Karaelmas Science and Engineering Journal
Journal home page: http://fbd.beun.edu.tr
DOI: 10.7212/zkufbd.v10i1.1516

Review
Received / Geliş tarihi : 20.08.2019
Accepted / Kabul tarihi : 18.10.2019

Similarity Matching of XML Schema

XML Şemalarda Benzerlik Eşleştirme

Ayse Salman*

Maltepe University, Faculty of Engineering, Computer Engineering, Maltepe, Istanbul, Turkey

Abstract

XML Schema similarity matching problem is a key task in numerous applications, particularly to support data exchange and integration. Solving such match problem has found considerable interest among researchers and several matching approaches have been proposed. However, to achieve the best match, and because the problem is hard several approaches must be combined. This paper describes a system that can find accurate matches by combining matching on two levels. First finding the best mapping between the schema elements by comparing their internal properties, then comparing elements similarity based on their context inside the schema.

Keywords: Context similarity, Internal similarity, Schema matching, Similarity measure, XML Schema

Öz

XML şemalarının benzeşim problemi bir çok uygulamada ortaya çıkmakta olan önemli bir hususdur. Özellikle de veri alışverişi ve entegrasyonu söz konusu olduğunda kilit rol oynamaktadır. Bu gibi durumlarda, benzeşim eşleştirme çözümleri bulmak, araştırmacıların ilgisini çeken bir konu olmus ve buna istinaden bir takım eşleştirme önerileri ortaya çıkmıştır. Fakat en başarılı eşleştirme tespitinin tek başına uygulanmasi yerine, genel olarak problemin karmaşıklığından dolayı birden fazla yöntemin beraber uygulanması gereklidir. Bu makale iki farklı aşamada benzeşimleri birleştirerek, doğru eşleştirmeler yapabilen bir sistemi tariflemiştir. Bu aşamalardan ilki, şemanın elemanlarını, yapıları bakımından kendi aralarında karşılaştırarak gruplamak, diğeri ise elemanları şemanın genel bağlamı doğrultusunda değerlendirmek şeklindedir.

Anahtar Kelimeler: İç benzeşim, Bağlamsal benzeşim, Şema eşleştirme, Benzeşim ölçeği, XML şema

1. Introduction

Extensible Markup Language, W3C XML, is now considered as the de facto standard for representing and exchanging data on the Internet (Yergeau et al., 2004). XML documents are created to follow certain structure rules defined through XML Schema Definition (XSD) or XML Document Type Definition (DTD). Similarity between XML documents can thus be conducted effectively using their relevant structure definition whether XSD or DTD (Bex et al. 2004).

XSD, or XML Schema, is widely used to describe XML documents than DTD because of its advantages. XML Schema is itself an XML document that uses XML syntax and is also more expressive than DTD. It has a rich type system and uses namespaces to support for schemas distribution (Bex et al. 2004, Lee et al. 2001). XML Schema has generally replaced XML DTD and became the standard used to define the structure of the XML data. This current work thus deals only with XML Schemas. However, data described using XML Schemas bring a challenging issue when integrating and exchanging such data. One of the main problems is how to assess the similarity of XML Schema documents and find metrics for measuring the resemblance between them (Thuy et al. 2012, Wu and Palmer 1994, Do and Rahm 2002, Lee et al. 2002, Tekli et al. 2007-2009, Algergawy et al. 2010, Nierman and Jagadish 2002).

In this work, we measure the similarity of XML schemas by combining schema elements similarity irrelevant of their context with their context similarity. The similarity measure is normalized to be constrained to the interval [0,1]. Similarity can be seen as the numerical distance between data objects with 0 means not similar at all (i.e. has nothing in common) and 1 means completely similar (i.e. logically identical).

*Corresponding author: ayse.salman@maltepe.edu.tr

Ayse Salman orcid.org/0000-0003-2649-3061

1.1. Similarity Measurement

XML Schema represents hierarchically structured information and thus is generally modelled as an Ordered Labelled Trees (OLT) (Yergeau et al. 2004). Each node in the tree represents either an XML element labelled with corresponding element tag name or an element attribute labelled with corresponding attribute name. XML Schema elements are either atomic or complex. Atomic element is either simple element or attribute and is thus represented in the tree by single leaf node. Complex element on the other hand is represented by an internal node in the tree. Attribute nodes appear as children of their encompassing element nodes. A single item that is the basis of the similarity measure in the XML schema tree may be an element node or an attribute node. Example of the XML Schema is shown in Figure 1 that is encoded as tree graph in Figure 2, where the nodes of the tree represent schema elements and attributes.

Our similarity process is done in two phases (Thuy et al. 2012, Tekli et al. 2007, Algergawy et al. 2010). In the first phase we compute the similarity coefficients of nodes exploiting the main properties of elements irrespective of the context. We base this on three properties linguistics (names of Schema elements), on datatypes (uses data types properties), and on cardinalities (elements cardinality). This can be called element internal matching (Algergawy et al. 2010). In the second phase we compute the similarity coefficients of schema elements based on the similarity of their contexts i.e. positions in the tree structure. This can be called element context matching.

