Semantic Similarity Measure of Fuzzy XML DTDs with Extreme Learning Machine

ZONGMIN MA[1, 2, *] AND ZHEN ZHAO[2,3]
[1]*College of Computer Science & Technology, Nanjing University of Aeronautics and Astronautics*
Nanjing, 211106, China
[2]*College of Computer Science and Engineering, Northeastern University*
Shenyang, 110819, China
[3]*College of Information Science and Technology, Bohai University*
Jinzhou, 121013, China

Data integration for distributed and heterogeneous XML data sources is still an open challenging, and XML DTD matching is crucial task in this process. A considerable amount of algorithms for comparing XML DTDs have been proposed in the literature. Yet, the existing approaches fall short in ability to identify semantic similarities in fuzzy XML DTDs. To fill this gap, in this paper, we provide an approach to cope with semantic similarities in the fuzzy XML DTDs. The present paper makes two major contributions. First, we propose a novel fuzzy XML DTD tree model to represent fuzzy XML DTD. Second, based on the proposed tree model, we present an effective algorithm based on Extreme Learning Machine (ELM) to synthesize the semantic similarities between fuzzy XML DTDs. The corresponding computational experimental results demonstrate that our proposed approach has a prominent high performance.

Keywords: Data integration; Fuzzy XML; Semantic similarity; Extreme Learning Machine (ELM).

1. INTRODUCTION

With the development of the Internet, the management of data available on the Internet becomes ever more significant. XML (Extensible Markup Language) has become a de-facto standard for representing and manipulating rapidly increasing amount of Web data in numerous applications, for example, data description and storage, information interchange, and so on. As the number of applications that utilize heterogeneous and distributed data source grows, the importance of XML data integration mechanisms increases greatly [4, 44]. Due to a valid XML data is one that has a XML Document Type Definitions (DTD) and conforms to it. It was natural that there is a growing demand in the research of XML DTD matching before XML data integration. In the case of measuring the similarity of two XML DTDs, we consider the problem of similarity of two sets of regular expressions, typically called a XML DTD matching problem. XML DTD matching takes two XML DTDs as input and produces a mapping between elements of the two XML DTDs that correspond semantically to each other. XML DTD matching plays a critical role in XML data integration. Various systems and approaches have been designed to perform XML DTD matching automatically [29, 37, 6, 5, 25, 40, 52]. The majority of the approaches exploit functions which evaluate similarity of a particular feature of the given XML DTDs, such as similarity of labels [29, 37, 6], similarity of

Received June 21, 2016.
* Corresponding author. Work is supported by the National Natural Science Foundation of China (61572118 and 61370075).

context [5, 25] or similarity of paths [37, 40].

Previous similarity measuring methods of XML DTDs assume that data is deterministic. In fact, in many domains, fuzzy information is often included in some practical applications. It is therefore difficult to denote all information with crisp data. Some data are inherently vague rather than definite since their values are subjective in real-world applications. There have a few efforts in the area of fuzzy XML data processing. Some simple model for representing and querying XML with fuzzy information are proposed in [31, 2, 12]. With the increasing of fuzzy XML data on the Web, it is necessary to provide a suitable method for comparing and measuring similarity of fuzzy XML DTDs and then to integrate multiple fuzzy XML DTDs to a single one. Although XML DTD matching has been extensively investigated, the proposed approaches cannot be applied to process fuzzy XML DTDs matching due to the lack of an effective fuzzy XML DTD model. To our best knowledge, so far, there are not any reports discussing the similarity measure of the fuzzy XML DTDs. This motivate us to fill this gap.

Concentrating on the similarity comparison of heterogeneous fuzzy XML DTDs which are collected from different data sources, this paper devotes to develop an integrate approach to support the identification of semantic similarity measure in the fuzzy XML DTDs. For this purpose, we take a first step in construction of a new fuzzy XML DTD tree model, which makes it easier to describe fuzzy data and capture the feature information in fuzzy XML DTDs. We identify the corresponding feature information (i.e., element/attribute labels) in tree nodes and then decide the similarity of nodes come from discrete fuzzy XML DTD trees. Based on the proposed fuzzy XML DTD tree model, we develop an effective algorithm based on Extreme Learning Machine (ELM) to synthesize the individual similarity measure between fuzzy XML DTD tree nodes. Here the ELM-based algorithm is a kind of fuzzy XML clustering algorithms, which is categorized as fuzzy web mining. The issues of fuzzy web mining mainly include web usage mining (e.g., [21, 43]), clustering for web document classification (e.g., [19, 26]), domain ontology mining (e.g., [32, 12]) and so on. A comprehensive survey on fuzzy web mining is given in [9, 47]. Actually many useful techniques for fuzzy data mining have been proposed in the literature, including fuzzy frequent itemsets mining (e..g, [[]]), fuzzy rules mining (e..g, [11]) and genetic-fuzzy data mining (e..g, [48]). A recent review on fuzzy data mining techniques can be found in [47].

Being a machine learning approach, ELM is invented for classification, regression and so on [45, 30]. In particular, ELM has been applied in XML documents classification [52] and even in probabilistic XML documents classification [53]. But this paper is the first effort to apply ELM in the semantic similarity measure of fuzzy XML DTDs. In addition, the fuzzy XML DTD tree proposed in the paper is different from the common fuzzy XML DTD tree model given in [56]. The major difference is that the common fuzzy XML DTD tree model directly comes from the corresponding fuzzy XML DTD and thereby contains redundant nodes; the fuzzy XML DTD tree proposed in the paper is a kind of simplified fuzzy XML DTD tree model, which removes the redundant nodes so that the complexity of similarity comparison can be reduced. To sum up, the main contributions of this paper include: (1) a novel fuzzy XML DTD tree model is proposed, which can concisely represent the feature information in fuzzy XML DTDs; (2) an ELM-based algorithm is developed to synthesize semantic similarities by combining multiple individual similarities of fuzzy XML DTDs tree nodes. In addition, the perfor-

mance evaluations of the proposed algorithm are demonstrated on different data sets in terms of fuzzy XML DTDs size, matching effectiveness and efficiency. The experimental results validate our approach and show the practicability of our matching algorithm.

The rest of the paper is organized as follows. After a presentation of fuzzy XML DTD tree model in Section 2, we introduce Extreme Learning Machine in Section 3. The semantic similarity measure approach in the fuzzy XML DTD based on ELM is proposed in Section 4. Experimental evaluations are given in Section 5, and Section 6 concludes this paper.

