Knowledge-Based Systems 35 (2012) 349–356

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Short Communication

S-Trans: Semantic transformation of XML healthcare data into OWL ontology

Pham Thi Thu Thuy, Young-Koo Lee *, Sungyoung Lee

Department of Computer Engineering, Kyung Hee University, Yong-in si, Gyeonggi-do 446-701, Republic of Korea

ARTICLE INFO

Article history:
Received 11 December 2011
Received in revised form 10 March 2012
Accepted 1 April 2012
Available online 10 April 2012

Keywords:
XML healthcare
Semantic transformation
OWL ontology
Semantic measures
XSD
DTD

ABSTRACT

Most healthcare data are available in XML format, which mainly focuses on the structure level and lacks support for data representation. Therefore, a variety of medical applications and medical semantic search engines have difficulty understanding and integrating healthcare data in a highly heterogeneous environment. OWL (Web Ontology Language) and Semantic Web technologies provide an infrastructure that can solve these problems. The aim of our study is to present a mechanism to ease the interpretation and automate the semantic transformation of XML healthcare data into the OWL ontology (S-Trans), which allows an easier and better semantic communication among hospital information systems. On the basis of the XML schemas (XSD or DTD), we extract the document structure and add more descriptions for XML elements. Moreover, to classify the semantic level of duplicate elements in an XML schema, we propose novel metrics to measure the similarity between them. Experimental results show that the proposed method reliably predicts semantic similarity of duplicates and produces a better-quality OWL ontology.

1. Introduction

The rise of XML (eXtensible Markup Language) in patient care has been driven by the needs for communication among health professionals and between healthcare organizations such as hospitals and health insurance companies [1,2]. The main advantage of XML is its flexibility, as it allows creators to describe any content easily by generating their own tags. However, this freedom can cause a lack of understanding and confusion for applications. Because an object can be described by different vocabularies or a vocabulary can express many objects, it is difficult for computers to recognize and differentiate the meaning of given data. Moreover, XML is at a disadvantage when it comes to the semantic interoperability because it focuses primarily on syntax, with no way to describe the semantics of the data [3]. This lack of description creates problems when semantic medical agents seek to understand and reason about these XML healthcare data.

To solve this problem, there is high interest in mapping or transforming XML [4] healthcare data into a semantic supporting language, such as OWL [5]. Furthermore, because DTD (Document Type Definition) and XSD (XML Schema Definition) are syntactic specifications used as models for XML documents, it is necessary to map DTD or XSD to an OWL model to harvest a general structure for the resulting ontology. Although some approaches have been developed to transform XSD or DTD into the OWL ontology, several problems must be resolved. One is the problem of duplicate elements in an XSD or DTD document. Most transforming approaches provide a unique identifier for each schema (XSD or DTD) element by adding a new key element or changing the source element's name [6–10]. However, this solution may lead to data redundancy because duplicate elements may represent the same information. The perfect XML transformation should create a correct, complete, and unique representation of every concept. To obtain this data quality, a similarity computation of duplicate elements is used. In this computation, if two elements have highly similar semantics, they are transformed into one representation.

This paper presents novel metrics to measure the similarity between duplicate elements in an XML schema (XSD or DTD) and proposes a transforming strategy for each similarity level. Compare with the previous studies on transforming XSD or DTD into the OWL ontology, our method is a new technique that solves the duplicate problem efficiently and reliably. Furthermore, the proposed method produces a syntactically legal OWL ontology, which is easily processed and interpreted by semantic applications.

The remainder of the paper is organized as follows. Section 2 presents an overview of the related approaches for transforming XML data into the OWL ontology and measuring the similarity between concepts. Section 3 describes the details of S-Trans method, including the semantic similarity measurement for duplicate elements and the transformation of XML schema into the OWL ontology. Section 4 presents the experimental setup and results. Finally, Section 5 concludes the paper.

* Corresponding author. Tel.: +82 31 201 3732.
E-mail addresses: tttpham@khu.ac.kr (P.T.T. Thuy), yklee@khu.ac.kr (Y.-K. Lee), sylee@oslab.khu.ac.kr (S. Lee).

0950-7051/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.04.009

2. Related work

In this section, we present two research trends that are related to our paper: (1) transforming XML data into OWL, including solutions for the duplicate problem and (2) measuring concept similarity within a single document and computing the element similarities between two different documents.

Several approaches related to schema mapping and transformation from XML data into the OWL ontology have been proposed. Ferdinand et al. [6] described mappings from XML to RDF as well as from XSD to OWL. However, the OWL instances could not suit the OWL model because elements in XML documents were mapped to different OWL concepts depending on the users' judgment. Hannes et al. [7] and Tsinaraki et al. [8] employed XSLT to interpret XSD as OWL ontology. Hannes prevented the duplicate problem by adding prefixes such as *has* and *dtp* before class and property elements, respectively, whereas Chrisa concatenated the ancestor names with the current element name. Similarly, Bernd et al. [9] attached the *key paths* to each nested element, whereas Toni et al. [10] and Cruz et al. [11] used XPath expressions to express the XSD elements. These techniques prevented XML duplicates from occurring in the OWL ontology, but they altered most of the XML element names. Our previous approach [12] only changed the name of duplicate elements by adding their corresponding ancestor name, but we did not consider the duplicate similarity.

