Research on Ontology Matching Approach in Semantic Web

Rubo Zhang, Ying Wang, Jing Wang
College of Computer Science and Technology
Harbin Engineering University, Harbin, 150001, China
yingwang@hrbeu.edu.cn

Abstract

The major purpose of applying ontology is to share and reuse knowledge. However, a large amount of heterogeneous ontologies are constructed in Semantic Web, because there is not a common criterion for building ontology. Under the background, an ontology matching approach is put forward. It converts the ontology matching to the problem of building RDF (Resource Description Framework) graph matching tree. Moreover, the approach presents a structural similarity measure based on the entities of nodes from the matching tree, compensating the inadequacies of the linguistic similarity measure. Our implementation and experimental results are given to demonstrate the effectiveness of the matching approach.

1. Introduction

Ontology matching is the operation that produces a set of semantic correspondence between some entities of one ontology and some entities of the other. It plays a central role in many application domains, such as Semantic Web, data warehouses, e-commerce, query mediation, etc. Many researches have been taken to pursue good algorithms and tools for (semi-)automatic ontology matching. For example, S-Match [1], GLUE [2], COMA [3] are very famous systems.

The work presented in this paper adopts two ways to compute similarity of the ontologies. As ontologies and knowledge-representation languages evolve, the structure-based similarity measures are required to help linguistic similarity measures to improve the precision of matching. The approach can be outlined in the following points:

- Compute linguistic similarity considering both morphological and semantic of the ontology entities.
- Covert the ontology matching to RDF graph matching, and use matching tree to express the matching state in order to compute the structural similarity.
- Decompose the RDF graph to a set of statement, on this basis, establish the index structure for matching.

The rest of this paper is organized as follows. Section 2 presents an approach to compute linguistic similarity. Section 3 outlines the process how to obtain structural similarity by matching the RDF graphs, and discusses the evaluation result in Section 4. Finally, Section 5 concludes and discusses some future works.

2. Linguistic Similarity

The most intuitive and basic method to discover the matching may be that of exploiting the similarity based on the linguistic information. Generally, linguistic similarity between two entities relies on both morphological and semantic of the word. This paper employs Edit Distance to compute the morphological similarity. Edit Distance is proposed scale the distance between two strings (later extended to the statement). It applies a Dynamic Programming algorithm to calculate the minimum operation number of insertions, deletions, and substitutions required to transform one string into the other. The function that applies Edit Distance to compute similarity is shown as follows:

$$sim_{dis\tan ce}(e_{i1}, e_{j2}) = 1 - \frac{dis\tan ce(e_{i1}, e_{j2})}{\max(|e_{i1}|, |e_{j2}|)}$$

where $sim_{dis\tan ce}(e_{i1}, e_{j2})$ denotes the Edit Distance similarity between entities e_{i1} and e_{j2} of the two ontologies, and $dis\tan ce(e_{i1}, e_{j2})$ is the Edit Distance of two strings.

The Edit Distance similarity ignores a problem: two entities with similar meaning might be absolutely differently spelled. Therefore the semantics of the entity should be considered. It is common to use WordNet as external resources to compute similarity. Unlike other traditional lexicon, WordNet organizes the word according as semantics not morphology. WordNet is a semantic network of word senses, in which each node is a synset. A synset contains words with the same sense and a word can occur in different synsets indicating that the

978-0-7695-3112-0/08 $25.00 © 2008 IEEE
DOI 10.1109/ICICSE.2008.51

word has multiple senses. There are many methods to use WordNet, however, the most simple is to compute the path connected two synset. If the path is short, the similarity is high, and vice versa. Lin et al. define the similarity between two senses in WordNet [4] as:

$$sim_{wordnet}(e_{i1}, e_{j2}) = \frac{2 \times \log(p(s))}{\log(p(s_1)) + \log(p(s_2))}$$

where $sim_{wordnet}(e_{i1}, e_{j2})$ denotes the WordNet similarity, $p(s) = count(s)/total$ is the probability of a randomly selected word occurring in the synset s or any sub synsets of it, and $total$ is the number of word in WordNet. In addition, $e_{i1} \in s_1, e_{j2} \in s_2$ represent e_{i1} and e_{j2} in synsets s_1 and s_2, respectively. The synset s is the common hypernym of s_1 and s_2 in WordNet.

