JACET Journal of Advances in Computer Engineering and Technology, 1(1) 2015

An Improved Semantic Schema Matching Approach

Zahra Sheikhnajdy[1], Mehran Mohsenzadeh[2], Mashalah Abbasi Dezfuli[3]

Received (5-9-2011)
Accepted (22-10-2011)

Abstract **- Schema matching is a critical step in many applications, such as data warehouse loading, Online Analytical Process (OLAP), Data mining, semantic web [2] 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, we present a new improved solution for schema matching problem. An improvement hybrid semantic schema matching algorithm which semi automatically finds matching between two data representation schemas is introduced. The algorithm finds mappings based on the hierarchical organization of the elements of a term WordNet dictionary.**

Index Terms **- schema matching, schema integration, ontology, RDF Schema, element level matcher, structural level matcher.**

1- Science and Research Branch, Islamic Azad University, Khouzestan, Iran (z.sheikhnajdy@khouzestan.srbiau.ac.ir)
2-Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran. (mohsenzadeh@srbiau.ac.ir)
3-Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Khouzestan, Iran, (mabbasi@khuzestan.srbiau.ac.ir)

I. INTRODUCTION

SCHEMA matching is the identification of database elements with similar meaning as preparation for subsequent database integration [3], [4], [6], [7] and [10]. 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 [20]. 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 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.

Schema matching is primarily studied as a piece of other applications. For example, schema integration uses matching to find similar structures in heterogeneous schemas. Schema matching is used 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.

Manual 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 [14].

The semantic algorithm introduced in this paper, creates an improvement generic schema matching approach for RDF and OWL schemas. This is an improvement hybrid semantic schema matching algorithm that uses both element and

structural level matchers for finding best matches between entities. Another important component of this work is the WordNet Lexical Database that helps finding matching. This implementation finds matching with different cardinalities (1:1, 1: n and n: 1).

The paper is organized as follows. Section II presents previous work and the basic characteristics of known matchers. Section III introduces our semantic algorithm and describes its operation. Also this section introduces the WordNet Lexical Database [11]. The evaluation results based on quality measures and comparison with other approaches are given in section IV. Conclusions and future work are discussed in section V.

II. RELATED WORK

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

1. A Classification Of Schema Matching Approaches

Schema matching support by using dictionaries, thesauri and other kind of domain knowledge is useful for identifying correspondences. Several approaches and tools were developed for supporting schema matching [12]. A good survey of these approaches is given in [13]. The authors classify schema matching approaches into three classes. Here, we briefly summarize this work.

• *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 [17].

• *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 [5].

Individual matchers are 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.

• *Language vs. constraints:* Language-based matchers use textual information and linguistic techniques for matching. 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: n mapping means that a single attribute is mapped to a set of other attributes.

• *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. 1 shows classification of schema matching approaches.

2. Prevalent Approaches

In this section we introduce most important implementation of schema matching approaches.

COMA/COMA++ is a generic, composite matcher with very effective match results [6] and [7]. It can process the relational, XML, RDF schemas as well as ontologies. The COMA++ supports a number of other features like merging, saving and aggregating match results of two schemas.

Clio [16] 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.

Cupid [10] 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.

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) and on the reuse of already identified matching.

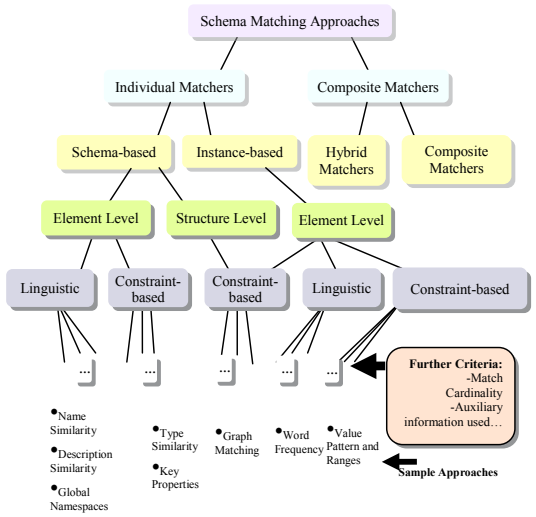

Fig.1. Classification of schema matching approaches

*-algorithm was compared with other 6 known approaches and only COMA++ and an approach of XML and ontology matching were able to provide better results in some points. The results are shown in Table II [1]. Notice that the set of schemas and their characteristics are illustrated in Table I.

