World of Computer Science and Information Technology Journal (WCSIT)
ISSN: 2221-0741
Vol. 1, No. 5, 193-197, 2011

Improving Semantic Schema Integration

Zahra Sheikhnajdy
Department of Computer Engineering
Islamic Azad University – Science & Research, Khouzestan, Iran
z.sheikhnajdy@khouzestan.srbiau.ac.ir

Mehran Mohsenzadeh
Department of Computer Engineering
Islamic Azad University - Science & Research Center, Tehran, Iran
mohsenzadeh@srbiau.ac.ir

Mashalah Abbasi Dezfuli
Department of Computer Engineering
Islamic Azad University – Science & Research, Khouzestan, Iran
Abbasi_masha@yahoo.com

Abstract—Schema matching is a critical step in many applications, such as data warehouse loading, Online Analytical Process (OLAP), Data mining, semantic web and schema integration. This task is defined for finding the semantic correspondences between elements of two schemas. Recently, schema matching has found considerable interest in both research and practice. In this paper, some approaches for supporting semantic schema matching compared and then we suggest three solutions for improving semantic schema matching problem.

Keywords—schema matching; element level matcher; structural level matcher; semantic ambiguities; step word; word sense disambiguation.

I. INTRODUCTION

Schema matching is the identification of database elements with similar meaning as preparation for subsequent database integration. A schema consists of a set of related elements, such as classes, or XML elements or attributes. The result of a Match operation is a mapping. A mapping consists of a set of mapping elements, each of which indicates that certain elements of schema S1 are related to certain elements of schema S2 [11].

Match is a schema manipulation operation that takes two schemas as input and returns a mapping that identifies corresponding elements in the two schemas [2], [3], [4], [5], [6], [7]. Generic schema matching system architecture shows in Fig. 1.

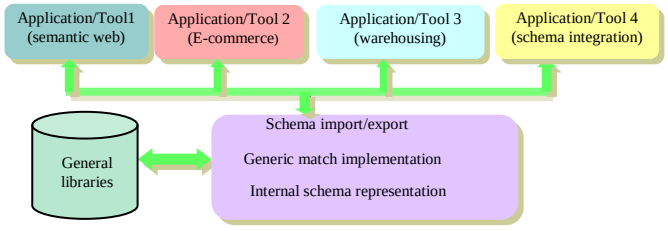

Figure 1. Generic schema matching system architecture

Schema matching is primarily studied as a piece of these other applications. For example, schema integration uses matching to find similar structures in heterogeneous schemas, which are then used as integration points. Data translation uses matching to find simple data transformations. Schema matching use in several application domains such as database application domain, for instance Data integration, Data warehousing, Data mining, E-commerce, Query processing, Peer data management, Model management and so on. Another application domain of schema matching is semantic web like Semantic web services and Xml/html to ontology.

Manually schema matching is a time-consuming, error-prone, and therefore expensive process. Thus, a faster and less labor-intensive integration approach that does this job automated is needed.

Over the past 20 years, different schema matching methods have been proposed and have been shown to be successful to various degrees. However, schema matching is an ongoing research area and the problem is not yet considered to be solved [12]. In this paper, several exiting schema matching introduced and compared with together, then three solutions for improving this approach introduced.

The paper is organized as follows. Section II presents previous work and the basic characteristics of known matchers. Section III introduces our solutions and finally, conclusions and future work are discussed in section IV.

II. EXITING SCHEMA MATCHING APROACHES

In this section we present a classification of the major approaches to schema matching and describe the most popular ones.

A. A Classification Of Schema Matching Approaches

Schema matching is an important subtask of data integration. The core of schema matching is the operator Match which takes two schemas as input and produces a mapping between the elements of these schemas based on semantic correspondences. Implementing this operator requires an

internal representation to which imported schemas are translated and which allows a generic solution. This can be further supported by using dictionaries, thesauri and other kind of domain knowledge useful for identifying correspondences.

Several approaches and tools were developed for supporting schema matching in a semi-automatic way which combines techniques from schema translation, graph transformation, machine learning and knowledge representation [9]. A good survey of these approaches is given in [10]. Here, we briefly summarize this work and discuss it with regard to a rule-based approach.

