Inference Rules Generation Algorithm for Ontology Mapping

Zin Thu Thu Myint\textsuperscript{1}, and Kay Khaing Win\textsuperscript{3}

Abstract—The integration of data sources faces with several semantic heterogeneity problems. So, web applications use the ontology as an ideal solution to handle the semantic conflicts in data integration. Ontology is organized with the name and the description of the domain specific entities by using the predicates that represent relationships between these entities. Ontology integration and mapping creates the mapping rules to construct the equivalent between heterogeneous domain ontology concepts when integrating multiple data sources. These mapping rules can be established after an analysis of the similarity in order to detect the correspondences between different ontologies, describing the overlapping domains.

There are many approaches to integrate the existing ontologies based on ontology merging and ontology mapping. However, the integration of these data sources is still trying to become automated. In this paper, we proposed cluster based ontology integration and mapping approach that can build the integrated ontology and mapping rules automatically for same structured ontologies. The solution is provided with the help of SemWeb library for ontology merging in C#.Net.

Keywords—Cluster, Ontology, Ontology integration, and Semantic similarity.

I. INTRODUCTION

The Standard terminology and conceptual hierarchy for a domain description is necessary for the interoperability in the intelligent web. So, ontology is used to define the concepts about the sorts of objects, and relations between objects that are possible in specified domain knowledge and it enables the knowledge sharing [1]. The map of concepts and relationships between ontologies provides the usable resources for semantic data integration. Ontology based data Integration has two general approaches such as Global-as-View (GaV) approach and Local-as-View (LaV) approach [2]. In the GaV approach, every concept in the global ontology is obtained by collecting the concepts over the ontology at each local source. Therefore it has a problem to adjust the evolution of the local source ontologies but the benefit is that the querying strategies are simple. In contrast, the LaV approach allows the changes to local source ontologies without affecting the global schema, but query processing can be complex. In this paper, we will focus on the former approach GAV.

As the architecture of this system follows the framework of Global-as-View (GAV) approach of ontology, it pre-builds the local ontology at each local source to represent the relational structure of database. At the cooperative source, it dynamically builds the global ontology by using the data received from the local site and creates the mapping scheme to map the concepts of global and local ontologies. This mapping scheme contains the rules that are constructed by referring to the semantic similarity between the concepts of global ontology and the view of local ontologies with the help of machine readable dictionary WordNet [3] that is widely used for discovery of semantic relationships between concepts.

When users make queries and submit to the system, the global ontology and mapping schema are used to retrieve the information from the sources.

There are three main processing phases in ontology based data integration system such as ontology creation, ontology mapping and query service [4]. Ontology mapping is initiated after creating the local ontologies to form the global view and querying strategies are more interoperable by utilizing the information obtained from the mapping schemes. It can see that ontology mapping is the core process of ontology based data integration system. In this proposed system, we used the cluster based approach to compact the concepts of global view when integrating the multiple local sources. The former process of this cluster based approach is ontology merging in SemWeb [5] and it generates the ontology as the graph structure. So, this proposed system can easily build the clusters with the data contained in the graph.

Related research includes research on ontology generation, semantic similarity and query processing. An ontology can be generated manually using an authoring tool (e.g., protege) or (semi-)automatically from the knowledge sources (e.g., database schemas) [6]. The Techniques used for ontology mapping, including ontology merging, overlap to get a compact entities of global view with those techniques for semantic matching in ontology integration. This semantic similarity measurement is also applied in the triplet extraction process and in the query generation procedure. Finally, query processing involves triplet extraction from input sentence, SPARQL query generation and after the SPARQL query has been generated, the data included in the ontology can be retrieved and returned to the user.

This paper is organized as follows. In Section II, this paper presents related research of cluster based ontology integration and mapping system. The system design detail is presented in

\textsuperscript{1}Zin Thu Thu Myint is with the University of Technology (Yadanarpon Cyber City), Myanmar (e-mail: zinthumyint@gmail.com).

\textsuperscript{3}Kay Khaing Win is with the University of Technology (Yadanarpon Cyber City), Myanmar. She is now with the Department of Advance Science and Technology (Nay Py iDaw) (e-mail: kkhaingwin@gmail.com).
Section III. It gives in Section IV the description of cluster based ontology integration and mapping. Section V concludes with some remarks.

