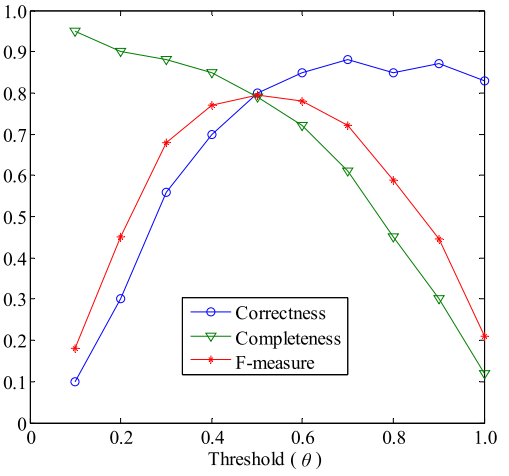

Fig. 6. Quality of integrated data based on θ.

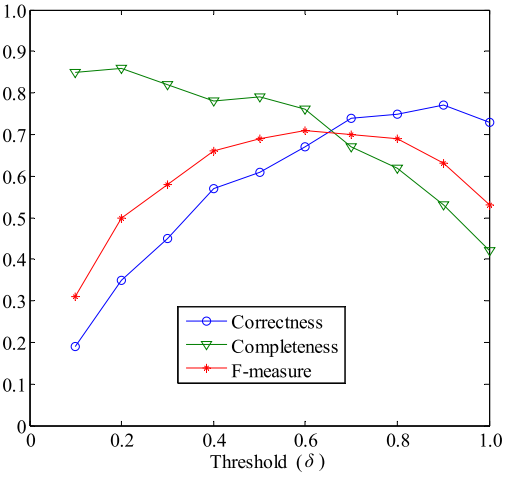

Fig. 7. Quality of integrated data based on δ.

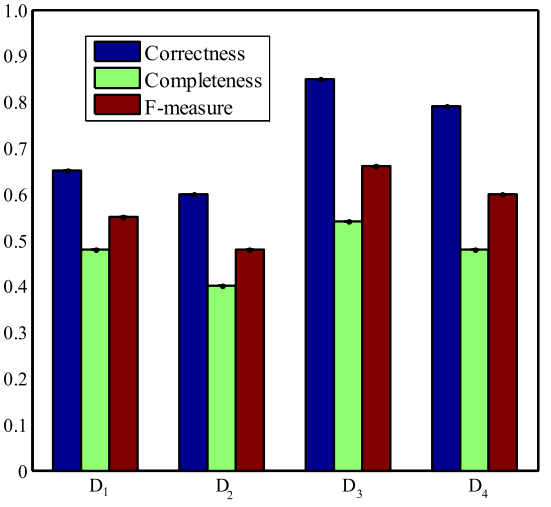

Fig. 8. Quality comparison of data integration over different data sets.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

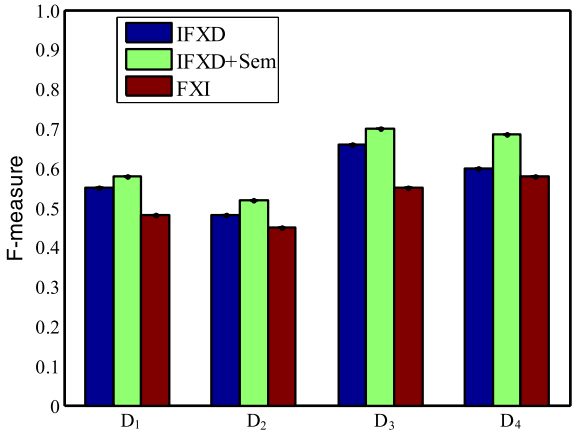

Fig. 9. Quality comparison of data integration with different approaches.