2. Representation Model of Fuzzy XML DTD

2.1 Fuzzy XML DTD

A fuzzy XML DTD serves as a grammar for a fuzzy XML document, determining its internal structure. A fuzzy XML DTD is a structure composed of a set of elements and attributes, linked together via the containment relation. It identifies element/ attribute structural positions, data-types, and constraint operators, which describe constraints on the repeatability of elements/ attributes. In addition, in order to represent fuzzy information in fuzzy XML data, a representation model based on "membership degree and possibility distributions" is developed in [56, 35]. In this model, an element may be involved in a membership degree. The membership degree of an element indicates the possibility of being its parent's child element. The attribute values of elements may be presented as possibility distributions in this representation model. Note that it is possible that some attributes can take multiple (conjunctive) values. In contrast, some attributes can only take unique (disjunctive) value. That is to say, we have two kinds of fuzziness in the fuzzy XML data: one is the fuzziness in elements associated with membership degrees; another is the fuzziness in attribute values of elements represented with possibility distributions. The *Type* attribute as a child of element *Dist* is used to indicate the type of possibility distribution, having values of disjunctive or conjunctive. In addition, each *Dist* element has a *Val* element as its child. The *Poss* attribute as child of element *VAL* indicates the membership degrees of a given element.

Here we do not present the detailed definitions of fuzzy XML DTD representation model and only give an example fragment of fuzzy XML DTD shown in Fig. 1. One can refer to [56, 35] for more details.

```
1.  <!ELEMENT College (Val+)>
2.  <!ATTLIST College Cname   IDREF #REQUIRED>
3.  <!ELEMENT Val (Department*)>
4.  <!ATTLIST Val Poss CDATA "1.0">
5.  <!ELEMENT Department (Teacher*, Student*)>
6.  <!ATTLIST Department Dname IDREF #REQUIRED>
7.  <!ELEMENT Teacher (Dist)>
8.  <!ATTLIST Teacher TID IDREF #REQUIRED>
9.  <!ELEMENT Dist (Val+)>
```

```
10.     <!ATTLIST Dist Type (disjunctive)>
11.     <!ELEMENT Val (Tname?, Title?)>
12.     <!ATTLIST Val Poss CDATA “1.0”>
13.     <!ELEMENT Student (Age?, (Email|Phone)?)>
14.     <!ATTLIST Student Sid IDREF #REQUIRED>
15.     <!ELEMENT Age (Dist)>
16.     <!ELEMENT Dist (Val+)>
17.     <!ATTLIST Dist Type (disjunctive)>
18.     <!ELEMENT Email (Dist)>
19.     <!ELEMENT Dist (Val+)>
20.     <!ATTLIST Dist Type (conjunctive)>
21.     <!ELEMENT Phone (Dist)>
22.     <!ELEMENT Dist (Val+)>
23.     <!ATTLIST Dist Type (conjunctive)>
24.     <!ELEMENT Val (#PCDATA)>
25.     <!ATTLIST Val Poss CDATA “1.0”>
```

Fig. 1. A fragment of fuzzy XML DTD

In the fuzzy XML DTD shown in Fig. 1, *Poss* is adopted together with a fuzzy construct denoted by *Val* to specify the possibility of a given element department in the fuzzy XML DTD in lines 3-4. Another fuzzy construct called *Dist* can be utilized to express possibility distribution of attribute values of elements. Since we have two types of possibility distribution, *Type* is adopted to indicate the type of possibility distribution, being disjunctive or conjunctive. Consider the disjunctive *Dist* construction in lines 7-10, which express the multiple possible statuses for the teacher. Lines 15-17 are the disjunctive *Dist* construct for the age of the student. Lines 18-23 are the conjunctive *Dist* construct for the email or phone of the student. A *Dist* element may have multiple *Val* elements as children. Each *Val* has an associated *Poss* as a child node.

2.2 Fuzzy XML DTD tree model

Our objective is to offer a useful method to captures semantic similarity of fuzzy XML DTDs. For this purpose, we need first to establish a suitable model and then compare fuzzy XML DTDs based on this model. XML DTDs are presented as ordered labeled trees in most existing XML DTD comparison approaches in [29, 37, 6, 5]. In this ordered labeled tree, the nodes represent elements/attributes and are labeled with the corresponding element/attribute label names, which are ordered following their orders of appearance.

Generally speaking, a fuzzy XML DTD which represents hierarchically structured information also can be presented as a rooted ordered labeled tree. But the fuzzy XML DTD is clearly different from the crisp XML DTD because the fuzzy XML DTD contains several special fuzzy construction, which is attribute *Poss* and *Type*, in addition to element *VAL* and *Dist*.

In order to reduce the complexity of the similarity comparison, the redundant elements/attributes are deleted and the pertinent information of the elements/attributes is

encapsulated in the node of the fuzzy XML DTD tree. We adopt the following considerations in order to reach such a goal.

We note that *Type* always appears as the first child attribute of the *Dist* and *Poss* always appears as the first child attribute of the *VAL*. They are considered redundant data and increase the computational complexity. Therefore all of *Type* and *Poss* attribute are no longer reserved when a fuzzy XML DTD is mapped into a fuzzy DTD tree. But *Type* values (conjunctive/disjunctive) are remained because they need to be considered while calculating the nodes similarity degree (cf. Section 4.1). So we need to copy *Type* values into its sibling (element/attribute) nodes in processing of transformation. Similarly, we can disregard the *Val* and *Dist* elements if it is not affecting the tree structure and the depth of the other node.

Based on the discussion above, in order to improve the efficiency of comparison, we present a new **F**uzzy **X**ML **D**TD **T**ree model (FXDT for short). It is defined as follows.

Definition 1 (Fuzzy XML DTD tree). Formally, we model a fuzzy XML DTD as a rooted ordered labeled tree FXDT = {*N, E, L, T, FT, CC, AC*}. Here

- *N is the set of nodes in tree FXDT.*
- *E is the set of edges, which reflect the hierarchical structure of the tree FXDT.*
- *L is the set of labels of the elements and attributes corresponding to the nodes in N.*
- *T is the set of data types, including the basic element data types and attribute data types.*
- *FT is the set of fuzzy data types of nodes N.*
- *CC* is the set of cardinality constraints associated with the elements and attributes of FXDT ('?', '*', '+' and Null).
- *AC* is the set of alternative constraints associated with the elements and attributes of FXDT (',', '|').

We need to hold related information of elements/attributes (e.g. label, constraint, the fuzzy value) for computing of the similarity (cf. Section 4.1) when the fuzzy XML DTD is transformed into the fuzzy XML DTD tree model (FXDT) we propose. Hence, following our tree representation model, a fuzzy XML DTD tree node is modeled as follows:

Definition 2 (Fuzzy XML DTD tree node). A node $n \in N$ of FXDT is represented by a sextuplet *n* = {*NodeLabel*, *NodeDepth*, *NodeDataType*, *NodeFuzzyType*, *NodeCardConstraint*, *NodeAlterConstraint* }. Here

- $NodeLabel \in L$ is the label name of the node.
- *NodeDepth* is the nesting depth of the node in the fuzzy XML DTD. The depth of the root node is defined to be 1. If the parent of node n is at depth d, then the depth of node n is d + 1.
- $NodeDataType \in T$ is data-types. It may be "#PCDATA", "String", "Decimal", and so on.
- $NodeFuzzyType \in FT$ denotes the type of possibility distribution, disjunctive or conjunctive distributions. For a crisp node, its value is equal to Null.
- $NodeCardConstraint \in CC$ is the cardinality constraints of the node.
- $NodeAlterConstraint \in AC$ is the alternative constraints of the node.

