Refinement of Correspondences in EXSMAL for XML Document Transformation

Herzi Khaled
National Institute of Applied Sciences
LIRIS, Bât Blaise.Pascal,
7, Avenue J.Capelle, 69621
Villeurbanne, France
herzi.khaled@yahoo.fr

Aïcha-Nabila Benharkat
National Institute of Applied Sciences
LIRIS, Bât Blaise.Pascal,
7, Avenue J.Capelle, 69621
Villeurbanne, France
nabila.benharkat@insa-lyon.fr

Youssef Amghar
National Institute of Applied Sciences
LIRIS, Bât Blaise.Pascal,
7, Avenue J.Capelle, 69621
Villeurbanne, France
youssef.amghar@insa-lyon.fr

Abstract:

Schema matching is an important prerequisite to the transformation of XML documents with different schemas. In this work, we are interested in the process of matching between data schemes in order to transform documents XML. After explaining related works in the domain, we choose the EXSMAL Algorithm to generate a set of correspondences. Then we try to filter this set in order to obtain 1-1 correspondences. In this purpose, two calculations of similarity are applied: Path similarity and internal similarity. This refinement helps to facilitate the transformation of the documents XML. We also base on a dynamic ontology updated by a user feedback which describes the semantic relation between nodes like IsA, PartOf, Similar, etc. These semantic relations are then expressed in LIMXS data model. The transformation will use operations such as: Connect and Rename for the simple matching, Merge and Split for the complex ones.

1.Introduction

Today, data is generally translated to XML (EXtensible Markup Language) in order to be exchanged between applications belonging to different businesses or organizations.

The XML flexibility has also stimulated XML schema designers to define their own tags, and so, several different schemas are used in the same application domain. For example, companies in an e-business field use different schemas for invoice information. A company should transform its XML documents into ones that conform to the schema of other company for the invoice interaction. An automated transformation of XML documents has therefore become an important issue.

So, the process of matching is unavoidably required. It is focused on the research of the semantic similarities between data schemata. In this work, we are interested especially in the process of matching in order to improve XML documents transformation.

In this domain, semantic schema matching research has extensively grown, and many matching systems have been developed [3], [4], [5], [6], [7], [8],[10],[11].

We have proposed in [3][4] EXSMAL, a similarity algorithm able to calculate simple and complex matchings. Therefore, this paper proposes a schema matching algorithm that refines correspondences in EXSMAL on the basis of two new similarities described in [1] in order to obtain 1-1 correspondences between elements of XML schema. Then, the use of an ontology updated by user feedback constitutes a second part of the refinement benefiting from the various relationships (Is part of…) between the domain concepts that the ontology represents to give a clearer meaning for relationships.

The paper is organized as follows: The section 2 presents related works in the schema Matching and transformation domain. In section 3, we explain how the EXSMAL algorithm generates a set of correspondences based on its two similarity types (basic and structural). In Section 4, we propose a refinement of the latter set by means of new calculations aiming at obtaining 1-1 correspondence between elements of XML schema. In this purpose, we apply: Path similarity between leaf nodes and internal similarity between internal nodes. In Section 5, we explain the ontology role in the framework. In Section 6, the model LIMXS is described in detail in order to find the transformation operations between the given pairs of elements. In Section 7, the conclusion and future works are summarized.

Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA'06)
0-7695-2641-1/06 $20.00 © 2006 IEEE

IEEE COMPUTER SOCIETY

2.Related works

Matching is the task of finding semantic correspondences between elements of two schemas. Various approaches have recently been developed to determine schema matches (semi-)automatically. The underlying approaches can be classified as follows:

Schema based matchers: rely only on schema level information and consider data information (DTD, data base schema…). Schema information includes names, descriptions, relationships, constraints. Cupid [5] derives the similarity of elements based on the similarity of their components hereby emphasizing the name and data type similarities present at the finest level of granularity (leaf level). This algorithm matches elements, names, datatypes, constraints structures and integrates linguistic matching. In [6], SF converts schemas (SQL DDL, RDF, XML) into labelled graphs and uses fix-point computation to determine correspondences of 1:1 local and m: n global cardinality between corresponding nodes of the graphs.