2. Tools and Methods

Similarity matching themes exploit the attributes being possesed by the target object being matched. Essentially there are 3 components under which available matching approaches are grouped; element internal similarity matching, contex similarity matching and schema similarity matching. In this chapter, the different kinds of similarity matching procedures that fall under the aforementioned groups will be discussed.

2.1. Linguistic or Name Similarity

The earliest methods used in this field takes the linguistic or name similarity of each pair of element based on their syntactic and semantic relationships. Syntactic similarity can be done by comparing the character strings representing the names, and semantic similarity by comparing their meanings. In order to compare between two names, we should first prepare their two strings for the process of comparison by a tokenization algorithm (Ramasubramanian and Ramya 2013). Tokenization is the process of separating the string into a sequence of character units that are useful for processing called tokens by using delimiters such as punctuation and upper case. Some characters might have

```
<?xml version="1.0"?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="Books">
 <xsd:complexType> <xsd:sequence>
   <xsd:element name="Book" maxOccurs=" unbounded ">
   <xsd:complexType> <xsd:sequence>
    <xsd:element name="Name">
    <xsd:complexType> <xsd:sequence>
      <xsd:element name="Title"    type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string"/>
      <xsd:element name="Edition" type="xsd:string"/>
    </xsd:sequence> </xsd:complexType>
    </xsd:element>
    <xsd:element name="Publisher">
    <xsd:complexType> <xsd:sequence>
      <xsd:element name="PubName" type="xsd:string"/>
      <xsd:element name="ISBN"       type="xsd: unsignedInt"/>
     </xsd:sequence> </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="BookID" type="xsd:string" use="required"/>
  </xsd:complexType>
  </xsd:element>
 </xsd:sequence> </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Figure 1. Example of XML schema for books.

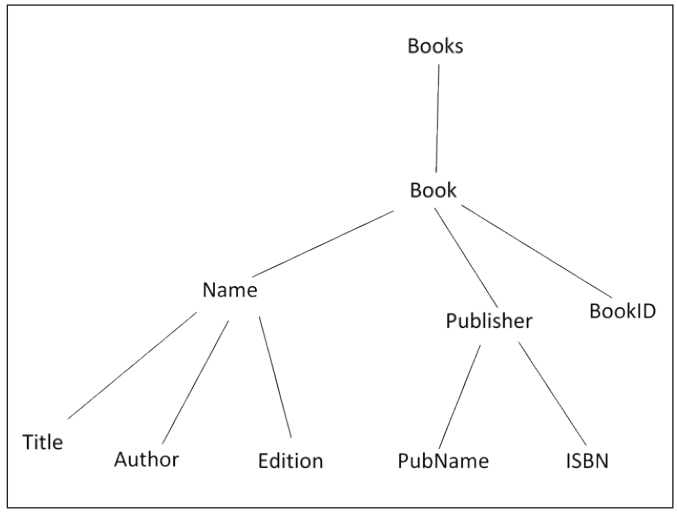

Figure 2. Tree structure for XML schema for books.

to be ignored e.g. some punctuation marks, and some units might have to be expanded to their full text form such as abbreviations (e.g. Pub for Publisher) and acronyms. Once tokenization is done linguistic similarity is measured as follows.

1. The syntactic relationship can be measured by using a string edit distance, the most commonly used is the Levenshtein distance (Rice et al. 1997). In this method, the similarity value of two strings is measured as the optimal (i.e. minimal) cost of transforming one string to another through a sequence of three edit single character operations: deletion, insertion, and substitution with each edit operation has a unit cost. Thus, the normalized syntax similarity measure SynSim is given by

$$SynSim(e_1,e_2) = 1 - \frac{EDist(e_1,e_2)}{\max(|e_1|,|e_2|)} \quad (1)$$

Where $EDist(e_1, e_2)$ is the edit distance algorithm for the two element name strings e_1 and e_2.

2. To measure the semantic similarity between two names e_1 and e_2 we measure the similarity between their two sets of tokens T_1 and T_2 by using knowledge resource (e.g. thesaurus) such as WordNet thesaurus. The lexical database WordNet organizes nouns and verbs (of English) into a taxonomy of is–a relations. One natural way to compare between two words for similarity is to base it on the distance i.e. length of the path between them in the is-a taxonomy. The shorter the path from one node to the other, the more similar they are. Several WordNet-based similarity measures were based on path lengths represented by the number of edges between concepts or names (Wu and Palmer 1994) and the works in (Leacock and Chodorow 1998, Resnik 1995). Wu and Palmer's proposed a measure of the semantic similarity between two concepts based on their depths in the taxonomy and that of their least common super-concept that subsumes them both called the Least Common Subsumer (LCS). These properties are combined into a similarity measure using the Wu and Palmer's equation (3) below and also illustrated in Figure 3.

