Ontology-Based Partner Selection in Business Interaction

Jingshan Huang
Computer Science and Engineering Department
University of South Carolina
Columbia, SC 29208, USA
+1-803-777-3768
huang27@sc.edu

Jiangbo Dang
Siemens Corporate Research
Princeton, NJ 08540, USA
+1-803-318-8169
jiangbo.dang@siemens.com

Michael N. Huhns
Computer Science and Engineering Department
University of South Carolina
Columbia, SC 29208, USA
+1-803-777-5921
huhns@sc.edu

Abstract. Traditional businesses are finding great advantages from the incorporation of E-business capabilities, especially for participation in the global economy. The global economy is inherently open and dynamic, which imposes a requirement that businesses must coordinate with each other if they are to be most efficient and successful. To aid in this coordination and achieve seamless and autonomic interoperation, e-business partners are choosing to be represented by service agents. However, before service agents are able to coordinate well with each other, they need to understand each others' service descriptions. Ontologies developed by service providers to describe their service can help in this. Unfortunately, due to the heterogeneity implicit in independently designed ontologies, distributed e-businesses will encounter semantic mismatches and misunderstandings. In this chapter, we introduce a compatibility vector system, created upon a schema-based ontology-merging algorithm, to determine and maintain ontology compatibility, which can be used as a basis for businesses to select candidate partners with which to interoperate.

1 Introduction

Because of its potential to provide new opportunities and unparalleled efficiencies, e-business is increasingly being utilized by enterprises. Broadly speaking, e-business can be regarded as any business process that relies on an automated information system, which typically incorporates Web-based technologies. The Web-based technologies within e-business enable companies to link their internal and external data processing systems in more efficient and flexible ways, so that they can be more agile and responsive to their customers.

E-business is usually conducted using the dynamic environment of the Internet and the World-Wide Web. Therefore, to introduce agents into e-business, i.e., to represent services or business partners by agents, might increase the extent to which the data process is automated. This possible advantage results from agents' autonomy and proactiveness.

It has been discovered that exposing formerly internal activities to external business collaborators can yield increased value. Although there is value in accessing the service provided by a single agent through a semantically well-founded interface, greater value is derived through enabling a flexible composition of e-businesses, which not only creates new services, but also potentially adds value to existing ones (Singh and Huhns, 2005). As the first step of communication and integration of e-business activities, mutual understanding of semantics among services plays an important role in the composition process.

An ontology serves as a declarative model for the knowledge and capabilities possessed by an agent or of interest to an agent. It forms the foundation upon which machine-understandable service descriptions can be obtained and, as a result, it makes automatic coordination among agents possible. By providing a more comprehensible and formal semantics, the use of and reference to ontologies can help the functionalities and behaviors of agents to be described, advertised, discovered, and composed by others. Eventually, these agents would be able to interoperate with each other, even though they have not been designed to do so.

However, because it is impractical to force all agents to adopt a global ontology that describes every concept that is or might be included as part of their services, ontologies from different agents typically have heterogeneous semantics. Due to this basic characteristic, agents need to reconcile ontologies and form a mutual understanding when they interact with each other. Only via this means will agents be able to comprehend and/or integrate the information from different sources, and enhance process interoperability thereafter.

In this chapter, we focus on an important but mostly neglected research topic—how to select suitable business partners with which to interact. More compatible ontologies are likely to yield better understanding among the partners. In this sense, ontology compatibility is used as a basis for businesses to select candidate partners. Based on this insight, we design a compatibility vector system, built upon an ontology-merging algorithm, to measure and maintain ontology compatibility.