In [10] the authors classify schema matching approaches into three classes:

• *Individual matchers* compute a mapping using only a single match criterion,

• *Hybrid matchers* support multiple criteria by using a fixed combination of individual matching techniques [11].

• *Composite matchers* combine the results of individual matchers depending on schema characteristics, application domain or even results of previous steps, e.g., by applying techniques from machine learning.

Individual matchers as building blocks for hybrid and composite matchers can be further classified into:

• *Schema vs. instance level*: Schema-level matchers consider only schema information such as structures (data types, classes, attributes) as well as properties of schema elements like name, type etc. In contrast, instance-level matchers consider data contents, too. This allows a more detailed characterization of data, especially in cases with incomplete or unknown schema information.

• *Element vs. structure matching*: Element matchers consider matching between atomic schema elements such as attributes whereas structure-level matchers can deal with combinations of elements, e.g., by comparing sets of attributes of two classes.

• *Language vs. constraints*: Language-based matchers use textual information and linguistic techniques for matching. Examples are equality or similarity of element names as well as a thesauri-based identification of synonyms and hypernyms. A second approach is to consider constraints defined as part of the schema, e.g., data types, cardinalities of relationships or key characteristics.

• *Matching cardinality*: Another kind of characterization is the cardinality of matches. For example, a 1:1 match means that an attribute for one schema is mapped to another attribute of the second schema. A 1: *n* mapping means that a single attribute is mapped to a set of other attributes, e.g., by computing a value from the other values or the extension of one class is computed by combining the instances from several other classes of the second schema.

• *Auxiliary information*: Often external information can be used to support the identification of matches. This can be provided in the form of user input, results from previous steps or by using thesauri, dictionaries, ontologies etc.

Fig. 2 shows classification of schema matching approaches.

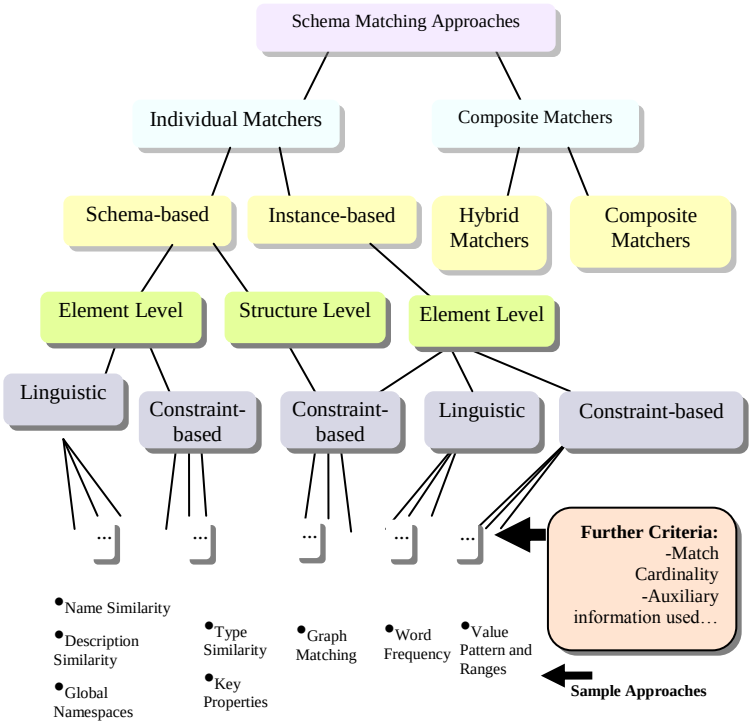

Figure 2. Classification of schema matching approaches

B. *Matching Quality Measures*

To provide a basis for evaluating the quality of an algorithm, the match task has to be performed manually first. The obtained real match result can be used to assess the quality of the result semi-automatically determined by the algorithm. False negatives, A, are matches needed but not semi-automatically identified, while false positives are matches falsely detected by the semi-automatic match operation. True negatives, D, are false matches, which have also been correctly discarded by the automatic match operation. Intuitively, both false negatives and false positives reduce the match quality. Fig. 3 shows this classification.