To sum up, FXDT is a rooted ordered tree, in which the nodes represent the fuzzy XML elements/attributes. Nodes are ordered following their order of appearance in the fuzzy XML DTD. And FXDT representation model that we propose here considers the

most common characteristics (e.g., label, data-type, constraint, and the fuzzy values) of fuzzy XML DTDs.

After the fuzzy XML DTD mentioned in Section 2.1 is converted into the FXDT tree model in this way, we can measure the individual similarity of nodes and apply the approach based on ELM to synthesize the semantic similarities of nodes represented with the proposed FXDT tree model. Fig. 2 shows a FXDT tree instance that basically comes from the corresponding fragment of a fuzzy XML DTD is described in Fig. 1.Without loss of generality, in the following, we use "node" which means "element node" or "attribute node".

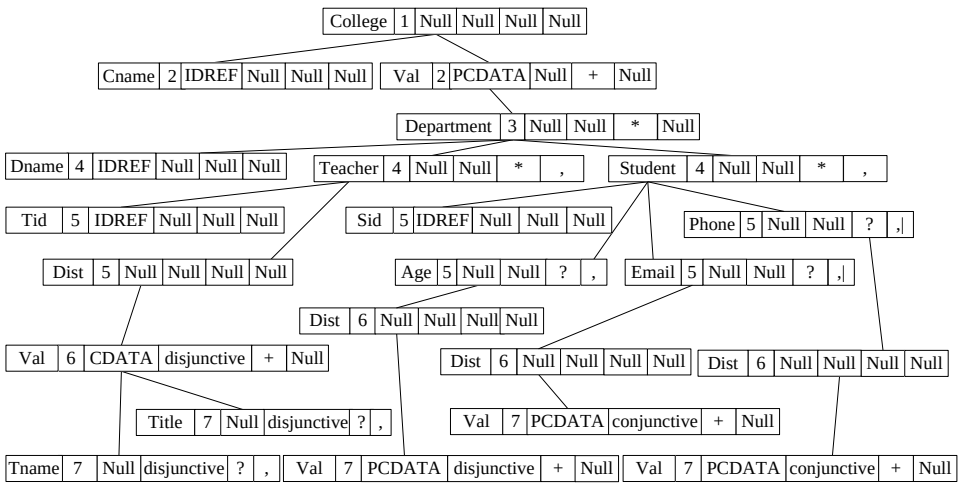

Fig. 2. A FXDT tree corresponding to the fuzzy XML DTD in Fig. 1.

3. EXTREME LEARNING MACHINE (ELM)

The Extreme Learning Machine (ELM) model was originally proposed in [18] and developed in [17], which implements a single-hidden layer feed forward neural network (SLFN) with N mapping neurons. The biggest advantage of ELM is that it can provide extremely fast learning speed and good generalization performance compared with the traditional neural network. The essence of ELM is that the network's hidden layer weights and bias values can be randomly initialized and the output weights can be calculated by matrix operations without iteratively tuning.

Consider N arbitrary samples $(x_i, t_i) \in R^{n \times m}$. Then ELM is modeled as

$$\sum_{i=1}^{L} \beta_i g(W_i \cdot x_j + b_i) = o_j \quad (j = 1, \ldots, N) \tag{1}$$

Here L is the number of hidden layer nodes, g(.) is activation function, W_i is the weight

vector between the *i*-th hidden node and the input nodes, β_i is the weight vector between the *i*-th hidden node and the output nodes, and b_i is the bias of the *i*-th hidden node.

Learning goal of ELM tends to reach the smallest training error. That is

$$\sum_{j=1}^{L} \left\| o_j - t_j \right\| = 0 \text{ and}$$

o_j is the actual output value of the SLFN. There must exist W_i, β_i, b_i such that

$$\sum_{i=1}^{L} \beta_i g(W_i \cdot x_j + b_i) = t_j \quad (j = 1, \ldots, N) \tag{2}$$

The equation above can be expressed compactly as follows:

$$H\beta = T \tag{3}$$

Here

$$H = \begin{bmatrix} h(x_1) \\ \cdot \\ \cdot \\ \cdot \\ h(x_n) \end{bmatrix} = \begin{bmatrix} g(W_1 \cdot x_1 + b_1) & \cdots & g(W_L \cdot x_1 + b_L) \\ \cdot & & \cdot \\ \cdot & \cdots & \cdot \\ \cdot & & \cdot \\ g(W_1 \cdot x_N + b_1) & \cdots & g(W_L \cdot x_N + b_L) \end{bmatrix}_{n \times L}$$

$$\beta = \left[\beta_1^T, \ldots, \beta_L^T \right]_{L \times m}$$

$$T = \left[t_1^T, \ldots, t_n^T \right]_{n \times m}$$

Training of a single hidden layer neural network is equivalent to simply finding the least-squares solution of the linear system. The output weight β is calculated as

$$\beta = H^{\dagger} T \tag{4}$$

Here $H^{\dagger} = (H^T H)^{-1} H^T$ is the Moore–Penrose generalized inverse of matrix H.

Note that fast and accurate predictions are allowed in ELM. After the calculation of the network output weights β, the response of the corresponding to a new data points (vector x_t) can be predicted by

$$o_t = \beta\, h(x_t) \tag{5}$$

Here $h(x_t) = [g(W_1 x_t + b_1), \ldots, g(W_L x_t + b_L)]^T$ is the network hidden layer output for x_t.

4. SEMANTIC SIMILARITY MEASURE OF FUZZY XML DTDS USING ELM

Semantic similarity measure in the fuzzy XML DTDs can be viewed as detecting the similarity matching of fuzzy XML DTD trees. So how to calculate similarity matching of the fuzzy XML DTD trees is a key task. To match the fuzzy XML DTD trees, firstly, we need to synthesize semantic similarity of nodes by utilizing ELM and to find out the similar nodes being compared from the fuzzy XML DTD trees, and then to calculate the similarity of the fuzzy XML DTDs according to the number of matching nodes. In other words, we need to first identify the similarity of nodes before comparing the similarity of fuzzy XML DTD trees. Our approach consists of the following several parts.

- Computing the individual similarity of nodes
- Synthesizing node similarity using ELM
- Computing the similarity of fuzzy DTDs

In this section, we detail each of these parts and give the corresponding algorithms.

4.1 Node similarity measures

Given a fuzzy XML DTD tree, a node is a fundamental data item for the similarity measures. A node may be an element or attribute one. We use Sim_{Node} (N_1, N_2) to represent the similarity degree of two nodes N_1 and N_2, where come from different fuzzy XML DTDs. Sim_{Node} (N_1, N_2) = 0 means strong dissimilarity between N_1 and N_2, while Sim_{Node} (N_1, N_2) = 1 means N_1 and N_2 are exactly the same.

