Ontology Matching Using an Artificial Neural Network to Learn Weights

Jingshan Huang[1], **Jiangbo Dang**[2], **José M. Vidal**[1], **and Michael N. Huhns**[1]
{huang27@sc.edu, jiangbo.dang@siemens.com, vidal@sc.edu, huhns@sc.edu}

[1]Computer Science and Engineering Department, University of South Carolina
[2]Siemens Corporate Research

Abstract

Ontologies are a formal, declarative knowledge representation model. They form a semantic foundation for many domains, such as Web services, E-commerce, and the Semantic Web, where applications can mutually understand and share information with each other. However, because ontologies reflect their designers' conceptual views of part of the world, heterogeneity is an inherent characteristic for ontologies. During (semi)automated matching among ontologies, different semantic aspects, i.e., concept names, concept properties, and concept relationships, contribute in different degrees to the matching result. Therefore, a vector of weights are needed to be assigned to these aspects. It is not trivial to determine what those weights should be, and current research work depends a lot on human heuristics. In this paper, we take an artificial neural network approach to learning and adjusting the above weights, and thereby support a new ontology matching algorithm, with the purpose to avoid some of the disadvantages in both rule-based and learning-based ontology matching approaches.

1 Introduction

As a formal and declarative knowledge representation model, ontologies are becoming increasingly important as a foundation for many kinds of Web applications. With the Semantic Web gaining attention as the next generation of the Web, reconciling different views of independently developed and exposed data sources becomes a critical issue. While ontologies serve as a basis for solving this problem, the heterogeneity among different ontologies poses an inherent problem. This is due to the fact that Web applications are developed by different parties that can design ontologies according to their own conceptual views of the world.

To handle this heterogeneity issue in ontologies, many approaches have been proposed. Generally speaking, there are two different kinds of solutions. The first is a centralized solution, in which a central ontology that is agreed-upon, global, and unique and includes every concept that can satisfy the needs of different parties is built. Cyc [Cyc, 2006] is a typical example of this centralized solution. However, a central ontology will never be large and compatible enough to include all concepts of interest to every individual ontology designer, so it will have to be modified and extended. Each new extension will be different and likely incompatible. Therefore, it is more common to turn to a second kind of solution, a distributed one. Basically, this solution focuses on the ability for individual ontologies to match and reconcile with each other, and possibly reuse each other as well. The ontology matching is initially carried out by a human. There are some critical drawbacks to this manual effort, including that it is time-consuming and error-prone, and does not scale. Therefore, to develop tools that are either automatic or semi-automatic, and can help people in matching ontologies is of great need and value.

Most ontology matching techniques belong to either rule-based or learning-based ones. Both categories have their disadvantages. In brief, the former ignore the information obtained from instance data. A more severe problem is the way this technique treats different semantic aspects during the ontology matching process. In general, ontologies are characterized by the aspects of concept names, concept properties, and concept relationships. These aspects have different contributions in understanding ontologies' semantics. Therefore, it is essential to assign different weights to these semantic aspects, if a more accurate and meaningful matching result is favored. However, current research work has made use of human intervention and/or prior domain knowledge to define these weights. The main problems for learning-based matching algorithms include a relatively longer running time, and the difficulty in getting enough and/or good-quality data.

In this paper, we present a new approach to match ontologies that combines both rule-based and learning-based algorithms. Our contributions are in the following. (1) Our approach integrates an artificial neural network (ANN) technique in our algorithm, such that the weights mentioned above can be learned instead of being specified in advance by a human. (2) Moreover, our learning technique is carried out based on the ontology schema information alone, which distinguishes it from most other learning-based algorithms.

The rest of this paper is organized as follows. Section 2 introduces the related work in ontology matching. Section 3 gives an overview of our approach, and discusses the challenges in applying machine learning techniques without the

help from instance data information. Section 4 presents the details of our algorithm. Section 5 reports the experiments conducted and analyzes the results. Section 6 concludes with future work.