Some methods have been introduced to measure the similarity between concepts within a document. Two traditional methods in calculating this similarity were proposed by Rada et al. [16] and Wu & Palmer [17]. They used *conceptual distance* to measure the length of the shortest path that connects concepts in the taxonomy. Leacock and Chodorow [18] and Li et al. [19] also considered degrees of similarity between concepts based on the path length between them, and if two concepts had the same name, their similarity value was 1. By contrast, in our approach, duplicate concepts may have different semantics depending on their cardinality constraints and outside relationships. On the other hand, some approaches measure the element similarity of two different schemas. Do et al. [13] and COMA++ [14] proposed a method to compute the similarity of two XSD elements or between an XSD element and an OWL concept with string matching. Yang and Powers [20] used linguistic taxonomy based on concept definitions in WordNet [21] to gain the most accurate semantics for element names. Recently, some researchers [22–24] have employed additional functions to calculate the similarity of a particular feature of a given schema, such as the similarities of leaf nodes, root nodes, data types, and constraints. All of the partial results are then combined into a final similarity value using a weighted sum function.

In general, approaches in the first category avoid the problem of duplicates by giving a unique identifier to each element, whereas most approaches in the second category assume that the duplicate elements are similar to each other. The S-Trans method has the same purpose as the first category by transforming XML into OWL, but we apply the semantic similarity computation to assess the similarity values of XML duplicate elements before transformation. This computation is also different from the second category because our computation focuses on duplicates within a single schema, and all the similarity values are determined by our proposed metrics without any user intervention.

3. S-Trans

3.1. Semantic similarity of duplicate elements

To illustrate the S-Trans method, we will first restrict ourselves to the hierarchical schemas. The XML schema is displayed as a graph, where the nodes represent the schema elements. We motivate S-Trans with the real XML data set, *prescription.dtd* [15]. The DTD document and its corresponding XSD are displayed in Table 1.

Assuming that all forms of XML data, such as XSD, DTD, and XML instances, can be represented as a tree, we draw a tree of *prescription.dtd* as in Fig. 1. As presented in Fig. 1, the element *name* of the *physician* is similar to the element *name* of the *patient* and is different from the element *name* of the *drug* because *name* of the *drug* contains two children, and the siblings of *name* of the *drug* are slightly different from those of *name* of the *physician* and *name* of the *patient* elements. In contrast, two duplicate elements, *phone* of the *physician* and *phone* of the *patient*, do not contain any children, but they are different in their cardinality constraint. The first *phone* element is declared with '+', whereas the second one is defined with '*'. Furthermore, *month*, *date*, and *year* of *prescribed.date* are repeated in the *DOB (Date Of Birth)* element. These elements have the same cardinality constraint, the same siblings, and children, but they have different parents.

On the basis of the above mentioned observations, we can conclude that there are four main factors that affect the similarity between duplicate elements: the children, the siblings, the parent, and the cardinality constraint. Therefore, our S-Trans similarity measure is the combination of these four factors using a weighted function, which is determined by Definition 1.

Definition 1. The duplicate similarity (*DupSim*) between duplicate elements, e_1 and e_2, in XSD or DTD document is defined as the weighted sum of their parent similarity (*PrSim*), their child similarity (*ChSim*), their sibling similarity (*SbSim*), and their cardinality constraint similarity (*CaSim*):

$$DupSim(e_1,e_2) = \alpha_1^* PrSim(e_1,e_2) + \alpha_2^* ChSim(e_1,e_2) \\ \alpha_3^* SbSim(e_1,e_2) + \alpha_4^* CaSim(e_1,e_2) \quad (1)$$

where α_i ($i = 1, \ldots, 4$) is the weight of the ith property, and

$$\sum_{i=1}^{4} \alpha_i = 1 \quad (2)$$

If the *PrSim* property contributes more than the other properties to the similarity of duplicates, then the weight α_1 of the *PrSim* is greater than the other weights. Without loss of generality, in our demonstration, we assume that all similarity properties have an equivalent role; thus, the weights are $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.25$.

Because the parent similarity between elements depends not only on the parent name and cardinality constraint but also on the distance of the parent to the root node, the parent similarity is determined by the following definition.

Definition 2. Given that the parent name of the element E_1 is e_1, the parent name of the element E_2 is e_2, then the parent similarity (*PrSim*) between two duplicate elements E_1 and E_2 is defined as the weighted sum of the name similarity (*NaSim*) of the parent, the parent cardinality constraint similarity (*CaSim*), and the distance similarity of two parents to the root node (*DtSim*):

$$PrSim(E_1,E_2) = \beta_1^* NaSim(e_1,e_2) + \beta_2^* * CaSim(e_1,e_2) \\ + \beta_3^* DtSim(e_1,e_2) \quad (3)$$

where β_1, β_2, and β_3 are the weight parameters and are similar to the Eq. (2). In this paper, 0.34 is assigned to $\beta 1$ and 0.33 is assigned to both β_2 and β_3.

The name similarity (*NaSim*) computes the string similarity between two elements within a schema. If the element name is declared as a set of words or the short form of some words, the normalization and tokenization steps are required. These steps remove genitives, punctuation, capitalization, stop words (such as, *of*,

Table 1
Example of a DTD document and a part of its corresponding XSD document.

