SAXM : Semi-automatic XML Schema Mapping

Nuttakan Amarintrarak[†], Kanda Runapongsa Saikeaw[‡],
Sissades Tongsima[§], Nuwee Wiwatwattana[*]
[†,‡]*Department of Computer Engineering, Khon Kean University, Thailand*
[§]*National Center for Genetic Engineering and Biotechnology, Thailand*
[*]*Pollution Control Department, Bangkok, Thailand*
Email: a.nuttakan@gmail.com[†], krunapon@kku.ac.th[‡],
sissades@biotec.or.th[§], nuwee.w@pcd.go.th[*]

Abstract

Data integration technology becomes essential in processing many types of data. This paper proposes a new method called Semi-Automatic XML schema Mapping (SAXM). Semantic similarity is firstly used to separate unmatchable nodes. Structural similarity and data type compatibility are then applied to ascertain the matching of two nodes. The output of SAXM is a set of similarity relationship values between two XML schemas elements. Moreover, a user can verify the mapping to provide a more accurate result. Experimental results show that our proposed method provides the highest recall and comparable precision among well known approaches.

Key Words: XML schema mapping, schema matcher, semantic matching, structural matching

1. Introduction

Nowadays, data integration is a crucial step in processing many types of data. The integration process comprises of various tasks including data matching, data transformation, and schema/semantic matching. This paper focuses on the schema mapping task and proposes a new method for XML schema mapping without the requirement of XML instances. This is to support the scenario where developers have access to only XML schema but not its XML instances. The output of our solution is a value that indicates the similarity relationship between two XML schemas. The system has an option for users to verify the mapping to get more accurate results. Related work is briefed in Section 2. In Section 3, we discuss about the crux of schema mapping – similarity equations. Then, we describe the system implementation in Section 4 and present experimental results in Section 5.

2. Related Work

Several methods have been presented to solve the schema matching problem. For example, COMA [1] presented a flexible and integrated technique on mapping by combining several techniques. Unlike COMA, our work has been used with real-world applications, particularly in Bioinformatics. Xing [2] presents a mapping algorithm which considers DTD to create mapping results. Unfortunately, DTD is easy to manage but does not support complex data types and user-defined data types.

3. Similarity Equations

For each pair of source and target schema elements, semantic similarity is first computed. Then, only compared nodes that are similar semantically are further computed to find data type compatibility. Structural similarity then is used to detect any difference between the given nodes in source and target schemas. Finally, the overall similarity between the given nodes is computed. Figure 1 shows two input XML schemas which are to compute the similarity value. Throughout this paper, we will use these two schemas in our approach explanation.

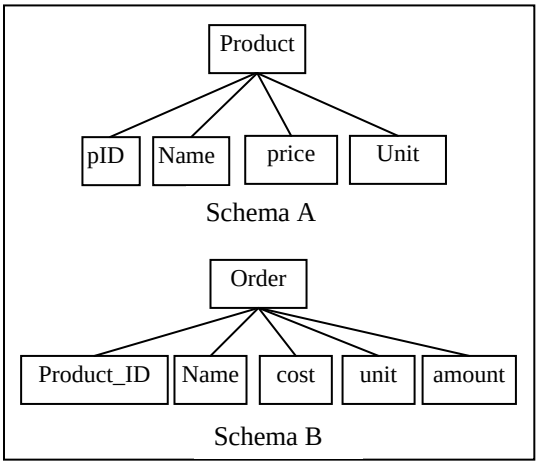

Figure 1. Example input schemas

3.1. *Semantic similarity* is similarity between element names based on WordNet, a lexical database for English [3]. We use WordNet to define a tree relationship of the words. The tree relationship is then used in subsequent computation of semantic similarity. Among the proposed semantic similarity formulas [4,5,6], we adopt the formula from RESNIK

[4] as the default setting because it captures the meaning relationship (either synonym or hypernym) and has been used in many practical domains as shown in Equation 1.

$$SeSim(n_1, n_2) = \max_{C1, C2} \left[\max_{c \in S(c_1, c_2)} [-\log p(c)] \right] \quad (1)$$

where p is the probability of encountering an instance of concept c , $S(c1,c2)$ is the set of concept that subsume both c_1 and c_2 , C is the set of concepts in the taxonomy that are sense of word n.

xsd A / xsd B	Product	pID	Name	price	unit
Order	0.2	0.19	0.52	0.38	0.37
Product_ID	0.56	0.67	0.18	0.15	0.31
Name	0.2	0.19	1	0.38	0.51
cost	0	0.13	0.18	0.94	0.18
unit	0.48	0.19	0.51	0.24	1.0
amount	0.48	0.17	0.18	0.41	0.48

Table 1.Semantic similarity result of the two schemas

Table 1 Shown the semantic similarity value of all element pairs from *schema A* to *schema B* which shown in Figure 1. Just the element pairs which there are semantic similarity value more than 0.5 will be continue compute the next equations.