Instance based matchers: consider data instance (i.e., data contents): An element-level matcher computes a mapping between instances of data schema. In [7], the SEMINT system is instance-based . It associates attributes in the two schemas with match signatures. These consist of 15 constraint-based and 5 content-based criteria derived from instance values and normalized to the [0,1] interval, so each attribute is a point in 20-dimensional space. Attributes of one schema are clustered with respect to their Euclidean distance. A neural network is trained on the cluster centers and then is used to obtain the most relevant cluster for each attribute of the second schema. SEMINT is a hybrid element-level matcher. It does not use schema structure, as the latter cannot be mapped into a numerical value. In [8], learning techniques are used to associate elements of input schema and elements of mediated schema. The LSD algorithm generates as output atomic mapping 1-1.

Ontology based matchers: Algorithms in this case compute complex matchings using ontologies with a hierarchical structure. The ontology includes possible concepts of a specific domain and represents the relationships among them in a form of a directed graph. Each concept is associated with a list of words. However, since the ontology is managed by a human expert, it becomes too costly to update the ontology. In [9], a dynamic matching between ontologies is described, it is developed in the context of ontology network in which each node is a peer with a list of concept characterized by properties. This algorithm bases on linguistic affinity and contextual affinity between two concepts of two peers.

3.Matching Algorithm EXSMAL

We have proposed in [3][4] EXSMAL, a similarity algorithm, taking into consideration the specific characteristics of EDI message's branching diagram expressed with XML Schema. Our choice for the XML schema was motivated by its potential to define the structure and semantics of an EDI message. We aim at using this algorithm in larger contexts.

EXSMAL realizes matching of XML schemas taking into account textual description, data types and context description of schema elements. The similarity computation is based on a pair wise element similarity (basic and structural) and a filtering process eliminating the correspondences with a value below the threshold *thraccept* given by the user. The algorithm consists of three steps:

3.1 The Basic similarity

basicSim is computed as the weighted sum of the textual description and data type similarity.

BasicSim(s, t)
=descSim(s,t)*coeff_desc+coeff_type*dataTypeSim(s,t)

Where s and t are respectively elements of is a source and a target schema where *(coeff_desc + coeff_type = 1) and (0≤coeff_desc≤1 and 0≤ coeff_type≤ 1).*

The information retrieval technique is used to compute the textual description and a static matrix defining the XML schema primitive data type affinity is used for the second item.

3.2 The Structural similarity

structSim is computed by using two modules: the structural neighbors computing and the aggregation function **agg**. The structural neighbors of an element e is a quadruplet *<ancestor(e), sibling(e), immediateChild(e), leaf(e)>* in which: ancestor(e) is the set of parent elements from the root until the direct parent of the element e, sibling(e): the set of sibling elements (the elements sharing the same direct parent element) of e, immediateChild(e): the set of direct descendants of the element e, and leaf(e): the set of leaf elements of the sub-tree rooted at e.

The structural similarity value of two elements s and t depends on the similarity value resulting from the comparison of each pair of structural neighbors items *(ancSim(s, t), sibSim(s, t), immCSim(s, t) and leafSim(s, t))*. Therefore, the structural similarity value is computed in function of the ancestor, sibling, immediate child item and leaf item's similarity. The similarity value of each structural neighbors items' pair is computed by using the function *agg(M, thr)* which takes a matrix M and a

IEEE COMPUTER SOCIETY

threshold value thr in [0, 100] as input. It returns the aggregated value of the input matrix M . Finally, the structural similarity value between two elements s and t, structSim(s, t), is computed with the following formula:

*structSim(s,t)=ancSim(s,t)*coeff_anc+sibSim(s,t)*coeff_sib+immCSim(s,t)*coeff_immC+ leafSim(s,t)*coeff_leaf, where 0 ≤ coeff_anc ≤ 1, 0 ≤ coeff_sib ≤ 1, 0 ≤ coeff_immC ≤ 1 , 0 ≤ coeff_leaf ≤ 1, and coeff_anc+coeff_sib+coeff_immC+coeff_leaf=1*