<!ELEMENT prescription	<?xml version="1.0" encoding="UTF-8" ?>
(prescribed.date, patient, drug, physician, interchange?)>	<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<!ELEMENT prescribed.date (month, day, year)>	<xs:element name="prescription">
<!ELEMENT patient	<xs:complexType><xs:sequence>
(name, id+, gender, DOB, address+, phone*)>	<xs:element ref="prescribed.date"/> <xs:element ref="patient"/>
<!ELEMENT drug (drug.name, strength, quantity, sig)>	<xs:element ref="drug"/> <xs:element ref="physician"/>
<!ELEMENT name (#PCDATA)>	<xs:element ref="interchange" minOccurs="0" maxOccurs="1"/>
<!ATTLIST name liquid NMTOKEN #IMPLIED>	</xs:sequence></xs:complexType>
<!ATTLIST name tablet NMTOKEN #IMPLIED>	</xs:element>
<!ELEMENT id (SSN|DEA)>	<xs:element name="patient">
<!ELEMENT SSN (#PCDATA)>	<xs:complexType><xs:sequence><xs:element ref="name"/>
<!ELEMENT DEA (#PCDATA)>	<xs:element ref="id" minOccurs="1" maxOccurs="unbounded"/>
<!ELEMENT gender (#PCDATA)>	<xs:element ref="gender"/> <xs:element ref="DOB"/>
<!ELEMENT DOB (month, day, year)>	<xs:element ref="address" "minOccurs="1" maxOccurs="unbounded"/>
<!ELEMENT address (#PCDATA)>	<xs:element ref="phone" minOccurs="0" maxOccurs="unbounded"/>
<!ELEMENT phone (#PCDATA)>	</xs:sequence></xs:complexType></xs:element>
<!ELEMENT physician (name, address+, phone+, id+)>	<xs:element name="name">
<!ELEMENT month (#PCDATA)>	<xs:complexType mixed="true">
<!ELEMENT day (#PCDATA)>	<xs:attribute name="liquid" type="xs:NMTOKEN" use="optional"/>
<!ELEMENT year (#PCDATA)>	<xs:attribute name="tablet" type="xs:NMTOKEN" use="optional"/>
<!ELEMENT interchange (#PCDATA)>	</xs:complexType></xs:element>...
	</xs:schema>

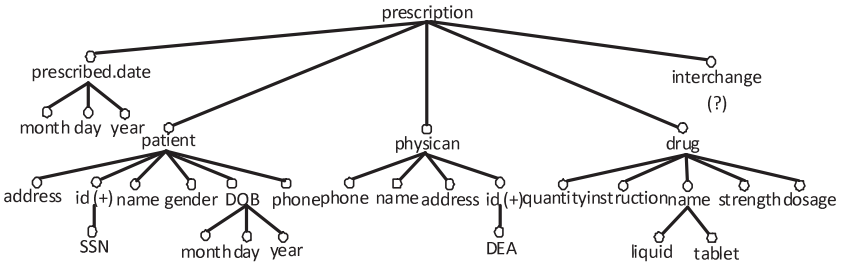

Fig. 1. The corresponding tree of XML schema (XSD/DTD) for Table 1.

and, *with*, *for*, *to*, *in*, *by*, *on*, and *the*), and inflection (plurals and verb conjugations), and replace the short word by its full name. For example, *prescribed.date* becomes *prescribed* and *date*; *DOB* becomes *date*, *of*, and *birth*. The name similarity between two elements e_1 and e_2 is:

$$NaSim(e_1, e_2) = \frac{n_{e_1 \cap e_2}}{\max(n_{e_1}, n_{e_2})} \quad (4)$$

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. For example,

$$NaSim(cure, cured) = \frac{n_{cure \cap cured}}{\max(n_{cure}, n_{cured})} = \frac{4}{5} = 0.8$$

In the case that one of the two elements is processed by the tokenization step, then the name similarity of these two elements is presented as in matrices (5) and (6):

$$NaSim(E_1, E_2) = \begin{bmatrix} NaSim(e_{1_1}, e_{2_1}) & \cdots & NaSim(e_{1_1}, e_{2_n}) \\ \vdots & \ddots & \vdots \\ NaSim(e_{1_m}, e_{2_1}) & \cdots & NaSim(e_{1_m}, e_{2_n}) \end{bmatrix}, \quad m \geqslant n \quad (5)$$

$$NaSim(E_2, E_1) = \begin{bmatrix} NaSim(e_{2_1}, e_{1_1}) & \cdots & NaSim(e_{2_1}, e_{1_m}) \\ \vdots & \ddots & \vdots \\ NaSim(e_{2_n}, e_{1_1}) & \cdots & NaSim(e_{2_n}, e_{1_m}) \end{bmatrix}, \quad m < n \quad (6)$$

where m and n are the number of words in the token sets of the elements E_1 and E_2, respectively; $NaSim(e_1, e_2)$ is the name similarity between elements e_1 and e_2, determined by Eq. (4). The name similarities of two elements E_1 and E_2 in the matrices (5) and (6) are determined by the following Eqs. (7) and (8), respectively:

$$NaSim(E_1, E_2) = \frac{\sum_{i=1}^{m} max_{j=1}^{n}(NaSim(e_{1_i}, e_{2_j}))}{m} \quad (7)$$

$$NaSim(E_2, E_1) = \frac{\sum_{i=1}^{n} max_{j=1}^{m}(NaSim(e_{2_i}, e_{1_j}))}{n} \quad (8)$$

where *max* is the maximum similarity value of each row in the matrices.

The next factor that affects the semantic similarity between two elements is the cardinality constraint. It is declared as *minOccurs* and *maxOccurs* in the XSD document. These terms define the minimum and maximum number of occurrence times of an element in the XML instances.

We propose a novel metric to measure the constraint similarity (*CaSim*) between two duplicate elements e_1 and e_2. For the determined values of *minOccurs* and *maxOccurs*, we use the following equation to compute their cardinality constraint similarity:

$$CaSim(e_1(i_1, a_1), e_2(i_2, a_2)) = \frac{\left(1 - \frac{|e_1.i_1 - e_2.i_2|}{e_1.i_1 + e_2.i_2}\right) + \left(1 - \frac{|e_1.a_1 - e_2.a_2|}{e_1.a_1 - e_2.a_2}\right)}{2} \quad (9)$$

In (9), i_1 and i_2 are short forms of *minOccurs* for elements e_1 and e_2, whereas a_1 and a_2 are similarly short forms of *maxOccurs*. The *minOccurs* values are usually either 0 or 1, and *maxOccurs* values are either 1 or *unbound*. In most cases, the value of *maxOccurs* is undetermined (*unbound*). To compute the value of *CaSim* for the *unbound* value, we define the following function:

$$e_1[maxOccurs = unbound] = 5 * \max(e_2[maxOccurs]) \quad (10)$$

where *max* is the determined value of *maxOccurs* of the element e_2.