2 Related Work

According to the classification in [Doan and Halevy, 2005], most schema matching techniques can be divided into two categories: rule-based and learning-based. The former considers schema information only, and different algorithms distinguish from each other in their specific rules; while the latter considers both schema information and instance data, and various kinds of machine learning techniques have been adopted.

2.1 Rule-Based Ontology Matching

PROMPT [Noy and Musen, 2000] provides a semi-automatic approach to ontology merging. By performing some tasks automatically and guiding the user in performing other tasks, PROMPT helps in understanding and reusing ontologies.

[Dou *et al.*, 2003] views ontology translation as ontology merging and automated reasoning, which are in turn implemented through a set of axioms. The authors regard the ontology merging as taking the union of the terms and the axioms defining them, then adding bridging axioms through the terms in the merge.

Cupid [Madhavan *et al.*, 2001] discovers mappings between schema elements based on their names, data types, constraints, and schema structure. Cupid has a bias toward leaf structure where much of the schema content resides. The experimental results show a better performance than DIKE and MOMIS.

[Giunchiglia *et al.*, 2005] views match as an operator that takes two graph-like structures and produces a mapping between the nodes. F. Giunchiglia et al. discover mappings by computing semantic relations, determined by analyzing the meaning which is codified in the elements and the structures.

The hypothesis in [Stephens *et al.*, 2004] is that a multiplicity of ontology fragments can be related to each other without the use of a global ontology. Any pair of ontologies can be related indirectly through a semantic bridge consisting of many other previously unrelated ontologies. [Huang *et al.*, 2005] extends this work to incorporate: extended use of WordNet; use of the Java WordNet Library API for performing run-time access to the dictionary; and reasoning rules based on the domain-independent relationships and each ontology concept's property list to infer new relationships.

2.2 Learning-Based Ontology Matching

GLUE [Doan *et al.*, 2003] employs machine learning techniques to find semantic mappings between ontologies. After obtaining the results from a Content Learner and a Name Learner, a Metalearner is used to combine the predictions from both learners. Then common knowledge and domain constraints are incorporated through a Relaxation Labeler, and the mappings are finally calculated. In addition, the authors extend GLUE to find complex mappings.

[Williams, 2004] introduces a methodology and algorithm for multiagent knowledge sharing and learning in a peer-to-peer setting. DOGGIE enables multiagent systems to assist groups of people in locating, translating, and sharing knowledge. After locating similar concepts, agents can continue to translate concepts and then are able to share meanings.

[Soh, 2002] describes a framework for distributed ontology learning embedded in a multiagent environment. The objective is to improve communication and understanding among the agents while agent autonomy is still preserved. Agents are able to evolve independently their own ontological knowledge, and maintain translation tables through learning to help sustain the collaborative effort.

[Wiesman and Roos, 2004] presents an ontology matching approach based on probability theory by exchanging instances of concepts. During each step of the matching process, the likelihood that a decision is correct is taken into account. No domain knowledge is required, and the ontology structure plays no role.

[Madhavan *et al.*, 2005] shows how a corpus of schemas and mappings can be used to augment the evidence about the schemas being matched. Such a corpus typically contains multiple schemas that model similar concepts and their properties. They first increase the evidence about each element being matched by including evidence from similar elements in the corpus. Then they learn statistics about elements and their relationships to infer constraints.

2.3 Other Ontology Matching Algorithms

COMA [Do and Rahm, 2002] is a platform to combine multiple matchers in a flexible way. It provides a large spectrum of individual matchers and several mechanisms to combine the results of matcher executions.

Based on two distinguishing characteristics observed on the "deep Web", a statistical schema matching is presented in [He and Chang, 2003]. All input schemas are matched by finding an underlying generative schema model. In addition, the authors specialize their general framework to develop Algorithm MGSsd.

[Gruninger and Kopena, 2005] describes how model-theoretic invariants of an ontology can be used to specify semantic mappings translation definitions between application ontologies and an interlingua. The sets of models for the core theories of PSL are partitioned into equivalence classes, each of which is axiomatized using a definitional extension of PSL.

