Journal of Southeast University (English Edition) Vol. 22, No. 3, pp. 348 – 352 Sept. 2006 ISSN 1003—7985

New automated ontology mapping algorithm

Li Xuanru He Jieyue

(School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: A new algorithm for automated ontology mapping based on linguistic similarity and structure similarity is presented. First, the concept of WordNet is turned into a vector, then the similarity of two entities is calculated according to the cosine of the angle between the corresponding vectors. Secondly, based on the linguistic similarity, a weighted function and a sigmoid function can be used to combine the linguistic similarity and structure similarity to compute the similarity of an ontology. Experimental results show that the matching ratio can reach 63% to 70% and it can effectively accomplish the mapping between ontologies.

Key words: ontology; ontology mapping; semantic web

Ontology plays a crucial role in the emerging semantic web. It is necessary to establish inter-operation among semantic web application using different ontologies by ontology mapping. Many researchers agree that one of the major bottlenecks in semantic integration is how to achieve ontology mapping. In the past, ontology mapping was performed manually, but the manual mapping is tedious work. Hence, the task of finding mappings (semi-) automatically has been an active area of research in the ontology community. Currently, the major approach to (semi-) automatically ontology mapping is based on heuristics or matching learning techniques that use various characteristics of ontologies. FCA-Merge[1] performs ontology mapping by using techniques from formal concept analysis. GLUE[2] uses multiple learners and a probabilistic model to do ontology mapping. These approaches need a large number of examples for training, which is, unfortunately, currently difficult to obtain. The tools PROMPT and Anchor PROMPPT[3] use labels and the structure of ontologies for ontology mapping. Cupid[4] implements a hybrid matching algorithm comprising linguistic and structural schema matching techniques. In this paper, a new automated ontology mapping algorithm, which combines the linguistic similarity with the structure similarity of entities of an ontology via a sigmoid function and a weighted function, is described. This algorithm improves the calculating module of structure similarity and experiments show encouraging results, yielding 63% to 70% of the nodes on several real-world domains.

1 Definitions

In this section, we will give the definitions concerning ontology, similarity and ontology mapping.

1.1 Ontology

There is no common formal definition of ontology. In 1995, Studer et al. [5] gave the most popular definition of ontology: "an ontology is an explicit, partial account of a conceptualization/the intended model of a logical language." An ontology can be expressed by the function: $OT = \langle ID, C, R, F, T, I \rangle$, where ID is the label of the ontology; C is the set of concepts; R is the relation between concepts, R: $C_1 \times C_2 \ldots \times C_N$; F is a function relation, F: $C_1 \times C_2 \ldots \times C_{n-1} \rightarrow C_n$; T is true assertion; I is the instance of concept.

1.2 Ontology mapping

Now we describe our understanding of the term "ontology mapping". Su[6] gives the definition of the ontology mapping: "Given two ontologies A and B, mapping one ontology with another means that for each concept (node) in ontology A, we try to find a corresponding concept (node), which has the same or similar semantics, in ontology Band vice versa." According to this definition, an ontology mapping function can be defined as follows:

- $\lambda_{Map}: O_1 \rightarrow O_2$;
- $\lambda_{Map}(e_1) \rightarrow e_2$: If sim $(e_1, e_2) > s$ with sim (e_1, e_2) being the similarity coefficients between e_1 and e_2, s being

Received 2006-04-12.

Foundation item: The High Technology Research and Development Program of Jiangsu Province (No. BG2004034).

Biographies: Li Xuanru(1980—), female, graduate; He Jieyue(corresponding author), female, professor, julie_hjy@yahoo.com.

the threshold, e_1 and e_2 being the entities in two ontologies.

In this paper, we only consider one-to-one mappings between single entities. We do not cover mappings of whole ontologies or sub-trees, nor do we complex mappings as concatenation of literals (e. g. name corresponds to first name plus last name) or functional transformation of attributes (e. g. currency conversions).

2 Calculating Similarity

We solve the problem by computing similarity coefficients between entities of the two ontologies and then deducing a mapping from those coefficients. The coefficients, in the [0, 1] range, are calculated in three steps. The first step is linguistic similarity, mapping entities based on their names. The result is a linguistic similarity coefficient, lsim, between each pair of entities. The second step is the structural similarity of ontology entities based on the similarity of their contexts. The third step, called combining similarity, uses a weighted function and a sigmoid function to combine the above two similarities.