II. RELATED RESEARCH

Related research considered in the proposed system includes the following processing phases: ontology generation, semantic similarity between concepts and querying strategy.

A. Ontology Generation

Ontology generation phase is prepared to assist the ontology based data integration system with existing ontologies. Here, we use the semi-automatic ontology editor (i.e. protege) to build the domain specific ontology [7]. This domain specific ontology is created by referring to the relational database schema of data sources and the OWL-DL language is used to generate the ontology [8].

For ontology schema, table names in database schema are recognized as the ontology classes, column names in each table are recognized as the data-type properties of each class that define for the corresponding table and the relation between the classes are defined by the object-type properties. The records contained in each table are collected as the individual instance of each class. And then it can build the ontology as database schema by applying the rules described in the following [9].

- If the primary key of more than one relation is the same, they should be merged in one ontological class, and their attributes should be merged.
- If the primary key of one relation is unique for that relation, and not contain the primary key in another relation, then that relation will be considered as separate ontological class.
- If the foreign in a relation Ri is a primary key in another relation Rj, then there is an object property from Ri to Rj (named by its name in R1), and the domain is Ri and range is Rj.
- If the primary key of one relation consists of two other primary keys, then the relation is a primary key between two classes, the class are the two relations denoted by the two primary key.

B. Semantic Similarity

After creating the local ontologies, it is needed to detect the correspondences between different ontologies that are describing the overlapping domain concepts to integrate them as the global ontology and to create the mapping schema [10]. It uses the machine readable dictionary WordNet to define the senses of each entity of ontology. According to the sense such as words or names, it uses the different similarity measurement methods to estimate the closest entities.

WordNet is the machine readable dictionary and it is widely used for confirming the semantic relationships between overlapping domain concepts of ontology. In this system, WordNet is used to discover the form of words (such as noun, adjective, name, etc.) that are contained as the stream of words in the user input query. System finds the information of the input words whether it is unknown or has form. The rules for deciding on each word according to its form are as follows:

- If the word has the unknown form, assumes these words as name stream and assigns null in its form.
- If the word has the form, assigns this form type in its form.

And then, it finds the similarity for the words which have form by applying the simple word similarity measurement method and estimates the similarity of naming streams by applying the Edit Distance similarity measurement method.

1) Edit Distance Similarity Measurement: Here, it is used to compute the similarity between the words which have no meaning (e.g. the name of the person) and containing spaces. The following recurrence relations define the edit distance, \(d(s_1, s_2)\), of two strings \(s_1\) and \(s_2\) [11].

\[
a) \quad d(\varepsilon, \varepsilon) = 0 \quad // \varepsilon \text{ represents an empty string} \\
b) \quad d(s, \varepsilon) = d(\varepsilon, s) = |s| \quad // |s| \text{ is the length of string } s \\
c) \quad d(s_1+c_1, s_2+c_2) = min(\, d(s_1-s_2+c_1, c_2), d(s_1+c_1-s_2+c_2) + 1, \, d(s_1, s_2) + 1, \, d(s_1+c_1, s_2+c_2) + 1) \\
\]

Where \(c_1\) and \(c_2\) are the last characters of \(s_1\) (= \(s_1+c_1\)) and \(s_2\) (= \(s_2+c_2\)) respectively, and \(p(c_1, c_2) = 0\) if \(c_1 = c_2\); \(p(c_1, c_2) = 1\), otherwise. The threshold value for Edit Distance similarity of two concepts is defined as (distance < name-Length).

2) Simple Word Similarity Measurement: It is used to estimate the similarity for each word that has the form. The similarity between the two concepts is calculated by using the following equation.

\[
sim(s_1, s_2) = \sum_{i=1}^{len} \beta_i sim_i(s_1, s_2) \\
\]

Where \(\beta_i (1 \leq i \leq len)\) is an adjustable parameter and \(len\) is the number of characters in each word. Moreover, \(\beta_1 = 1/len\), which reflects the degree contributions to the overall semantic similarity from \(sim_1\) to \(sim_n\). \(sim_i(s_1, s_2)\) is respectively semantic similarity of each character contained in a word. The threshold value for semantic similarity of two concepts is defined as 0.8.