3 Overview of Our Approach

In our opinion, the semantics of an ontology concept is determined by three aspects: (1) the name of the concept; (2) the properties of the concept; and (3) the relationships of the concept. These three features together specify a conceptual model for each concept from the viewpoint of an ontology designer. For example, in the `University` ontology from mondeca.com[1], a concept has "Faculty" as its name, three properties ("emailAddress", "homePhone", and "workPhone"), and four relationships (*subClassOf* concept "Em-

[1] http://www.mondeca.com/owl/moses/univ.owl

ployee", *superClassOf* concepts "Lecturer", "PostDoc", and "Professor").

3.1 Problems with Existing Matching Algorithms

The rule-based matching algorithms usually have the advantage of relatively fast running speed, but share the disadvantage of ignoring the additional information from instance data. In addition, there is a more serious concern for this type of algorithms. In order to obtain a helpful matching result from (semi)automated tools, more than one of the three semantic aspects mentioned above should be considered. If only one aspect is taken into account then a meaningful matching result is unlikely to be acquired. Once two or more aspects are considered, it is unavoidable to determine the corresponding weights for different aspects, reflecting their different importances (or contributions) in ontology matching. To the best of our knowledge, most existing rule-based algorithms make use of human heuristics and/or domain knowledge to predefine these weights. Moreover, once weights are determined, they are unlikely to be updated, or at most by trial-and-error.

While taking advantages of extra clues contained in instance data, the learning-based matching algorithms are likely to be slower. In addition, the difficulty in getting enough good-quality data is also a potential problem. On the other hand, it is very challenging for machines to learn to reconcile ontology structures by only providing with schema information. The most critical challenge is that, because ontologies reflect their designers' conceptual views of part of the world, they exhibit a great deal of diversities. Identical terms can be used to describe different concepts, or vice versa, different terms can be assigned to the same concept. A more complicated situation is, even if the same set of terms are adopted, which is almost impossible in the real life, different designers can still create different relationships for the same concept, corresponding to their different conceptual views for this concept. Compared with schemas, instance data usually have a lot less varieties.

3.2 Our Solution

Based on the insight of the pros and cons of these two approaches, we present a new matching algorithm, Superconcept Formation System (SFS), which combines rule-based and learning-based solutions. We integrate machine learning techniques, such that the weights of a concept's semantic aspects can be learned from training examples, instead of being ad-hoc predefined ones. In addition, in order to avoid the problem of lacking instance data (either quality or quantity), which is common for real-world ontologies, our weight learning technique is carried out at the schema level, instead of the instance level.

Our main idea is, given a pair of ontologies being matched, although it is true that many design diversities might exist, it is still reasonable to assume that the contributions of different semantic aspects to ontology understanding should be independent of specific concepts. In fact, different contributions, which are the foundation for different weights, are characteristics of ontologies viewed as a whole. That is, during ontology matching, weights are features with regard to ontologies, rather than individual concepts. Therefore, it is possible to learn these weights for **all** concepts by training examples from a **subset** of concepts.

The following assumptions are made in this paper for two ontologies being matched.

1. They are from the same or similar general domain, and they have "Thing" as a built-in common root.
2. Each is correct or reasonable in itself.

Ontology matching consists of many mapping tasks, SFS concentrates on finding pairs of equivalent concepts as the first step. In addition, after the successful discovery of equivalent concept pairs, it is not difficult to design an algorithm to merge/align corresponding ontologies.

There are many different kinds of relationships in ontologies, e.g., *superClassOf*, *subClassOf*, *partOf*, *contains*, etc. In this paper, we consider only the *super/subClassOf* relationships, which are the most common ones in most real-world ontologies. We plan to extend SFS to include other relationships later.

4 Details of SFS

We build a 3-dimension vector for each concept, and each dimension records one semantic aspect, i.e., concept name, concept properties, and concept relationships. When we match two concepts, we compare their contents in these three dimensions, and acquire the corresponding similarity in each dimension. Recall that our goal is to find the equivalent concept pairs.