$$TokSim(n_1,n_2) = \frac{2 \times depth(LCS)}{depth(n_1) + depth(n_2)} = \frac{2 \times L_0}{L_1 + L_2 + 2 \times L_0} \quad (2)$$

where L_1 and L_2 are the numbers of is-a edges from tokens n_1 and n_2 to their LCS n_0, and L_0 is the number of is-a edges from n_0 to the root of the hierarchy.

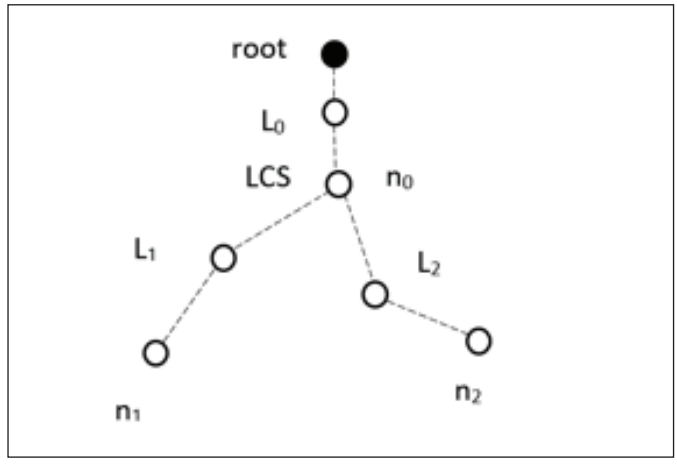

Figure 3. The hierarchy structure for semantic similarity measure

The semantic name similarity of two elements e_1 and e_2 is measured by the semantic similarity of their two sets of tokens T_1 and T_2. This is measured as the average of the best similarity of each token in one set with a token in the other set. It is calculated from (Algergawy et al. 2010, Niwattanakul et al. 2013) as follows:

$$SemSim(e_1,e_2) = \frac{\sum t_1 \in T_1 \left[\max_{t_2 \in T_2} TokSim(t_1,t_2)\right] + \sum t_2 \in T_2 \left[\max_{t_1 \in T_1} TokSim(t_1,t_2)\right]}{|T_1| + |T_2|} \quad (3)$$

Name similarity $NaSim$ is basically measured by semantic matching between two elements. To combine semantic and syntactic similarity values the semantic matching is first applied. If semantic similarity is below certain predefined *threshold*, syntactic matching is then applied and similarity is taken as the average of semantic $SemSim$ and syntax $SynSim$ similarity measures.

2.2. Data Type Similarity

XSD has a rich type system and any element has a type. Hence for effective schema matching, in addition to the similarity of element names the similarity of their datatypes must be measured. Using type information is then the key factor for schema matching and for estimating the degree of similarity between different types (Thuy et al. 2012, Ye at al. 2011). Datatypes in XML form a hierarchy defined by WC3 (Yergeau et al. 2004). An element datatype has one of two main properties, that is either simple or complex. There are 43 built-in simple types (e.g. string, integer), which can be used for element and attribute declarations. Complex types on the other hand are user-defined and can be used for element declarations, they can have elements in

their content and may carry attributes. In addition, there is a set of 12 constraining facets (e.g. length, minLength, maxLength) used to define the valid values for a simple datatype. Associated with each simple datatype is its own subset of facets from that set.

The two most reliable approaches used to establish data type similarity are either based on constraining facets (Algergawy et al. 2010), or the datatypes hierarchy (Dongo et al. 2017, Niwattanakul et al. 2013). However, we should notice that simple datatypes are used to describe leaf element nodes and attribute values of complex datatypes. But there is no link of simple datatype to complex datatype they are of two different categories. Even if two elements have high similarity names (or even identical) but one simple and the other complex the meaning and usage are different. Also, as a complex element contains children, to measure the similarity between two complex elements, we have to measure the similarity of their children which eventually leads to attribute elements. Hence if the two elements are such that at least one is of complex type, their datatype similarity can simply be taken to be 0. To evaluate the similarity between built-in simple types and user-defined simple types the two methods used are:

- Method 1: we can base this on the facets of each datatype. The similarity value between two different datatypes is based on the number of their common facets; the more common facets the more similar they are. Hence the similarity measure between two datatypes nodes d_1 and d_2 can be calculated by their number of common facets divided by their total facets using Jaccard similarity coefficient (Niwattanakul et al. 2013):

$$DaSim(d_1, d_2) = \frac{|D_1 \cap D_2|}{|D_1 \cup D_2|} \quad (4)$$

  Where D_1 and D_2 are the facet sets of elements e_1 and e_2 datatypes respectively

- Method 2: or to base it on datatype hierarchy, usually by using an extension to the one defined by WC3 and that still preserves the relationships between datatypes, as shown by the works in (Hong-Minh and Smith 2007, 29. Dongo et al. 2017). The work in (Hong-Minh and Smith 2007) for example uses the extension shown in Figure 4. The authors propose five new datatype groups: Calendar, Text, Logic, Numeric, and Other. With this hierarchy the datatype similarity measure of two nodes is based on the distance separating them and also their depth. Obviously, datatypes similarity increases as the distance between them decreases. On the other hand, as we go deeper in the hierarchy the differences between datatypes becomes less significant. Hence datatypes similarity also increases as the depth in the hierarchy increases. The similarity measure between two datatypes nodes d_1 and d_2 as shown by the authors in (Hong-Minh and Smith 2007) is given by the following equation:

$$aSim(d_1, d_2) = \begin{cases} e^{-\beta l} \times \dfrac{e^{\alpha h} - e^{-\alpha h}}{e^{\alpha h} + e^{-\alpha h}}, d_1 \neq d_2 \\ 1, otherwise \end{cases} \quad (5)$$

Where l is the shortest path length between d_1 and d_2; h is the depth of the Least Common Subsumer (LCS) datatype which subsumes datatypes d_1 and d_2; and α and β are user-defined parameters.

Table 1 presents the datatype similarity results of some attribute types in the XML Schema using the two methods. As expected, we can see that integer and decimal or int and short have more similarity than with integer and string. The user-defined parameters α and β in equation (5) are taken as α = β = 0.3057 from the work in (Hong-Minh and Smith 2007).

2.3. Cardinality Constraint Similarity

The other key information of an element is its cardinality constraints: *minOccurs* and *maxOccurs*. Their values define the minimum and maximum number of times an element occurs in XML instances. The average of the differences between the minimum number of occurrences and maximum number of occurrences can be taken to measure the similarity of cardinality constraints of two elements as suggested by (Thuy et al. 2012) and this is shown by equation (6):

$$CaSim(e_1, e_2) = \frac{(1 - E_{\min}) + (1 - E_{\max})}{2} \quad (6)$$

where,

$$E_{\max} = \frac{|e_1.\max - e_2.\max|}{e_1.\max + e_2.\max}$$

Table 1. Datatype compatibility table for attribute types in the two methods.

Datatype	Datatype	Similarity Method 1	Similarity Method 2
integer	decimal	0.89	0.62
integer	string	0.25	0.06
date	dateTime	1.0	0.30
boolean	string	0.33	0.09
int	short	1.0	0.70

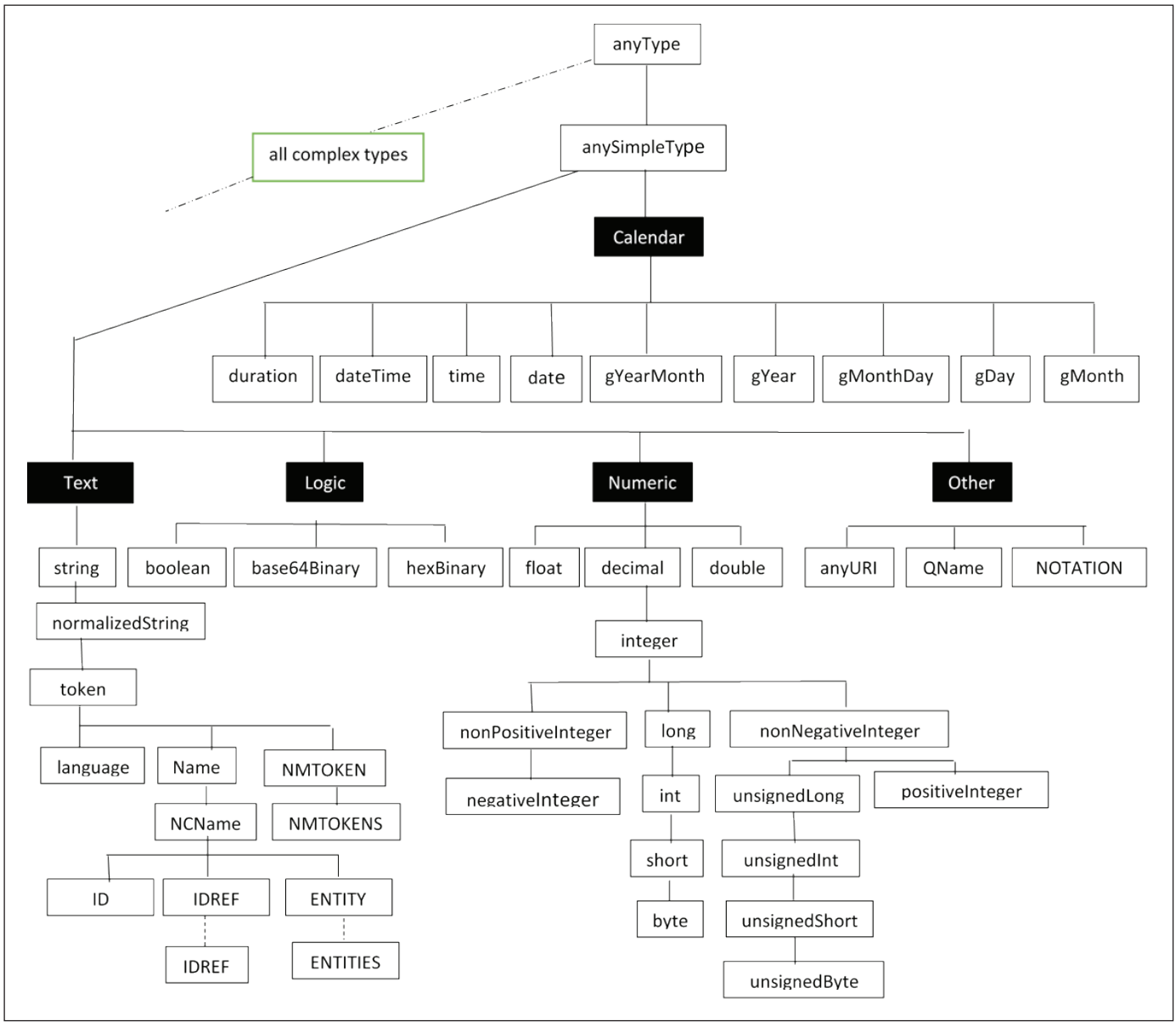

Figure 4. WC3 datatypes hierarchy with extensions from Hong-Minh and Smith 2007.

and

$$E_{\min} = \begin{cases} \dfrac{|e_1.\min - e_2.\min|}{e_1.\min + e_2.\min} \\ 0, e_1.\min = e_2.\min = 0 \end{cases}$$

where *CaSim*(e_1, e_2) denotes the cardinality constraint similarity between two elements e_1 and e_2 and *min* and *max* represent *minOccurs* and *maxOccurs* of an element, respectively. Obviously, the similarity measure is high if the two elements cardinalities are closer. Usually, *minOccurs* is assigned 0 or 1, but *maxOccurs* can be any number ≥ 1 or assigned "unbounded", meaning the element can occur any number of times without specific limit. The default values for both is 1.