2 Related Work

2.1 Related Work in Ontology Matching

The need for automatic or semi-automatic mapping, matching, and merging of ontologies from different sources has prompted considerable research, such as GLUE (Doan et al., 2003), PROMPT (Noy and Musen, 2000), Cupid (Madhavan et al., 2001), COMA (Do and Rahm, 2002), Similarity Flooding (Melnik et al., 2002), S-Match (Giunchiglia et al., 2005), and Puzzle (Huang et al., 2005). Here we briefly describe these systems. The more detailed information about their comparisons can be found in our other publications about ontology matching algorithms, http://www.cse.sc.edu/∼huang27/paper/JPCC.pdf for example.

PROMPT is a tool that uses linguistic similarity matches between concepts for initiating the merging or alignment process, and then uses the underlying ontological structures of the Protégé-2000 environment to inform a set of heuristics for identifying further matches between the ontologies. PROMPT has good performance in terms of precision and recall. However, user intervention is required, which is not always available in many applications.

Similarity Flooding utilizes a hybrid matching technique based on a measure of similarity spreading from similar nodes to their adjacent neighbors. Before a fix-point is reached, alignments between nodes are refined iteratively. This algorithm considers only simple linguistic similarity between node names, ignoring node properties and inter-node relationships.

Cupid combines linguistic and structural schema matching techniques, as well as the help of a precompiled dictionary. But it can only work with a tree-structured ontology instead of a more general graph-structured one. As a result, there are many limitations to its application, because a tree cannot represent multiple-inheritance, an important characteristic in ontologies.

COMA provides an extensible library of matching algorithms, a framework for combining results, and an evaluation platform. According to their evaluation, COMA is performing well in terms of precision, recall, and other measures. Although being a composite schema matching tool, COMA does not integrate reasoning and machine learning techniques.

S-Match is a modular system into which individual components can be plugged and unplugged. The core of the system is the computation of relations. Five possible relations are defined between nodes: equivalence, more general, less general, mismatch, and overlapping. Giunchiglia et al. claim that S-Match outperforms Cupid, COMA, and Similarity Flooding in measurements of precision, recall, overall, and F-measure. However, like Cupid, S-Match uses a tree-structured ontology.

GLUE introduces well-founded notions of semantic similarity, applies multiple machine learning strategies, and can find not only one-to-one mappings, but also complex mappings. However, it depends heavily on the availability of instance data. Therefore, it is not practical for cases where there is an insufficient number of instances or no instance at all.

2.2 Related Work in Ontology Application in E-Service

The application of ontologies in e-service environments has been studied widely. In (Honavar et al., 2001), Honavar et al. describe several challenges in information extraction and knowledge acquisition from heterogeneous, distributed, autonomously operated, and dynamic data sources when scientific discovery is carried out in data-rich domains. They outline the key elements of algorithmic and systems solutions for computer-assisted scientific discovery in such domains, including ontology-assisted approaches to customizable data integration and information extraction from heterogeneous and distributed data sources. Ontology-driven approaches to exploratory data analysis from alternative ontological perspectives are also discussed.

An ontology-based information retrieval model for the Semantic Web is presented in (Song et al., 2005). The authors generate an ontology through translating and integrating domain ontologies. The terms defined in the ontology are used as metadata to markup the Web's content; these semantic markups are semantic index terms for information retrieval. The equivalent classes of semantic index terms are obtained by using a description logic reasoner. It is claimed that the logical views of documents and user information needs, generated in terms of the equivalent classes of semantic index terms, can represent documents and user information needs well, so the performance of information retrieval can be improved when a suitable ranking function is chosen.

Tijerino et al. introduce an approach (TANGO) to generate ontologies based on table analysis (Tijerino et al., 2005). TANGO aims to understand a table's structure and conceptual content; discover the constraints that hold between concepts extracted from the table; match the recognized concepts with ones from a more general specification of related concepts; and merge the resulting structure with other similar knowledge representations. The authors claim that TANGO is a formalized method of processing the format and content of tables that can serve to incrementally build a relevant reusable conceptual ontology.