4.1 Similarity in Concept Names

The similarity s_1 between a pair of concept names is a real value in the range of $[0, 1]$. Some pre-processing on these two strings is performed before the calculation of s_1. For example, the removal of hyphens and underscores. Another example is the transformation of a noun from its plural form to a single form, which is accomplished automatically through WordNet JWNL API [JWNL-1.3, 2006].

If two names have an exact string matching or are synonyms of each other in WordNet then s_1 has a value of 1. Otherwise, s_1 is calculated according to

$$s_1 = 1 - \frac{d}{l}, \tag{1}$$

where d stands for the edit distance between two strings, and l for the length of the longer string.

4.2 Similarity in Concept Properties

Given two lists of concept properties (including those inherited from ancestors), p_1 and p_2, their similarity s_2 is a real value in the range of $[0, 1]$, and s_2 is calculated according to

$$s_2 = \frac{n}{m}, \tag{2}$$

where n is the number of pairs of properties matched, and m is the smaller cardinality of lists p_1 and p_2.

In order for a pair of properties (one from p_1 and the other from p_2) to be matched, their data types should be the same

or compatible with each other (*float* and *double* for example), and the property names should have a similarity value greater than a threshold. Notice that here we use the same procedure as in Section 4.1 to calculate the similarity between a pair of property names. In addition, we adopt the idea of "stable marriage" in determining the matched property pairs. That is, once two properties are considered matched, it means that they both find the best matched one from the other property list. Imagine a similarity matrix built between p_1 and p_2, each time we pick up a pair with the maximum value in the matrix, say cell $[i, j]$, and then discard row i and column j.

4.3 Similarity in Concept Relationships

As mentioned before, we take into account only the *super/subClassOf* relationships. In order to obtain a better matching result, we try to make use of as much information as we can. For example, suppose there are two pairs of equivalent concepts, and the numbers of concepts in-between are different from each other, i.e., the ontology with more detailed design tends to have more intermediate concepts. If the direct parent alone is considered, the information from this multilayered parent-child hierarchy will be ignored. Therefore, we not only consider the direct parent of a concept, but also all ancestors (concepts along the path from a concept up to the root "Thing") of this concept as well. Descendants (direct and indirect children of a concept) are not taken into account as that would lead to an infinite loop.

Given two lists of concept ancestors, a_1 and a_2, their similarity s_3 is a real value in the range of $[0, 1]$, and is obtained by first calculating the similarity values for pairwise concepts (one from a_1, the other from a_2, considering all combinations), then assigning the maximum value to s_3. Notice that this is a recursive procedure but is guaranteed to terminate, because (1) the number of concepts is finite; (2) we are traversing ontologies bottom-up; and (3) "Thing" is a common root for the two ontologies being matched.

4.4 Concept Similarity Matrix

After s_1, s_2, and s_3 between two concepts, C_1 and C_2, are calculated, the similarity value s between C_1 and C_2 is obtained as the weighted sum of s_1, s_2, and s_3:

$$s = \sum_{i=1}^{3}(w_i s_i), \tag{3}$$

where $\sum_{i=1}^{3} w_i = 1$. Notice that w_i are randomly initialized, and will be adjusted through a learning process (see Sec. 4.5).

For two ontologies being matched, $\mathcal{O}_1$ and $\mathcal{O}_2$, we calculate the similarity values for pairwise concepts (one from $\mathcal{O}_1$, the other from $\mathcal{O}_2$, considering all combinations). Then we build a $n_1 \times n_2$ matrix $\mathcal{M}$ to record all values calculated, where n_i is the number of concepts in $\mathcal{O}_i$. The cell [i, j] in $\mathcal{M}$ stores the similarity value between the i^{th} concept in $\mathcal{O}_1$ and the j^{th} concept in $\mathcal{O}_2$.