3.3 The Pair-wise element

This similarity is then computed as the weighted sum of the basic similarity value and the structural similarity value. It's proposed as the final similarity value for a pair of elements in our approach.
The pair wise element similarity of s and t is computed by the following formula:

*Similarity(s, t) = basicSim(s, t)*coeff_base + structSim(s, t)*coeff_struct, where 0 ≤ coeff_base ≤ 1, 0 ≤ coeff_struct ≤ 1, and coeff_base + coeff_struct = 1*
Finally, a filtering can be done according to a threshold thraccept

This example fig 1 shows the result of EXSMAL matchings (1-m, 1-1).

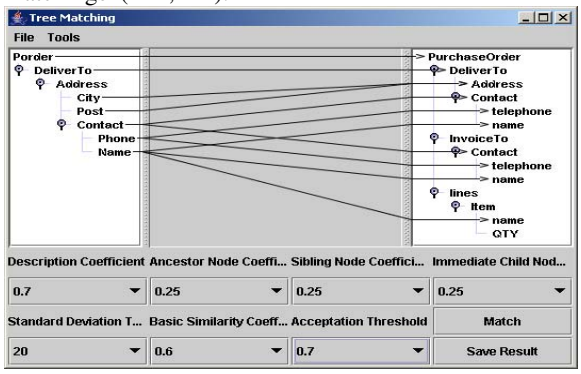

Fig 1 : Result of matching with EXSMAL.

4.Extracting One-To-One Matchings from EXSMAL

Results of the node matching in EXSMAL are many-to-many matching relationships between source and target nodes. In this phase, we extract one-to-one matchings based on a proposed path similarity in leaf nodes and internal similarity in internal nodes.

4.1 The Path similarity

.For all pairs of matchings between leaf nodes, path similarities are computed to extract one-to-one matchings. Path similarity (Ns, Nt) = (number of matched nodes between Ps and Pt)/ $|Ps|+|Pt|$
Where Ps is the path from the root to source node and Pt is the path from the root to target node

4.2 The internal similarity

Internal similarity (Ns, Nt) = (number of matched nodes between LNs and LNt)/ $|LNs|+|LNt|$.
Where LNs is a set of leaf nodes of the subtree rooted at Ns and LNt is a set of leaf nodes of the subtree rooted at Nt . The higher structural similarity means that the corresponding internal nodes include more matched leaf nodes.
After the calculus of path and internal similarities, we can extract the one to one correspondence from the result of matching in EXSMAL as illustrated hereafter.

For both of the calculus, similarities below a threshold (fig 2) ***Thpath*** or ***ThInternal*** are eliminated. If there remains more than one whose similarity is higher than ***Thpath*** or ***ThInternal***, the pair with the highest value is chosen. If the two values are similar, we chose the pair whose basic similarity (*in EXSMAL Algorithm*) is the highest.

4.3 Illustration

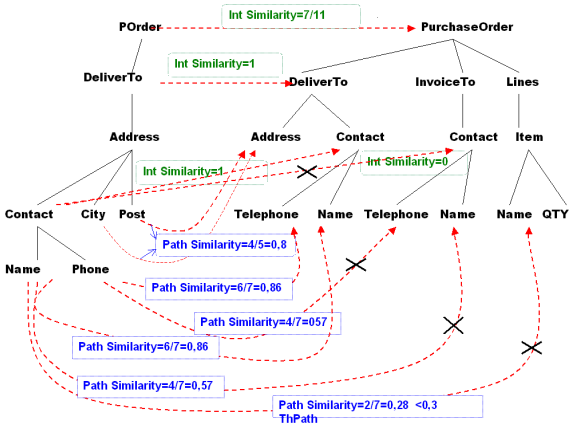

Fig 2 : Path and internal similarities calculus

After computing path similarities for all pairs of leaf nodes matchings, if the path similarity of a preliminary matching is smaller than the value of threshold *ThPath*, we consider the matching as not meaningful and remove it to improve accuracy.

IEEE COMPUTER SOCIETY

Then, matchings between internal nodes on their paths should be found. To find internal node matchings, we compute the internal node similarity. If the similarity is higher than the value of threshold *ThInternal*, they are matched.