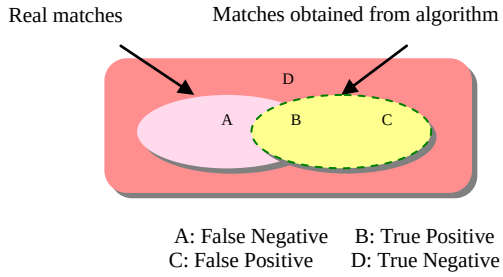

Figure 3. Classification of matches

The following quality measures [5], which we use in our evaluation can be computed as below:

$$\text{Precision} = \frac{|B|}{|B|+|C|} \quad (1)$$

$$\text{Recall} = \frac{|B|}{|A|+|B|} \quad (2)$$

$$\text{Fmeasure} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

$$\text{Overall} = \text{Recall} * \left(1 - \frac{1}{\text{Precision}}\right) \quad (4)$$

C. Prevalent Approaches

In this section we introduce most important implementation of schema matching approaches. This is necessary work for how use of these methods in our implementation. Each of these implementations are shown in Table I.

COMA/COMA++ is a generic, composite matcher with very effective match results [4]. It can process the relational, XML, RDF schemas as well as ontologies. Internally it converts the input schemas as trees for structural matching. For each source element, elements with similarity higher then than threshold are displayed to the user for final selection. The COMA++ supports a number of other features like merging, saving and aggregating match results of two schemas.

Clio consists of a set of Schema Readers, which read a schema and translate it to an internal representation, a Correspondence Engine (CE), which is used to identify matching parts of the schemas or databases, and a Mapping Generator, which generates view definitions to map data in the source schema into data in the target schema. The correspondence engine uses n: m element level matching that have been gained by knowledge or that have been given by the user via a graphical interface.

Cupid is a hybrid schema matcher, combining a name matcher and a structure-based matcher. This tool finds the element matching of a schema, using the similarity of their names and types at the leaf level.

SF uses no external dictionary, but offers several filters for the best matching selection from the result of the structure-based matcher. After the end of the structural matching, the user can choose which matching he wants to keep by using certain filters provided by the system.

Finally, [1], that we refer to it with *-algorithm, is a hybrid Semantic Schema Mapping Algorithm. This algorithm finds mappings based on the hierarchical organization of the elements of a term dictionary (WordNet [8]) and on the reuse of already identified matching. This algorithm uses tokenization techniques for finding semantic matches between multi term entities.

Theses 7 known approaches compared together. The results are shown in Table III [1]. Notice that the set of schemas and their characteristics are illustrated in Table II.

TABLE I. SPECIFICATIONS OF SOME SCHEMA MATCHING APPROACHES

Criteria	COMA++	CUPID	SF	CLIO	*-algorithm	SEMINT
Architecture						
GUI	yes	no	no	yes	Yes	no
Approach	composite	hybrid	hybrid	hybrid	Hybrid	hybrid
Schema Representation						
Schema	XSD, XDR, SQL, OWL	XDR, SQL	SQL, RDF	SQL, XSD	XDR, XSD, OWL, RDF, Relational Databases	SQL
Internal Rep	directed graph	tree	RDF graph	directed graph	Directed Acyclic Graph	attributed-based
Elements	Entities/ relationships/ attributes	Entities/ relationships/ attributes	Entities/ relationships/ attributes	Entities/ relationships/ attributes	Entities/ relationships/	attributes
Result Representation						
Local/global card	1:1/*m*:*n*	1:1/*m*:*n*	1:1/1:1	1:1/1:1	1:1/*n*:1, *m*:*n*/*m*:*n*	*m*:*n*/*m*:*n*
Directionality	unidirectional	source-target	unidirectional	source-target	source-target	unidirectional
Input Information						
Auxiliary	synonyms	synonyms	-	-	synonyms, hyponyms, hypernyms	-
Match Algorithms						
Reuse	previous mappings, synonyms	synonyms	-	-	constraints, synonym Dictionaries	-

TABLE II. CHARACTERISTICS OF THE EVALUATED SCHEMAS [16]

Schema	*Nodes*	*Attributes-Links*	*Schema*	*Nodes*	*Attributes-Links*
Relational *PO2*	4	20	Relational *PO1*	8	37
Ontology *Order1.owl*	33	36	Ontology *Order2.owl*	32	29
XML *CIDX*	7	27	XML *Excel*	12	36
Relational *bibliography*	2	8	Ontology *Bibliographic*	75	749
Relational *PO2*	4	20	XML *Noris_xdr*	11	54
XML *mondial_xsd*	27	93	Ontology *Mondial_owl*	214	93