2.3 Related Work in Quality of Service

Quality of service (QoS) is becoming a significant factor with the widespread deployment of Web services. By QoS, we refer to the non-functional properties of services, such as reliability, availability, and security. Ontology quality consists of many aspects, of which the compatibility is one of the most important ones, because better compatibility leads directly to better understanding, which is critical during business interactions. Notice that the quality of services themselves is a separate research topic that will not be covered in this chapter.

(Bilgin and Singh, 2004) proposes a Service Query and Manipulation Language (SWSQL) to maintain QoS attribute ontologies and to publish, rate, and select services by their functionality as well as QoS properties. Based on SWSQL, they extend the UDDI registry to a service repository by combing a relational database and an attribute ontology.

Zhou et al. (Zhou et al., 2004) provide a DAML-QoS ontology as a complement to a DAML-S ontology in which multiple QoS profiles can be attached to one service profile. In addition, they present a matchmaking algorithm for QoS properties.

One widely used QoS attribute is user rating, but it is subjective to the perception of an end user and is limited by the lack of an objective representation of performance history. Kalepu et al. (Kalepu et al., 2004) introduce reputation, a composition of user rating, compliance, and verity as a more viable QoS attribute. Ontologies are applied to QoS-aware service selection, execution, and composition. A selected ontology itself can adopt some QoS measures to facilitate mutual ontology understanding as discussed in this paper.

3 Example Scenario and Our Solution

3.1 A Running Example

An example scenario of the business interaction within an e-business environment can be envisioned as follows. E-business partners are represented by service agents, then:

1. A number of agents form an e-business community (EBC) within which services provided by different agents might be integrated and have the ability to render a more complete and functional service. This integration requires the mutual understanding of the individual ontologies underlying each agent.

2. The agents outside this EBC can request help from the community and make use of its services, either the original ones or the integrated one. This request requires not only an understanding of the related ontologies, but also the ability to choose suitable agent(s), especially under the situations where resources are limited.

Consider the travel business as an example. Many websites provide services for this business area, e.g., Expedia.com, Orbitz.com, and Hotels.com. When a customer makes his/her travel plan, it is very possible that, for some specific dates, one website would provide a flight ticket with the lowest price, while another website would have the best offer in car rentals, and a third website would offer the cheapest hotel reservations. Therefore, it is preferable for the services from all these websites to be integrated to render the best vacation package for each customer. On the other hand, some customers might be interested in one service alone, only to buy a flight ticket for example. In this case, these customers would find it beneficial if there is an agent gathering and comparing information from all related websites. In either case, the mutual understanding among agents representing different websites is necessary.

Two major problems need to be solved. First, during the formation of an EBC, how can it be ensured that all agents within the community have no problem in understanding each other's ontology? Second, an agent seeking coordination from outside this community would like to choose those agents that understand its ontology best. How can it ensure this selection is a correct one?

3.2 Solution Overview

We design an ontology compatibility vector system to tackle the above challenges. This vector system is built upon an ontology-merging algorithm. Our main idea is: along with the formation of an EBC, we create a *center* ontology by merging all original ones; then the *distances* (dissimilarities) from original ontologies to this center are suitably encoded in the compatibility vectors stored in the center. Based on the information contained in the vectors, partners are supposed to understand the ontology from each other without trouble, and the partner from outside this community will have no difficulty in choosing candidate partners that have ontologies with good compatibilities. In addition, these vectors can be adjusted efficiently and dynamically during the period in which the EBC is formed. In the following sections, we will first briefly introduce our algorithm in merging ontologies, then we present the compatibility vector system in detail.

4 A Schema-Based Ontology-Merging Algorithm

Our goal is to develop a methodology for constructing a merged ontology from two original ones. This methodology can then be applied iteratively to merge all original ontologies within an EBC. Our methodology extends the ontology-merging algorithm presented in (Huang et al., 2005), and is summarized next.