Extracting final matchings using the path similarities consists of two steps. First, if a source node has one-to-many matching relationships, the matching with the highest path similarity is chosen, else matching with the highest basic similarity is chosen. The same treatment is done for the internal similarity.

After the calculus of path similarity in leaf nodes and internal similarity in internal nodes we obtain at the end a matching between two schemas XML with one to one matching.

5.Ontology Role

To comprehend better different representations (e.g. supplier product names) it is necessary to map into a common semantic representation. This representation is called taxonomy or other research communities call it ontology [12]. Some studies such as [13], [14] treats the business integration with ontologies (RDFT, DAML+OIL, etc.)

We think that using an ontology provides specific relationships between concepts that can serve as an interpretation in the matching algorithm.

Actually, emerging research in ontology languages provide formal structures that link concepts through semantic relations. The taxonomy relations as hypernyms (i.e., super-class) and hyponyms (i.e., sub-class) are substantially the most popular. But relations as meronyms (a term that refers to a part of another word, i.e. part-of a whole eg: lank,leg) and holonyms (the inverse relation. ie., whole of its parts) are very important ontological features as well because of 1:n and n:1 matchings. Other relations as synonyms(same or very similar meaning), antonyms (term pairs that are opposite in meaning eg: in, out) can help us to refine 1:1 matchings.

Our interest resumed by identifying how ontologies help the semi-automatic translation to resolve ambiguities with the above mentioned relationships.

Ontology operations are selected by user feedback which is formulated according to the model. Feedback = (SourceNodeName, TargetNodeName, IsA | PartOf | Similar | Remove).

Then, we refer to a data model described in [2] Layered Interoperability Model for XML Schemas (LIMXS) for the mappings expression.

Algorithm of the mapping process:

Input: S, T: two XML Schemata

Output: set of triplets <S_i, T_j, V_{sim}>

With (S_i: an element of S, T_j: an element of T and V_{sim}: the similarity value between S_i and T_j

Matching(S, T) {

Convert S and T to tree

For each pair of elements <S_i, T_j>, compute

{

Basic similarity value.

Structural similarity value.

Pair-wise element similarity value.

}

Filtering: eliminate the element pairs having their V_{sim} below an acceptation threshold value, and produce the set $EC_1=\{S_i,T_j\}/i,j=1,n$.

Refinement: refines EC1 with two new calculations (Path and internal similarity and produce the set $EC_2=\{S_i,T_j\}/i,j=1,n$.

Mapping generation: produces the mappings expressed in LIMXS on the basis of the EC2 set and the ontology content.

}

6.Layered Interoperability Model for XML Schemas

To find the transformation operations that may be applied between elements we refine the semantics of the filtered matchings in order to give a sense to these similarity values.

The LIMXS model presented in [1] is a good candidate for this purpose. The model defines the Generalization (specialization) mechanism, the association (semantic relationship between concepts at the same level), the aggregation (relationship composition between two concepts), and Constraints (set of constraints on concepts and relationships). The primitive operations proposed by this model are: Add operation (adds an entity to the target schema : relationship, concept, property), Merge (fusion concatenation 1-n or n-1 matching) Split (performs the inverse transformation), Rename (changes the name of a concept or property) and finally Connect : (1-1 matching binding two entities without transformation).

Finally, the EC2 set and the ontology are used to express in LIMXS operations such as: Connect and Rename for the simple matching (1-1), Merge and Split for the complex matching (1-n, n-1).