If the value of *maxOccurs* is not specific number i.e. *maxOccurs* = "unbounded", its value is undetermined in XML Schemas. If the two matching elements' *maxOccurs* is unbounded, we can take their *maxOccurs* cardinality as equal i.e. $e_1.max=e_2.max$ and E_{max}=0. If only one element's *maxOccurs* is unbounded a value can be chosen as representative and used to calculate E_{max}. One way to achieve this is to survey the dataset (XSD and XML instances) for the particular application considered, as suggested for example in (Thuy

Table 2. Cardinality constraint compatibility table

	minOccurs = 0, maxOccurs = 1	minOccurs = 0, maxOccurs = unbd	minOccurs = 1, maxOccurs = 1	minOccurs = 1, maxOccurs = unbd
minOccurs = 0, maxOccurs = 1	1.00	0.59	0.50	0.09
minOccurs = 0, maxOccurs = unbd	0.59	1.00	0.09	0.50
minOccurs = 1, maxOccurs = 1	0.50	0.09	1.00	0.59
minOccurs = 1, maxOccurs = unbd	0.09	0.50	0.59	1.00

unbd = "*unbounded*"; "*unbounded*" is taken as 10.

et al. 2012), and a value is assigned as a result. However, any approximation in such process is going to affect only E_{max} which contributes to half of the value of the coefficient *CaSim*. The cardinality constraint similarity illustrated in Table 2 represents the most common values of *minOccurs* and *maxOccurs*.

Internal Similarity of two elements e_1 and e_2 is a combination function of name similarity (*NaSim*), data type similarity (*DaSim*), and constraint similarity (*CaSim*) (Thuy et al. 2012, Algergawy et al. 2010). For simplicity we consider linear combination function and internal similarity coefficient is given as the weighted average of the above three similarity coefficients of two elements as follows:

$$IntSim(e_1,e_2) = a \times NaSim(e_1,e_2) + b \times DaSim(e_1,e_2) + c \times CaSim(e_1,e_2) \quad (7)$$

where; $a + b + c = 1$

The value given to a certain weight determines the importance of the measure in determining the node similarity. However different values could be determined by the set of data under consideration. Usually a is given the highest value which reflects the importance of name similarity in deciding the value of element internal similarity coefficient.

2.4. Element Context Similarity Matching

Context similarity matches the schema elements based on the similarity of their XML Schema trees structure. Classical *tree edit distance* measures similarity by the minimal cost of editing operations that transform one tree into the other one (Zhang K. and Shasha 1997). This similarity measuring approach focus on the structural and geometrical characteristics of the trees and does not consider the conceptual semantics of the tree nodes (Allali and Sagot 2004, Guda et al. 2002). Measuring similarity of XML Schema trees requires a more comprehensive method which matches the schema nodes *context*, i.e. its position, in the XML Schema tree. The notion of *context similarity* has been used in many works such as in XClust, Cupid and Similarity Flooding (Lee et al. 2002, Madhavan et al. 2001, Melnik et al. 2002). The *context* of a node in a tree is naturally realized by its ancestors to the root, its descendants to the leaves and its siblings. Hence the context of an element node can be defined by the following four components (Lee et al. 2002, Algergawy et al. 2010, Madhavan et al. 2001) Figure 5:

The ancestor: this is defined by the path from the root node to the element node.

The descendant: this is defined by set of immediate children nodes.

The leaf: this is defined by the set of leaves of the subtrees rooted at the element node. This can also be described by the set of paths from the element node to its leaves. Leaves in XML Schema tree represent the element data content.

The sibling: this is defined by the set of nodes sharing the same immediate ancestor of the element node.