C. Querying Strategy

This strategy is supported for the web query service of ontology based data integration system. For accessing the data on ontology, ontology understanding query such as SPARQL is needed. However, end users enter the unstructured sentence (words, statements, etc.) as an input when they wanted to search the required information on the web. So, it is needed to extract the triplets (i.e. subjects, predicates and objects) from the input query to build the ontology browsing query SPARQL [12]. Extracting the triplets from the user input query is the first step of query service. In this proposed system, triplet extraction process extract the triple patterns included in the user input query with the help of domain specific ontology and machine readable dictionary WordNet [13].

After the triplets had extracted from the input sentence,
these are used to generate the ontology understanding query SPARQL. Finally, the generated SPARQL query browses each entities of global ontology to obtain the required information from the ontology and return the retrieving results to end users [14].

III. SYSTEM OVERVIEW

Using ontology in data integration systems is an ideal solution to handle the semantic conflicts between various data sources [15]. There are two trends to use the ontology in data integration system: one use for translating query or their result and the other uses ontology for the generation of global schema [16]. The system presented in this paper uses both of these two trends for data integration and accessing data on integrated ontology. The system architecture is depicted in fig. 1.

IV. INFERENCE RULES GENERATION

It is needed to consider the two relevant operations such as ontology mapping and ontology integration to obtain the interoperability between the local source ontologies [17, 18, 19, 20]. In proposed system, it uses the staffs’ profile and history records of organizations to build the local source ontologies. In this section, how to integrate the local source ontologies and generate the mapping rules is explained and the algorithm is shown in fig. 2.

In this algorithm, the cooperative source merges the local source ontologies as the graph structure using the SemWeb merge library for ontological processing in C#.Net. This processing step is shown in line 1. The resulting ontology is organized in the graph structure. And then it initializes each entity of merged ontology including data-type properties and of local sources in global view. This cluster based ontology integration and mapping step will be fully explained in the next section. When user submits the query to the system, it extracts the triplets from the user input query and sends these triplets to SPARQL query generator. Finally, it retrieves the required information from the global ontology by confirming the mapping rules.

Here, this system rebuilds the global ontology from merged ontology by changing the prefix name with the global prefix name. If the whole URL is same with the other URL in one cluster, it adds one of them to global view. So, the global view is more compact and it can get the simple querying strategy to browse the global view.

Algorithm OntologyIntegration (O1, O2)

Input: Consistent ontologies O1 to On
Output: Integrated ontology Oint
1. Compute O = O1 U O2
2. Initialize each cluster c with an e ∈ C U P U I
3. Merge clusters containing equivalent entities based on Class-Type URL
4. Build initial inference rules upon the entities contained in the same cluster
5. Repeat
6. Find relation(object-type-properties) of separate class to its Domain/Range class
7. inference rule => \{Z, q, X\}, \{X, owlsameAs, X\}\n8. If it has no relation, then Find super/sub class of separate class
9. inference rule => \{Z, q, X\}, \{X, owlsameAs, X\} and \{Z, rdfs:subClassOf, X\}\n10. inference rule => \{Z, q, X\}, \{X, rdfs:subClassOf, X\}\n11. mappingRules <= mappingRules U inference rule
12. Until no more separate class

In this system, it firstly creates the source ontology at each local site. And then it starts to build the global ontology by collecting the data obtained from the local ontologies. It simply merges the local ontologies and clusters the same entities of local ontologies. According to the resulted clusters, it generates the mapping rules to match the semantic conflicts.
individual instances of this entity in each cluster. It is shown in line 2. The step of merging the clusters that contain equivalent entities based on the class-type URL is written in line 3. Here, this proposed system takes the additional processing stage to get the compact global view by changing the prefix name with the global prefix name and using one of the whole overlapping domain ontology concepts in building global view. And then, the initial inference rules are built at line 4 by considering the entities contained in the same cluster. The rules generation contained in this processing step is illustrated in fig. 3 and fig. 4.

Fig. 3 shows the rule generation for the same classes in one cluster. It uses the sameAs property of owl language to refer to these classes are same. And then it continues to define the relations (object-type properties) mapping of these same class. It is shown in the following figure fig. 4.

\[
\frac{(X, Y, owl:sameAs, X')}{(X, owl:sameAs, X')}
\]

**Inference rule for same cluster**

\[
\frac{(Y, owl:sameAs, Y)}{(Y, owl:sameAs, Y')}
\]

**Inference rule for same cluster**

If the clusters which contain the separate classes are still remaining, it needs to find the relations to organized classes obtained from the previous steps. These relations may be object type properties or super-class properties or sub-class properties. These steps are shown at line 5 to line 12 in algorithm.