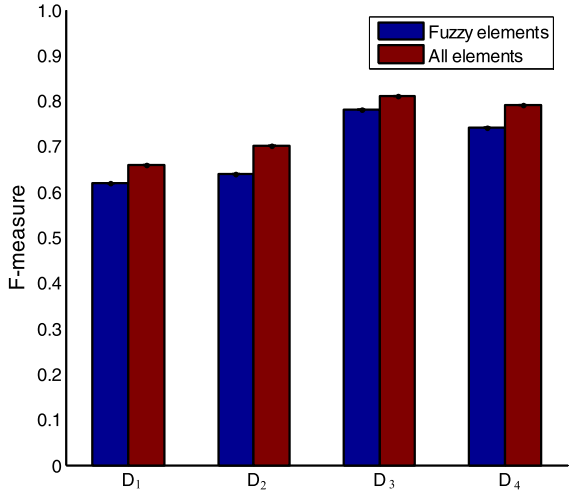

Fig. 10. Quality comparison of integrated data at the element level.

quality generally decreases with increasing data set size when the fuzzy XML documents in the same data sets are homogeneous. The quality of data integration over D_3, for example, is better than the quality of data integration over D_4.

Also, we compare our integration approach (IFXD) with the fuzzy XML integration approach in [40] (abbreviated FXI) in the context of F-measures. For the IFXD, we identify two cases: the first one is the IFXD with only structural identification for each node pair, denoted *IFXD*, and the second is the IFXD with both structural and semantic identification for each node pair, denoted *IFXD+Sem*. It is shown in Fig. 9 that (1) both the IFXD and the IFXD+Sem have higher F-measure values than the FXI, and (2) the difference of F-measure values between the IFXD and the IFXD+Sem is small.

First, compared with the IFXD and the IFXD+Sem, the FXI lacks a mechanism to process structural similarity. As a result, it fails to effectively recognize duplicate subtrees. So, the FXI has relatively low F-measure values. Second, the data sets contain a relatively small number of synonyms. So there is not a remarkable difference between the node pairs identified by semantic approaches and the node pairs identified by both structural and semantic approaches. The F-measure values of the IFXD are close to the F-measure values of the IFXD+Sem.

To further investigate the quality of integration at the level of elements, we select 20 document pairs with a similar structural complexity from each data set. Then we can obtain the average values of the integration quality of (fuzzy) elements. It can be seen from Fig. 10 that the elements of fuzzy XML documents are effectively integrated. Com-

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

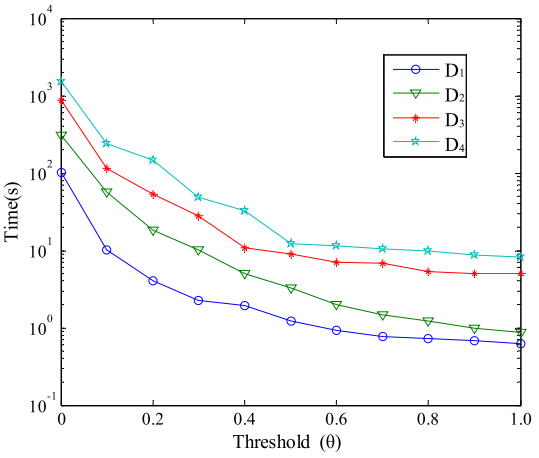

Fig. 11. Execution time for different data sets and threshold θ.

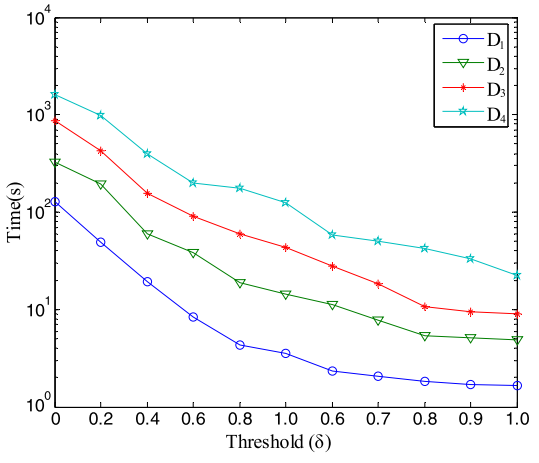

Fig. 12. Execution time for different data sets and threshold δ.

pared with the fuzzy XML documents from the synthetic data sets, the fuzzy XML documents from the real data sets have better integration quality. This is mainly because the fuzzy XML documents from the real data sets have the same schema. In addition, it is shown in Fig. 10 that the F-measure of fuzzy elements is lower than the F-measure of all the elements. The main reason is that the integration of fuzzy elements needs to consider more element features.