Eq. (10) is proposed based on our survey on the dataset (XSD and XML instances). The appearance time of an attribute with *maxOccurs = unbound* is usually approximately 5 times greater than the maximum values of the determined *maxOccurs*. In the case that all *maxOccurs* in the XSD document are undetermined (*unbound*), we assign them values of 5. Applying Eq. (9) to this value, we obtain the similarity of the attribute's cardinality constraint.

Sometimes the values of the *minOccurs* and *maxOccurs* are retrieved from other indicators. For instance, *maxOccurs* and *minOccurs* values of *Order* and *Group* indicators (*any*, *all*, *choice*, *sequence*, *group name*, and *group reference*) are 1.

For the DTD document, the cardinality constraints are declared by the characters "?" (*min* = 0, *max* = 1), " + " (*min* = 1, *max* = *unbound*), and "∗" (*min* = *0*, *max* = *unbound*). Similarly, we can determine the value *CaSim* of an element in a DTD document by applying Eqs. (9) and (10). For elements without constraint declarations, we assume that their *min* and *max* values are 1.

The datatype similarity (*DtSim*) property employs the distance measure [17] to compute the distance similarity of each datatype pair in the hierarchy tree proposed by W3C as follows:

$$DtSim(e_1, e_2) = \frac{2^* D_3}{D_1 + D_2 + 2^* D_3} \tag{11}$$

where D_3 is the number of nodes from the common node of elements e_1 and e_2 to the root node, and D_1 and D_2 are the number of nodes from the elements e_1 and e_2, respectively, to their common node.

The child similarity of duplicate elements in a schema is computed based on Definition 3, as follows:

Definition 3. Given that the child name of the element E_1 is e_1 and the child name of the element E_2 is e_2, the child similarity (*ChSim*) between two duplicate elements E_1 and E_2 is defined as the weighted sum of the name similarity (*NaSim*) and the cardinality constraint similarity (*CaSim*).

$$ChSim(e_1, e_2) = \delta_1^* NaSim(e_1, e_2) + \delta_2^* CaSim(e_1, e_2) \tag{12}$$

where δ_1 and δ_2 are the weight parameters, $\delta_1 = \delta_2 = 0.5$; *NaSim* is determined by Eqs. (4), (7), (8); *CaSim* is measured by Eqs. (9) and (10).

Usually, one element contains more than one child, thus the child similarity of duplicate elements can be presented as matrices (13) and (14):

$$ChSim(E_1, E_2) = \begin{bmatrix} ChSim(e_{1_1}, e_{2_1}) & \cdots & ChSim(e_{1_1}, e_{2_s}) \\ \vdots & \ddots & \vdots \\ ChSim(e_{1_r}, e_{2_1}) & \cdots & ChSim(e_{1_r}, e_{2_s}) \end{bmatrix}, \quad r \geqslant s \tag{13}$$

$$ChSim(E_2, E_1) = \begin{bmatrix} ChSim(e_{2_1}, e_{1_1}) & \cdots & ChSim(e_{2_1}, e_{1_r}) \\ \vdots & \ddots & \vdots \\ ChSim(e_{2_s}, e_{1_1}) & \cdots & ChSim(e_{2_s}, e_{1_r}) \end{bmatrix}, \quad r < s \tag{14}$$

where r and s are the numbers of child elements of E_1 and E_2, respectively; $ChSim(e_1, e_2)$ is the child similarity of elements e_1 and e_2, and it is determined by Eq. (12). The child similarities of two elements E_1 and E_2 in the matrices (13) and (14) are determined by Eqs. (15) and (16), respectively:

$$ChSim(E_1, E_2) = \frac{\sum_{i=1}^{r} max_{j=1}^{s}(ChSim(e_{1_i}, e_{2_j}))}{r}, \quad r \geqslant s \tag{15}$$

$$ChSim(E_2, E_1) = \frac{\sum_{i=1}^{s} max_{j=1}^{r}(ChSim(e_{2_i}, e_{1_j}))}{s}, \quad r < s \tag{16}$$

In the case that one of the duplicate elements is a leaf node (meaning it contains no child node), the child similarity of this twin element is 0.

Similar to child similarity, sibling similarity between duplicate elements is computed based on the following equation:

$$SbSim(e_1, e_2) = \varepsilon_1^* NaSim(e_1, e_2) + \varepsilon_2^* CaSim(e_1, e_2) \tag{17}$$

where ε_1 and ε_2 are the weight factors, $\varepsilon_1 = \varepsilon_2 = 0.5$.

Given that the number of sibling elements in E_1 is x and the number of sibling elements in E_2 is y, the matrix of sibling similarity is similar to (13) if $x \geqslant y$ and similar to (14) if $x < y$. The sibling similarities of duplicate elements in these two cases are computed by Eqs. (18) and (19):

$$SbSim(E_1, E_2) = \frac{\sum_{i=1}^{x} max_{j=1}^{y}(SbSim(e_{1_i}, e_{2_j}))}{x}, \quad x \geqslant y \tag{18}$$

$$SbSim(E_2, E_1) = \frac{\sum_{i=1}^{y} max_{j=1}^{x}(SbSim(e_{2_i}, e_{1_j}))}{y}, \quad x < y \tag{19}$$

Depending on the expected similarity value, the duplicate elements can be classified into two groups, similar and non-similar. The transforming strategies in Section 3.2 are then applied to transform these duplicates into the appropriate OWL concepts. In this paper, we use the threshold value 0.7 to classify the duplicate elements. The value of 0.7 was chosen based on our observation of experimental results: At the threshold of 0.7, the error rate of classification is greatly smaller than other thresholds. See Section 4.2 for details.

