Wireless Pers Commun
DOI 10.1007/s11277-013-1256-z

A Semantic Approach for Transforming XML Data into RDF Ontology

Pham Thi Thu Thuy · Young-Koo Lee · Sungyoung Lee

© Springer Science+Business Media New York 2013

Abstract This paper deals with the problem of transforming eXtensible Markup Language (XML) data into the Resource Description Language (RDF) which can be understood by the computer. While it is not difficult to customize XML for arbitrary data, the effective transformation is not straightforward and the result may be not semantically richer than the source document since the redundancy data resulted from the duplicate elements in XML schema. To cope with this problem, we propose an approach to measure the similarity between these duplicates before giving the transforming strategy. The similarity measure is the combination of the children and ancestor factors, which describe the relationship of elements. The experimental results show that the proposed method gives the high degree of accuracy and produces better quality of RDF ontology.

Keywords XML · RDF · Duplicate · Transformation · Similarity measure

This work is partly done while P. T. T. Thuy was a Ph.D. student at Kyung Hee University.

P. T. T. Thuy (✉) · Y.-K. Lee · S. Lee
Department of Computer Engineering,
Kyung Hee University, Seoul, Republic of Korea
e-mail: thuthuypht@gmail.com

Y.-K. Lee
e-mail: yklee@khu.ac.kr

S. Lee
e-mail: sylee@oslab.khu.ac.kr

Present Address
P. T. T. Thuy
Faculty of Information Technology, Nha Trang University,
Nha Trang, Vietnam
e-mail: tttpham@oslab.khu.ac.kr

1 Introduction

Recently, XML has been widely used as a representation language on the Web [1]—a standard format for exchanging information among applications and services [2]. The structure of XML data [3,4] is described by using Document Type Definition (DTD) [5] or XML Schema Definition (XSD) [6,7]. However, XML only supports data structure and lacks formal semantics and reference information for computer understanding [8]. Therefore, the exchange of XML among systems is not yet fully automatic. To solve this semantic problem of XML data, many researchers have proposed methods to transform XML data into higher semantic supporting languages, such as Web Ontology Language (OWL) [9–11] and Resource Description Framework (RDF) [12–16]. The development of algorithms that automate the transformation is highly beneficial for many domains, such as for XML messaging and component-based development, to applications and services in e-business, e-science, and e-learning.

In this paper, we focus on the transformation of XML into RDF since RDF recommended by W3C is the standard model for describing the semantics and reasoning about information on the Web. RDF presents data by using graphs of resources [17], so computer applications can achieve information through the Uniform Resource Identifiers (URIs). The essential structure of RDF is an object-property-property value triple. This concept makes RDF become an appropriate choice for expressing metadata in heterogeneous resource, especially for vast amount of data on the Web. However, the traditional format of data is not in RDF, but in XML form. Therefore, to reuse the valuable existing XML data, it is needed to transform XML data into RDF. This paper addresses the problem of semantic lacking of XML. In particular, we have developed an approach to the transformation of heterogeneous XML data sources.

Though, our target output is RDF, we use RDF Schema as an intermediary stage of transforming process. Because RDF only cannot describe classes and properties in structured documents [18], they are depicted by the RDF Vocabulary Description Language 1.0: RDF schema, shortly, RDF schema. Therefore, our procedure interprets valid XML documents as RDF model and uses vocabularies of the RDF schema.

There are some difficulties associated with using DTD or XSD to extract RDF Schema. For instances, DTD and XSD do not distinguish between classes and properties while RDF does. Our solution is to rely on the content of the element in order to decide it is a class or property. Moreover, most transforming approaches give a unique identifier for each XSD or DTD element by adding a new key element or changing the source element's name. However, this may lead to data redundancy, when duplicate elements represent the same information. The ideal result of an XML transformation is a correct, complete, and unique representation of every object. To achieve this data quality, a similarity computation of duplicate elements is used, where if two elements are highly similar in semantics, they are transformed into one representation.

In this paper, we propose a novel technique to measure the similarity of duplicate element and based on the similarity results, duplicates are transformed into appropriate RDF concepts. Our contributions are as following:

1. We propose several novel metrics to measure the semantic similarity of duplicate elements in XML schema (XSD or DTD)
2. We present a set of rules that derive classes, properties, and data types from DTD/XSD and interpret XML data as RDF statements by using RDF schema vocabularies.
3. We conduct a set of experiments to evaluate our computations and compare our method with the related approaches.

The remainder of the paper is organized as follows. In Sect. 2, we briefly present the related work. Section 3 describes the duplicate measuring metrics and transforming notations from XSD to RDF Schema. Section 4 presents the validation of RDF result and experimental comparison between our work and related approaches. Finally, Sect. 5 concludes this paper.

2 Related Work

In this section, we present two research directions that are related to our paper: (1) Transforming or mapping XML data to RDF ontology; (2) Measuring the element similarity between concepts in different documents.

First, there are several approaches related to mapping or transforming XML data into RDF. Klein [12] proposes a transformation procedure for converting XML data to RDF data by using RDF Schema vocabularies. However, the proposed method only translates some selected information in the document. Moreover, elements in XML document are decided to be classes or properties depending on user's opinion. Amann et al. [13] rely on DTD to define the meaning for every XML element and use XPath to map information in XML documents to ontology. However, beside referencing to XML document and its DTD, it refers to the specification of rules which is not needed in our approach.