In addition to the quality of data integration, its execution time is also a critical factor to evaluate an integration strategy. We observe how parameters $\theta$and δ impact the performance of integration over different data sets. We apply all synthetic and real data sets to conduct our experiments. It is shown in Figs. 11 and 12 that the times for integrating fuzzy XML documents are determined by the size and complexity of the data sets. Compared with the synthetic data sets D_1 and D_2, the real data sets D_3 and D_4 contain more nodes and hold more complex structures. So the integration over D_3 and D_4 costs more in integration time than integration over D_1 and D_2. Also, it is shown in Figs. 11 and 12 that the integration times are reduced, depending on the increasing values of thresholds θ and δ. This is because the increasing thresholds could result in a smaller number of the fuzzy XML documents to be integrated.

To evaluate our method of integrating fuzzy XML documents, we apply a scale-up performance to evaluate the integration processing time when the number of documents/nodes increases. First, we use the synthetic data sets D_1 and D_2 to conduct our experiments over different numbers of documents (from 2 to 150). Fig. 13 shows that the scale-up performance is good when the number of documents increases. Compared with D_1, the integration over D_2 costs

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

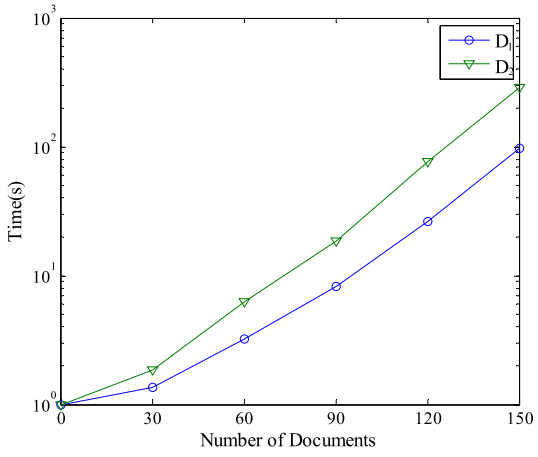

Fig. 13. Integration time comparison for synthetic sets.

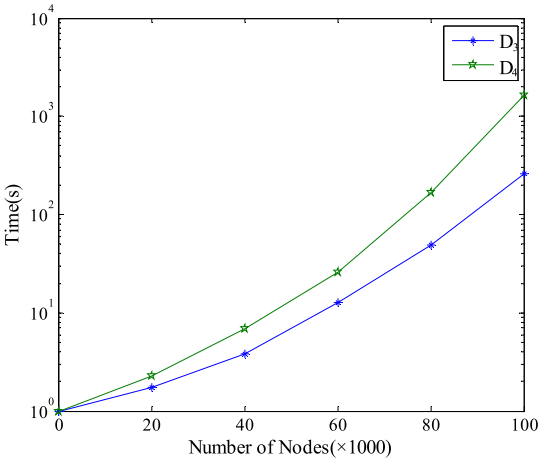

Fig. 14. Integration time comparison for real data sets.

more in integration processing time. This is mainly because D_2 is more complex and contains more elements/attribute nodes, which has a great impact on the integration performance. Second, we use the real data sets of D_3 and D_4, which contain massive document instances, to conduct our experiments over different numbers of nodes. Fig. 14 shows that the integration processing times linearly increase as the number of nodes number increases. It is also shown that the integration over D_3 costs less in integration processing time than integration over D_4 because of the much greater average number of nodes in D_4.