3.2. After computing semantic similarity, we then compute the *structural similarity* as shown in our Equation 2. Structural similarity is structural matching between sub trees in source and target schemas thus we take into account the position of sibling and ancestor nodes of source node n1 and target node n_2.

$$StrSim(n_1, n_2) = 1/p * (sls / sib) \quad (2)$$

where p is shortest path length of the parent node level of n_1 which has semantic similarity to the parent node of n_2. The shortest path length which is considerable will be less than 10 level, If p is more than 10 or not found, then we assume this parent node similarity relation are very small. The value of p will set to 10. *sls* is the number of sibling nodes of n_1 which have semantic similarity to sibling nodes of n_2. Assume that $s(n_1)$ is the number of sibling nodes of n_1 and $s(n_2)$ is the number of sibling nodes of n_2, then *sib* is $s(n_1)$ when $s(n_1) > s(n_2)$. Otherwise, *sib* is $s(n_2)$.

Figure 2 shows the XML schema trees of *schema A* and *schema B* with their semantic similarity relation. Consider at price element, P = 10 because of, it's parent node which is semantic similarity related with parent node of cost node is not found. *sls* is 3 {pID, Name and unit} and it's amount of sibling nodes is 4 ,then structural similarity between *price element* and *cost element* is

StrSim(price,cost) = 1/10 * (3/4) = 0.075

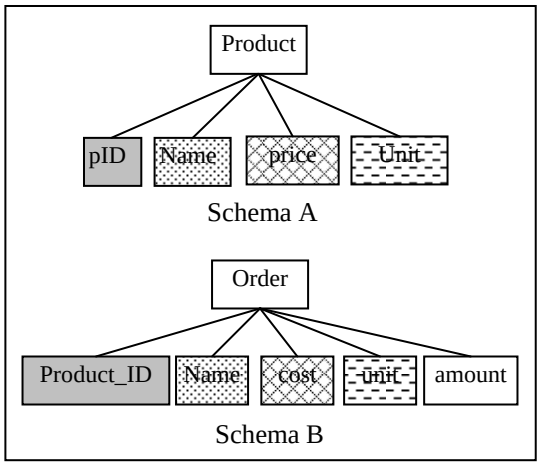

Figure 2 Semantic similarity relation of two schemas

3.3. *Data type compatibility* is the shortest path of each data type node in the data type hierarchy of XML schema presented in Figure 3. We evaluate data type compatibility function by using Equation 3. [7]

$$c(d_1, d_2) = \begin{cases} e^{-\beta(l)} \times \dfrac{e^{\alpha h} - e^{-\alpha h}}{e^{\alpha h} + e^{-\alpha h}} & d_1 \neq d_2 \\ 1, & d_1 = d_2 \end{cases} \quad (3)$$

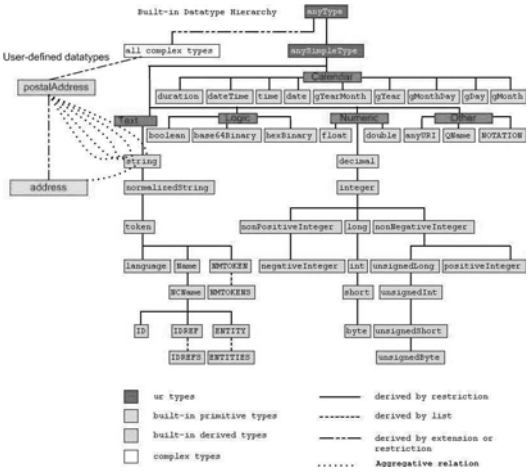

Figure 3. Hierarchy of standard data type.

where d_1 is the data type of n_1 and d_2 is the data type of n_2. $c(d_1,d_2)$ is the compatibility between d_1 and d_2. l is the shortest path length between d_1 and d_1 and h is the depth of subsuming node of d_1 and d_2. Tuning parameter is $\beta = \alpha = 0.3057$

for example, if data type of *price element* is double and data type of *cost element* is float. In Figure 3, l of double and float is 2 and h of double and float is 2 from cost Then $c(price, cost)$ will be

$$c(double, float) = e^{-.3057(2)} \times \frac{e^{.3057(2)} - e^{-.3057(2)}}{e^{.3057(2)} + e^{-.3057(2)}}$$

$$= 0.29$$

3.4 We proposed the *overall similarity* value between the source node n_1 and the target node n_2 in Equation 4 below.

$$Sim(n_1, n_2) = w_1 \times Sesim(n_1, n_2) + w_2 \times Strsim(n1, n2) + w_3 \times c(d_1, d2) \quad (4)$$

where w_1, w_2 and w_3 are similarity weights. $w_1+w_2+w_3$ is default to 1 while user can later modify these parameters. By default, w_1 = 0.5 because of semantic similarity are the most important to consider; w_2 = 0.35 because the structural similarity shows the right or wrong position of an element in the schema tree; w_3 = 0.15 because data type compatibility is the least important. The $sim(n_1,n_2)$ value is between 0 and 1; 0 means that n_1 and n_2 are completely different while 1 means that n_1 are very similar to n_2. If a source XML element matches with multiple elements in the target XML schema, the system will display all possible mapping pairs so that users can choose the most appropriate matching pair.