TABLE I. CHARACTERISTICS OF THE EVALUATED SCHEMAS [1]

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	740
Relational PO2	8	20	XML Noris_xdr	11	54
XML mondial_xsd	27	93	Ontology Mondial_owl	214	93

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

	Algorithm	Coma/ Coma++	Cupid	SF	SemInt [9]	LSD [6]	XML-OWL
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, n:n	1:1, n:m	1:1, n:1	1:1, n:n	n:m, n:m	1:1, n:1	1:1, n:1
Mean Precision	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

III. AN IMPROVED HYBRI SCHEMA MATCHING APPROACH

The algorithm's inputs are two schemas in OWL, RDF file format. Each schema consists of classes, properties and set of their attributes such as constraints and hierarchical relations between them. The output is an ontology file that shows each match entity pair mid their match measure.

Our goal is to implement an improved hybrid schema matching. For this reason, we studied several algorithms in schema matching domain and then analyzed and compared these algorithms. Finally we conclude the algorithm in [1] that we call it *-algorithm, is good and feasible and in most cases, offers better quality in the result than other approaches. On the other hand, it may be modified and extended in order to get an even better performance. This algorithm uses only element level mapping detection and in some cases it not able detection semantic ambiguity. The structure of our algorithm is shown in Fig. 2.

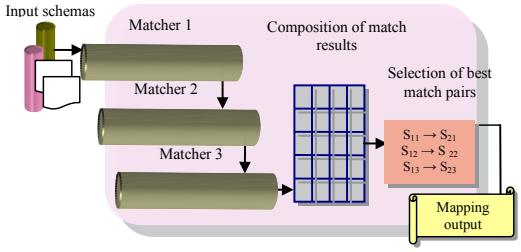

Fig.2. The structure of improved hybrid schema matching approach

1. Operation Of Matcher1

Operation of matcher 1 is based on *-algorithm [1]. Matcher 1 after getting its input, each entity of the first schema compare with all the entities of the second schema (target), and the existence of a matching with an entity of the target schema is examined. If such a matching exists and satisfies the similarity thresholds declared by the user, then this entity is overlooked and the algorithm continues with the next entity of the source schema.

If there is no matching that satisfy the given thresholds, then each entity of both schemas are tokenized based on the delimiters ' ', '.', '_', '-'. For each token of the source entity, the existence of a matching with a token of the target entity is examined using WordNet dictionary. The maximum similarity value for each token combination is kept as long as it is greater than the threshold. In this way, a matrix with similarity values between tokens of the source entity and tokens of the target entity is generated; with dimensions N×M where N, M are the amounts of source and target entities tokens respectively. For each such matrix, the sums of the maximum similarity values of each row (Sum1) and of

each column (Sum2) are calculated. Maximum similarity value of the source and the target entity is determined as the mean value of these two sums:

MaxSim(Schema1_element,Schema2_element) =(Sum1+Sum2)/(M+N) (1)

Input: 2 Schemas, S1, S2, (OWL, RDF or combination of them) in OWL or RDF files format

Operation:
after getting their inputs, each schema parses and all important data extract from here. (Such as classes, properties, constraints and hierarchical relation between them)
While (∃ next S1.entity)
{
Get next S1.entity
If (Each similarity value X of current S1.entity with an S2.entity ∉ [minimum threshold, maximum threshold]) then
{
While (∃ next S2.entity)
{
Get next S2.entity
Tokenize S1.entity based on delimiters ' ', '.', '_', '-', upper case.
Tokenize S2.entity based on delimiters ' ', '.', '_', '-', upper case.
While (∃ S1.tokens)
{
Get next S1.entity and set it to i
While (∃ S2.tokens)
{
Get next S2.token and set it to j
Max Sim (i, j) = 0
If (i == j) then
Max Sim (i, j) = 1
Else
For each (Lexicon based matching i with j)
If (Lexicon based Similarity (i, j) > Max Sim (i,j)) then
Max Sim (i, j) = Lexicon based Similarity (i, j)
If (Max Sim (i, j) > Threshold) then
Store Max Sim (i, j) in Similarity_value_Matrix
}
}
Calculate the sum of maximum similarity values for each row of the Similarity_value_Matrix and set it as Sum1
Calculate the sum of maximum similarity values for each column of the Similarity_value_Matrix and set it as Sum2
MaxSim (S1.entity, S2.entity) = (Sum1 + Sum2) / (amount (S1.tokens) + amount (S2.tokens))