2.1 Linguistic similarity

The linguistic similarity (LS) function includes calculating the similarity between the name of concepts, properties and instances based on their names[4,7]. The linguistic similarity coefficient is in the [0, 1] range. The higher the coefficient is, the more similar the entities are. In our algorithm, we calculate the element similarity according to the following two steps:

Step 1 Normalization Similar entities in different ontologies often have names that differ due to the use of abbreviations, acronyms, punctuation, etc. So, as part of this step we perform tokenization[4] (parsing names into tokens based on punctuation, case, etc.). For example: Corporate_Body = {Corporate, Body}. The second part of this step is expansion[4] (identifying abbreviation and acronyms), For example: prof = professor. For each of these steps, we use a thesaurus defined by us, which includes domain — specific references.

Step 2 Comparison Compute the linguistic similarity[7]

$$\text{lism} = \frac{\sum_{t_1 \in T_1} \left[\max_{t_2 \in T_2} \text{related}(t_1, t_2) \right] + \sum_{t_2 \in T_2} \left[\max_{t_1 \in T_1} \text{related}(t_1, t_2) \right]}{|T_1| + |T_2|}$$

where T_1 is the tokenization set of entit y_1, and T_2 is the tokenization set of entit y_2.

The function: related(x, y) transfers the concept of Wordnet into a vector, then the similarity of two entities is calculated according to the cosine of the angle between the corresponding vectors[8].

According to the linguistic similarity function, we can obtain the initial similarity vectors, which describe, respectively: ① Concept similarity: concept_lism = LS (C_1, C_2); ② Property similarity: property_lism = LS (P_1, P_2); ③ Instance similarity: instance_lism = LS (I_1, I_2)

2.2 Structural similarity

The semantic information of the ontology is not only represented by its own linguistic characteristics, but also by its structure characteristics[2-4,9]. For example, the similarity of classes is related to their super classes, sub classes, properties, instances and brother classes.

In the rest of this section, we will set some rules manually to describe the structure characters of the ontology. The structural similarity is based in part on linguistic similarity calculated in the above phase.

Rule 1 If the super classes are similar, the actual classes are also similar.

$$\text{super\_sim} = \frac{\sum_{c_1 \in \text{super}(C_1)} \left[\max_{c_2 \in \text{super}(C_2)} \text{LS}(c_1, c_2) \right] + \sum_{c_2 \in \text{super}(C_2)} \left[\max_{c_1 \in \text{super}(C_1)} \text{LS}(c_1, c_2) \right]}{|\text{super}(C_1)| + |\text{super}(C_2)|}$$

Rule 2 If the sub classes are similar, the actual classes are also similar.

$$\text{sub\_sim} = \frac{\sum_{c_1 \in \text{sub}(C_1)} \left[\max_{c_2 \in \text{sub}(C_2)} \text{LS}(c_1, c_2) \right] + \sum_{c_2 \in \text{sub}(C_2)} \left[\max_{c_1 \in \text{sub}(C_1)} \text{LS}(c_1, c_2) \right]}{|\text{sub}(C_1)| + |\text{sub}(C_2)|}$$

Rule 3 If the related properties are similar, the actual classes are also similar.

$$\text{pro\_sim} = \frac{\sum_{p_1 \in \text{pro}(C_1)} \left[\max_{p_2 \in \text{pro}(C_1)} \text{LS}(p_1, p_2) \right] + \sum_{p_2 \in \text{pro}(C_2)} \left[\max_{p_1 \in \text{pro}(C_1)} \text{LS}(p_1, p_2) \right]}{|\text{pro}(C_1)| + |\text{pro}(C_2)|}$$

Rule 4 If the instances are similar, the actual classes are also similar.

$$\text{ins\_sim} = \frac{\sum_{i_1 \in \text{ins}(C_1)} \left[\max_{i_2 \in \text{ins}(C_1)} \text{LS}(i_1, i_2) \right] + \sum_{i_2 \in \text{ins}(C_2)} \left[\max_{i_1 \in \text{ins}(C_1)} \text{LS}(i_1, i_2) \right]}{|\text{ins}(C_1)| + |\text{ins}(C_2)|}$$