The example of using Equation 4 is as follows:

sim(price,cost)=0.5(0.94)+0.35(0.075)+0.15(0.29)
= 0.536

4. Implementation

SAXM is implemented in Java. An Advanced users can choose an appropriate semantic similarity formula and can tune similarity weights. A user also has an option to save mapping results which can be used in the next schema matching.

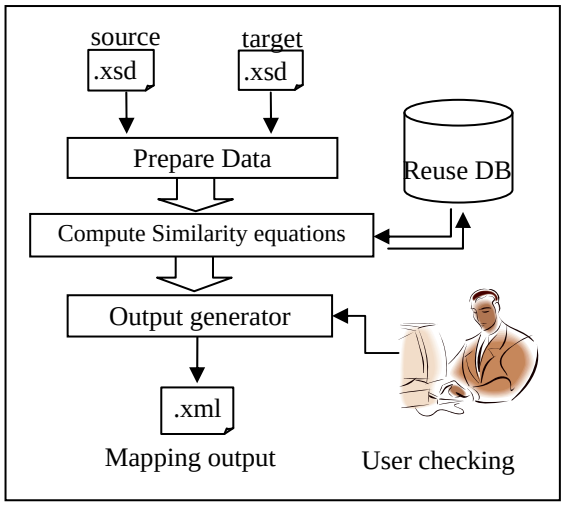

Figure 4. the SAXM architecture

Figure 4 shows the overall architecutre of SAXM. This system requires a source and a target schema for mapping. Data preparation add attribute name “nID” to identify every node in input schemas and create reference id for a node which is refer to other. The reuse database stores the history of semantic mapping results. For each pair of source and target element, automatic mapping is firstly searched for semantic mapping in the reuse database. If this element pair has been mapped previously, its similarity value will be found. Otherwise, the system will compute the similarity between this element pair.

The mapping result is shown in Graphical User Interface (GUI) mode. The input xml schemas are displayed as a tree with the similarity relation line for each element pair as shown in Figure 5. Mapping information section describes the element information and the similarity relation information. A user can check the result add/edit or delete any mapping relations. The mapping output is in xml format, which is generated by the output generator. Mapping produces details of all mapping pairs with its similarity values. The example of mapping output is shown in Figure 6.

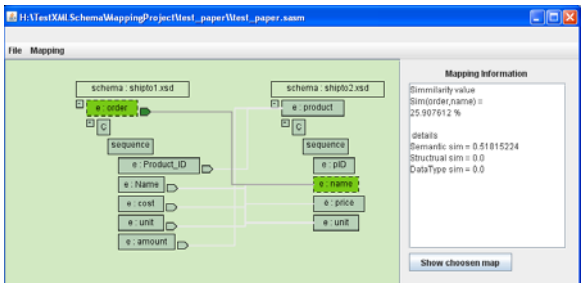

Figure 5. Example of mapping result.

In Figure 6, *source* and *target* elements contain the file locations of input xml schema. The *m* element describes a mapping result. The *sourceNode* element stores the element path of source node and the *targetNode* element stores element path of target node. In the *sim* element, *value* attribute saves the computed overall similarity while semantic, structural and data type attributes correspond to the semantic similarity, structural similarity and data type compatible values respectively.

```
<?xml version="1.0" ?>
<!--Generate by Semi-Automatic schema maping project-->
<output>
  <source>C:\mappingProject\employee.xsd</source>
  <target> C:\mappingProject\employee_1.xsd</target>
  <map>
    <m>
      <sourceNode>Employee/Name/</sourceNode>
      <targetNode>Teacher/teacher_name/</targetNode>
      <sim value="44.872%" semantic="0.617"
        structrual="0.4" datatype="0.0"/>
    </m>
    <m>
      <sourceNode>Employee/Name/</sourceNode>
      <targetNode>Student/student_name/</targetNode>
      <sim value="27.31%" semantic="0.546"
        structrual="0.0"datatype="0.0"/>
    </m>
  </map>
</output>
```

Figure 6. Example of mapping output

5. Experimental Results

To test the efficiency of our proposed method, we used XBenchMatch [8] which provides a standard dataset to test the mapping and allows us to evaluate results by the quality of mapping equations. The tested methods include SAXM (automatic version – no help from a user), COMA [1], and Approximate [2], which were executed in three different scenarios.