So the linguistic similarity by combining the above two methods can be defined as:

$$sim_{element}(e_{i1}, e_{j2}) = \alpha \cdot sim_{distance}(e_{i1}, e_{j2}) + \beta \cdot sim_{wordnet}(e_{i1}, e_{j2})$$ w

here, $\partial, \beta \in [0,1]$, $\partial + \beta = 1$.

Compared with other methods, this approach which combines the Edit Distance and WordNet technologies to compute similarity, not only performs well when the entities names are completely or partially same, but also works effectively when the entities are completely different in name but have some semantic links. The reason is that it can deal with both the morphological and semantic of entities.

3. Structural Similarity

Linguistic similarity measure considers only the information on the labels of the entities; meanwhile structural similarity provides the potential semantic of ontology structure. RDF model, a foundation of Semantic Web, has the nature of graph structure. Web ontology can be mapped to an RDF graph [5]. Thus, the ontology matching can be converted to RDF graph matching. The matching state between two graphs can be shown in the form of tree, called the matching tree, while the matching process will be expressed as the creation of the matching tree. Then adopt the matching tree to compute structural similarity between ontology entities with semantic correspondence.

3.1. RDF graph

The underlying structure of any expression in RDF is a collection of triples, as a statement, each consisting of a subject, a predicate and an object. A set of such triples is called an RDF graph (a directed labeled graph), in which each triple represents as a node-arc-node link. Additionally, the nodes come in three varieties: URI reference, blank nodes, and literals, and the predicate are a property type of the resource, such as an attribute, a relationship, or a characteristic.

Definition 1. The RDF graph of the ontologies to be matched can be denoted as a triple: $G = (N, A, n_0)$, where

(1) N is the set of the nodes which is subject or object.

(2) A is the set of the arcs. The arcs are directional and labeled with the RDF predicates.

(3) $n_0 \in N$ is the home node of the RDF graph, which describe the global information of the document, such as document type, time, etc. Note that, there exists path from home node to every other node.

(4) Every arc of the graph is a statement. Furthermore, the arc is directional and labeled with the RDF predicate starting from the subject node and terminating at the object node.

3.2. Matching Process

Given two ontologies as input, by applying the matching approach, the matching pairs will be generated. The matching process of two RDF graphs is outlined as follows:

Step 1. Convert the two ontologies to RDF graphs G_1 and G_2. Extract the direct information from every entity of the ontologies before matching.

Step 2. Parse the RDF graphs. Beginning from the home vertex in the graph, traverse every arc by the Breadth First Search algorithm, and make sure no loop existed in the path from the home node to any other. Every arrived arc generates one statement, while subject is the starting node, object is the ending node, and predicate is the linking arc. Then put the set of statement according to the RDF graph into an array called the statement-array. The statements in the array are to be ordered by dictionary sort for convenient searching, and each one has a waiting–match list to save the other statement related to this. Initially, waiting-match every list only contains the statement of home node.

Step 3. Create matching tree. The matching tree is created by the rule of that: if the subjects of two statements are similar, by the same predicate, then the objects are also similar. Matching begins from the home nodes of every RDF graphs, and establishes the node of matching tree gradually by searching of two statement-arrays. When finishing the search, one or multiple matching tree(s) will be produced.

Step 4. Calculate the structural similarity by the following formula:

$$Sim_{structure}(e_{i1}, e_{j2}) = w_1 Sim_{element}(e_{i1}, e_{j2}) + w_2 Sim_{element}(F(e_{i1}, e_{j2})) + w_3 \frac{\sum_{i=1}^{n} Sim_{element}(S(e_{i1}, e_{j2}))}{n}$$

where, $w_1, w_{2,}, w_3 \in [0,1]$ and $w_1 + w_2 + w_3 = 1$.