Now we show the execution performance of different integration strategies on a given data set. We apply three integration strategies that are respectively based on the FXI, the IFXD, and the IFXD+Sem, and conduct our experiments over D_4 with similar experimental settings. It is clearly shown in Fig. 15 that the three integration strategies have exponential processing times. As mentioned in Section 4.1, the IFXD+Sem needs extra time to process all semantic information of the nodes. So, the IFXD costs less than the IFXD+Sem. Basically, the performance of both strategies is acceptable for a personal computer to handle massive XML documents. Also, it is shown in Fig. 15 that both the IFXD and the IFXD+Sem cost less than the FXI. This is because the FXI needs more time on candidate selection.

We further show the average execution times that the three integration strategies are applied over data sets D_1, D_2, D_3, and D_4. It is shown in Fig. 16 that the data integrations over the synthetic data sets clearly cost less than the data integrations over the real data sets. The reason is that the number of subtrees that need to be identified and then integrated over the real data sets is much more than the number that need to be identified and then integrated over the

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

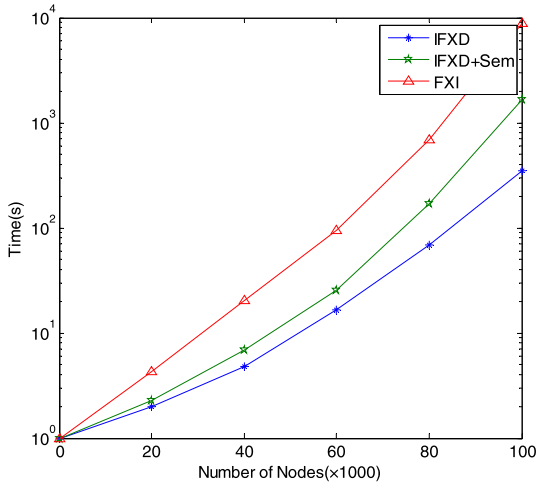

Fig. 15. Execution time comparison for different strategies.

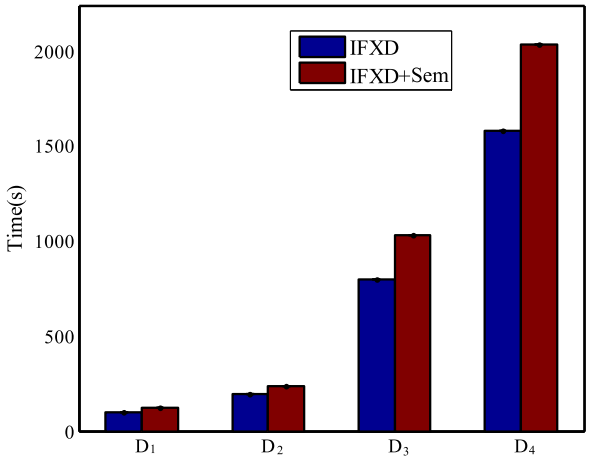

Fig. 16. Average execution time comparison for different data sets.

synthetic data sets. In addition, it is shown in Fig. 16 that the IFXD+Sem costs more than the IFXD over the same (synthetic or real) data sets.

To show the execution performance of our integration approach, we extract 20 document pairs from each data set and then respectively calculate the average execution time for integrating each document pair. It is shown in Fig. 17 that the integration of fuzzy XML documents from the synthetic data sets needs more execution time than the integration of fuzzy XML documents from the real data sets. This is mainly because the fuzzy XML documents from the synthetic data sets are heterogeneous. Also, it is shown in Fig. 17 that the average execution time for integrating the fuzzy XML document pair over D_2 is more than the average execution time for integrating the fuzzy XML document pair over D_1; the average execution time for integrating the fuzzy XML document pair over D_4 is more than the average execution time for integrating the fuzzy XML document pair over D_3. The main reason is that the fuzzy XML documents over D_2 and D_4, respectively, have a more complex structure than the fuzzy XML documents over D_1 and D_3, respectively.