To accurately assess the similarity between node-pairs, a similarity measure should exploit the features of nodes as well as their relationships. The characteristics associated with each node in a fuzzy XML DTD tree are called the node features. *NodeLabel, NodeDepth, NodeDataType* are the most commonly used features for elements/ attributes. Consequently, these features are commonly used for the similarity measure of two nodes. That is to say, the node similarity measures exploit their own features value to compare with nodes from different fuzzy XML DTD trees. According to the different exploited features, some of the commonly used similarity measures were proposed.

- Label name similarity measure

The label name (*NodeLabel*) is important for node matching. Label similarity measures take the advantage of the strings representation of label names to deal with the similarity of two nodes. There are many commonly used methods to compare strings, for example, the Jaro similarity measure [51]. Here we adopt Lin's similarity measure method in [15] based on the edit distance. The Lin's similarity measure between two strings is given by the minimum cost of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. To compute the similarity degree between the nodes based on values of *NodeLabel* L_1 and L_2, the following formula is used:

$$Sim_{Label}(N_1, N_2) = 1/(1+\text{editDistance}(L_1, L_2)) \quad (6)$$

Here editDistance (L_1, L_2) is the minimum number of character insertion, deletion, and substitution operations that is needed to transform L_1 to L_2. Each edit operation is assigned a unit cost. The similarity value given by the Lin's similarity formula is in [0, 1], with zero value denoting a dissimilarity and 1 denoting a total similarity.

- Depth similarity measure

Sometimes it is insufficient that the node label name is considered an only vital source for determining the node similarity. The depth of a node is also a feature must be considered.

Depth similarity measure has been proposed to assess the depth of nodes and their nearest common ancestor node. The similarity of two nodes is defined by how closely they are in the hierarchy and it can be calculated using the following equation:

$$Sim_{Depth}(N_1, N_2) = (d_{01}+d_{02})/(d_1+d_2) \quad (7)$$

Here d_1 and d_2 are the depth of N_1 and N_2 in local fuzzy XML DTD tree. N_0 is the most specific common ancestor of $N_1$and N_2, and d_{01} and d_{02} is the depth of N_0 in the fuzzy XML DTD tree respectively.

- Data type similarity measure

The node data type is another information source that makes a contribution in determining the node similarity [1]. Nodes having the same data types or belonging to the same data type category have the possibility to be similar and their data type similarity measure ($Sim_{DataType}$) is high. Table 1 illustrates that nodes having the same data types, the data type similarity is set to 1.0, while the data type similarity of nodes having different data types but belonging to the same category is set to 0.5.

Table 1. Data type similarity table.

Type1	Type2	SimDataType
#PCDATA	#PCDATA	1.0
CDATA	CDATA	1.0
#PCDATA	CDATA	0.5
CDATA	NMTOKEN	0.8
ID	IDREF	0.9
…	…	…

- Fuzzy type similarity measure

The node values of *NodeFuzzyType* are necessary feature information that contributes to determining the node similarity. Note that the *NodeFuzzyType* value of a node in a fuzzy XML DTD tree is of either disjunctive or conjunctive. So we distinguish the *NodeFuzzyType* similarity between them, i.e. the possibility that two disjunctive (or conjunctive) nodes come from two different fuzzy XML DTD trees are similar is higher than that one disjunctive (conjunctive) node and one conjunctive (disjunctive) are similar. Fuzzy type similarity is represented as $Sim_{FuzzyType}(N_1, N_2)$. Nodes having the same *NodeFuzzyType* value have the possibility to be similar and their fuzzy type similarity is set to be 1, while the fuzzy type similarity of the nodes having different *NodeFuzzyType* value is set to be 0.5. It can be shown in following formula.

$$Sim_{FuzzyType}(N_1, N_2) = \begin{cases} 1 & if \quad T_1 = T_2 \\ 0.5 & if \quad T_1 \neq T_2 \end{cases} \quad (8)$$

- Cardinality constraints similarity measure

Another available feature of the node that makes a contribution to assessing the node similarity is its cardinality constraint [29]. Nodes having the same cardinality constraints have the more high possibility similarity measure. We denote the cardinality constraint similarity of two nodes as $Sim_{CardConstraint}$ (N_1, N_2), which can be determined from the table of cardinality constraints similarity (Table 2).

Table 2. Cardinality constraints similarity table.

	*	+	?	NULL
*	1.0	0.9	0.8	0.5
+	0.9	1.0	0.8	0.7
?	0.8	0.8	1.0	0.8
NULL	0.5	0.7	0.8	1.0

- Alternative constraints similarity measure

Alternativeness constraint operators specify a node's disposition with regard to its parent and siblings [29]. For instance, in declaration ((a | b), c), Alternative constraints', |' is associated with both elements *a* and *b*, while Alternative constraints ',' is associated with element c. We denote the Alternative constraint similarity of two nodes as $Sim_{AlterConstraint}$ (N_1, N_2), which can be determined from the table of alternative constraints similarity (Table 3).

Table 3. Alternative constraints similarity table

	,	|	,|	|,	…
,	1.0	0	0.8	0.5	…
|	0	1.0	0.5	0.8	…
,|	0.8	0.5	1.0	0	…
|,	0.5	0.8	0	1.0	…
…	…	…	…	…	…

There are a lot of node similarity measure approaches [28, 39, 42] and we simply sign corresponding individual similarity measure which is calculated by those approaches as S_1, S_2, ⋯, S_N. It is clear that each individual similarity measure of node exploits a specific feature of the node. But assessing the similarity between nodes using an individual similarity measure is not sufficient. Therefore, it is necessary to consider a variety of node features along with a variety of measures to assess the similarity. This multi-measure nature is potent in that it makes a matching system highly flexible and adaptable to a particular application domain. However, it results in considerable challenges on how to combine these measures [23]. Without a proper means of combining, the node similarity measures fail to produce correct correspondences across elements. In general, with regard to the purpose of the similarity measure, all the individual results are then combined into the resulting similarity value, usually using a kind of weighted sum [29]. To get node semantic similarity value between pairs of nodes, in the next section, we present strategies of combining individual similarity measures values resulted from dif-

ferent features similarity measures.

4.2 Synthesizing node similarity using ELM

Target of this paper is that designs a matching framework based on semantic similarity between two fuzzy XML DTDs. Two fuzzy XML DTDs, which are collected from different data sources, can have similar elements/ attributes. To get the similarity of fuzzy XML DTDs, firstly, we have to measure the similarity of nodes in fuzzy XML DTD trees. A detailed introduction to various node similarity measurement methods has been done in Section 4.1. But single measurement method is biased and partial. These similarity measure value need to be integrated. In some previous studies [29, 1], the method based on weight is commonly used. Duo to the allocation of weight is provided by the human expert, so that the ultimate integration value may be a relatively large deviation. So，in this section, we introduce a synthesizing similarity measure approach using ELM with regard to the similarity integration. Specifically, it presents an integrated similarity measure approach between nodes(elements/ attributes) come from different fuzzy XML DTD trees based on ELM, which judges the latent semantic similarity. More accurate results can be obtained comparing with those approaches based on weight given by a human expert mentioned in research [1].