The structural similarity between entities in one node of matching tree is related to their father-node and son-node. So denote $Sim_{element}(F(e_{i1}, e_{j2}))$ and $Sim_{element}(S(e_{i1}, e_{j2}))$ as the similarity of their father-node and son-node, respectively. Because they may have more than one son-node, it is ought to divide the total son-nodes similarity by the number n. Then assign the weights w_1, w_2 and w_3 to the linguistic similarity, father-node similarity and son-node similarity, and obtain the structural similarity.

Step 5. Output matching pairs. After the similarity calculation, it is time to select the entities to be match pairs by examining whether their similarity is higher than the threshold δ.

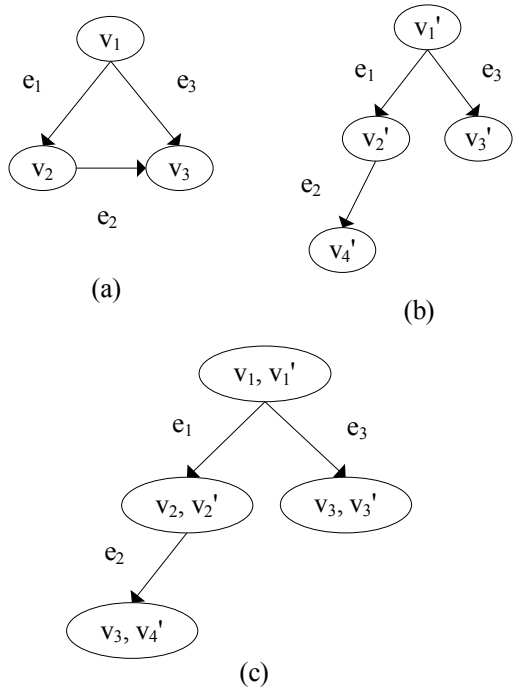

Figure 1. **The illustration of matching process**

Example 1 Let us consider the two simple RDF graphs displayed in Figure 1.a and Figure 1.b. To match these two RDF graphs, first establish two statement-arrays of them. Supposing the home node v_1 and v_1' of each graph is similar, and take the pair (v_1, v_1') as the root node of matching tree. Searching the statements of node v_1 and v_1' as subject in their statement-arrays, if they have the same predicate, the object of the two statements are similar. As a result, the node (v_2, v_2') will be created to be the son-node of (v_1, v_1') with predicate e_1 as arc. Then put the statements of v_2 and v_2' into the waiting-matching list. In the same way, the node (v_3, v_3') and (v_3, v_4') are created, then build the matching tree in figure 1.c.

4. Experiment and result analysis

The experiment environment is Redhat Linux 9.0/P4 3.0/1G/160G, and the adopted software tools are JDK 1.4.2, Jena 2.2, WordNet 2.1, eclipse 3.0, Protege 3.0.

In order to evaluate the proposed method, three datasets [6] are use to take tests separately. The statistical data of these datasets is shown in table 1. The ontologies in Course Catalog I and Course Catalog II describes course system in Cornell and Washington University, respectively. The former is a simple edition, including 34-39 concept nodes that there are some similarities among them. The latter is an extension edition, including 166-176 concept nodes that there are few similarities among them. What's more, the ontology in Company Profiles depicts the business information in Standard.com and Yahoo.com separately, and mentioned in table 1, the number of concept and instance in Standard.com and Yahoo.com is very large, and there exist many mapping relationships among them.

Table 1. **The statistical data of testing datasets**

Dataset	Ontology	Concept	Instance	Mapping
Course Catalog I	Cornell	34	1526	34
	Washington	39	1912	37
Course Catalog II	Cornell	176	4360	54
	Washington	166	6957	50
Company Profiles	Standard.com	333	13634	236
	Yahoo.com	115	9504	104

Generally speaking, it is often to adopt two evaluation indices Recall and Precision in the area of Information Retrieval to assess the matching algorithm, and these are denoted as follows [7]:

$$\text{Recall:} \quad R(A,R) = \frac{|R \cap A|}{|R|}$$

$$\text{Precision:} \quad P(A,R) = \frac{|R \cap A|}{|A|}$$

Where R and A denote the desired result and the actual result separately. The Recall is a proportion of the right matching number and the desired matching number. The Precision is a proportion of the right matching number and the actual matching number.