We conducted our experiments at two different levels: element level and document level. The experimental results demonstrate that the choices of parameters have a great impact on the quality and performance of data integration. Also, the output quality of data integration and the execution performance are closely related to the size of the data sets as well as the structural complexity of fuzzy XML documents. It is shown that our approach has an advantage in data integration quality.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

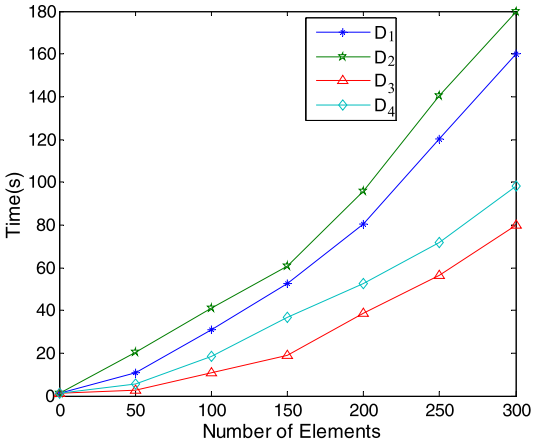

Fig. 17. Average execution time comparison for different numbers of elements.

7. Conclusions

To effectively deal with data integration of fuzzy XML documents, in this article, we proposed a novel fuzzy XML tree model named *fuzzy XML document tree model* (FXTM) to capture the structural and semantic information of fuzzy XML documents. On the basis of the FXTM, we proposed a similarity measure approach involving two-layer subtree structures. Moreover, we presented a framework to integrate heterogeneous fuzzy XML data. The experimental results show that our approach can efficiently perform fuzzy XML document integration. The novelty of our approach lies in the use of similarity measurements of two-layer subtrees that are applied to fuzzy XML documents.

We identified several issues that need to be further investigated in fuzzy XML document integration. The first issue is that of considering schema constraints (e.g., cardinality constraints and integrity constraints) when the integrated fuzzy XML document is created. Future work will extend the entity identification algorithm to cater for the situation that schema constraints occur in the two-layer subtrees. The second issue is that of considering an automatic tuning of the user-provided thresholds and weights so that better integration results can be obtained for a concrete application domain. Finally, it is assumed in the integration framework proposed in this article that each of the data sources being integrated is only a single fuzzy XML document. It is possible that a data source being integrated consists of multiple similar fuzzy XML documents. At this point, the proposed approach of integrating heterogeneous fuzzy XML data should be extended to handle such a problem.

Acknowledgements

The authors thank the anonymous referees for their valuable comments and suggestions, which improved the technical content and the presentation of the article. This work was supported by the National Natural Science Foundation of China (61772269 and 61370075).

References

[1] L. Zamboulis, XML data integration by graph restructuring, in: Proceedings of the 2004 British National Conference on Databases, 2004, pp. 57–71.
[2] F. Tseng, XML-based heterogeneous database integration for data warehouse creation, in: Proceedings of the 2005 Pacific-Asia Conference on Information Systems, 2005, p. 48.
[3] A. Thomo, S. Venkatesh, Rewriting of visibly pushdown languages for XML data integration, Theor. Comput. Sci. 412 (39) (2011) 5285–5297.
[4] N. Bikakis, et al., The XML and semantic web worlds: technologies, interoperability and integration: a survey of the state of the art, in: Semantic Hyper/Multi-media Adaptation, Springer, Berlin, 2013, pp. 319–360.
[5] S. Abiteboul, L. Segoufin, V. Vianu, Representing and querying XML with incomplete information, ACM Trans. Database Syst. 31 (1) (2006) 208–254.
[6] A. Nierrman, H.V. Jagadish, ProTDB: probabilistic data in XML, in: Proceedings of the 28th International Conference on Very Large Data Bases, 2002, pp. 646–657.
[7] A.D. Keijzer, Data integration using uncertain XML, in: Soft Computing in XML Data Management, Springer, Berlin, 2010, pp. 79–103.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