Node matching is to look for similar nodes come from different fuzzy XML DTD trees. That is to say, node matching is a process of measuring similarity among nodes and then selecting the most plausible nodes based on the similarity measure. In this perspective, the node matching procedure is as follows:

Step 1: It is needed to extract features of nodes (e.g., label or depth) coming from two different fuzzy XML DTD trees, and represent them in an internal format can be applied to individual similarity measure like mentioned in Section 2.2.

Step 2: It is needed to compute individual similarity measures value respectively like mentioned in Section 4.1.

Step 3: It is needed to synthesize the individual similarity measures using ELM. This synthesis can be viewed as a prediction.

In Step 3, we need to synthesize various similarity measures (cf. Section 4.1) values to a single semantic similarity value. The ELM model of synthesizing various similarity measures in node matching is depicted in Fig. 3. $Node_i$ and $Node_j$ are node-pairs come from different fuzzy XML DTDs trees($FDXT_p$, $FDXT_q$) respectively. S_1, S_2, ⋯, S_n are individual similarity measures values.

The processing which synthesizes various similarity measures using ELM is divided into two phases, namely, training phase and prediction phase. In order to obtain reliable training data and to construct a robust prediction model, the training phase supports the prediction-phase by providing a supervised learning that predicts semantic similarity from various measures. On the other hand, the prediction-phase is semantic similarity computing of node-pairs.

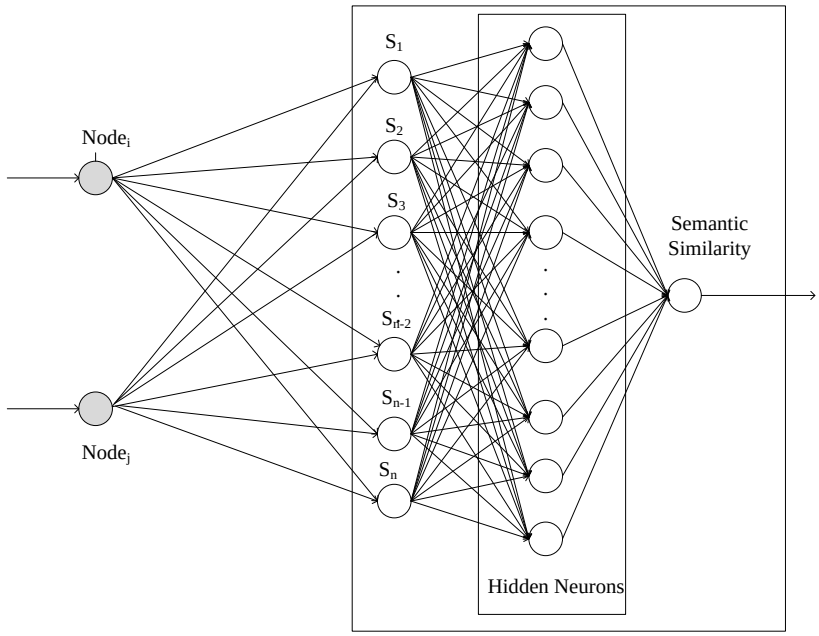

Fig. 3. ELM-based similarity synthesis

This prediction model based on ELM is built to map the relationship between the input variables (various similarity measures values) and output variables (semantic similarity value) with the sampling data. The processing of sampling is presented as follows. At first, randomly selecting a number of node-pairs($node_i$, $node_j$) from different fuzzy XML DTD trees ($FDXT_p$, $FDXT_q$) to be used as a training data set. Secondly, compute individual similarity measures value S_1, S_2, $\cdots$, S_n between two nodes of node-pairs using the approach mentioned in section 4.1 respectively. It is an input variable. Thirdly, label samples by human experts to determine semantic similarity measure between the two nodes. It is an output variable.

The use of the ELM model dramatically reduces the time required for building a prediction model. A similarity synthesizing algorithm based on ELM described in above is presented as follows.

Algorithm 1 $\mathbf{Sim_{Node}}$

Input: $Node_1$, $Node_2$ //node-pairs of different fuzzy XML DTD trees
Output: SimNodeValue //node similarity measure value
Begin
1 Node Feature extracting;
2 Compute individual similarity measure values S_1, S_2, $\cdots$, S_n, respectively;
3 Training ELM for calculate $\beta = H^{+}T$;
4 Calculate output SimNodeValue = βH;
5 Return SimNodeValue

First, the feature value of each node in the fuzzy XML document tree is extracted (with several linguistic terms) based on the FXDT model in Section 2.2 (Line 1). Then individual similarity measure values S_1, S_2, $\cdots$, S_n between two nodes of node-pairs are computed using the approach proposed in Section 4.1, respectively (Line 2). After obtaining enough node-pairs training sample, the value of β can be calculated by training ELM (Line 3). At last, the semantic similarity of node-pairs is calculated (Line 4). In the semantic similarities computation process of tree nodes, ELM algorithm is used to construct the prediction model in order to map the relationship between the various similarity measures values and semantic similarity value. This approach can thus speed up the later synthesizing process compared to the common similarity combining approach.

4.3 Fuzzy XML DTD similarity measure

Given a set of fuzzy XML DTDs FD = {$FDTD_1$, $FDTD_2$, …, $FDTD_N$}, we compute the similarity of their corresponding FXDT trees. For any two FXDTs, we sum up the number of all match pairs of nodes which similarity values are bigger than a given threshold (θ), and normalize the result. Algorithm 2 gives the algorithm to compute the similarity matrix of a set of fuzzy XML DTDs.

Algorithm 2 FDTDSimilarity

Input: FD={$FDXT_1$,⋯,$FDXT_N$} //fuzzy XML DTD trees set
Output: FDSimMatrix //FDTD similarity matrix FDSimMatrix
Begin
1 For (p = 1 to N-1)
2 {
3 For (q=p+1 toN)
4 {
5 For each node $node_{pi} \in FDXT_p$, $node_{qj} \in FDXT_q$
6 {
7 if (Sim_{Node} ($node_{pi}$, $node_{qj}$) >θ)
8 $NumSimNode_{pq}$= $NumSimNode_{pq}$+1; //number of similarity node $NumSimNode_{pq}$
9 }
10 $FDSimMatrix_{pq}$ = $NumSimNode_{pq}$ /min (|$FDXT_p$|, |$FDXT_q$|)
11 }
12 }
13 Return FDSimMatrix

The FDTDSimilarity algorithm is a fuzzy XML DTDs clustering algorithm. In this algorithm, we select any of the two fuzzy XML DTDs in the fuzzy XML DTDs set FD (Lines 1–4), traveling each node in the two fuzzy XML DTDs (Lines 5–6) and computing node similarities (Line 7). If the similarity degree of any two nodes is greater than a given threshold (say θ), the number of similar nodes adds 1 (Lines 8-9). Finally a similarity matrix that represents the semantic similarity values between all nodes is obtained (Line 10). After we obtain the fuzzy XML DTDs similarity matrix, clustering of fuzzy XML DTDs can be carried out. Fuzzy XML DTDs from the same application domain tend to be clustered together and form different clusters. The integration process of such

fuzzy XML DTDs within each cluster will become easier in the future.