4.5 Weight Learning through ANN

The main purpose of SFS is to try to learn the different weights for three semantic aspects during the ontology matching process. We design our learning problem as follows.

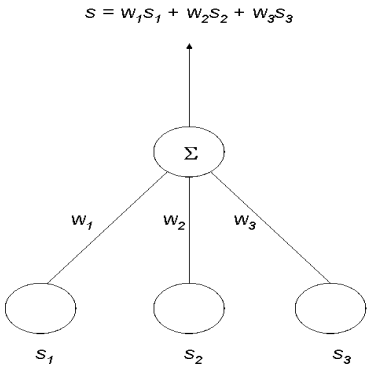

Figure 1: Neural Network Structure

- Task T: match two ontologies (in particular, find equivalent concept pairs)
- Performance measure P: *Precision* and *Recall* measurements with regard to manual matching
- Training experience E: a set of equivalent concept pairs by manual matching
- Target function V: a pair of concepts $\rightarrow \Re$
- Target function representation: $\hat{V}(b) = \sum_{i=1}^{3}(w_i s_i)$

We choose ANN as our learning technique, based on the following considerations.

- Instances are represented by attribute-value pairs
- The target function output is a real-valued one
- Fast evaluation of the learned target function is preferable

Network Design

We adopt a two-layer 3×1 network in SFS, as shown in Figure 1. The input into this network is a vector $\vec{s}$, which consists of s_1, s_2, and s_3, representing the similarity in name, properties, and ancestors, respectively, for a given pair of concepts. The output from this network is s, the similarity value between these two concepts as given by Formula (3). Notice that a linear function might not be powerful enough to reflect the true relationships among w_i. However, the delta rule converges toward a best-fit approximation to the target concept even when the training examples are not linearly separable [Mitchell, 1997]. If more relationships among ontology concepts are to be considered then one or more layers of hidden units might need to be added to express a rich variety of nonlinear decision surfaces.

Initially, we obtain a concept similarity matrix $\mathcal{M}$ for $\mathcal{O}_1$ and $\mathcal{O}_2$, with w_i being initialized randomly. Then we randomly pick up a set of concepts from $\mathcal{O}_1$, and find the corresponding equivalent concepts by a manual matching with $\mathcal{O}_2$. Each of such manually matched pairs will be processed

by SFS, and the similarity values in name, properties, and ancestors for these two concepts are calculated and used as a training example to the network in Figure 1.

Hypothesis Space and Our Searching Strategy

We regard the hypothesis space in this learning problem as a 3-dimensional space consisting of w_1, w_2, and w_3, that is, a set of weight vectors $\vec{w}$. Remember that our objective is to find the weights that best fit the training examples. We adopt gradient descent (delta rule) as our training rule, and our searching strategy within the hypothesis space is to find the hypothesis, i.e., weight vector, that minimizes the training error with regard to all training examples. According to [Mitchell, 1997], a standard definition of the training error E of a hypothesis is given by

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2, \tag{4}$$

where D is the set of training examples, t_d is the target output for training example d, and o_d is the output of the network for d.

We customize the above formal definition according to the characteristics of our learning problem as follows. For any training example d, instead of a given target value t_d, we need some other values. The intuition is that a given pair of manually matched concepts corresponds to a cell $[i, j]$ in $\mathcal{M}$, therefore, the value of cell $[i, j]$ should be the maximum one in both row i and column j. Suppose the maximum value for row i and column j are t_r and t_c, respectively, then our customized description of E is

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} [(t_r - o_d) + (t_c - o_d)]^2. \tag{5}$$

Accordingly, the weight update rule for gradient descent in SFS is

$$\Delta w_i = \eta \sum_{d \in D} [(t_r - o_d) + (t_c - o_d)] s_{id}, \tag{6}$$

where η is the learning rate, and s_{id} is the s_i value for a specific training example d.