Rule 5 If the brother concepts are similar, the actual classes are also similar.

$$\text{bro\_sim} = \frac{\sum_{c_1 \in \text{bro}(C_1)} \left[\max_{c_2 \in \text{bro}(C_1)} \text{LS}(c_1, c_2) \right] + \sum_{c_2 \in \text{bro}(C_2)} \left[\max_{c_1 \in \text{bro}(C_1)} \text{LS}(c_1, c_2) \right]}{|\text{bro}(C_1)| + |\text{bro}(C_2)|}$$

Rule 6 If the super properties are similar, the actual properties are also similar.

$$\text{superpro\_sim} = \frac{\sum_{p_1 \in \text{superpro}(P_1)} \left[\max_{p_2 \in \text{superpro}(P_1)} \text{LS}(p_1, p_2) \right] + \sum_{p_2 \in \text{super}(P_2)} \left[\max_{p_1 \in \text{super}(P_1)} \text{LS}(p_1, p_2) \right]}{|\text{super pro}(P_1)| + |\text{super pro}(P_2)|}$$

Rule 7 If the sub properties are similar, the actual properties are also similar.

$$\text{subpro\_sim} = \frac{\sum_{p_1 \in \text{subpro}(P_1)} \left[\max_{p_2 \in \text{subpro}(P_1)} \text{LS}(p_1, p_2) \right] + \sum_{p_2 \in \text{subpro}(P_2)} \left[\max_{p_1 \in \text{subpro}(P_1)} \text{LS}(p_1, p_2) \right]}{|\text{subpro}(P_1)| + |\text{subpro}(P_2)|}$$

Rule 8 If the domain classes are similar, the actual properties are similar.

$$\text{domain\_sim} = \frac{\sum_{c_1 \in \text{domain}(P_1)} \left[\max_{c_2 \in \text{domain}(P_1)} \text{LS}(c_1, c_2) \right] + \sum_{c_2 \in \text{domain}(P_2)} \left[\max_{c_1 \in \text{domain}(P_1)} \text{LS}(c_1, c_2) \right]}{|\text{domain}(P_1)| + |\text{domain}(P_2)|}$$

Rule 9 If the range classes are similar, the actual properties are similar.

$$\text{range\_sim} = \frac{\sum_{c_1 \in \text{range}(P_1)} \left[\max_{c_2 \in \text{range}(P_1)} \text{LS}(c_1, c_2) \right] + \sum_{c_2 \in \text{range}(P_2)} \left[\max_{c_1 \in \text{range}(P_1)} \text{LS}(c_1, c_2) \right]}{|\text{range}(P_1)| + |\text{range}(P_2)|}$$

Rule 10 If the classes are similar, the actual instances are similar.

$$\text{class\_sim} = \frac{\sum_{c_1 \in \text{class}(I_1)} \left[\max_{c_2 \in \text{class}(I_1)} \text{LS}(c_1, c_2) \right] + \sum_{c_2 \in \text{class}(I_2)} \left[\max_{c_1 \in \text{class}(I_1)} \text{LS}(c_1, c_2) \right]}{|\text{class}(I_1)| + |\text{class}(I_2)|}$$

So, the similarity value between two entities of two ontologies should be related to the above similarity value. In the following section, we will give a general formula to combine linguistic similarity and structure similarity.

2.3 Combining similarity

Now we can calculate entity similarity by combining the linguistic similarity with the structure similarity. In general, we can use a weighted function to combine all kinds of similarities[4]; however, here we assume that low similarities are not available, while high similarities should be enhanced. Hence, we also use a sigmoid function[4,9] to combine the above similarities. The sigmoid function (see Fig. 1) can transfer a value in the high range into a value in the [0, 1] range. In our algorithm, the similarities are in the [0, 1] range, so we must shift the sigmoid function to fit our range. We define $\text{sig}(x) = \frac{1}{1 + e^{-10(x-0.5)}}$. If the similarity is below 0.5, it is unlikely that the similarity can be used; above 0.5, they probably can be used(see Fig. 2).