Melnik [14], Yin/Yang Web [15], and Ferdinand et al. [9] also describes a set of mapping notations from XML to RDF. However, they only focuss on how to map all XML elements to RDF concepts and do not concern about the similarity of duplicate elements in XML document. Therefore, the issues follow the structures of XML but bear little meaning and do not fit well into RDF model. There are other approaches giving new XML syntax for RDF. These approaches use XML to define a language to represent RDF triples, such as, the "strawman unstriped syntax" of Berners-Lee [19] and Borden's syntax [20].

In the previous work [16], we present a procedure for transforming valid XML documents into RDF via RDF Schema. However, since we only rely on the general definitions of DTD to draw classes and properties, sometimes there is a conflict between class and sub-class elements. Further, we assign each XML element a unique identifier, so we do not solve the duplicate problem.

Second, some approaches are proposed to measure the element similarity between schemas. Do et al. [21] compute the name similarity between elements of two XML Schemas. Yang et al. [22] use linguistic taxonomy based on concept definitions in WordNet [23] to gain the most accurate semantics for the element names. Some researchers [1,24] employ supplemental functions to calculate the similarity of a particular feature of a given schema. Our previous work on measuring the similarity between XML Schemas [25,26] computes both the structural and semantic aspects of XML elements. All the partial results are then combined into the final similarity value using a weighted sum function.

In general, approaches in the first direction try to transform or map a source XML element to a destination RDF element. All proposed methods consider each XML element as a separated concept and give it a unique RDF identifier. However, this assumption is not completely true, since most duplicates in XML Schema are highly similar. Therefore, the data redundancy happens if each dulicate is assigned a separated identifier. In order to avoid the data redundancy problem, it is necessary to measure the similarity of duplicates in XML Schemas before transforming them into RDF ontology. However, at the time of this research, there is no approach solving the duplicates problem during the transformation process from XML into RDF. Moreover, since our approach concentrates on measuring the similarity of duplicates, our method is most similar to the second direction of related approaches, although our computation focusses on the similarity between duplicates within XML document.

3 Duplicate Similarity and RDF Transformation

3.1 Duplicate Similarity Measure

In order to introduce the metrics for measuring the semantic similarity of duplicated element, let us consider the duplicates in DTD document [27] presented in Fig. 1.

In Fig. 1, four duplicate elements *xsitype* of *Description*, *MultimediaContent*, *Audio*, and *AudioDescriptor* are quite similar since they are leaf nodes (without children elements) but they are different in their ancestors. Based on these observations, we can conclude that there are two main factors that affect the similarity between duplicated elements, particularly the children, and the ancestor. Therefore, we define the following definition, which combines two these factors, to assess the similarity of the duplicates.

Definition 1 The duplicate similarity (DupSim) between duplicated elements, e_1 and e_2, in XSD or DTD document is defined as the weighted sum of their children similarity (ChildSim), and their ancestor similarity (ASim):