5 Experiments

Our hypothesis is that the three weights for semantic aspects learned from a subset of concepts are representative of all concepts in the same ontology. In order to verify this, we need to show by our experiments: (1) the learning process itself is a correct one, i.e., three weights converge to certain values; and (2) when we apply the learned weights to other concepts, the matching result from our approach has a good performance.

We adopt two **real-world** ontologies built by professionals: `akt`[2] and `iswc`[3]. The characteristics of these two ontologies are summarized in Table 1.

[2]http://www.csd.abdn.ac.uk/∼cmckenzi/playpen/rdf/akt_ontology_LITE.owl

[3]http://annotation.semanticweb.org/iswc/iswc.owl

Features	akt	iswc
Max Depth of Ontology	8	4
Number of Concepts	83	33
Number of Properties	105	68
Number of Relationships	116	49
Number of *super/subClassOf* Relationships	54	32
Percentage of *super/subClassOf* Relationships	71%	65%

Table 1: Characteristics of Source Ontologies

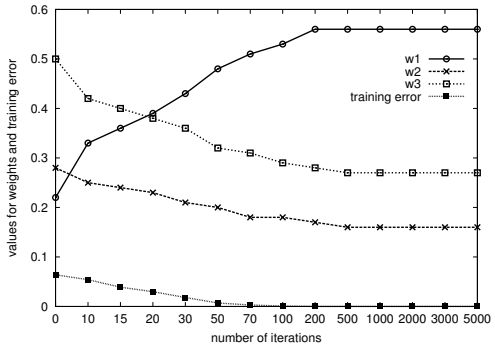

Figure 2: Values for Weights and Training Error – Setting 1

Results and Analysis

We randomly set $w_1 = 0.22$, $w_2 = 0.28$, $w_3 = 0.5$, learning rate η is set to 0.05, and we provide with 3 pairs of equivalent concepts by human. Figure 2 shows the values of the training error and w_i after different iterations of the gradient descent algorithm. After setting η to 0.2 and keeping all other parameters the same as before, the experiment result is shown in Figure 3.

We then use the learned weights (w_1 = 0.56, w_2 = 0.16, w_3 = 0.27) to match the remaining concepts. Out of 13 pairs of equivalent concepts suggested by SFS, 11 are agreed by two ontology experts; and there are 4 pairs not output by SFS but suggested by experts.

From the evolution of weights in both Figure 2 and 3, it is clear that **all three weights for semantic aspects converge**. At the same time, the training error also converges, along with the change of these weights. Moreover, after the learning rate

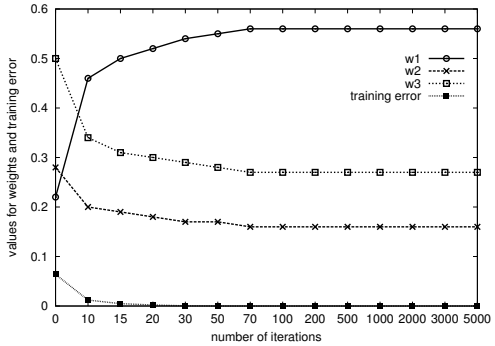

Figure 3: Values for Weights and Training Error – Setting 2

changes from 0.05 to 0.2, the convergence speeds up. The number of iterations needed drops dramatically from 500 in the former to 70 in the latter.

By comparing the matching result from SFS with that from ontology experts, we calculate both *Precision* (the percentage of correct predictions over all predictions) and *Recall* (the percentage of correct predictions over correct mappings) as follows:

- *Precision* = $\frac{11}{13}$ = 0.85
- *Recall* = $\frac{11}{15}$ = 0.73

Notice that this is achieved with the help from human efforts playing a very small portion—only 3 pairs of equivalent concepts, which are 20% of 15 (the total number of equivalent pairs agreed by human experts), are input into SFS. Also notice that ANN usually requires a large number of training examples, which is not the case in this paper. The possible reason could be the simple (but justified via experiments) structure we choose for our learning problem.