3.2. Transforming DTD/XSD into the OWL ontology

According to the characteristics of OWL definitions and the functionality of DTD elements, we define suitable transforming notations from DTD elements into OWL concepts. In OWL, a class defined by *owl:class* identifies a class or a non-instance item of the ontology. Because *DOCTYPE* defining the root element of the document usually contains other elements and attributes, it is converted to an *owl:class*. If *ELEMENT* includes other elements or attributes or refers to an entity notion, it is mapped to the *owl:class* as well. *ATTLIST*, which normally defines the attributes of the document, is transformed into *owl:DatatypeProperty* by default. However, if *ATTLIST* contains other entity references, we map it to an *owl:class* because it has the same functionality of the class. This notion was not mentioned in our previous work [12].

In OWL, a property is divided into two types, *ObjectProperty* and *DatatypeProperty*. Because *ObjectProperty* specifies the relationship between two instances that belong to the same or different classes [7], we use *ObjectProperty* to describe the relationship among OWL classes. The *DatatypeProperty* indicates the relationship between instances and RDF literals. The *rdfs:domain* and *rdfs:range*, which restrict the anterior and posterior values of a property, respectively, are used as a supplement for the *DatatypeProperty* and *ObjectProperty*. Moreover, to prevent a member of one class from being a member of another class, we used *owl:disjointWith*. Other OWL descriptions such as *owl:unionOf, owl:DataRange, owl:oneOf, rdf:first, and rdf:rest* are also utilized to improve the expressiveness of OWL attributes. Details of the transformation model from DTD and XSD into OWL are presented in Table 2.

Furthermore, unlike XML, OWL does not allow identical names. Thus, when our procedure meets an element or attribute that has the same name as the previous node, two solutions are proposed:

Table 2
The transforming correspondences between DTD/XSD and OWL.

DTD	XSD	OWL representation		
		Type	rdfs:domain	rdfs:range
DOCTYPE (root)	element@name, complexType	owl:class, owl:disjointWith owl:ObjectProperty	Class name	Child name
ELEMENT contains				
Other elements	element@name, complexType	owl:class, owl:disjointWith owl:ObjectProperty	Class name	Child name
ENTITY reference only	element@ref	owl:class, owl:disjointWith		
Data type only	element@name, complexType mix="true"| simpleType|element@type	owl:DatatypeProperty	Attribute name	Datatype
ENTITY contains				
>1 attributes	element@name, complexType mix="true" simpleType|>1 attribute@name	owl:DatatypeProperty, owl:subPropertyOf	Attribute name	Datatype
One attribute	element@name, complexType mix="true"| simpleType|element@type	owl:DatatypeProperty	Attribute name	Datatype
ATTLIST contains				
Other property	>1 attribute@name, extension@base| restriction@base	owl:DatatypeProperty rdfs:subPropertyOf	Attribute name	Datatype
Data type only	1 attribute name	owl:DatatypeProperty	Attribute name	Datatype
ENTITY reference	attribute@ref	owl:class, owl:ObjectProperty	Class name	Child name
ELEMENT element-name (child1, child2, ...)	sequence	owl:intersectionOf		
ELEMENT element-name (child1|, child2|, ...)	choice	owl:unionOf		
+, ∗, ?	maxOccurs| minOccurs	owl:maxCardinality, owl:minCardinality		

(1) If this duplicate element is highly similar to the previous element, the procedure uses *owl:unionOf* to connect the parent nodes of these duplicates in the same domain.
(2) Otherwise, the procedure renames the duplicated element by adding the parent element's name along with an underscore '_' character between the parent's name and the duplicate's name.

For example, because the value of *DupSim* between elements *name* of the *patient* and *name* of the *physician* is 0.8, which is higher than our threshold, the two elements *name* are combined to one element *name* and their domain consists of the parents of the two elements. The OWL description of two elements *name* is as follows:

```
<owl:DatatypeProperty rdf:about="#name">
  <rdfs:domain> <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#physician"/>
      <owl:Class rdf:about="#patient"/>
    </owl:unionOf> </owl:Class>
  </rdfs:domain>
<rdfs:range rdf:resource=
        "http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

However, because the value of *DupSim* between elements *name* of the *drug* and *name* of the *physician* (or *patient*) is 0.48, which is lower than our threshold (0.7), the element *name* of the *drug* is renamed by adding its parent name before its name. Its OWL definition is as follows:

```
<owl:DatatypeProperty rdf:ID="drug_name">
  <rdfs:range rdf:resource=
      "http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#drug"/>
</owl:DatatypeProperty>
```

4. Experiment evaluation

4.1. Evaluation setup

In this section, we describe our experimental setup. We first discuss the implementation language for transformation. Then, we present performance metrics for evaluating the quality of transformation, and then describe the real-world schemas used in the experiment.

The transformation language used in this experiment is XSLT (eXtensible Stylesheet Language Transformation) [29]. XSLT is applied to an XML schema with Visual C#. We choose XSLT because its natural purpose is to act as a transformation tool for extensible languages.

We evaluate the proposed transforming strategies by matching an XSD document with an OWL ontology to determine the true matches, and compare our results with related methods. To assess the quality of the matching system, we use *precision* and *recall* [13,25]. Given the reference matching, R (such as that resulted by related methods), the *Precision* of the matching produced by
S-Trans, T, is computed as the following equation:

$$Precision(R,T) = \frac{|R \cap T|}{|T|} \tag{20}$$

Recall specifies the share of real correspondences:

$$Recall(R,T) = \frac{|R \cap T|}{|R|} \tag{21}$$

Although precision and recall are the most widely used measures, when comparing matching systems, one may prefer to have only a single measure. For this reason, two measures, *F-measure* and *Overall* [13], are introduced to aggregate the precision and recall.

$$F\text{-}measure = 2^{*}\frac{precision^{*}recall}{precision + recall} \tag{22}$$

Table 3
The characteristics of the tested schemas.

#	Schema name	File size (KB)	# Nodes	Max depth	# Duplicates
1	Drug_medicament	180	683	9	0
2	Patient-admission	40	240	4	7
3	Healthcaremetadata	5523	1071	11	16
4	Pathology.report	328	778	5	14

Overall is defined as follows:

$$overall = recall^{*}\left(2 - \left(\frac{1}{precision}\right)\right) \quad (23)$$

To obtain practical evidence, we applied our transformation to four healthcare XML documents: particularly, (1) *drug_medicament.xml* [26], (2) *genertic_risk.xml* [26], (3) *healthcaremetadata.xsd* [27], (4) *pathology.report.xml* [27]. The corresponding schemas of the test cases numbered (1), (2), and (4) are generated with a schema converter [28]. The characteristics of the four schemas are presented in Table 3.

Because there is no supporting tool for matching between DTD and the OWL model, in this section, we only compare our transformation of XSD into the OWL ontology with one transforming method proposed by Hannes et al. [7] and with two matching methods introduced by Toni et al. [10] and COMA++ [14].

4.2. Results

This section provides evaluation results for the test cases in Section 4.1. First, we conducted an experiment to determine the best threshold value for classifying duplicate elements. Second, we carried out another set of experiments to compare our work with related approaches. Lastly, we assessed the quality of each proposed measuring factor with S-Trans.

Fig. 2 shows how error rates for classifying duplicate elements change as we use different threshold values for the *DupSim* measure. For the three schemas including duplicate elements in Table 3, we manually classified into two groups: similar and non-similar, then computed the classification error rate at each threshold (in the range 0.1–1.0). The weighted average of the error rates for the three schemas is computed by using the number of duplicate pairs in each schema as the weighted factor.

Fig. 2 shows that very small or very large threshold values result in a large number of error rates. Particularly, at the threshold values ranging between 0.1 and 0.3, the average error rate of the classification is nearly 38%. This number decreases to approximately 27% at the threshold values between 0.45 and 0.55 and declines to 2% at the threshold value of 0.7. From the threshold values of 0.75 and higher, the error rate values steadily climb up and are highest (62%) at the threshold values of 0.95 and 1.0. Because the error rate of classification achieves the minimum value at the threshold of 0.7, we use 0.7 as the classifying value to separate the duplicates into two groups, similar and non-similar.