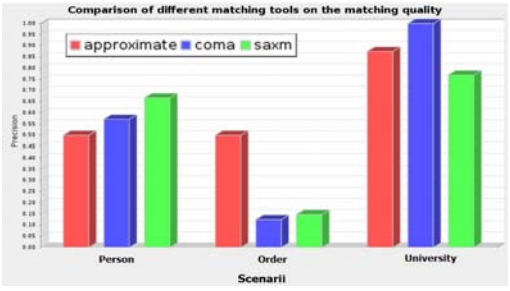

Figure 7. Comparison on Precision Values

Figure 7 shows the comparison result on precision value which indicates the correct mappings (a set of node matches between two schemas) among the mappings produced by the mapping tools. Precision equation is show in Equation 5. When T_{ex} is mappings provided by experts and T_{map} is mappings provide by matcher. SAXM achieved good precision in nearly all scenarios except in the order scenario. This is because the order scenario considers the mapping between two schemas with different sizes on which SAXM have not focused yet.

$$precision = \frac{\left| T_{map} \cap T_{ex} \right|}{\left| T_{map} \right|} \quad (5)$$

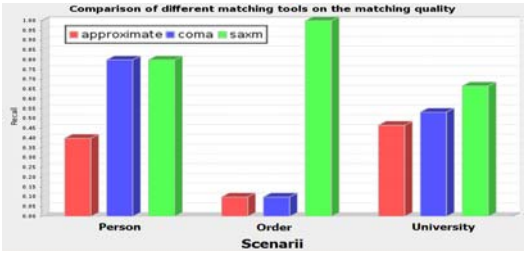

Figure 8. Comparison on recall values

Equation 6 shows recall value calculation, and Figure 8 shows recall values comparison. SAXM has the highest recall value in all scenarios because it considers all structural semantic and data types rendering to match all possible mappings.

$$recall = \frac{\left| T_{map} \cap T_{ex} \right|}{\left| T_{ex} \right|} \quad (6)$$

F- measure indicates the compromise between recall and precision value which is shown in Equation 7. As we see in Figure 9, SAXM has the best value among the compared approaches.

$$F - measure = \frac{2 \times precision \times recall}{precision + recall} \quad (7)$$

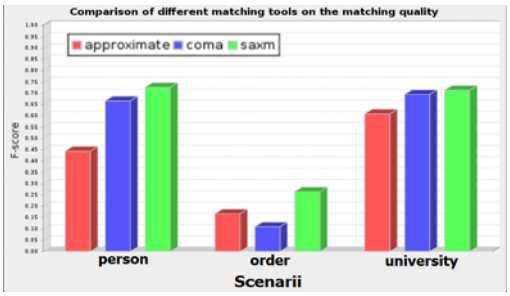

Figure 9. Comparison on F-measure Values

Conclusion.

We have presented SAXM which is a new approach for XML schema mapping with the two new equations to compute the structural similarity and evaluate the overall similarity between XML schema elements. Compared with other approaches, SAXM provides the best mapping result. It also has been used to solve the similarity problem between large schema files in Bioinformatics domain.

Acknowledgement

This project has been funded and supported by the National Center for Genetic Engineering and Biotechnology (BIOTEC). We would like to thank Dr. Cong Yu for his constructive comments to improve this paper.

References

[1] Hong-Hai Do and Erhard Rahm, "COMA: a system for flexible combination of schema matching aproaches", the VLDB Journal, P. 610-621, 2002.

[2] Guangming Xing, "Fast approximate Matching between XML Documents and Schemata", Lecture Notes in Computer Science, P. 425-436, 2006.

[3] WordNet – a Lexical Database for the English Language. Retrieved March 2, 2009, from http://www.cogsci.princeton.edu/wn/

[4] Resnik, "Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language", Volume 11, P. 95-130, 1999.

[5] Dekang Lin, "An Information-Theoretic Definition of similarity", Proceedings of the Fifteen International Conference on Machine Learning, P. 296 – 304, 1998.

[6] Jay J.Jiang,et al., "Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy", Proceedings of ROCLING X, Taiwan, 1997.

[7] Tran Hong-Minh and Smith. D**, "**Hierarchical Approach for Data type Matching in XML Schemas**",** BNCOD apos07. 24th British National Conference on Volume, P.120 – 129, 2007.

[8] Fabien Duchateaue,et. al, "XBenchMatch : a Benchmark for XML Schema Matching", Proceedings of the 33rd International Conference on VLDB,P. 1318-1321,2007