TABLE III. RESULTS OF THE COMPARISON BETWEEN APPROACHES' EVALUATION IN [1]

	Algorithm	*Coma/ Coma++*	*Cupid*	*SF*	*SemInt [9]*	*LSD [6]*	*XML-OWL [1]*
Schema Types	XML, Relational, OWL, a combination of the above	XML	XML	XML, Relational	Relational	XML	XML, OWL
Matching Cardinality	1:1, 1:n, n:1, n:m, m:n	1:1, n:m	1:1, n:1	1:1, m:n	m:n, n:m	1:1, n:1	1:1, n:1
Mean Presicion	0.86	0.93	0.83	0.84	0.78	0.8	0.6
Mean Recall	0.88	0.89	0.48	0.5	0.86	0.8	0.9
F-Measure	0.86	0.90	0.42	0.65	0.81	0.8	0.72
Mean Overall	0.74	0.82	~0.2	~0.5	0.48	0.6	0.3

The selected schemas vary in size (from 10 up to 824 elements) and kind and system evaluation was not covered different kinds of schemas. In other word this comparisons between homogeneous (i.e., relational vs.

relational) and heterogeneous (i.e., relational vs. ontology) schemas.

In *-algorithm, three out of four measures (Precision, Recall, FMeasure) exceed 0.85, while Overall, which is the most pessimistic measure, is round about 0.75. This algorithm is as good as other approaches and in some cases is even better. The algorithm was compared with other 6 known approaches [16], [4], [5], [7], [17] and only COMA++ and an approach of

XML and ontology matching were able to provide better results in some points. In *-algorithm and COMA++, *Overall* has a great significance. Although the XML-OWL approach has the best *Recall*.

III. SOME SOLUTIONS FOR BETTER QUALITY

As presented above, in most cases, [1] and COMA++ give best quality for semantic schema matching. For this reason, we used *-algorithm [1] as a basis for presenting our suggestions.

A. Using Content

*-algorithm uses only element level matcher for finding matches between entities. This leads to some semantic ambiguities considered as true matches and so increasing false positive matches. If we use structural level matcher too, several homonym's ambiguity disregards and not lead's to false result.

For example suppose that schema1 and schema2 have an entity with the same name as Fig. 4:

Schema 1:

```
<owl:Class rdf:ID="Like">
  <rdfs:subClassOf rdf:resource="#Similar"/>
    <rdfs : subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource=”#ToObject”/>
      </owl:Resctriction>
    </rdfs:subClassOf>
</owl:Class>
```

Schema 2:

```
<owl:Class rdf:ID="Like">
  <rdfs:subClassOf rdf:resource="#Love"/>
    <rdfs : subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource=”#ToPerson”/>
      </owl:Resctriction>
    </rdfs:subClassOf>
</owl:Class>
```

Figure 4. An example of semantic ambiguity

In element level matcher, these entities match with similarity value 1. But these entities have different means; it means extract from their attributes. "Like" in schema 1 refers to similar attribute and another "Like" in schema 2 refers to love. For this example, similarity related to an object while the love attribute related to a person. So these entities are different and need not be matched.

Another hand, perhaps two namesake entities that not refer to unify concept, haven't any property. For example suppose that in Fig. 4, there is no onProperty field exists for both "Like" entities. In this case, similarity value equal to 1 while these entities are different in semantic.

In structural level, several attributes should be consider, such as properties of a class, parent and child's a class, and for properties, domain, range, parent and child's a property.

Solution A suggests to discarding these semantic ambiguities with using structural level matchers. For this reason we must use some attributes and measurement parameters in content of each entity such as properties of each entity, all share properties between two entities and similarity value between parents of entities.

So with considering content of entities, relations and attributes between them, several semantic ambiguities discard from matches and leads to increasing precision and overall parameters.

B. Step Word Omitting Approach

In *-algorithm, because of step words between multi term words, such as "of", "for", "with" and etc, some true matches discard or match with low similarity value. For example "PairOfNodes" and "NodePair" matches with 0.8 match

degree and its reason is preposition "Of". These step words lead to increasing true negative matches. In solution B, we suggest deleting this step words with step word omitting approaches. By this technique, degree match between these entities increase to 1.