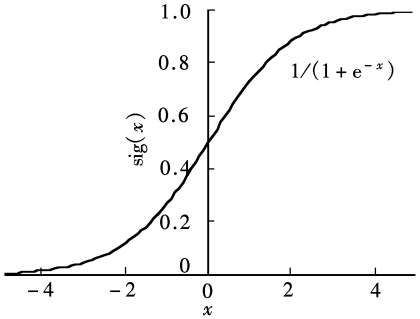

Fig. 1 Sigmoid function

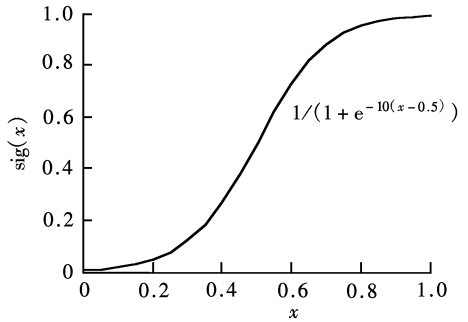

Fig. 2 The proposed sigmoid function

Concept similarity

$$w\mathrm{sim}_{\mathrm{concept}} = w_{\mathrm{lism}}\mathrm{sig}(\mathrm{concept\_lism}) + w_{\mathrm{super}}\mathrm{sig}(\mathit{super}\_\mathrm{sim}) + w_{\mathrm{sub}}\mathrm{sig}(\mathrm{sub}\_\mathrm{sim}) + w_{\mathrm{pro}}\mathrm{sig}(\mathrm{pro}\_\mathrm{sim}) + w_{\mathrm{ins}}\mathrm{sig}(\mathrm{ins}\_\mathrm{sim}) + w_{\mathrm{bro}}\mathrm{sig}(\mathrm{bro}\_\mathrm{sim})$$

with $w_{\mathrm{lism}} + w_{\mathit{super}} + w_{\mathrm{sub}} + w_{\mathrm{pro}} + w_{\mathrm{ins}} + w_{\mathrm{bro}} = 1$.

Property similarity

$$w\mathrm{sim}_{\mathrm{property}} = w_{\mathrm{lism}}\mathrm{sig}(\mathrm{property}\_\mathrm{lism}) + w_{\mathit{super}\mathrm{pro}}\mathrm{sig}(\mathrm{superpro}\_\mathrm{sim}) + w_{\mathrm{subpro}}\mathrm{prosig}(\mathrm{subpro}\_\mathrm{sim}) + w_{\mathrm{domain}}\mathrm{sig}(\mathrm{domain}\_\mathrm{sim}) + w_{\mathrm{range}}\mathrm{sig}(\mathrm{range}\_\mathrm{sim})$$

with $w_{\mathrm{lism}} + w_{\mathrm{superpro}} + w_{\mathrm{subpro}} + w_{\mathrm{domain}} + w_{\mathrm{range}} = 1$.

Instance similarity

$$w\mathrm{sim}_{\mathrm{instance}} = w_{\mathrm{lism}}\mathrm{sig}(\mathrm{instance}\_\mathrm{lism}) + w_{\mathrm{class}}\mathrm{sig}(\mathrm{class}\_\mathrm{sim})$$

with $w_{\mathrm{lism}} + w_{\mathrm{class}} = 1$.

3 Mapping Algorithm

In our methodology, we combine the linguistic similarity with the structure similarity, and use an iteration to obtain significantly better results (see Fig. 3).

Step 1 Calculating the linguistic similarity based on the name of entities;

Step 2 Judging the iteration time, we use iteration to obtain significantly better results, the iteration time is the high depth of class node in the two ontologies in our algorithm;

Step 3 Calculating the structure similarity based on the context of entities;

Step 4 Combining the linguistic similarity and the structure similarity;

Step 5 Outputing the mapping pairs.

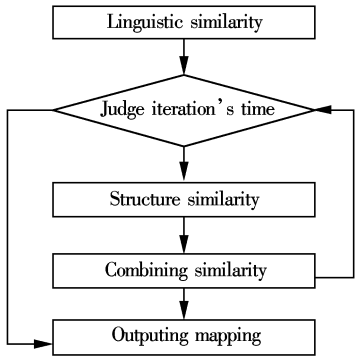

Fig. 3 The flow of algorithm

4 Experiment

We evaluate our algorithm on several real-world domains. Our goals are to evaluate the matching accuracy and to verify that our algorithm can work well in different domains.