Store [S1.entity, S2.entity, MaxSim (S1.entity, S2.entity)] in List1
}
}
}
Output: a list (List1) containing all entity pair from two schemas with matching greater than threshold (we suggest 0.92)

Fig. 3. Operation of matcher 1

These values are calculated for all the entities of the source schema and then, a list containing all entity pair from two schemas with matching greater than threshold (we suggest 0.92) is returned. Matcher 1 algorithm is shown in Fig. 3.

Matcher 1 uses English dictionary WordNet which is widely used in the information retrieval domain. In next section we introduce this dictionary and how use of WordNet in algorithm.

2. WordNet Dictionary

Information in WordNet is organized in logical groups called "synsets". Every "synset" consists of a list of synonymous words or collocations and pointers which describe the relations between this "synset" and the other ones. A word can exist in more than one "synset".

Pointers represent two kinds of relations: lexical and semantic. Lexical relations hold between semantically related word forms. Semantic relations hold between word meanings. These relations consist of (but they are not limited to) hypernyms (...Is kind of), hyponyms (is kind of...), antonymy, entailment, meronyms (parts of ...), holonyms (...is part of). Nouns and verbs are organized into hierarchies based on the hypernymy and hyponymy relation between "synsets".

The similarity value of two elements is calculated based on the depth that the elements appear in the different Lexicon's hierarchies. This similarity value calculated below:

Distance = ((dept1-dept)/dept1+ (dept2-dept) /dept2)/2 **(2)**

Sim=1-(Distance × Distance) **(3)**

Where dept1 is depth of the first entity, dept2 is depth of the second entity, dept is common parent depth and finally sim is similarity value between entity1 and entity2.

Finally, user select threshold and algorithm ignores the matching with similarity value below this (We suggest 0.92 for threshold and this quantity selected after examine this approach with many real world ontology data sets).

For example suppose that we want to match "FatherLove" entity from schema 1 with "MotherLove" entity from schema 2. Since there is no matching exists between them, each entity tokenizes and so matcher 1 execute for these tokens. The step by step procedure shown in below:

We have a similarity matrix with dimensions 2×2. based on dictionary maximum similarity for "Father" and "Mother" is 0.99, for "Father" and "Love" is 0, for "Love" and "Mother" is 0.22 and finally for "Love" and "Love" is 1. So similarity matrix for these entity pair is $\begin{bmatrix} .9 & 0 \\ .2 & 1.0 \end{bmatrix}$.

And then:

Sim1 = .99 + 1 = 1.99
Sim2 = .99 + 1 = 1.99
MaxSim ("FatherLove" , "MotherLove") = (1.99 + 1.99) / (4) = 0.995

First we explain how computed 0.99 (i=0, j=0) for “Father” and “Mother” tokens. WordNet try to find their depth (dept1 and dept2 respectively). For each token, the hierarchical structure from token to its root represented below:

abstract_entity → object → living_thing → organism → person
relative → ancestor → progenitor → genitor →parent → Father

abstract_entity → object → living_thing → organism → person
relative → ancestor → progenitor → genitor →parent → Mother

Therefore, base on equations 2 and 3:

dept1=10, dept2=10, dept=9
distance= 0.1 and then sim=0.99

And so on, for “Love” and “Mother” tokens:

entity → abstract_entity → abstraction → attribute → state
feeling → emotion → Love