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="Audio">
      <xs:complexType>  <xs:sequence>
            <xs:element ref="AudioDescriptor"/>
      </xs:sequence>
<xs:attribute name="xsitype"
                        type="xs:NMTOKEN" use="required"/>
</xs:complexType> </xs:element>
<xs:element name="AudioDescriptor">
<xs:complexType><xs:sequence>
  <xs:element ref="SeriesOfVector" />
   </xs:sequence>
   <xs:attribute name="loEdge"
                         type="xs:NMTOKEN" use="required" />
   <xs:attribute name="hiEdge"
                         type="xs:NMTOKEN" use="required" />
   <xs:attribute name="xsitype"
                          type="xs:NMTOKEN" use="required" />
  </xs:complexType>  </xs:element>
<xs:element name="Description"> <xs:complexType><xs:sequence>
  <xs:element ref="MultimediaContent" />
   </xs:sequence>
   <xs:attribute name="xsitype"
                         type="xs:NMTOKEN" use="required" />
  </xs:complexType>  </xs:element>
<xs:element name="Mpeg7">
     <xs:complexType> <xs:sequence>
          <xs:element ref="Description" />
  …
  </xs:complexType>  </xs:element>  </xs:schema>
```

Fig. 1 A part of XSD document of audio *MPEG7*

$$DupSim(e_1, e_2) = \alpha * ChildSim(e_1, e_2) + (1 - \alpha) * ASim(e_1, e_2) \tag{1}$$

where α is the weight parameter, between *0* and *1*. If the *ChildSim* property contributes more than *ASim* property for the similarity of duplicates, then the weight of the *ChildSim* is greater than that of *ASim*.

However, the main problem is how to determine which similarity factor has more important role than the other. In this paper, we propose a method to obtain the weight value. According to the experiment in Sect. 4, the value of *DupSim* is close to the manually similarity value at the value 0.55 of α. Therefore, the weight value of *ChildSim* is 0.55 and weight value of *ASim* is 0.45.

To compute the children similarity of duplicated elements, we pick up in turn one child of the first element to compare with each child of the second element. For instance, children of an element e_1 is $Ce_1 = [e_{11}, e_{12}, \ldots, e_{1k}]$, and children of an element e_2 is $Ce_2 = [e_{21}, e_{22}, \ldots, e_{2t}]$, which k and t are the numbers of children of the element e_1 and e_2, respectively. If $k \le t$, we take one after another each element in Ce_1 to compare with each element in the set Ce_2. The highest value of the measurement is chosen. Otherwise, we compare each element in Ce_2 to Ce_1. Therefore, the children similarity of twin elements is the average function of the similarity of each children pair. The children similarity is computed based on the following Eq. (2):

$$ChildSim\,(e_1, e_2) = \begin{cases} \frac{\sum_i max_j(NameSim(e_1.child[i], e_2.child[j]))}{k}, k \le t \\ \frac{\sum_j max_i(NameSim(e_1.child[i], e_2.child[j]))}{t}, k \ge t \end{cases} \tag{2}$$

where *NameSim* is the semantic similarity of children elements, which is determined by Eq. (3). In the case that one of duplicated elements is the leaf node (that means it contains no child node), the children similarity of this twin element is *0*. Otherwise, if both elements are leaf nodes, their children similarity is *0*.

To compute the semantic similarity of children elements, we measure the meaning of the element names by referring them in the WordNet [23]. We reuse the distance-based method [28] to measure the distance similarity of children elements in the WordNet taxonomy. The semantic similarity between a children element e_1 in the first schema and a children element e_2 in the second schema is determined by the following Eq. (3):

$$NameSim(e_1, e_2) = \frac{2 * depth(LCS)}{depth(e_1) + depth(e_2)} \tag{3}$$

where *depth(LCS)* is the number of nodes from the common super-concept of element e_1 and e_2 to the root node; $depth(e_1)$ and $depth(e_2)$ are numbers of nodes from element e_1 and e_2 to the root node.

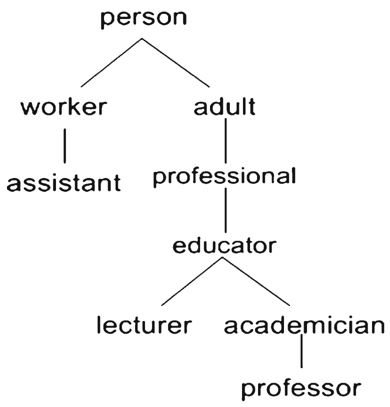
For example, Fig. 2 is a fragment of WordNet. The semantic similarity between two elements *lecturer* and *professor* is computed as following:

$$NameSim(lecturer, professor) = \frac{2 * 4}{5 + 6} = 0.73$$

In some cases, the element name is a combination of words or the short form of some words. In those cases, the normalization steps are required. These steps remove genitives, punctuation, capitalization, stop words (such as, *of*, *and*, *with*, *for*, *to*, *in*, *by*, *on*, and *the*), and inflection (plurals and verb conjugations), and replace the short word by its full name.

Assume that m and n are the numbers of tokenized words of two elements E_1 and E_2, respectively. The name similarity between two these elements are computed as following Eq. (4) or Eq. (5):

Fig. 2 A fragment of WordNet



$$NameSim(E_1, E_2) = \frac{\sum_{i=1}^{m} max_{j=1}^{n}(NameSim(e_{1_i}, e_{2_j}))}{m}, m \geq n \tag{4}$$

$$NameSim(E_2, E_1) = \frac{\sum_{i=1}^{n} max_{j=1}^{m}(NameSim(e_{2_i}, e_{1_j}))}{n}, n > m \tag{5}$$

Further, in some cases, a name of the element is not included in the WordNet, we define a metric to measure the similarity of those elements as following:

$$NamSim(e_1, e_2) = LingSim(e_1, e_2) = \frac{n_{e_1 \cap e_2}}{\max(n_{e_1}, n_{e_2})} \tag{6}$$

where $n_{e_1 \cap e_2}$ is the number of matching characters between elements e_1 and e_2; *max* is the maximum value; n_{e_1} and n_{e_2} are the lengths of the elements e_1 and e_2, respectively.

In order to measure the ancestor similarity of duplicate elements, the important thing is to find their common super concept [29]. Starting from two duplicate elements, the algorithm traverses each of their ancestor elements in turn until it finds their common node. Suppose that C is the least common super concept of the duplicate elements e_1 and e_2, and $level_{e1}$ and $level_{e2}$ are the numbers of nodes on the path from e_1 and e_2, respectively, to C. Then their ancestor similarity is:

$$ASim(e_1, e_2) = \begin{cases} 1 & if \quad level_{e1} = level_{e2} = 1 \\ 0.85^{\max(m,n)-1} & otherwise. \end{cases} \tag{7}$$

Details of ancestor similarity measure are presented in Fig. 3.

For example,

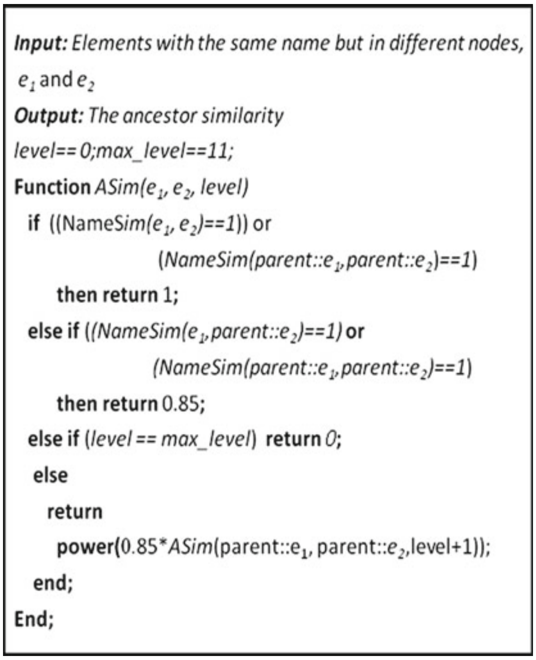
$$DupSim(MultimediaContent.xsitype, Audio.xsitype) = 0.5 * 1 + 0.5 * 0.85 = 0.93$$
$$DupSim(MultimediaContent.xsitype, AudioDescriptor) = 0.5 * 1 + 0.5 * 0.72 = 0.86$$

Depending on the expected similarity value, the duplicated elements can be divided into two groups, high similarity and low similarity. In this paper, we use the threshold value *0.7* to differentiate the duplicated elements. Duplicates having the similarity value at *0.7* and higher are transformed into unique RDF element, otherwise they are renamed.

3.2 XML Schema to RDF Transformation

In this stage, the collection of classes and properties is created from the given XML Schema. The general idea of this step is as follows:

Fig. 3 Ancestor similarity algorithm