4.1 Top-Level Procedure

The ontology merging is carried out at the schema level. Internally we represent an ontology using a directed acyclic graph $G(V, E)$, where V is a set of ontology concepts (nodes), and E is a set of edges between two concepts, i.e., $E = \{(u, v) | u, v \in V$ and u is a superClass of $v\}$. In addition, we assume that all ontologies share "Thing" as a common built-in root. In order to merge two ontologies, G_1 and G_2, we try to relocate each concept from one ontology into the other one. We adopt a breadth-first order to traverse G_1 and pick up a concept C as the target to be relocated into G_2. Consequently, at least one member of C's parent set $Parent(C)$ in the original graph G_1 has already been put into the suitable place in the destination graph G_2 before the relocation of C itself. The following pseudocode describes this top-level procedure, whose time complexity is $O(n^2)$, with n the number of concepts in the resultant merged ontology.

```
Input: Ontology $G_1$ and $G_2$
Output: Merged Ontology $G_2$
begin
    new location of $G_1$'s root = $G_2$'s root
    for each node $C$ (except for the root) in $G_1$
        Parent($C$) = $C$'s parent set in $G_1$
        for each member $p_i$ in Parent($C$)
            $p_j$ = new location of $p_i$ in $G_2$
            relocate($C$, $p_j$)
        end for
    end for
end
```

Top-Level Procedure - $merge(G_1, G_2)$

4.2 Relocate Function

The *relocate* function in the top-level procedure is used to relocate C into a subgraph rooted by p_j. The main idea is: try to find the relationship between C and p_j's direct child(ren) in the following descending priorities: equivalentClass, superClass, and subClass. Because equivalentClass has most significant and accurate information, it is straightforward that equivalentClass has been assigned the highest priority. For superClass and subClass, since we adopt a top-down procedure to relocate concepts, the former has been given a higher priority than the latter. If we cannot find any of these three relationships, the only option is for us to let C be another direct child of p_j.

4.3 Extensions to Puzzle

In this chapter, two extensions have been applied to Puzzle system. The purpose is to increase the performance of this ontology-merging algorithm.

4.3.1 Enriched Contextual Matching

$$relocation\ value = w_{linguistic} \times v_{linguistic} + w_{contextual} \times v_{contextual}. \tag{1}$$

In (Huang et al., 2005), Equation (1) is used to figure out the likelihood of correctly relocating a concept. $v_{contextual}$, calculated based on a concept's contextual feature, considers a concept's properties and its $subClassOf$ relationship. We enrich the contextual matching by including a concept's other relationships, such as $disjointWith$, $partOf$, and $contains$, etc. By taking into account more relationships, we gain more complete semantics of concepts of interest.

4.3.2 Weight Learning through Artificial Neural Networks

In (Huang et al., 2005), different weights for the linguistic matching and the contextual matching, i.e., $w_{linguistic}$ and $w_{contextual}$, are specified by a developer based on trial-and-error. Here we apply an artificial neural network (ANN) technique to learn these weights. For simplicity, we rewrite $w_{linguistic}$ as w_1, $w_{contextual}$ as w_2, $v_{linguistic}$ as v_1, and $v_{contextual}$ as v_2. Our learning problem is designed as follows.

- Task T: match two ontologies
- Performance measure P: *Precision* and *Recall* 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: $\hat{V}(b) = \sum_{i=1}^{2}(w_i v_i)$

In this learning problem, the hypothesis space is a two-dimensional space consisting of w_1 and w_2. For every weight vector $\vec{w}$ in our hypothesis space, our learning objective is to find the vector that best fits 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. The training error E and the weight update rule are given in Equations (2) and (3), respectively.

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

$$\Delta w_i = \eta \sum_{d \in D} [(t_r - o_d) + (t_c - o_d)] v_{id}. \tag{3}$$

D is the set of training examples; o_d is the output of the network for a specific training example d; η is the learning rate; and v_{id} is the v_i value for d. t_r and t_c are explained here. We have a matrix $\mathcal{M}$ recording the *relocation values* for pairwise concepts (one from G_1, the other from G_2). A given pair of manually matched concepts corresponds to a cell $[i,\ j]$ in $\mathcal{M}$, and t_r and t_c are the maximum value for row i and column j, respectively.