[8] E. Hung, L. Getoor, V.S. Subrahmanian, PXML: a probabilistic semistructured data model and algebra, in: Proceedings of the 19th IEEE International Conference on Data Engineering, 2003, pp. 467–478.
[9] B. Kimelfeld, Y. Kosharovski, Y. Sagiv, Query efficiency in probabilistic XML models, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 2008, pp. 701–714.
[10] S. Abiteboul, B. Kimelfeld, Y. Sagiv, P. Senellart, On the expressiveness of probabilistic XML models, VLDB J. 18 (5) (2009) 1041–1064.
[11] M.V. Keulen, A.D. Keijzer, Qualitative effects of knowledge rules and user feedback in probabilistic data integration, VLDB J. 18 (5) (2009) 1191–1217.
[12] B. Kimelfeld, Y. Kosharovsky, Y. Sagiv, Query evaluation over probabilistic XML, VLDB J. 18 (5) (2009) 1117–1140.
[13] B. Kimelfeld, P. Senellart, Probabilistic XML: models and complexity, in: Advances in Probabilistic Databases for Uncertain Information Management, Springer, Berlin, 2013, pp. 39–66.
[14] A. Gaurav, R. Alhajj, Incorporating fuzziness in XML and mapping fuzzy relational data into fuzzy XML, in: Proceedings of the 2006 ACM Symposium on Applied Computing, 2006, pp. 456–460.
[15] K. Turowski, U. Weng, Representing and processing fuzzy information – an XML-based approach, Knowl.-Based Syst. 15 (1) (2002) 67–75.
[16] B. Oliboni, G. Pozzani, Representing fuzzy information by using XML schema, in: Proceedings of the 19th International Conference on Database and Expert Systems Application, 2008, pp. 683–687.
[17] J. Lee, Y.-Y. Fanjiang, Modeling imprecise requirements with XML, Inf. Softw. Technol. 45 (7) (2002) 445–460.
[18] Z.M. Ma, L. Yan, Fuzzy XML data modeling with the UML and relational data models, Data Knowl. Eng. 63 (3) (2007) 972–996.
[19] L. Yan, Z.M. Ma, F. Zhang, Fuzzy XML Data Management, Springer, Berlin, 2014.
[20] G. Panic, M. Rackovic, S. Skrbic, Fuzzy XML with implementation, in: Proceedings of the 2012 Balkan Conference in Informatics, 2012, pp. 58–62.
[21] G. Panić, M. Racković, S. Škrbić, Fuzzy XML and prioritized fuzzy XQuery with implementation, J. Intell. Fuzzy Syst. 26 (1) (2014) 303–316.
[22] X. Yang, M.L. Lee, T.W. Ling, Resolving structural conflicts in the integration of XML schemas: a semantic approach, in: Proceedings of the 2003 International Conference on Conceptual Modeling, Springer, Berlin, 2003, pp. 520–533.
[23] H. Köpcke, E. Rahm, Frameworks for entity matching: a comparison, Data Knowl. Eng. 69 (2) (2010) 197–210.
[24] X.L. Zhang, T. Yang, B.Q. Fan, Novel method for measuring structure and semantic similarity of XML documents based on extended adjacency matrix, Phys. Proc. 24 (2012) 1452–1461.
[25] A. Nierman, H.V. Jagadish, Evaluating structural similarity in XML documents, in: Proceedings of the 5th International Workshop on the Web and Databases, 2002, pp. 61–66.
[26] A. Poggi, S. Abiteboul, XML data integration with identification, in: International Workshop on Database Programming Languages, 2005, pp. 106–121.
[27] W. Liang, H. Yokota, LAX: an efficient approximate XML join based on clustered leaf nodes for XML data integration, in: British National Conference on Databases, 2005, pp. 82–97.
[28] L. Ribeiro, T. Härder, Entity identification in XML documents, Grundl. Datenbanken (2006) 130–134.
[29] Y. Qi, et al., XML data integration: merging, query processing and conflict resolution, in: Advanced Applications and Structures in XML Processing: Label Streams, Semantics Utilization and Data Query Technologies, IGI Global, Hershey, 2010, pp. 333–360.
[30] S. Agreste, P.D. Meo, E. Ferrara, D. Ursino, XML matchers: approaches and challenges, Knowl.-Based Syst. 66 (2014) 190–209.
[31] J. Tekli, R. Chbeir, A novel XML document structure comparison framework based-on sub-tree commonalities and label semantics, J. Web Semant. 11 (2012) 14–40.
[32] J. Tekli, et al., Approximate XML structure validation based on document–grammar tree similarity, Inf. Sci. 295 (2015) 258–302.
[33] D.D.B. Saccol, C.A. Heuser, Integration of XML data, in: Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web, Springer, Berlin, 2003, pp. 68–80.
[34] A.M.D. Nascimento, C.S. Hara, A model for XML instance level integration, in: Proceedings of the 23rd Brazilian Symposium on Databases, Sociedade Brasileira de Computação, Porto Alegre, 2008, pp. 46–60.
[35] A.M. Kade, C.A. Heuser, Matching XML documents in highly dynamic applications, in: Proceedings of the 8th ACM Symposium on Document Engineering, 2008, pp. 191–198.
[36] M. van Keulen, A. de Keijzer, W. Alink, A probabilistic XML approach to data integration, in: Proceedings of the 21st International Conference on Data Engineering, 2005, pp. 459–470.
[37] T. Pankowski, Reconciling inconsistent data in probabilistic XML data integration, in: Proceedings of the 2008 British National Conference on Databases, 2008, pp. 75–86.
[38] A. Hamissi, B.B. Yaghlane, Belief integration approach of uncertain XML documents, in: Proceedings of IPMU'08, 2008, pp. 370–377.
[39] M.L. Ba, et al., Integration of web sources under uncertainty and dependencies using probabilistic XML, in: Proceedings of the 2014 International Conference on Database Systems for Advanced Applications, 2014, pp. 360–375.
[40] J. Liu, X.X. Zhang, Data integration in fuzzy XML documents, Inf. Sci. 280 (2014) 82–97.
[41] L. Yan, Z.M. Ma, J. Liu, Fuzzy data modeling based on XML schema, in: Proceedings of the 2009 ACM Symposium on Applied Computing, 2009, pp. 1563–1567.
[42] G. Nicol, et al., Document object model (DOM) level 3 core specification, W3C Working Draft 13 (2001) 1–146.
[43] R.A. Wagner, M.J. Fisher, The string-to-string correction problem, J. ACM 21 (1) (1974) 168–173.
[44] W. Cohen, P. Ravikumar, S. Fienberg, A comparison of string metrics for matching names and records, in: KDD Workshop on Data Cleaning and Object Consolidation, 2003, pp. 73–78.
[45] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, Sov. Phys. Dokl. 10 (8) (1966) 707–710.
[46] G. Navarro, A guided tour to approximate string matching, ACM Comput. Surv. 33 (1) (2001) 31–88.
[47] J.J. Jiang, D.W. Conrath, Semantic similarity based on corpus statistics and lexical taxonomy, in: Proceedings of the International Conference on Research in Computational Linguistics, 1997.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018