```
Input: Elements with the same name but in different nodes,
 $e_1$ and $e_2$
Output: The ancestor similarity
level== 0;max_level==11;
Function ASim($e_1$, $e_2$, level)
  if ((NameSim($e_1$, $e_2$)==1)) or
                  (NameSim(parent::$e_1$,parent::$e_2$)==1)
      then return 1;
  else if ((NameSim($e_1$,parent::$e_2$)==1) or
                  (NameSim(parent::$e_1$,parent::$e_2$)==1)
      then return 0.85;
  else if (level == max_level) return 0;
  else
     return
      power(0.85*ASim(parent::$e_1$, parent::$e_2$,level+1));
  end;
End;
```

- The $\langle schema \rangle$ element is the root element of the XML Schema. It can contain some attributes. Value of the attribute *xmlns*: is interpreted as a namespace [30].
- The first element declared by element name is the root-class of document.
- For each XML Schema *xs:complexType* which is described using *sequence*, *all* and *choic*e, we map to a *rdf:class*. Every element or attribute declared within it is mapped to a contained class or sub-property.
- For each sub-element (elements in brackets or following the first element), we decide whether they are subclasses (contained class) or properties of the parent class.
- Since element which is declared by *xs:simpleType* does not contain any other elements or attributes, it is mapped to a *rdf:Property*.
- Global element and attribute definitions are mapped similarly to the local ones.

By observing the XSD and its corresponding XML instances, we recognized that only XSD definitions, such as *xs:element* and *xs:attribute*, appear in the XML document. It means that XML instances contain two main components, such as elements and attributes. Therefore, our goal concentrates on transforming XML constructs that are related to these components.

We consider five main definitions in an XML Schema: element name, element ref, attribute name, attribute ref, and datatype. An element definition has two following syntaxes:

$$\langle xs : elementname = \text{“}xxx\text{”}\rangle or \langle xs : elementref = \text{“}xxx\text{”}\rangle$$

where *xxx* is the given name of the element. The prefix *xs* can be changed depending on the namespace declaration.

Because $\langle xs\text{:}\ element\ name \rangle$ is used to describe elements of a document and each element can contain children elements, the function of these elements is like a class in a structural program, therefore, we treat element-name as a RDF class. However, two cases are considered for $\langle xs :\ element\ name \rangle$ definition. If $\langle xs :\ element\ name \rangle$ contains another

$\langle xs$: $element\ name\rangle$, which has not only literal, we can assume that is a "part-of" relationship. The transformation procedure considers it as a subclass of the previous class. On the contrary, if an XML element containing the declaration $\langle xs$: $element\ name\rangle$ and without any attribute and element, this element is mapped to a property.

Moreover, there are two kinds of class definition in XML Schema, $\langle xs$: $element\ name\rangle$ and $\langle xs$: $element\ ref\rangle$. The second one refers to the name of another element. This reference to an element or an attribute is comparable to cloning an object. Therefore, our procedure considers $\langle xs$: $element\ ref\rangle$ as $\langle xs$: $element\ name\rangle$ and both are applied the same mapping rules. If an element has element ID, our procedure considers it as unique identifier for this element. Hence, if there are other elements with the same name and same level, we use this ID instead of creating new attribute for class element.

Furthermore, since the attribute element gives additional information about its parent node, we consider it as a property of the class. Therefore, the declaration such as $\langle xs : attributename =$ "xxx"$\rangle$ or $\langle xs : attributeref =$ "xxx"$\rangle$ is converted to the RDF property of its parent class.

Specially, difference from XML Schema, RDF Schema does not allow the duplicate name. Thus, when our procedure meets an attribute which has the same name as the previous node, we use the metric defined in Definition 1 to measure their similarity. If their similarity is equal and higher than our threshold *(0.8)*, they are transformed into one RDF element, otherwise the second duplicate element is renamed by adding its parent name in ahead of its name.

In above schema files, *Mpeg7* is a root node since it contains other elements. It is converted to the RDF class as following:

```
<rdfs:Class rdf:about="&mpeg7;Mpeg7"
            rdfs:label="Mpeg7">
</rdfs:Class>
```

For other classes, besides the definition for class, the *rdfs:Container* is added to describe the relationship between parent and child element. For instance, the class *Description* is expressed in RDF Schema as below:

```
<rdfs:Class rdf:about="&mpeg7;Description" rdfs:label="Description">
              <rdfs:Container rdf:resource="&mpeg7;Mpeg7"/>
</rdfs:Class>
```

For node considered as attribute, it is defined by *rdf:Property*. The *rdfs:domain* and *rdfs:range* are used to describe the corresponding resource and data type of that attribute. Furthermore, in order to depict the attribute's appearing times, our procedure borrows the OWL expressions, such as *owl:maxCardinality* and *owl:minCardinality*. For example, the attribute *xmlnsmpeg7* of the class *Mpeg7* is interpreted as following:

```
<rdf:Property rdf:about="&mpeg7;xmlnsmpeg7" owl:maxCardinality="1"
              owl:minCardinality="1" rdfs:label="xmlnsmpeg7">
       <rdfs:domain rdf:resource="&mpeg7;Mpeg7"/>
       <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
```

The XML data allow duplicate element and attribute names but the RDF does not. To solve this problem, we measure the similartiy of these duplicates. If their similarity value is lower than the given threshold value (in this paper, the threshold value is *0.7*), we change the second name by adding its parent's name and underscore ahead of its name, otherwise we combine these duplicates as one element. For instance, in the motivating XSD file, the attribute *xsitype* appears four times inside four classes: *Description, MultimediaContent, Audio,* and *AudioDescriptor*. The *DupSim* computation shows that all duplicates are highly similar, (above 0.7), therefore those duplicates are transformed into one RDF element as following:

```
<rdf:Property rdf:about="&mpeg7; xsitype" owl:maxCardinality="1"
                    owl:minCardinality="1" rdfs:label="xsitype">
        <rdfs:domain rdf:resource="&mpeg7;Description"/>
        <rdfs:domain rdf:resource="&mpeg7;MultimediaContent"/>
        <rdfs:domain rdf:resource="&mpeg7;Audio"/>
        <rdfs:domain rdf:resource="&mpeg7;AudioDescriptor"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
```

After extracting RDF Schema from the XML Schema file, the next step is to transform the XML instances into RDF statements. The inputs of this stage are RDF Schema and XML document. If the changing element happens in the previous step, the corresponding element in the XML file is also renamed.

3.3 Complexity

We begin with a complexity consideration by determining the computation costs at each step in the process. There are two main step in our scheme. The first step is to measure the similarity of duplicates within an XML Schema. For each pair of duplicates, we need to compare m different similarity functions (*comp*), with a cost of sim_1 to sim_k and aggregate them with the cost of *aggr*. The second step is to transform XML Schema elements into appropriate RDF concepts, with a cost of *trans*. The worst-case runtime complexity c is determined by the following equation:

$$c = comp \cdot \left(\sum_{i=0}^{m} Sim_i + aggr \right) + trans \tag{8}$$

To compare the ancestor and children elements of two compared duplicates, we have to retrieve two subtrees, $O(log(n))$, and use the set similarity function for those sub-elements again. Therefore, the complexity of this single similarity determination is $C(sim_i) = O(log^2(n))$. Then, for each duplicate pair an aggregation operation is performed once with $aggr = O(1)$. Finally, the cost of the transformation step is $O(n)$, since this step interprets all elements in XML Schema in RDF ontology.

4 Experiment Evaluation

One of the most advantages of our approach to other methods is the independence program. It can perform automatically at all steps, such as mapping from XSD/DTD to RDF Schema, transforming XML document into RDF individuals. Moreover, our procedure can be applied as the intermediate module between multimedia [31] XML-based files and the Semantic Web applications. Users can choose any XML data file to transform it to the RDF destination.

Our program results in two data files including RDF Schema file and RDF file, such as *Mpg7.rdfs* and *Mpeg7.rdf*. The RDF Schema stores descriptions of classes, properties, and the relationships between properties and classes as well as the data types of these properties.

The program language to be used is C# with the help from the library. Net 2.0. We choose C# because it is a strong language supported for building application related to data processing and windows interface. In order to validate the RDF result, we use the RDF Validation Service given by the W3C [32,33]. The RDF results always pass the validation of this service.

In order to compare our method with the related approaches on the transforming XML data into RDF, we implement the matching between the source XML Schema and the destination RDF document. The criteria for evaluating the quality of matching system are *precision* and *recall* [34]. *Precision* reflects the share of real correspondences among all found correspondences. Given the reference matching, R, the *precision* of the matching produced by our method, T, is computed as the following equation:

$$precision(R, T) = \frac{|R \cap T|}{|T|} \tag{9}$$

Recall specifies the share of real correspondences:

$$recall(R, T) = \frac{|R \cap T|}{|R|} \tag{10}$$

Although *precision* and *recall* are the most widely used measures, when comparing matching systems, one may prefer to have only a single measure. Moreover, systems are not comparable based solely on *precision* and *recall*. For this reason, *F-measure* is introduced to aggregate the *precision* and *recall*. *F-measure* presents the harmonic mean of precision and recall.