The comparisons of our work with related methods are shown in Figs. 3–6.

In Fig. 3, S-Trans performs as well as COMA++ and outperforms Hannes and Toni's methods in terms of quality indicators. The main reason for this improved performance is that the *drug_medicament* schema does not contain any duplicate; thus, S-Trans does not change or integrate any XML element. Therefore, the COMA++ method can match 100% of XML elements with OWL concepts. Hannes's method, however, renames all complex and simple elements by adding "*has*" and "*dtp*" prefixes, and Toni's method changes the XML elements by replacing them with the XPath

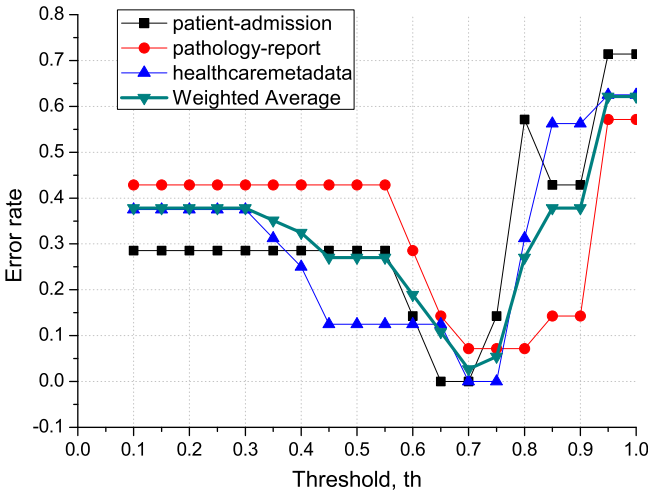

Fig. 2. The error rate of classification at different thresholds.

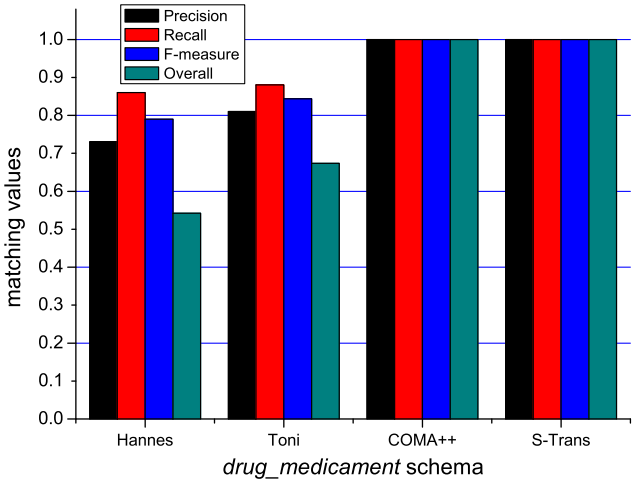

Fig. 3. Evaluation results, *drug_medicament* schema.

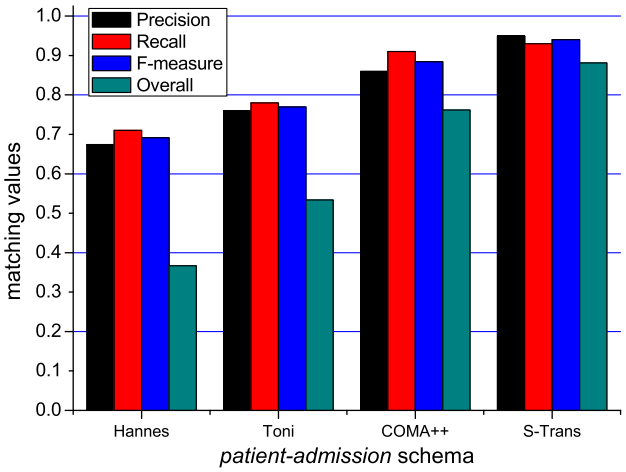

Fig. 4. Evaluation results, *patient_admission* schema.

expressions. Therefore, methods of Hannes and Toni result in fewer matches. However, Toni's method only changes the complex elements, whereas Hannes' method alters most of elements and properties. Therefore, Toni's approach is slightly better

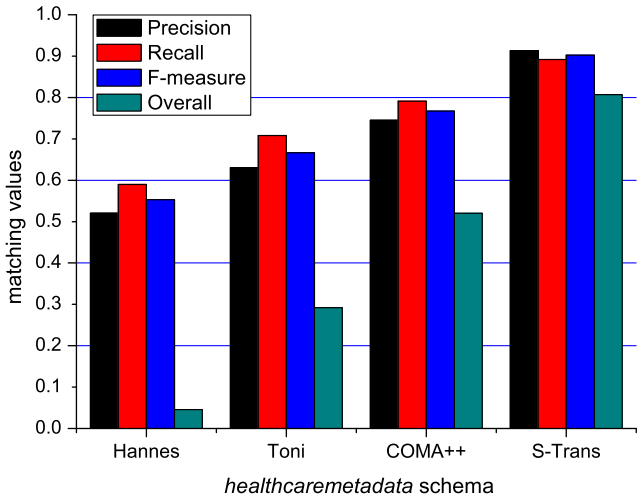

Fig. 5. Evaluation results, *healthcaremetadata* schema.

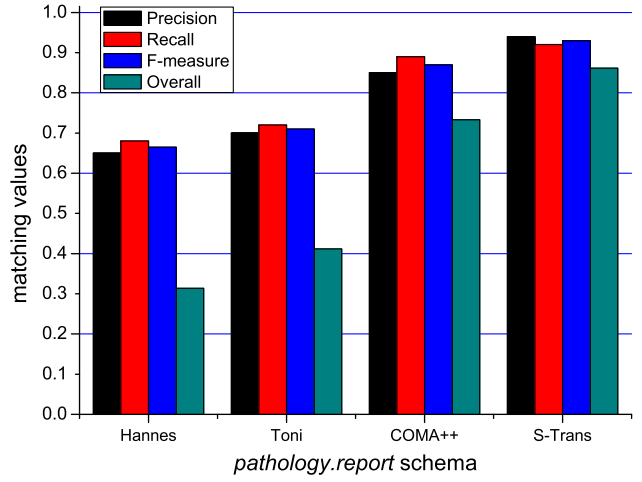

Fig. 6. Evaluation results, *pathology.report* schema.

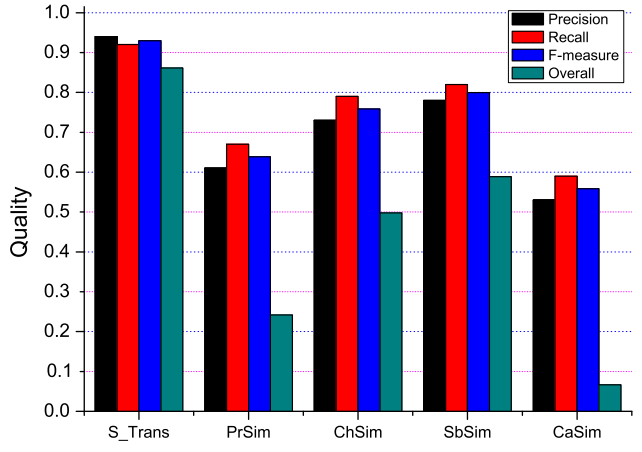

Fig. 7. Quality of *S-Trans*, *PrSim*, *ChSim*, *SbSim*, and *CaSim*.

than Hannes' approach in terms of matching quality between XML and OWL.

In the case of larger-sized XML schema, as well as cases with more duplicate elements, S-Trans still outperforms other methods. Particularly, S-Trans's overall scores are over 84% in three test cases; see Figs. 4–6. Hannes' method works very poorly in test cases #3 (Fig. 6); its overall value is only 4% compared with 31%, 51%, and 84% of Toni, COMA++, and S-Trans, respectively.

Furthermore, to determine the most important factor that affects the similarity values, we separated four similarity factors and compared them with their combination (*S-Trans*). The comparison results are presented in Fig. 7.

Fig. 7 shows that *CaSim* has the lowest quality; its overall quality is only 7% compared with 24% for *PrSim*, 50% for *ChSim*, and 59% for *SbSim*. The reason for this difference in quality is that most duplicates have similar *CaSim*. However, we can also observe that the combination of all similarity factors outperforms *SbSim*; therefore, it is preferable to use multiple similarity measures instead of a single measure.

5. Conclusions

The S-Trans method presented in this paper allows the automatic transformation of XML schema (XSD or DTD) into the OWL ontology. Our procedure outperforms the existing methods due to the following four reasons. First, when transforming all the elements of an XML document into OWL, our algorithm retains the original structure and captures the implicit semantics expressed in the XML document. Second, a component in DTD or XSD is considered as a class or a property or a data type based on its definition and its detail descriptions, which causes the result to be independent from users' opinions. Third, the languages used in our procedure are utilized according to their original functions. DTD or XSD is used for defining XML structure, XML is used for describing instances, and OWL is used for providing definitions and relationships between concepts. To solve the duplicate problem, we proposed several novel metrics to measure the semantic similarity between each duplicate pair. On the basis of the similarity results and the introduced transforming strategies, duplicate elements are transformed into the appropriate OWL concepts. Our experimental evaluation shows that a combination of similarity factors, such as parent, child, sibling, and cardinality constraints, provides the best similarity values of duplicates compared with using single similarity factor.

We hope that this research has created a bridge to narrow the gap between XML data and the OWL ontology in a semantic way. If this method is popularized, a large amount of the XML healthcare data on the Web today can be interpreted into the meaningful OWL ontology, which is particularly useful for the Semantic Web, and medical applications.