In a schema matching system, the context of a node is thus defined as a combination of the four contexts (Tekli et al. 2007-2009, Nierman and Jagadish 2002). In the following we present some of the commonly used methods for calculating context similarity between elements.

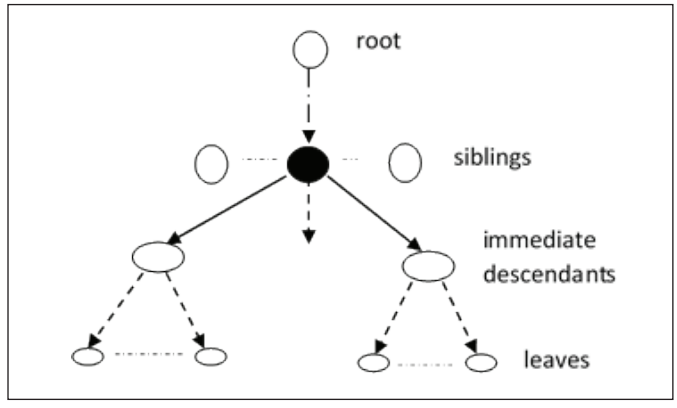

Figure 5. The context of an element.

2.5. Ancestor Context Similarity

The ancestor context for a given element e is the path, i.e. sequence of nodes, extending from the root to node e. To compare between paths, we can use edit distance algorithm as the measure proposed in (Choi et al. 2007). Here the string, the basis for comparison in the algorithm, is the sequence of strings representing element names on each path. The path similarity $ACPSim(e_1, e_2)$, can thus be computed as follows:

$$AnCPSim(e_1,e_2) = 1 - \frac{EDist(P_1,P_2)}{\max(|P_1|,|P_2|)} \quad (8)$$

Where P_1 and P_2 are the ancestor context paths for e_1 and e_2 respectively, $EDist(P_1,P_2)$ is the edit distance operation and $|P_1|$ and $|P_2|$ are the paths lengths as defined by the sequence of element name strings. If P_1 and P_2 are completely similar *EDist* will return 0 and *AnCPSim* will be 1. If P_1 and P_2 have nothing in common in their paths, *EDist* will be equal to $\max(|P_1|,|P_2|)$ and *AnCPSim* will be 0.

2.6. Descendant Context Similarity

Let e_1 and e_2 be the two elements to measure their descendant context similarity, and let their immediate children sets be $\{e_{11},e_{12},\ldots,e_{1K}\}$ and $\{e_{21},e_{22},\ldots,e_{2L}\}$ respectively. For each element in one set we find the maximum similarity value in matching with all elements in the other one. We can calculate the descendant context similarity, $DCSim(e_1,e_2)$ of two elements by taking the average of the maximum similarity values of all element pairs in the two sets (Do and Rahm 2002, Algergawy et al. 2010) as in the following equation:

$$LeCSim(e_1,e_2) = \frac{\sum_{i=1}^{i=K}\left[\max_{j=i}^{j=M} IntSim(e_{1i},e_{2j})\right]}{\max(K,M)} \quad (9)$$

Where K and L are the number of elements of the sets of immediate children of e_1 and e_2 respectively.

2.7. Leaf Context Similarity

For the leaf context similarity of two elements the two most commonly used methods are discussed below.

- We consider the two set of leaf nodes of the two elements e_1 and e_2. For each element we first define the set of leaf nodes of subtree rooted at that element; $L_1(e_1) = \{e_{11},e_{12},\ldots,e_{1K}\}$ and $\{e_{21},e_{22},\ldots,e_{2M}\}$. The internal similarity between each pair of leaves in the two sets is determined, and the matching pairs with maximum similarity values are selected. Hence, the leaf context similarity measure is computed by using equation similar to equation (9):

$$LeCSim(e_1,e_2) = \frac{\sum_{i=1}^{i=K}\left[\max_{j=i}^{j=M} IntSim(e_{1i},e_{2j})\right]}{\max(K,M)} \quad (10)$$

- Another way is to measure the leaf context of a node by the set of paths from the node to leaves in the subtree rooted at the node as suggested by the authors of (Lee et al. 2002, Choi et al. 2007). The leaf context similarity of two elements is then computed by averaging edit distances of the two nodes sets of paths as shown below in the same way as equation (8).

$$LeCSim(e_1,e_2) = \frac{\sum_{i=1}^{i=|P_1|}\left[\max_{j=i}^{j=|P_2|} EDist(P_{1i},P_{2j})\right]}{\max(|P_1|,|P_2|)} \quad (11)$$

where P_1 and P_2 are here the two sets of paths for e_1 and e_2 respectively, and $EDist(P_{1i},P_{2j})$ is the edit distance operation for two element paths, and $|P_1|$ and $|P_2|$ are the number of paths in each set.

2.8. Sibling Context Similarity