[48] P. Resnik, Using information content to evaluate semantic similarity in a taxonomy, in: Proceedings of the International Joint Conference on Artificial Intelligence, 1995, pp. 448–453.
[49] G.A. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.
[50] A. Marie, A. Gal, Boosting schema matchers, in: Proceedings of the OTM 2008 Confederated International Conferences, 2008, pp. 283–300.
[51] S. Madria, K. Passi, S. Bhowmick, An XML schema integration and query mechanism system, Data Knowl. Eng. 65 (2) (2008) 266–303.
[52] Z.M. Ma, L. Yan, Conflicts and their resolutions in fuzzy relational multidatabases, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 18 (2) (2010) 169–195.
[53] L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility, Fuzzy Sets Syst. 1 (1) (1978) 3–28.
[54] T. Dalamagas, et al., A methodology for clustering XML documents by structure, Inf. Syst. 31 (3) (2006) 187–228.
[55] Z. Ma, L. Yan, Modeling fuzzy data with XML: a survey, Fuzzy Sets Syst. 301 (2016) 146–159.

Please cite this article in press as: Z. Ma et al., Heterogeneous fuzzy XML data integration based on structural and semantic similarities, Fuzzy Sets Syst. (2018), https://doi.org/10.1016/j.fss.2018.04.018