Acknowledgment

This work was supported by a grant from the Kyung Hee University in 2011 (KHU-20111209).

References

[1] Mick Seals, The use of XML in healthcare information management, Journal of Healthcare Information Management 14 (2) (2000) 85–95.
[2] M.A. Casteleiro, J. Des, M.J.F. Prieto, R. Perez, H. Paniagua, Executing medical guidelines on the web: towards next generation healthcare, Knowledge-Based Systems 22 (7) (2009) 545–551.
[3] S. Hawke, XML with Relational Semantics: Bridging the Gap to RDF and the Semantic Web. <http://www.w3.org/2001/05/xmlrs/>.
[4] E. Pulvermueller, S. Feja, A. Speck, Developer-friendly verification of process-based systems, Knowledge-Based Systems 23 (7) (2010) 667–676.
[5] M.A. Casteleiro, J.J. Des Diz, Clinical practice guidelines: a case study of combining OWL-S, OWL, and SWRL, Knowledge-Based Systems 21 (3) (2008) 247–255.
[6] M. Ferdinand, C. Zirpins, D. Trastour, Lifting XML schema to OWL, in: Proceedings of 4th ICWE, 2004, pp. 354–358.

[7] Hannes Bohring, Sören Auer, Mapping XML to OWL ontologies, Marktplatz Internet: Von e-Learning bis e-Payment, Leipziger Informatik-Tage, Germany, 2005, pp. 147–156.
[8] C. Tsinaraki, S. Christodoulakis, XS2OWL: a formal model and a system for enabling XML schema applications to interoperate with owl-dl domain knowledge and semantic web tools, in: Proceedings of DELOS, 2007, pp. 137–146.
[9] A. Bernd, B. Catriel, F. Irini, S. Michel, Ontology-based integration of XML web resources, in: The First International Semantic Web Conference, Springer-Verlag, 2002, pp. 117–131.
[10] Rodrigues Toni, P. Rosa, J. Cardoso, Moving from syntactic to semantic organizations using JXML2OWL, Journal of Computers in Industry 59 (2008) 808–819.
[11] C. Cruz, C. Nicolle, Ontology enrichment and automatic population from XML data, in: ODBIS 2008, 2008, pp. 17–20
[12] P.T.T. Thuy, Y.-K. Lee, S.Y. Lee, DTD2OWL: automatic transforming XML documents into OWL ontology, in: International Conference on Interaction Science, ACM, 2009, pp. 125–131.
[13] H.-H. Do, E. Rahm, COMA – a system for flexible combination of schema matching approaches, in: VLDB, 2002, pp 610–621.
[14] Database group Leipzig, COMA++. <http://www.dbs.uni-leipzig.de/Research/coma.html>.
[15] Robin Cover, The Cover Pages: Schema for Patient Medical Record. <http://www.xml.coverpages.org/BordenASTM20010314.html>.
[16] R. Rada, H. Mili, E. Bicknell, M. Blettner, Development and application of a metric on semantic nets, IEEE Transactions on Systems, Man and Cybernetics 19 (1) (1989) 17–30.
[17] Z. Wu, M. Palmer, Verbs semantics and lexical selection, in: Proceedings of 32nd Computational Linguistics, 1994, pp.133–138.
[18] C. Leacock, M. Chodorow, Combining Local Context with WordNet Similarity for Word Sense Identification, MIT Press, 1998. pp. 305–332.
[19] Y. Li, Z. Bandar, D. McLean, An approach for measuring semantic similarity between words using multiple information sources, IEEE Transactions on Knowledge and Data Engineering 15 (4) (2003) 871–882.
[20] D.D. Yang, D.M.W. Powers, measuring semantic similarity in the taxonomy of WordNet, in: ACSC2005, 2005, pp. 315–322.
[21] Princeton University, WordNet_A lexical database for English. <http://wordnet.princeton.edu/wordnet>.
[22] R. Nayak, T. Tran, A progressive clustering algorithm to group the XML data by structural and semantic similarity, Pattern Recognition & Artificial Intelligence 21 (4) (2007) 723–743.
[23] R. Nayak, W. Iryadi, XML schema clustering with semantic and hierarchical similarity measure, Knowledge-Based Systems 20 (2007) 336–349.
[24] A. Algergawy, R. Nayak, G. Saake, Element similarity measures in XML schema matching, Journal of Information Sciences (2010) 4975–4998.
[25] Wikipedia, Precision and Recall. <http://en.wikipedia.org/wiki/Precision_and_recall>.
[26] OSOR Forge Hospital, SCM: Repository. <http://forge.osor.eu/plugins/scmsvn/viewcvs.php/?root=hospital&sortdir=down>.
[27] Robin Cover, The Cover Pages: Schema for Patient Medical Record. <http://xml.coverpages.org/BordenASTM20010314.html>.
[28] A BackOffice Associates, LLC Company, Hit Software. <http://www.hitsw.com/index.html> (retrieved 27.03.11).
[29] D. Foetsch, E. Pulvermueller, A concept and implementation of higher-level XML transformation languages, Knowledge-Based Systems 22 (3) (2009) 186–194.