5. EXPERIMENTAL EVALUATIONS

To evaluate the effectiveness of our approach for fuzzy XML DTDs matching, we present experiments conducted to compare the performance of the proposed approaches and report the results. We present the similarity measures evaluation criteria adopted in our experimental evaluation process in Section 5.1. Section 5.2 details our matching experiments results and performance analysis.

5.1 Evaluation metrics

There are two main performances that should be considered with regard to the matching process: *matching effectiveness* and *matching efficiency*.

First let us look at the effectiveness measures. Owing to the proficient use of predecessors, we make use of the *precision* and *recall* metrics defined in [46], to evaluate the effectiveness of our approach. Following Dalamagas *et al*. [46], it is needed to define some terms that are used in computing match effectiveness as follows.

- *A* is the number of fuzzy XML DTDs that are the correct matches and correctly identified by the system.
- *B* is the number of fuzzy XML DTDs that are the false matches but identified by the system.
- *C* is the number of fuzzy XML DTDs that should have been matches but not identified by the system.

Precision (*P* for short) determines the degree of correctness of the match result, which is defined as the number of correctly matching divided by the total number of identified matches.

$$P = A/(A+B) \tag{9}$$

Recall (*R* for short) assesses the degree of completeness of the matching system, which is measured as the number of correctly matching divided as the number of correct matches in fuzzy XML DTDs.

$$R = A/(A+C) \tag{10}$$

F-measure is the weighted harmonic mean of precision and recall. If an application requires stringent matches, it is necessary to obtain higher *F*-measure values.

$$F\text{-}measure = (2\times P\times R)/(P+R) \tag{11}$$

The effectiveness of matching is commonly determined with the standard measures precision (*P*), recall (*R*) and *F*-measure with respect to a manually determined "perfect" result. Therefore, as with traditional information retrieval evaluation, high precision and recall, *F*-measure characterizes a good matching method.

Now let us look at the efficiency measures. Efficiency of a matching system is usually determined by using several aspects, two of them are: response time (the time it takes for an operation to be completed), and space (memory or non-volatile storage used up by its construction). In this study, we make use of the response time as an indicator for the matching efficiency.

5.2 Experimental and results

In this section, the simulation results are discussed in detail to illustrate the effectiveness of the proposed method in evaluating fuzzy XML DTDs similarity. And we use a real-world data set[1] from four different domains as original datasets, from which we synthesized synthetic fuzzy datasets for the experimental evaluation.

We firstly add fuzzy nodes to the XML DTDs in real-world data sets. That is to say, we use our fuzzy XML DTDs generator to randomly generate the multiple corresponding fuzzy XML DTDs. After doing this, each domain has corresponding 50 fuzzy XML DTDs which show different characteristics and they represent different application domains, as shown in Table 4.

Table 4. Characteristics of the Synthetic data sets

Domain	No. of fuzzy XML DTDs	No. of element	Avg-depth
Auction	50	1365	4.76
University	50	975	4. 44
Protein Sequence	50	4732	5.35
Publication	50	2650	7. 14

In the following, we present several evaluation scenarios reporting on the experimental results.

- *Matching effectiveness*

The first scenario is to compare the quality to individual domain (Auction) in synthetic data sets. In the first step, the similarity degree between every fuzzy XML DTD pairs is first computed by using our method after sampling. And we obtain the fuzzy XML DTDs similarity matrix. Second Step, we started a series of classification tasks by varying the classification similarity threshold in the [0, 1] interval. The fuzzy XML DTDs which similarity degrees are greater than a given threshold will be grouped together. Lastly, according to the clustering results, we compute precision (P), recall (R) and F-measure for each of the classification sets in the multilevel classification phases, the results of these evaluations are reported in Fig. 4.

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

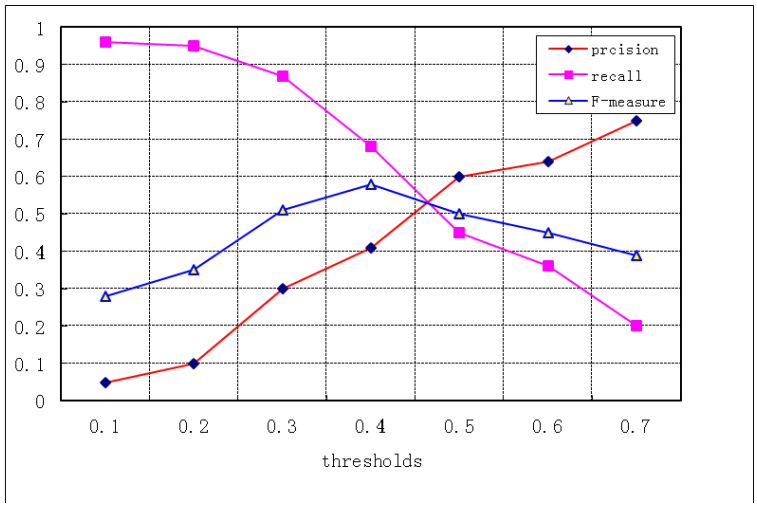

Fig. 4. Matching quality in synthetic data sets

From Fig. 4, we can see that inconsistent fuzzy XML DTDs are gradually filtered from the classification sets, while varying the classification threshold from 0 to 1. Results demonstrate that our algorithm yields optimal classes at a very early stage of the multi-level classification process (with classification thresholds < 0.5).

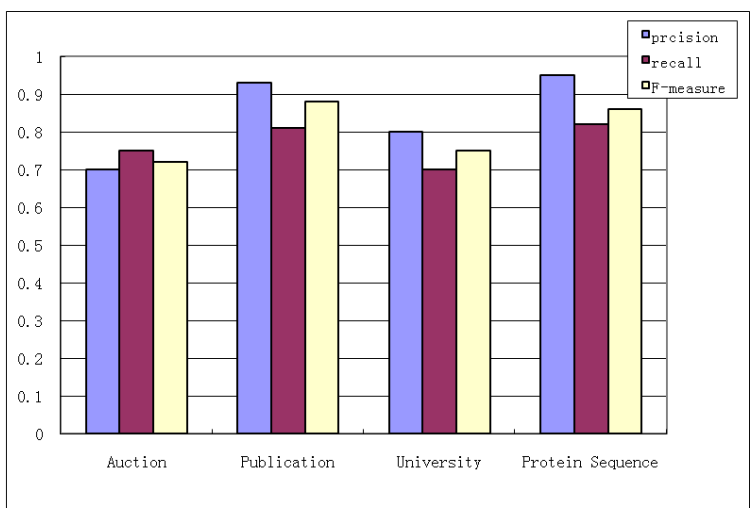

Fig. 5. Matching quality comparison for difference domain from synthetic data sets

The second scenario is to compare the quality with difference domain from synthetic data sets. In this scenario, the quality obtained in different threshold between [0, 1] for all fuzzy XML DTDs in the same domain is then averaged to obtain the ultimate quality for the domain. Results are summarized in Fig.5, and we find that matching quality over the publication and protein sequence higher than the matching quality over the auction

and university domains. This is mainly due to the fact that fuzzy XML DTDs in publication and protein sequence domains are more homogeneous than fuzzy XML DTDs in auction and university domains.