The first example has two ontologies describing two databases. These ontologies were created by us for semantic enhance information search system based on grid. These ontologies have approximately 150 entities, including concepts, properties and instances.

The second example has two ontologies describing the television guides. These ontologies were downloaded from the knowledge representation and reasoning group at the Vrije University. These ontologies have approximately 500 entities, including concepts, properties and instances.

The third example has two ontologies describing two university departments. These ontologies were also obtained from the knowledge representation and reasoning group at the Vrije University (http://wbkr.cs.vu.nl/). These ontologies have approximately 400 entities, including concepts, properties and instances. The accuracy of mapping is shown in Tab. 1.

Tab. 1 Result of experiments

Domain		Class nodes	Property nodes	Instance nodes	Depth	Accuracy/%
Database	Database 1	29	15	40	4	70.3
	Database 2	25	20	38	3	
Television guide	T. G1	29	26	189	3	63.2
	T. G2	32	17	134	2	
University department	U. D1	21	24	137	3	66.7
	U. D2	35	41	157	3	

5 Conclusion

The vision of the semantic web is grand. With the proliferation of ontologies on the semantic web, the development of automated techniques for ontology mapping will be crucial to its success. We have shown a methodology for identifying mapping between ontologies based on their linguistic similarity and their structure similarity, mean-

while we import a sigmoid function and a weighted function to combine these similarities. The structure similarity is calculated depending on the rules we set manually. The results of experiments show that we can accurately match 63% to 70% of the nodes on several real-world domains. This approach can be extensible in the future, with the deep research on the ontology, when we set more rules for calculating the structure similarity.

References

[1] Gerd S, Alexander M. FCA-Merge: bottom-up merging of ontologies [A]. In: *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* [C]. Seattle: Morgan Kaufmann, 2001. 225 - 230.

[2] Doan AnHai, Madhavan Jayant, Domingos Pedro, et al. Learning to map between ontologies on the semantic web [A]. In: *Proceedings of the 11th International Conference on World Wide Web* [C]. Hawaii: ACM Press, 2002. 662 - 673.

[3] Natalya F N, Mark A M. The prompt suite: interactive tools for ontology merging and mapping [J]. *International Journal of Human-Computer Studies*, 2003, **59**(6): 983 - 1024.

[4] Madhavan Jayant, Bernstein Philip A, Rahm Erhard. Generic schema matching with cupid [A]. In: *Proceedings of 27th International Conference on Very Large Data Bases* [C]. Roma: Morgan Kaufmann, 2001. 49 - 58.

[5] Studer R, Benjamins V R, Fensel D. Knowledge engineering, principles and methods [J]. *Data and Knowledge Engineering*, 1998, 25(1,2): 161 - 197.

[6] Su Xiaomeng. A text categorization perspective for ontology mapping [R]. Norway: Department of Computer and Information Science, Norwegian University of Science and Technology, 2002.

[7] Yuan W, Yu C. Semantic mapping algorithm between XML scheme and ontology [J]. *Min-Micro Systems*, 2005, **26**(11): 1988 - 1990.

[8] Siddharth P. Incorporating dictionary and corpus information into a context vector measure of semantic relatedness [D]. Duluth: University of Minnesota, 2003.

[9] Marc E, York S. Ontology mapping—an integrated approach [EB/OL]. (2005-02-14) [2006-03-01]. http://www. aifb. uni-karlsruhe. de/WBS/ysu/publications/2004_ mapping _TR. pdf/.

一种新的自动本体映射算法

李选如　　何洁月

(东南大学计算机科学与工程学院,南京 210096)

摘要:提出了一种自动完成本体映射的算法. 该算法通过计算本体概念之间元素层与结构层上的相似性来完成相似度的计算. 在元素层上,该算法引进了 WordNet,通过将 WordNet 中对应的概念转换为向量,计算向量间夹角的余弦得到元素层概念的相似度. 在结构层上,该算法通过加权函数和 sigmoid 函数,基于元素层的计算结果,将元素层的相似度和结构层的相似度结合起来,完成本体之间相似度的计算,最终完成映射. 实验结果表明,该算法的匹配准确率可以达到 63% ~ 70%,可以有效地完成本体之间的映射.

关键词:本体;本体映射;语义 web

中图分类号:TP391