The following Figure 2 and Figure 3 show the results of the approach applying on the dataset. For short, use A to denote the matching of ontology Cornell University to Washington University; B to denote Washington University matching to Cornell University in Course Catalog I; C and D denoted the matching of ontologies in

Course Catalog II; and E, F represent the matching of ontologies of Standard.com and Yahoo.com.

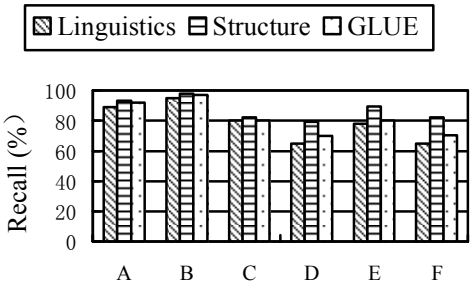

Figure 2. **Recall**

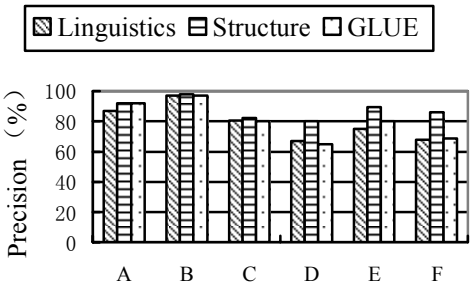

Figure 3. **Precision**

Experiments are taken on the datasets, such as single linguistic similarity measure, structural similarity combining linguistic similarity computation, and then this approach is compared with GLUE system. Through the experiment and result analysis, it can be found out that the structural matching result in Recall and Precision has advantage over the linguistic matching result, and this approach works better than GLUE system. The reason is that the ontology architecture contains much potential semantic information that influences the similarity among the ontologies greatly, which should be taken into account. In fact, the structural matching approach adopted in this approach is on the basis of the linguistic matching; and considers the ontology intrinsic architecture adequately. By this way, the proposed method can match the ontology more exactly.

5. Conclusions and Future works

This paper have presented here an approach to structure-based semantic similarity measurement that can be directly applied to ontologies modeled as RDF graphs, compensating the inadequacies of the linguistic similarity. The work is based on the intuitive idea that the similarity of two entities can be define in terms of how these two entities relate to the other entities. So it use matching tree to express the state of matching in order to obtain the structural similarity according to the entities position. The good results achieved in the tests have proved the value of the approach in situations in which structural similarities exist. The future works are to improve the approach in some aspects, e.g. efficiency problem and integration with other technique.

6. References

[1] S. Melnik, H. G. Molina, E. Rahm, Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching, *In Proceedings of the 18th International Conference on Data Engineering,* San Jose, CA, 2002, pp.117-128.

[2] F. Giunchiglia, M. Yatskevich, E. Giuchilglia, Efficient Semantic Matching, *In Proceedings of ESWC, Heraklion*, Greece, 2005, pp.272-289.

[3] A. Doan, J. Madhavan, P. Domingos, A. Halevy, Ontology Matching: A Machine Learning Approach, *Handbook on Ontologies in Information Systems*, Springer-Velag,2003, pp.397-416.

[4] P. Pantel, D. Lin, Discovering Word Senses from Text, *In Proceedings of the 2002 ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, 2002, pp.613-619.

[5] P. F. Patel-Schneider, P. Hayes, I. Horrocks (eds.), OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation 10 February 2004, Latest version is available at http://www.w3.org/TR/owl-semantics.

[6] http://anhai.cs.uiuc.edu/archive/summary.type.html.

[7] J. Euzenat, Semantic Precision and Recall for Ontology Alignment Evaluation, *In Proceedings of IJCAI*, 2007, pp.350.