- *Matching efficiency*

We experimented with synthetic data sets (chose all 200 fuzzy XML DTDs with a total size of 1MB). The timing experiments were implemented in JDK 1.6 and MATLAB R2014a, and performed on a system with Intel Core i7 processor, 8GB RAM and running on Windows 7. The results in Fig. 6 reflect that the time of matching fuzzy XML DTDs grows in a linear dependency on the size of fuzzy XML DTDs being compared. This figure indicates that the FTDT measure performs the worst among the other similarity measures. The reason behind this can be explained as follows: The FDTD measure is based on the other similarity measures. The synthesizing process consumes much time.

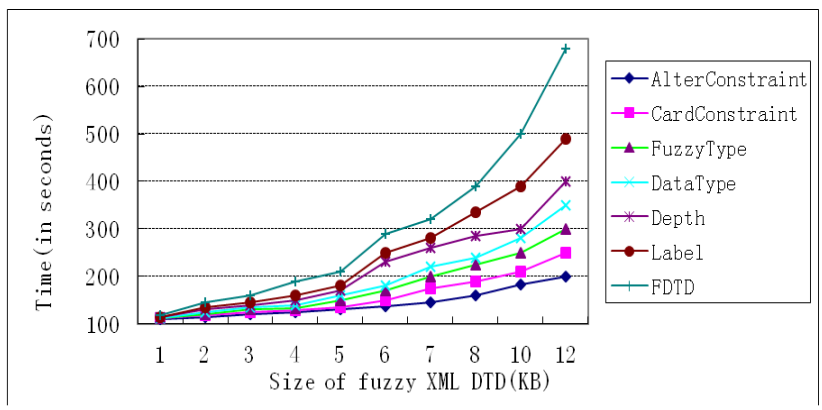

Fig. 6. Timing results to compare similarity measure for different size of fuzzy XML DTDs

5. CONCLUSION

In order to deal with the issue of semantic similarities in the fuzzy XML DTDs effectively, in this paper, we first propose a novel tree representation model to capture the node information of fuzzy XML DTDs, and then a effective solution based on Extreme Learning Machine is proposed, which synthesize the semantic similarities of fuzzy XML DTDs tree nodes represented with the proposed model. The experimental results show that our algorithms can efficiently perform matching on the fuzzy XML DTDs. This study provides two contributions: (1) a novel fuzzy XML DTD tree model is proposed, which eliminates the redundancy node in order to reduce the complexity of the similarity comparison; (2) an ELM-based algorithm is proposed for synthesizing semantic similarities of fuzzy XML DTDs trees by combining multiple individual similarities of fuzzy XML DTDs tree nodes.

Note that the ELM-based algorithm proposed in the paper is mainly for synthesizing semantic similarities of fuzzy XML DTDs trees. At this point, granular computing can

play an essential role. Currently granular computing has been widely investigated in the literature [20, 33, 55, 36, 27, 41, 3, 14, 50, 54, 13, 49, 22, 16, 34, 7, 38, 24]. It is worth of future research to apply granular computing techniques to solve the semantic similarity measure of fuzzy XML DTDs problem.

REFERENCES

[1] A. Algergawy, R. Nayak, G. Saake, Element similarity measures in XML schema matching, *Information Sciences*, 180 (24) (2010) 4975-4998.
[2] 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*, ACM, Dijon, France, 2006, pp. 456-460.
[3] A. Skowron, A. Jankowski, S. Dutta, Interactive granular computing, *Granular Computing*, 1 (2) (2016) 95-113.
[4] A. Thomo, S. Venkatesh, Rewriting of visibly pushdown languages for xml data integration, in: *Proc. of the 17th ACM Conference on Information and Knowledge Management*, ACM, Napa Valley, California, USA, 2008, pp. 521-530.
[5] A. Wojnar, I. Mlýnková, J. Dokulil, Structural and semantic aspects of similarity of document type definitions and XML schemas, *Information Sciences*, 180 (10) (2010) 1817-1836.
[6] A.G. Maguitman, F. Menczer, H. Roinestad, et al., Algorithmic detection of semantic similarity, in: *Proc. of the 14th International Conference on World Wide Web*, ACM, Chiba, Japan, 2005, pp. 107-116.
[7] B. Apolloni, S. Bassis, J. Rota, G. L. Galliani, M. Gioia, L. Ferrari, A neuro fuzzy algorithm for learning from complex granules, *Granular Computing*, 1 (2016) 1-22.
[8] B. Oliboni, G. Pozzani, Representing fuzzy information by using XML schema, in: *Proceedings of the 19th International Conference on Database and Expert Systems Application*, Turin, Italy, 2008, pp. 683-687.
[9] C. W. Lin, T. P. Hong, A survey of fuzzy web mining, *Wiley Interdisciplinary Re-views Data Mining & Knowledge Discovery*, 3(3) (2013) 190–199.
[10] C. W. Lin, T. P. Hong, T. C. Lin, A CMFFP-tree algorithm to mine complete multiple fuzzy frequent itemsets, *Applied Soft Computing*, 28 (2015) 431–439.
[11] C. W. Lin, T. P. Hong, T. C. Lin, Y. F. Chen, S. T. Pan, An integrated MFFP-tree algorithm for mining global fuzzy rules from distributed databases. *J. Univ. Comput. Sci*, 19(4) (2013) 521–538
[12] C. Y. Lu, S. W .Ho, J. M. Chung, et al., Mining fuzzy domain ontology based on concept vector from wikipedia category network. in： *Proc. of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, 3, 2011, pp. 249-252.
[13] D. Ciucci, Orthopairs and granular computing, *Granular Computing*, 1 (2016).
[14] D. Dubois, H. Prade, Bridging gaps between several forms of granular computing, *Granular Computing*, 1 (2) (2016) 115-126.
[15] D. Lin, An information-theoretic definition of similarity, in: *Proc. of the Inter. Conf. on Machine Learning*, Madison, Wisconsin, USA, 1998, pp. 296-304.
[16] F. Min, J. Xu, Semi-greedy heuristics for feature selection with test cost constraints, *Granular Computing*, 1 (2016) 1-13.
[17] G. B. Huang, An insight into extreme learning machines: random neurons, random