In similar way to the descendant context and leaf context, to compute the sibling context similarity between two elements, we compare their sibling context sets. $S_1(e_1) = \{e_{11},e_{12},\ldots,e_{1K}\}$ and $S_2(e_2) = \{e_{21},e_{22},\ldots,\}$ for elements e_1 and e_2. The internal similarity between each pair of siblings in the two sets is determined, and the matching pairs with maximum similarity values are selected. Finally, the average of best similarity values is computed. The sibling context similarity, $SiCSim(e_1,e_2)$, can be computed similar to descendant and leaf contexts above (equations (9) ,(10)) as follows

$$SiCSim(e_1,e_2) = \frac{\sum_{i=1}^{i=K}\left[\max_{j=i}^{j=N} IntSim(e_{1i},e_{2j})\right]}{\max(K,N)} \quad (12)$$

The context similarity of two elements e_1 and e_2 is the combination of the above four similarity measures: *AnCPSim* (ancestor context), *DeCSim* (descendant context), *SiCSim* (sibling context) and *LeCSim* (leaf context). We consider such combination to be linear for simplicity.

$$ConSim(e_1,e_2) = a \times AnCPSim(e_1,e_2) + b \times DeCSim(e_1,e_2) + c \times SiCSim(e_1,e_2) + d \times LeCSim(e_1,e_2) \quad (13)$$

where, $a+b+c+d=1$

2.9. XML Schema Similarity Measurement

Once the internal and context similarity values of two elements e_1 and e_2 in two XML Schema trees is calculated, a total element similarity value $Sim(e_1,e_2)$ can be determined using a weighted sum of these two components by the following equation:

$$Sim(e_1,e_2) = \omega \times IntSim(e_1,e_2) + (1-\omega) \times ConSim(e_1,e_2) \quad (14)$$

Where $0 < \omega \leq 1$. A reasonable value for ω is 0.5 which means equal weight for both internal and context similarity of the two elements.

When we have the element similarity of all element pairs in two schemas, we can then compute the similarity match of the two XML schemas *ScheSim* by the following equation (Nayak and Tran 2007):

$$ScheSim(T_1,T_2) = \frac{\sum_{i=1}^{i=|T_1|} \left[\max_{j=i}^{j=|T_1|} Sim(e_{1i},e_{2j})\right]}{\max(|T_1|,|T_2|)} \quad (15)$$

Where T_1 and T_2 are the two XML trees and their similarity is taken as the average of the sum of the best node similarity values Sim with respect to the maximum number of nodes $|T_1|$ and $|T_2|$ in the two trees.

3. Conclusions

In this paper, we have presented an overview of the existing research related to XML element similarity measures. The paper proposed a combination of schema matching techniques in order to produce best matching results. The matching method relies on both internal and context matchings of the elements of the XML schema. The overall similarity of XML schemas is measured by linearly combining the total similarity of the corresponding individual elements.

4. References

Algergawy, A., Nayak, R., Saake, G. 2010. Element similarity measures in XML Schema Matching. *Inf Sci*, 180(4): 4975–4998. https://doi.org/10.1016/j.ins.2010.08.022

Allali, J., Sagot, M. 2004. Novel Tree Edit Operations for RNA Secondary Structure Comparison. *Workshop on Algorithms in Bioinformatics 2004,* 412-425. https://doi.org/10.1007/978-3-540-30219-3_35

Bex, G. J., Neven, F., Bussche, J. 2004. DTDs versus XML Schema: A Practical Study. *WebDB 2004,* 79-84. https://doi.org/10.1145/1017074.1017095

Choi, I., Moon, B., Kim, H. 2007. A clustering method based on path similarities of XML data. *Data & Knowledge Engineering,* 60(2): 361–376. https://doi.org/10.1016/j.datak.2006.02.004

Do, H., Rahm, E. 2002. COMA-A system for flexible combination of schema matching approaches. *VLDB*, 28: 610–621. https://doi.org/10.1016/B978-155860869-6/50060-3

Dongo, I., Al Khalil, F., Chbeir, R., Cardinale, Y. 2017. Semantic Web Datatype Similarity: Towards Better RDF Document Matching. *International Conference on Database and Expert Systems DEXA,* 28: 189-205. https://doi.org/10.1007/978-3-319-64468-4_15

Guha, S., Jagadish, H. V., Koudas, N., Srivastava, D., Yu, T. 2002. Approximate XML Joins. *ACM SIGMOD 2002,* 287-298. https://doi.org/10.1145/564691.564725