If the remaining class has the relation to the organized classes, it generates the inference mapping rule as in equation 2. The relation can be detected by considering the entities contained in the same cluster.

\[
\frac{(Z, q, X), (X, owl:sameAs, X')}{(Z, q, X'), (Z, rdf:subClassOf, X')}
\]

**Inference rule for same cluster**

\[
\frac{(Z, q, X), (X, owl:sameAs, X')}{(Z, q, X), (Z, rdf:subClassOf, X')}
\]

If the remaining class has no relation of object-type properties with the organized classes, it is needed to fine the sub-class or super-class of this class. These steps are also described in line 6 and 7 in algorithm.

\[
\frac{(Z, q, X), (X, owl:sameAs, X')}{(Z, q, X')}
\]

**Inference rule for same cluster**

\[
\frac{(Z, q, X), (X, owl:sameAs, X')}{(Z, q, X), (Z, rdf:subClassOf, X')}
\]

If the remaining class has no relation of object-type properties with the organized classes, it is needed to fine the sub-class or super-class of this class. These steps are also described in line 8 and 9 in algorithm. And it generates the inference mapping rule for sub-class as in equation 3.

\[
\frac{(Z, q, X), (X, owl:sameAs, X')}{(Z, q, X', Z, rdf:subClassOf, X')}
\]

**Inference rule for same cluster**

\[
\frac{(Z, q, X), (X, owl:sameAs, X')}{(Z, q, X), (Z, rdf:subClassOf, X')}
\]

\[
\frac{(Z, q, X), (X, owl:sameAs, X')}{(Z, q, X), (Z, rdf:subClassOf, X')}
\]

The generated inference rule for object-type-property between these two classes is:

\[
\frac{(X, Y, owl:sameAs, X'), (Y, owl:sameAs, Y')}{(P, owl:sameAs, P')}
\]

**Inference rule for same cluster**

The inference mapping rule for super-class is generated as in equation 4.

\[
\frac{(Z, q, X), (X, owl:sameAs, X')}{(Z, q, X), (Z, rdf:superClassOf, X')}
\]

\[
\frac{(Z, q, X), (X, rdf:superClassOf, X')}{(Z, q, X), (Z, rdf:superClassOf, X')}
\]

Finally, these mapping rules are union and saved as the mapping table. This process is continued until no more separate class exists. The sample of resulting mapping table is shown in table 1.

**TABLE I**

<table>
<thead>
<tr>
<th>Mapping</th>
<th>Correspondences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Target</td>
</tr>
<tr>
<td>Map 1</td>
<td>Department(x)</td>
</tr>
<tr>
<td>Map 2</td>
<td>position(x)</td>
</tr>
</tbody>
</table>

This mapping table is continually used in query generation process to map the similar concepts of the global view and return all possible results to the users. the global view.

**V. CONCLUSION**

This paper discussed about the cluster based ontology integration and mapping approach. It also described the overview of the whole proposed system and the related research contained in this system. As the related research, it gave the necessary information about ontology generation.
from relation database structure, semantic similarity for discovering the correspondences between the concepts that can help the local source ontology mapping and query services, and useful query strategy. The aim of this paper is to fully present the cluster based ontology integration and mapping approach that can help the ontology based data integration system to be more compact and the querying strategy to be easier. This approach can be used in the integration of same structured ontologies. Although building the global view takes rich processing time, it can reduce the query processing time by changing the prefix name with global prefix name and removing the overlapping domain ontology concepts. So, this approach can give the great help in building the compact global view.

ACKNOWLEDGMENT

I am very grateful to Dr. Aung Win and Dr. Soe Soe Khine for fruitful discussions during the preparation of this paper and also specially thank to Rector, Professors and colleagues from Technology University (Yadanarpon Cyber City), Myanmar.

REFERENCES


[3] YANG Che-Yu, "Word sense disambiguation using semantic relatedness measurement", Journal of Zhejiang University SCIENCE A ISSN 1009-3095 (Print); ISSN 1862-1775 (Online), 2006 7(10).


[15] Deying Dou, Paea LePendu, Shiooong Kim and Peishen Qi, "Integrating Databases into the Semantic Web through an Ontology-based Framework", Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW’06) 0-7695-2571-7/06 © 2006 IEEE.