C. Word Sense Disambiguation

*-algorithm uses tokenization technique for matching multi term entities. As solution C, we suggest using Word Sense Disambiguation and Context Analysis approach [13], [14], [15] for finding semantic matching between multi terms entities. Thus instead of split a multi term entity, this analysis with context analysis techniques and concept graph and so get ever better quality match.

IV. Conclusion And Future Work

In this paper, three solutions for schema mapping introduced. Implementing each of theses solutions causes to improving semantic schema matching. Each of these solutions suggests focusing to semantically view and so, leads to semantically ever better quality.

In solution A, our focus was to using content of entities such as properties and attributes of them. This caused to decreasing false positive matches.

In solution B, step word omitting solutions caused to consider true negative matches and Solution C suggested using Word Sense Disambiguation and Context Analysis approach instead of tokenization technique. Therefore combination of these solutions, leads to increase measurement parameters such as precision and overall.

Our future work is implementing improvement semantic schema integration with RDF Schema metadata, using hierarchical relation between entities in WordNet dictionary. In this work, all solution that we suggest in this paper, implement and then schema integration runs base on type of matches.

References

[1] Manakanatas D., Plexousakis D., "A Tool for Semi-Automated Semantic Schema Mapping: Design and implementation", International Workshop Data Integration and the Semantic Web, pp. 290-306, June 5-9, 2009.

[2] Shvaiko P., Giunchiglia F., Yatskevich M.," semantic matching with s-match", Springer, 2009.

[3] Partyka J., Khan L., Thuraisingham B., "Semantic Schema Matching Without Shared Instances", IEEE International Conference on Semantic Computing, 2009.

[4] Aumueller D., Do H.H., Massmann S., Rahm E., "Schema and Ontology Matching with COMA++", Proc. ACM SIGMOD international conference on Management of data, pages 906-908, 2005.

[5] Do H. H., Rahm E., "COMA – A System for Flexible Combination of Schema Matching Approach", Proc. VLDB, pages 610-621, 2002.

[6] Li W., Clifton C., "Semantic Integration in Heterogeneous Databases Using Neural Networks". Proc. VLDB, pages 1-12, 1994.

[7] Madhavan J., Bernstein P.A., Rahm E., "Generic Schema Matching with Cupid", Proc. In VLDB : Proceedings of the 27th International Conference on Very Large Data Bases, pages 49-58, San Francisco, CA, USA, 2001.

[8] WordNet a Lexical Database for the English Language, http://wordnet.princeton.edu/

[9] Saake G., Sattler K.U., Conrad S., "Rule-based schema matching for Ontology-based mediators", Elsevier, 2005.

[10] Rahm E., Bernstein P., "A survey of approaches to automatic schema matching", VLDB J. 10 (4), pages 334–350, 2001.

[11] Milo T., Zohar S., "Using schema matching to simplify heterogeneous data translation", in: Int. Conference on Very Large Data Bases (VLDB) 98,, pages 122–133, 1998.

[12] Evermann J., "Theories of Meaning in Schema Matching: A Review", Journal of Database Management, 19(3), pages 55-82, July-September 2008.

[13] Soltanpoor R., Mohsenzadeh M., Mohaqeqi M., "Using concept graph and Naive Bayes to improve the classification of unknown documents", IEEECS, Conference on Information and Software Engineering, India, 2010.

[14] Teymoorian F., Mohsenzadeh M., "English-Persian Text Retrieval Using Concept Graph", IEEE International Conference on Computer Science and Information Technology (IACSIT), Singapore, 2009.

[15] Teymoorian F., Mohsenzadeh M., Seyyedi M., "Using Concept Graph to Increase Bilingual Text Retrieval Precision", IEEE International Conference on Digital Ecosystems and Technologies, Istanbul, Turkey, 2009.

[16] An Y., Borgida A. and Mylopoulos J., "Constructing Complex Semantic Mapping Be-tween XML Data and Ontologies", ISWC, 6-10, 2005.

[17] Yatskevich M., "Preliminary Evaluation of Schema Matching Systems", Technical Report, DIT-03-028, May 2003.