Example:

< *Mapping ID = "map1 "*>

<*SourceSchema source = "S1.xsd"*/>

IEEE COMPUTER SOCIETY

```
 <TargetSchema target = "S2.xsd"/>
 <HasMappings OneToOnemapping ID = "map11"/>
</Mapping>
....
< OneToOnemapping ID = "map11" >
 <Conceptmapping>
        <SourceConcept concept = "POrder"/>
        <TargetConcept concept = "PurchaseOrder"/>
        <Transformation>
              <Operation name ="rename"/>
        </Transformation>
 </Conceptmapping>
</OneToOnemapping>
....
< OneTomany mapping ID = "map14" >
 <Conceptmapping>
        <SourceConcept concept = "City"/>
        <SourceConcept concept = "Post"/>
        <TargetConcept concept = "Address"/>
        <Transformation>
              <Operation name ="Merge"/>
        </Transformation>
 </Conceptmapping>
</OneTomanymapping>
```

7.Conclusion

In this paper, we have presented a refinement of a schema matching algorithm EXSMAL. In this purpose, two new calculations of similarity were applied: Path similarity between leaf nodes and internal similarity between internal nodes. These similarities make it possible to refine the calculation of similarity in EXSMAL and to facilitate the transformation of the documents XML. Then, we explained how to use an ontology to provide specific relationships between concepts that can serve to interpret and give the necessary significance for connecting the right operation. The conjunction of the set of filtered correspondences and the significance found in the above ontology is expressed through the data model LIMXS. The latter model is used for the translation process.

The presented work takes part in a project that copes with a complete automated approach documents exchange using schema matching technique. Meanwhile, the front end of this framework has to be extended to deal with further aspects such as taking into account the constraints during the matching process.

8.References

[1] Jun-Seung Lee and Kyong-Ho lee. "XML Schema Matching Based on Incremental Ontology Update. "Dept. of Computer Science, Yonsei University134, Shinchon-dong Sudaemoon-kn, Seoul, 120-749 Korea.

[2] A.Boukottaya, C.Vanoirbeek, F.Paganelli, O.Abou Khaled. "Automating XML Transformations: A conceptual modelling based approach. " Media Research Group, EPFL (Swiss Federal institute of Technology) 1015 Lausanne, Switzerland Electronics and Telecommunications Departement, University of Florence, Italy.

[3] Uddam Chukmol, Rami Rifaieh, Aïcha-Nabila Benharkat: EXSMAL: EDI/XML Semi-Automatic Schema Matching ALgorithm. CEC 2005: 422-425

[4] Rami Rifaieh, Uddam Chukmol, Aïcha-Nabila Benharkat: A Matching Algorithm for Electronic Data Interchange. TES 2005: 34-47

[5] J.Madhavan, P. A. Bernstein and Rahm. "Generic Schema Matching with Cupid". Proceedings of the 27th VLDB Conference, 2001, Rome, Italy. pp.49-58.

[6] S.Melnik, H.Garcia-Molina and E. Rahm. "Similarity Flooding: A versatile Graph Matching approaches". In Proceeding (ICDE), 2002, San Jose, California, USA, 12 pages

[7] Li, W., C. Clifton: SemInt: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Network. Data and Knowledge Engineering, 33 (1), 2000.

[8] A. Doan, P. Domingos and A. Halevy. "Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach". In Proceedings of ACM SIGMOD International Conference on Management of Data, May 2001, Santa BarbaraUSA. pp. 509-520.

[9] S.Castano, A.Ferrara and S.Montelli. "H-MATCH: An Algorithm for Dynamically Matching Ontologies in Peer-based Systems ". In Proceedings of the SWDB 2003 Conference, 2003, Berlin, Germany.pp. 231-250.

[10] Barbara Staudt Lerner. "A Model for Compound Type Changes Encountered in Schema Evolution. "Computer Science Department University of Massachusetts, Amherst June 12, 1996

[11] Mong Li Lee, Liang Huai Yang, Wynne Hsu, Xia Yang. "XClust: Clustering XML Schemas for Effective Integration ".School of Computing, National University of Singapore 3 Science Drive 2, Singapore 117543

[12] M. Stonebraker, J. Hellerstein : "Content integration for e-business". In Proc. Of the ACM SIGMOD, 2001, Santa Barbara, CA, USA

[13] D. Fensel & al.: "Intelligent Information Integration for B2B Electronic Commerce", Kluwer 2002, pp. 50-70

[14] B. Omelayenko: "RDFT: A Mapping Meta-Ontology for Business Integration". In Proc. of the Workshop on Knowledge Transformation for the Semantic Web, (KTSW2002) , Lyon, France (2002), pp. 77-84

IEEE COMPUTER SOCIETY