$$F\text{-}measure = 2 * \frac{precision * recall}{precision + recall} \tag{11}$$

Before going to evaluate our proposed solution, we need to determine the weights for the *DupSim* function—Eq. (1). Most of the current papers related to weighted function, such as [1,24], assume that the value of weight parameters can be assigned by a user and, hence it is not discussed. The problem is how to prepare the dataset to determine the accurate value of weight parameter.

In this paper, to determine the weight parameter, we conduct the experiment on the real dataset presented in Table 1. Firstly, we determine the mutual similarity of each duplicate pair from user's perspective. Then, we set the respective parameters so that the error rate of similarity measure (*DupSim*) returns the lowest value. Since there are many duplicates in a XSD document, to get the reasonable value, we use the average function to obtain the final weight value.

Figure 4 shows how error rates of duplicate element similarities (*DupSim*) change as we use different α values. For the four schemas including duplicate elements in Table 1, we manually classified into two groups: similar and non-similar, then computed the classification error rate at each alpha values (in the range *0.1, 0.15, …, 1.0*). The weighted average of the error rates for the four schemas is computed by using the number of duplicate pairs in each schema as the weighted factors. For example, the weight values of the schemas number 1, 2, 3, and 4 are 16, 22, 32, and 58, respectively.

Figure 4 shows that very small or very large alpha values result in a large number of error rates. Particularly, at the alpha values ranging between *0.1* and *0.3*, the average error rate of

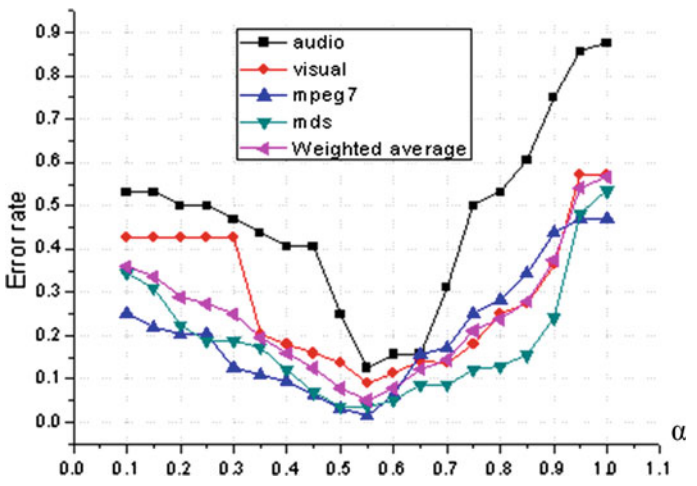


Fig. 4 The error rate of DupSim at different values of α

Table 1 The characteristics of the tested schemas

#	Schema name	File size (bytes)	# Nodes	Max depth	# Duplicates
1	audio-2001	40, 960	195	5	16
2	visual-2001	45, 056	273	6	22
3	mpeg7-udp-2004	69, 632	417	6	32
4	mds-2001	258, 048	1, 285	11	58

the classification is also ranging from 15 to *35* %. This number decreases to approximately *7%* at the alpha values *0.5* and declines to *5%* at the alpha value of *0.55*. From the alpha values of *0.55* and higher, the error rate values steadily climb up and are highest (*56* %) at the alpha value of *1.0*. Because the error rate of classification achieves the minimum value at the alpha of *0.55*, we use *0.55* as the value of weight alpha (α). Therefore, the weight value of children factor is *0.55* and weight value of ancestor factor is *0.45*. Those values can be used in the duplicate similarity computation for arbitrary datasets.

In order to obtain the practical evidence, we apply our transformation to four multimedia XSD documents from [24], particularly, *audio-2001.xsd*, *visual-2001.xsd*, *mpeg7-udp-2004.xsd*, and *mds-2001.xsd*. The characteristics of four schemas are presented in Table 1.

As presented in Table 1, the larger size of the schema file is, the higher number of duplicate elements has. Therefore, it can be conclude that, for the large dataset, our propose method will produce the big different from related transformation work. Because we conduct the experiments one four schemas, the finally results of precision and recall are the weighted average value of precision and recall of each schema. The weight factors in this case are also the numbers of duplicates of each schema. Particularly, weighted precision and recall are measured by following Eqs. (12) and (13), respectively:

$$precision = \frac{\sum_{i=1}^{n} w_i \times precision_i}{\sum_{i=1}^{n} w_i} \tag{12}$$

$$recall = \frac{\sum_{i=1}^{n} w_i \times recall_i}{\sum_{i=1}^{n} w_i} \tag{13}$$

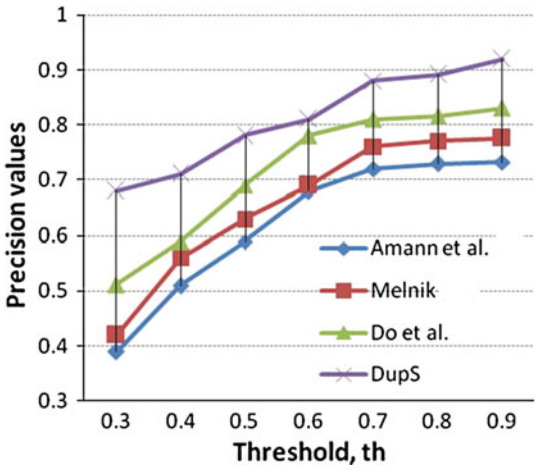


Fig. 5 Precision values of DupS and related approaches

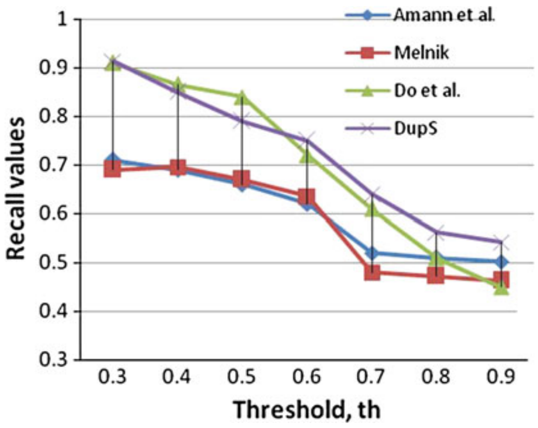


Fig. 6 Recall values of DupS and related approaches

where n is the number of schema files. In this experiment, n = 4; w_i is the number of duplicate elements in schema number i.

Since our approach focuses on the transforming XML Schema into RDF ontology, we compare our method to similar studies such as Klein [12], Amann et al. [13], and Ganguly et al. [34]. The precision, recall, and F-measure values among our method (***DupS***) and related work are shown in Figs. 5, 6, and 7, respectively.

Note that in this paper, the threshold values are chosen between 0.3 and 1.0, since the similarity values of duplicates are usually higher than 0.3. The precision and recall values presented in these figures are the average values among four tests on XML Schema presented in Table 1.