features and kernels, *Cognitive Computation*, 6 (3) (2014) 376-390.
[18] G. B. Huang, Q. Y. Zhu, C. K. Siew, Extreme learning machine: theory and applications, *Neurocomputing*, 70 (1) (2006) 489-501.
[19] G. E. Tsekouras, D. Gavalas, An Effective Fuzzy Clustering Algorithm for Web Document Classification: A Case Study in Cultural Content Mining. *International Journal of Software Engineering and Knowledge Engineering*, 23 (06) (2013) 869-886.
[20] G. Peters, R. Weber, DCC: A framework for dynamic granular clustering, *Granular Computing*, 1 (1) (2016) 1-11.
[21] G. Shivaprasad, N. S. Reddy, et al., Neuro-Fuzzy Based Hybrid Model for Web Usage Mining, *Procedia Computer Science*, 54 (2015) 327-334.
[22] G. Wilke, E. Portmann, Granular computing as a basis of human-data interaction: A cognitive cities use case, *Granular Computing*, 1 (2016) 1-17.
[23] H. Köpcke, E. Rahm, Frameworks for entity matching: A comparison, *Data & Knowledge Engineering*, 69 (2) (2010) 197-210.
[24] H. Liu, A. Gegov, M. Cocea, Rule-based systems: A granular computing perspective, *Granular Computing*, 1 (2016) 1-16.
[25] H. Su, S. Padmanabhan, M.L. Lo, Identification of syntactically similar DTD elements for schema matching, in: *Proc. of the Inter. Conference on Advances in Web-Age Information Management* , Springer, Xi'an, China, 2001, pp. 145–159.
[26] J. Dědek, P. Vojtáš, M. Vomlelová, Fuzzy ILP Classification of web reports after linguistic text mining, *Information Processing & Management*, 48 (3) (2012) 438-450.
[27] J. M. Mendel, A comparison of three approaches for estimating (synthesizing) an interval type-2 fuzzy set model of a linguistic term for computing with words, *Granular Computing*, 1 (1) (2016) 59-69.
[28] J. Tekli, R. Chbeir, et al., Approximate XML structure validation based on document–grammar tree similarity, *Information Sciences*, 295 (2015) 258-302.
[29] J. Tekli, R. Chbeir, Minimizing user effort in XML grammar matching, *Information Sciences*, 210 (2012) 1-40.
[30] J. Xin, Z. Wang, et al., A-ELM: Adaptive distributed extreme learning machine with MapReduce, *Neurocomputing*, 174 (2016) 368-374.
[31] K. Turowski, U. Weng, Representing and processing fuzzy information-an XML-based approach, *Knowledge-Based Systems*, 15 (1) (2002) 67-75.
[32] L. Dey, S. Bharadwaja, S. Bhat,. An Ontology-Based Mining of Consumer Feedbacks Using Fuzzy Reasoning, in：*Proc. Of the Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences* , 1, 2012, pp. 568-572.
[33] L. Livi, A. Sadeghian, Granular computing, computational intelligence, and the analysis of non-geometric input spaces, *Granular Computing*, 1 (1) (2016) 13-20.
[34] L. Maciel, R. Ballini, F. Gomide, Evolving granular analytics for interval time series forecasting, *Granular Computing*, 1 (2016) 1-12.
[35] L. Yan, Z.M. Ma, J. Liu, Fuzzy data modeling based on XML schema, in: *Proceedings of 2009 ACM Symposium on Applied Computing*, ACM, Honolulu, Hawaii, USA, 2009, pp. 1563-1567.
[36] M. Antonelli, P. Ducange, B. Lazzerini, F. Marcelloni, Multi-objective evolutionary design of granular rule-based classifiers, *Granular Computing*, 1 (1) (2016) 37-58.
[37] M. L. Lee, L. H. Yang, W. Hsu, et al., XClust: clustering XML schemas for effec-

tive integration, in: *Proc. of the 11th International Conference on Information and Knowledge Management*, ACM, McLean, VA, USA, 2002, pp. 292-299.
[38] M. Song, Y. Wang, A study of granular computing in the agenda of growth of artificial neural networks, *Granular Computing*, 1 (2016) 1-11.
[39] M. Szymczak, S. Zadrożny, et al., Coreference detection in an XML schema, *Information Sciences*, 296 (2015) 237-262.
[40] N. Tansalarak, K.T. Claypool, QMatch – using paths to match XML schemas, *Data & Knowledge Engineering*, 60 (2) (2007) 260–282.
[41] P. Lingras, F. Haider, M. Triff, Granular meta-clustering based on hierarchical, network, and temporal connections, *Granular Computing*, 1 (1) (2016) 71-92.
[42] S. Agreste, P. De Meo, et al., XML Matchers: approaches and challenges, *Knowledge-Based Systems*, 66 (2014) 190-209.
[43] S. G. Matthews, M. A. Gongora, A. A. Hopgood, S. Ahmadi, Web usage mining with evolutionary extraction of temporal fuzzy association rules, *Knowledge-Based Systems*, 54 (2013) 66–72.
[44] S. Guha, H. V. Jagadish, N. Koudas, D. Srivastava, Integrating XML Data Sources Using Approximate Joins, *ACM Transactions on Database Systems*, 31 (1) (2006) 161-207.
[45] S. Huang, B. Wang, et al., Parallel ensemble of online sequential extreme learning machine based on MapReduce, *Neurocomputing*, 174 (2016) 352-367.
[46] T. Dalamagas, T. Cheng, K. J. Winkel, T. Sellis, A methodology for clustering XML documents by structure, *Information Systems*, 31 (3) (2006) 187-228.
[47] T. P. Hong, C. H. Chen, C. W. Lin, A survey of fuzzy data mining techniques, *Fuzzy Statistical Decision-Making*, 343 (2016) 329-354.
[48] T. P. Hong, Y. C. Lee, M. T. Wu, An effective parallel approach for genetic-fuzzy data mining. *Expert Systems with Applications*, 41 (2) (2014) 655–662.
[49] V. Kreinovich, Solving equations (and systems of equations) under uncertainty: How different practical problems lead to different mathematical and computational formulations, *Granular Computing*, 1 (2016) 1-9.
[50] V. Loia, G. D'Aniello, A. Gaeta, F. Orciuoli, Enforcing situation awareness with granular computing: A systematic overview and new perspectives, *Granular Computting*, 1 (2) (2016) 127-143.
[51] W. Cohen, P. Ravikumar, S. Fienberg, A comparison of string distance metrics for name-matching tasks, in: *KDD Workshop on Data Cleaning and Object Consolidation*, Washington, DC, USA, 2003, pp. 73-78.
[52] X. Zhao, G. Wang, X. Bi, et al., XML document classification based on ELM, *Neurocomputing*, 74 (16) (2011) 2444-2451.
[53] X. Zhao, X. Bi, et al., Uncertain XML documents classification using extreme learning machine, *Neurocomputing*, 174 (2016) 375-382.
[54] Y. Yao, A triarchic theory of granular computing, *Granular Computing*, 1 (2) (2016) 145-157.
[55] Z. Xu, H. Wang, Managing multi-granularity linguistic information in qualitative group decision making: An overview, *Granular Computing*, 1 (1) (2016) 21-35.
[56] Z.M. Ma, L. Yan, Fuzzy XML data modeling with the UML and relational data models, *Data & Knowledge Engineering*, 63 (2007) 972-996.