And then base on equation 2 and 3:

dept1=7, dept2=10, dept=1
distance= 0.88 and then sim=0.22

3. Operation Of Matcher2

*-algorithm uses only element level matcher for finding matches between entities. This leads to increase 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>
```

Fig. 4. An example of semantic ambiguity

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.

Our algorithm tries to discard these semantic ambiguities with some measurement parameters. First, properties of each entity should be check. Therefore all properties of each entity and all share properties between two entities should be consider. A property in an entity refers to each property that exists in hierarchical relation of this entity, and a share property in an entity refers to a property that exists in another entity set of correspondences too. The similarity value for properties attributes is:

$$\text{Sim-property}=1-(((x-z)/(x+1) + (y-z)/(y+1))/2 \qquad (4)$$

That x equal to all properties of entity 1, y equal to all properties of entity 2 and z equal to all share properties between entity 1 and entity 2. Note that extra 1 in (x+1) and (y+1), only for avoiding division by zero in some cases (when both or one of entities haven’t any property) and not so effect on Sim-property function amount.

So, all matches that pass from stage1, with Sim-property less than .6, discard. This algorithm is shown in Fig. 5.

In our example x=1, y=1, and z=0, then Sim-property=.5 and this match discard.

Input: output of matcher 1 (a list (List1) containing all entity pair from two schemas with matching greater than threshold (we suggest 0.92)).
Operation:
this matcher uses for decrease false positive matches and cause to several ambiguity disregards.
i is a counter for List1 nodes, each node containing two entity pair and element level match value between them.
create a list (List2) for store result of matcher 2.
While (∃ next List1 [i])
{
  If type of entity pair is Class then
  {
    Compute all properties of entity 1(x), all properties of entity 2 (y), and all share properties between entity 1 and entity 2 (z).

    Sim-property=1-(((x-z)/(x+1)+ (y-z)/(y+1))/2)
  }

  If type of entity pair is Property then
  {
    Compute number of classes that entity1 exists in their as a property (domains of entity 1, x), number of classes that entity 2 exists in their as a property (domains of entity 2, y) and number of classes that both entity1 and entity2 exist in their as two properties (share domains of entity1 and entity2, z).

    Sim-property=1-(((x-z)/(x+1)+ (y-z)/(y+1))/2)
  }

  If (Sim-property >= 0.6)
    Add this node to List2.
}
Output: List2 that disregards several homonyms ambiguity.

Fig. 5. Operation of matcher 2

4. Operation Of Matcher 3

Maybe there are two namesake entities with different concepts which none of them have any property. For example suppose that in Fig. 4,

there is no onProperty field exists for both "Like" entities. In this case, Sim-property can't help to find semantic ambiguity, because x=0, y=0, z=0 (element level match value between "ToObject" and "ToPerson" equal to zero) and then Sim-property=1 and so this node passes successfully from matcher 2, while this entity pair not the same. Thus we have to solve ambiguity with matcher 3.

If two entities is match together, certainly these parents too match together. The similarity value between these parents can be with .2 tolerances. This get with about 100 match entities check. Thus, after running Sim-property, in matcher 3, only nodes with their Sim-parents greater than .7 accept.

Note that in our strategy Sim-parent as like as sim, and only different between them, is the entity pair types in hierarchical relations in schemas. This means that, in sim, two entities compared with together, while in Sim-parrent the following strategy used:

If both entities don't have any parent, obviously Sim-parent can't any help and default set to 1. If one entity has parent and another don't has, Sim-parent executes between this parrent and entity 2. And finally, if both entities have parent, Sim-parent execute between their parents.

For our example, matcher 3 execute for "Similar" and "Love", and then Sim-parent equal to zero. Thus this entity pair discards.

We trust that after running these three matchers, several true matches found and several semantic ambiguities discard from result.

5. Selecting Best Match Pairs

After acting three matchers, may each entity from schema 1, matches with some entities from schema 2. So in this step, best match pairs (i.e. match pairs that it's match degree more high than other) select.

IV. EVALUATION OF ALGORITHM

1. 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. 6 shows this classification.

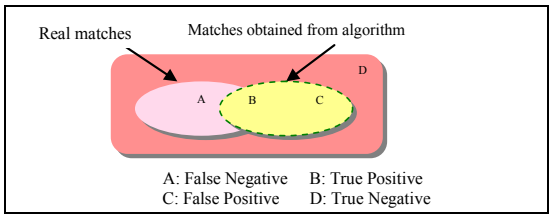

Fig. 6. Classification of matches

The quality measures [7], [8], [9], which we use in our evaluation can be computed as below:

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