The comparision results showed in Figs. 5, 6, and 7 indicate that our DupS significantly outperforms the other methods at all thresholds, followed by the methods of Do, Melnik, and Amann. Among related approaches, Do's method is different in purpose, its target is to find the matches between XML Schema and the ontology. Therefore, in order to compare with Do's method, we also find the matches between the XML input and the RDF output and then compare the number of true matches with Do's method.

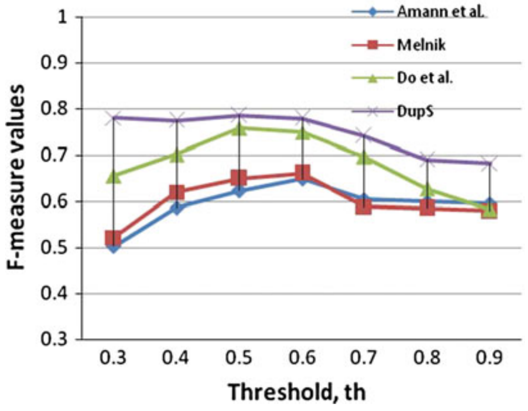


Fig. 7 F-measure values of DupS and related approaches

The F-measure values in Fig. 7 show that when the threshold values increase, the quality of Do's method decreases, especially from the threshold *0.7* and higher. The main reason is that when the number of duplicates in XSD increases, Do's method generates higher wrong matches, since it only relies on the name of elements during matching process. Melnik's method is better than Amann's method because when it transforms XML elements into RDF, it preserves the path of elements as the name in RDF ontology, therefore the element names in destination are still same with element names in source, whereas Amann's method renames most of XML elements by adding prefixes before these names.

From three above figures, we can see that from the threshold *0.7* and higher, most of values have little changes. Therefore, we can use the value *0.7* to divide the duplicate similarities into two groups: Those values are equal or higher than *0.7* are considered as highly similar, otherwise are less similar so we can renamed them in the RDF ontology.

5 Conclusions

In this paper, we have proposed a procedure to transform XML data into RDF statements by using RDF schema vocabularies. Our proposed method is a novel resolution method for current transformation approaches in converting XML data into RDF ontology. While other current methods consider each XML element including duplicates as separated elements, we measure the semantic similar of duplicate pair and group the highly similar duplicates as one representation.

In particular, our procedure outperforms the existing methods due to the following four reasons. Firstly, the duplicate element problem is solved by our proposing metrics, so the transforming result is a correct, complete and unique presentation of the duplicate elements having highly semantic similarities. Secondly, it transforms all the elements of the XML document into RDF while retaining the original structure and enriching the meaning for the source document. Thirdly, elements in XML are clarified in classes or properties based on their definition in XSD, making the result independent from the users' point of view. Finally, to balance the role of each similarity factor, we propose method to determine the weight factor which will improve the accuracy of duplicate similarity computation. If this procedure is executed, a large amount of the XML data will be interpreted as semantic RDF statements which are useful for the Semantic Web.

Acknowledgments This research was supported by the Ministry of Science, ICT&Future Planning (MSIP), Korea, under the Information Technology Research Center (ITRC) support program supervised by the National IT Industry Promotion Agency (NIPA)" [NIPA-2013-(H0301-13-2001)].

References

1. Nayak, R., & Tran, T. (2007). A progressive clustering algorithm to group the XML data by structural and semantic similarity". *Pattern Recognition and Artificial Intelligence*, *21*(4), 723–743.
2. W3C: Ubiquitous Web Domain. (2012). eXensible Markup Language (XML). http://www.w3.org/XML/.
3. Zhou, J., Chen, Z., Tang, X., Bao, Z., & Ling, T. (2012). Fast result enumeration for keyword queries on XML data. *Journal of Computing Science and Engineering (JCSE)*, *6*(2), 127–140.
4. Lee, T. Y., & Cheung, D. W. (2010). Formal models and algorithms for XML data interoperability. *Journal of Computing Science and Engineering (JCSE)*, *4*(4), 313–349.
5. W3School.com. (2012). XML DTD. http://www.w3schools.com/xml/xml_dtd.asp.
6. W3C. (2009). XML Schema. Part I: Structure. http://www.w3.org/TR/xmlschema11-1/.
7. W3C. (2009). XML Schema. Part 2: Datatypes. http://www.w3.org/TR/xmlschema11-2/.
8. Decker, S., van Hartmelen, F., Broekstra, J., Erdmann, M., Fensel, D., Horrocks, I., et al. (2000). The semantic web an the respective roles of XML and RDF. *Journal of IEEE Internet Computing*, *15*(3), 63–74.
9. Ferdinand, M. Zirpins, C., & Trastour, D. (2004). Lifting XML Schema to OWL. In *Web engineering 4th international conference, ICWE* (pp. 354–358).
10. Erdmann, M., & Studer, R. (2001). How to structure and access XML documents with ontologies. *Data and Knowledge Engineering*, *36*(3), 317–335.
11. Klein, M., Fensel, D., van Harmelen, F.,& Horrocks, I. (2001). The relation between ontologies and XML schemas. *Linkoping Electronic Articles in Computer and Information Science, 6*(4).
12. Klein, M. (2002). Interpreting XML via an RDF Schema. In 13th International workshop on personal and expert systems applications (pp. 889–893).