6 Conclusion

Being a semantic foundation for Web applications, ontologies help in reconciling different views of independently developed and exposed data sources. However, due to their inherent heterogeneity, ontologies need to be matched before they are able to be made better use of. We present SFS, a new matching algorithm aiming at overcoming some disadvantages of both rule-based and learning-based approaches. Our contributions are: (1) we exploit an approach to learn the weights for different semantic aspects of ontologies, through applying an artificial neural network technique during the ontology matching; and (2) we challenge the difficult problem of carrying out machine learning techniques without the help from instance data. We explain and analyze our algorithm in details, and a set of experiments verify the promising result of SFS.

Our focus has been on locating the equivalent concept pairs between two ontologies, leaving the other mapping tasks as future work, such as the discovery of parent-child concept pairs, the finding of sibling concept pairs, and so on. Another potential direction for the future work is to consider other relationships besides just the *super/subClassOf* relationships. In that case, multilayer networks might be more appropriate, owing to their representational power.

References

[Cyc, 2006] Cyc. http://www.cyc.com, June 2006.

[Do and Rahm, 2002] H. Do and E. Rahm. Coma c a system for flexible combination of schema matching approaches. In *Proceedings of the Twenty-eighth VLDB Conference*, Hong Kong, China, 2002.

[Doan and Halevy, 2005] A. Doan and A. Y. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine*, 26(1):83–94, 2005.

[Doan *et al.*, 2003] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to match ontologies on the semantic web. *The VLDB Journal*, 12(4):303–319, 2003.

[Dou *et al.*, 2003] D. Dou, D. McDermott, and P. Qi. Ontology translation on the semantic web. In *Proceedings of the International Conference on Ontologies, Databases, and Applications of Semantics*, Berlin, 2003. Lecture Notes in Computer Science, Springer-Verlag.

[Giunchiglia *et al.*, 2005] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. Semantic schema matching. In *Proceedings of the Thirteenth International Conference on Cooperative Information Systems (CoopIS 05)*, Agia Napa, Cyprus, November 2005.

[Gruninger and Kopena, 2005] M. Gruninger and J. B. Kopena. Semantic integration through invariants. *AI Magazine*, 26(1):11–20, 2005.

[He and Chang, 2003] B. He and K. C. Chang. Statistical schema matching across web query interfaces. In *Proceedings of SIGMOD 03*, San Diego, CA, June 2003.

[Huang *et al.*, 2005] J. Huang, R. Zavala, B. Mendoza, and M. N. Huhns. Reconciling agent ontologies for web service applications. In *Proceedings of Multiagent System Technologies: Third German Conference (MATES 05)*, Berlin, September 2005. Springer Verlag.

[JWNL-1.3, 2006] Java WordNet Library JWNL-1.3. http://sourceforge.net/projects/jwordnet, June 2006.

[Madhavan *et al.*, 2001] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proceedings of the Twenty-seventh VLDB Conference*, 2001.

[Madhavan *et al.*, 2005] J. Madhavan, P. A. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. In *Proceedings of the Twenty-first International Conference on Data Engineering (ICDE 05)*, Tokyo, Japan, April 2005.

[Mitchell, 1997] T. M. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., 1997.

[Noy and Musen, 2000] N. F. Noy and M. A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 00)*, Menlo Park, CA, USA, 2000. AAAI Press.

[Soh, 2002] L.-K. Soh. Multiagent distributed ontology learning. In *Working Notes of the second AAMAS OAS Workshop*, Bologna, Italy, July 2002.

[Stephens *et al.*, 2004] L. Stephens, A. Gangam, and M. N. Huhns. Constructing consensus ontologies for the semantic web: A conceptual approach. *World Wide Web Journal*, 7(4):421–442, 2004.

[Wiesman and Roos, 2004] F. Wiesman and N. Roos. Domain independent learning of ontology mappings. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 04)*, New York, NY, July 2004.

[Williams, 2004] A. B. Williams. Learning to share meaning in a multi-agent system. *Autonomous Agents and Multi-Agent Systems*, 8(2):165–193, 2004.