Hall, P., Dowling, G. 1980. Approximate String Matching. *Computing Survey*, 12(4): 381-402. https://doi.org/10.1145/356827.356830

Hong-Minh, T., Smith, D. 2007. Hierarchical approach for datatype matching in XML schemas. *British National Conference on Databases*, 24(1): 120–129. https://doi.org/10.1109/BNCOD.2007.10

Kasim, S., Omar, N., Akbar, N., Hassan, R., Murad, M. 2017. X-Similarity Comparison by Using WordNet. *JOIV*, 1(4-2): 188–191. http://dx.doi.org/10.30630/joiv.1.4-2.79

Leacock, C., Chodorow, M. 1998. Combining local context and WordNet similarity for word sense identification. *In: WordNet,* pp. 265–283. https://doi.org/10.7551/mitpress/7287.003.0018

Lee, M. L., Yang, L. H., Yang, X. 2002. XCLust: Clustering XML Schemas for Effective Integration. *CIKM*, 292–299. https://doi.org/10.1145/584792.584841

Lee, J. W., Lee, K., Kim, W. 2001. Preparations for Semantics-Based XML Mining. *ICDM*, 01: 345–352 https://doi.org/10.1109/ICDM.2001.989538

Madhavan, J., Bernstein, P.A., Rahm, E. 2001. Generic schema matching with Cupid. *VLDB,* 27: pp. 49–58.

Melnik, S., Garcia-Molina, H., Rahm, E. 2002. Similarity Flooding: a versatile graph matching algorithm and its application to schema matching. *ICDE,* 18: 117–128. 352 https://10.1109/ICDE.2002.994702

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. https://doi.org/10.1142/S0218001407005648

Nierman, A., Jagadish, H. V. 2002. Evaluating structural similarity in XML documents. *WebDB,* 5: 61-66. https://doi.org/10.1007/978-3-540-77018-3_31

Niwattanakul, S., Singthongchai J., Naenudorn, E., Wanapu, S. 2013. Using of Jaccard Coefficient for Keywords Similarity. *IMECS*, 01: 380-384. https://doi.org/10.1051/epjconf/20146800004

Princeton University, WordNet A lexical database for English, https://doi.org/10.1145/219717.219748

Ramasubramanian C., Ramya, R. 2013. Effective Pre-Processing Activities in Text Mining using Improved Porter's Stemming Algorithm. *IJARCCE,* 2(12): 4536-4538.

Resnik, P. 1995. Using information content to evaluate semantic similarity in a taxonomy. *IJCAI,* 14(1): 448–453.

Resnik, P. 1999. Semantic similarity in a taxonomy an information-based measure and its applications to problems of ambiguity in natural language. *Journal of Articial Intelligence Research,* 11: 95–130. https://doi.org/10.1613/jair.514

Rice, S. V., Bunke, H., Nartker, T. A. 1997. Classes of Cost Functions for String Edit Distance. *Algorithmica,* 18(2): 271-280. https://doi.org/10.1007/BF02526038

Tekli, J., Chbeir, R., Yetongnon, K. 2007. A hydrid approach for XML similarity. *SOFSEM,* 33: 783–795. https://doi.org/10.1007/978-3-540-69507-3_68

Tekli, J., Chbeir, R., Yetongnon, K. 2009. An overview on XML similarity: background, current trends and future directions. *Computer Science Review,* 3(3): 151–173. https://doi.org/10.1016/j.cosrev.2009.03.001

Thuy, P., Lee Y., Lee, S. 2012. Semantic and structural similarities between XML Schemas for integration of ubiquitous healthcare data. *Journal of Personal and Ubiquitous Computing,* 17(7): 1331–1339. https://doi.org/10.1007/s00779-012-0567-5

Wu, Z., Palmer, M. 1994. Verbs semantics and lexical selection. *Association for Computational Linguistics,* 32: 133–138. https://doi.org/10.3115/981732.981751

WWW Consortium, The Document Object Model, http://www.w3.org/DOM.

Yang, D., David, M. W. 2005. Powers, measuring semantic similarity in the taxonomy of WordNet. *ACSC,* 38: 315–322.

Ye, Y., Li, X., Wu, B., Li, Y. 2011. A comparative study of feature weighting methods for document co-clustering. *IJITCC* 1(2): 206–220. https://doi.org/10.1504/IJITCC.2011.039286

Yergeau, F., Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E. 2004. Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation.

Zhang, K. Shasha, D. 1997. Tree Pattern Matching. Pattern Matching Algorithms. *Oxford University Press-Chapter 11*.