13. Amann, B., Fundulaki, I., Scholl, M., Beeri, C., & Vercoustre, A.-M. (2001). Mapping XML fragments to community web ontologies. In *Fourth international workshop on the web and databases (WebDDB'2001)*.
14. Melnik, S. (1999, December) Bridging the gap between RDF and XML, Technical report. Stanford University. http://infolab.stanford.edu/melnik/rdf/fusion.html.
15. Patel-Schneider, P. F., & Siméon, J. (2002). The Yin/Yang Web: XML syntax and RDF semantics. In *Proceedings of the 11th international world wide web conference (WWW2002)* (pp. 443–453).
16. Thuy, P. T. T., Lee, Y. K., Lee, S. Y., & Jeong, B. S. (2007). Transforming Valid XML Documents into RDF via RDF Schema. *IEEE CS: International conference on next generation web services practices* (pp. 35–40).
17. Goyal, S., & Westenthaler, R. (2004). RDF Gravity (RDF Graph Visualization Tool). http://semweb.salzburgresearch.at/apps/rdf-gravity/.
18. Valencio, C. R., Oyama, F. T., Scarpelini, P., Colombini, A. C., Cansian, A. M., de Souza, R. C. G., et al. (2012). MR-Radix: A multi-relational data mining algorithm. *Human-Centric Computing and Information Sciences (HCIS)*, *2*, 4.
19. Berners Lee, T. (2007, March). *A strawman unstriped syntax for RDF in XML*. W3C.
20. Boden, J. (2001, June). Simplified XML syntax for RDF. Available at: http://www.openhealth.org/RDF/RDFSurfaceSyntax.html.
21. Do, H.-H., & Rahm, E. (2002). COMA: A system for flexible combination of schema matching approaches. In *Proceedings of the very large data bases conference (VLDB)* (pp. 610–621).
22. Yang, D. D., & Powers, D. M. W. (2005). Measuring semantic similarity in the taxonomy of wordNet. In *The 28th Australasian computer science conference (ACSC2005), Australia* (pp. 315–322).
23. Princeton University. (2006). WordNet A lexical database for English. http://wordnet.princeton.edu/wordnet.
24. Algergawy, A., Nayak, R., & Saake, G. (2010). Element similarity measures in XML schema matching. *Journal of Information Sciences*, *180*, 4975–4998.
25. Thuy, P.T.T., Lee, Y.-K., & Lee, S. (2012, April). S-Trans: Semantic transformation of XML healthcare data into OWL ontology. *Knowledge-Based Systems*. doi:10.1016/j.knosys.2012.04.009.
26. Thuy, P. T. T., Lee, Y.-K., & Lee, S. (2012, July). Semantic and structural similarities between XML Schemas for integration of ubiquitous healthcare data. *Personal and Ubiquitous Computing*. doi:10.1007/s00779-012-0567-5.
27. NIST MPEG-7 Validation Service. (2002). http://m7itb.nist.gov/M7Validation.html.

28. Rada, R., Mili, H., Bicknell, E., & Blettner, M. (1989). Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, *19*(1), 17–30.
29. Pan, R., Xu, G., Fu, B., Dolog, P., Wang, Z., & Leginus, M. (2012). Improving recommendations by the clustering of tag neighbours. *Journal of Convergence (JoC), 3*(1), 13–20.
30. Gonzalez, J. L., & Marcelnez, R. (2011). Phoenix: Fault-tolerant distributed web storage based on URLs. *Journal of Convergence (JoC), 2*(1): 79–86.
31. Luo, H., & Shyu, M.-L. (2011). Quality of service provision in mobile multimedia: A survey. *Human-centric Computing and Information Sciences (HCIS)*, *1*, 5.
32. Bechhofer, S. van Harmelen, F. Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., et al. (2004). OWL web ontology language reference. W3C. http://www.w3.org/TR/owl-ref/.
33. Prud'hommeaux, E. (2007, February). *Documentation of the RDF validation service*. W3C Recommendation. http://www.w3.org/RDF/Validator/documentation.
34. Ganguly, P., Rabhi, F. A., & Ray, P. K. (2002). Bridging semantic gap. In *Third Asian Pacific conference on Pattern languages of Program*.

Author Biographies

Pham Thi Thu Thuy received her B.S. from Hanoi University of Science and Technology, Hanoi, Vietnam. She earned her M.S. and Ph.D. in Computer Engineering from Kyung Hee University, Korea. She is a lecturer in the Department of Information Technology at Nha Trang University, Vietnam. Her research interests include database oriented technique, computational statistics, XML technology and the Semantic Web.

Young-Koo Lee got his B.S., M.S. and Ph.D. in Computer Science from Korea advanced Institute of Science and Technology, Korea. He is a professor in the Department of Computer Engineering at Kyung Hee University, Korea. His research interests include ubiquitous data management, data mining, and databases. He is a member of the IEEE, the IEEE Computer Society, and the ACM.

Sungyoung Lee received his B.S. from Korea University, Seoul, Korea. He earned his M.S. and Ph.D. in Computer Science from Illinois Institute of Technology (IIT), Chicago, Illinois, USA in 1987 and 1991, respectively. He has been a professor in the Department of Computer Engineering, Kyung Hee University, Korea since 1993. He is a founding director of the Ubiquitous Computing Laboratory, and has been the director of the Neo-Medical Ubiquitous-Life Care Information Technology Research Center, Kyung Hee University since 2006. He is a member of ACM and IEEE.