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

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

$$\text{Overall} = \text{Recall} * (1- \frac{1}{\text{Pr}\,ecision}) \quad (4)$$

2. Comparision Results

Mean values of quality measures for our algorithm are shown in Fig. 7.

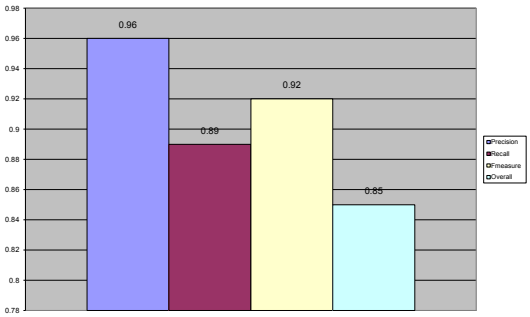

Fig. 7. Mean value of quality measures

Because of inaccessibility implementation of other schema matching algorithms except *-algorithm, we compare our algorithm with *-algorithm with the same real world standard RDF schemas (e.g. Animal schemas, Food schemas, Network schemas, Book schemas and etc). Table III shows result of this comparison.

Note that because of inaccessibility *-algorithm data set, mean value in Table II and Table III was compute with different real

world schemas. So via these two tables, we can compare our solution with other approaches such as COMA++, Cupid, SF and so on.

TABLE III. RESULTS OF THE COMPARISON BETWEEN OUR APPROACH AND *-ALGORITHM EVALUATION

Quality Measures	*-Algorithm	Our Algorithm
Mean Precision	0.88	0.96
Mean Recall	0.89	0.89
Mean Fmeasure	0.88	0.92
Mean Overall	0.77	0.85

Our solution founds several true matches with high match degree. Matcher 2 and matcher 3 cause to omitting several semantic ambiguities, with analyzing content of entities.

But, in some cases, our algorithm is not able to find correct matches with high degree, and it cause to decreasing of recall and overall quality measures.

For example in element level, “PairOfNodes” and “NodePair” matches with 0.8 match degree and its reason is preposition “Of”, that we can solve its with step word omitting approaches. By this technique, degree match between these entities increase to 1.

V. CONCLUSION AND FUTURE WORK

In this paper a new improved solution for schema mapping was presented and an improvement hybrid semantic schema matching algorithm which semi-automatically finds matching between two data representation schemas was introduced. This algorithm tried to find best quality matches and overcome to semantic ambiguity.

In element level, matcher 1 can finds correct matches with high degree such as (child, parent, 0.81), (animal, tiger, 0.89), (male, female, 0.96) and finally (humanBeing, person, 0.79).

Structural level matchers (matcher 2 and matcher 3), are use for remove several semantic ambiguity (with two different meaning), such as like (similarity and love), look (see and search) and etc. therefore these two matchers cause to decrease false positive matches and so increase correctness of matches. We trust that after running these three matchers, several true matches found and several semantic ambiguities discard from result.

In future work, our goal is to use Word Sense Disambiguation and Context Analysis approach [15], [18], [19] for finding semantic matching between multi terms entities. Thus the algorithm instead of split a multi term entity, analysis entity with context analysis techniques and concept graph and so get ever better quality match.

Another goal for our future works is implementing improvement semantic schema integration with RDF Schema metadata, using hierarchical relation between entities in WordNet dictionary.

At the end, this algorithm can be used in different application domain such as semantic integration, query processing system and data warehouses.

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] Dou D., Qin H., LePendu P., "ontograte: towards automatic integration for relational databases and the semantic web through an ontology-based framework", International Journal of Semantic Computing Vol. 4, No. 1 pages 123–151, 2010.

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

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

[5] Doan A., Domingos P., Halevy A., "Reconciling schemas of disparate data sources: a machine-learning approach", SIGMOD conference, pages 09–520, 2001.

[6] 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.

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

[8] Do H. H., Rahm E., Melnik S., "Comparison of Schema Matching Evaluations". Proc. GI – Workshop "Web and Databases", Oct. 2002.

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

[10] [10] 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.

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

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

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

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

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

[16] Chiticariu L., Mauricio A. Andez H., Kolaitis P. G., Popa L., "Semi-Automatic Schema Integration in Clio", ACM – September, pages 23-28, 2007.

[17] 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.

[18] 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.

[19] 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.

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