5 Compatibility Vector System

We first create a center ontology, then we calculate the concept distance from original ontologies to this center. Finally, compatibility vectors are created and stored in the center. Notice that this whole process is carried out incrementally, along with the joining of partners into an EBC.

5.1 Center Ontology and Concept Distance

5.1.1 Formation of a Center

As mentioned before, the center is generated by merging all original ontologies, step by step, as each new partner joins an EBC. At the beginning, when there is only one partner, its ontology is regarded as the center. With new partners join the community, the new ontologies are merged with the current center. The resultant merged ontology is the newly obtained center.

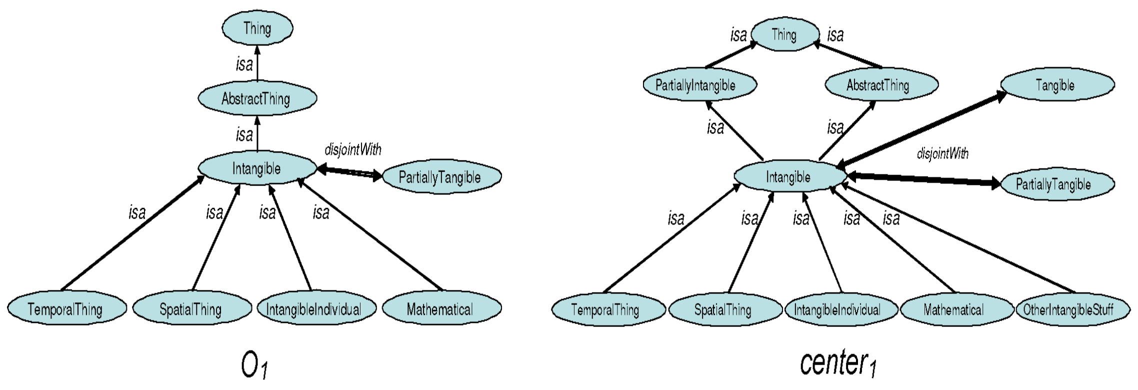

Figure 1: Graphical Representations for O_1 and $center_1$

5.1.2 Concept Distance Calculation

Being the result of merging original ontologies, the center contains information from all sources. With respect to whether or not a specific original ontology, O_i, understands each concept in the center, there are two situations. The first one is that for one specific concept in the center, O_i can understand it, but possibly with less accurate and/or complete information. The second situation is that O_i is not able to recognize that concept at all. In either case, the concept distance is represented by the amount of information missing, i.e., the number of relationships not known in O_i. Equation (4) formalizes the concept distance d.

$$d = \sum_{i=1}^{2}(w_i n_i), \tag{4}$$

where n_1 is the number of $sub/superClassOf$ relationships not known in O_i, n_2 is the number of other relationships not known; w_i's are corresponding weights, and $w_1 + w_2 = 1$. Notice that w_i's could be specified by users, or learned through an ANN similar to the one in Section 4.3.2.

Consider the ontologies in Figure 1. In O_1 (left part), concept “Intangible” has one $subClassOf$ (“AbstractThing”); four $superClassOf$ (“TemporalThing”, “SpatialThing”, “Mathematical”, and “IntangibleIndividual”); and one $disjointWith$ (“PartiallyTangible”). On the right part, merged $center_1$ (center is built incrementally, therefore, we have different $center_i$'s), the concept “Intangible” has more information from other ontologies: one more $subClassOf$ (“PartiallyIntangible”); one more $disjointWith$ (“Tangible”); and one more $superClassOf$ (“OtherIntangibleStuff”). Thus, the concept distance from “Intangible” in O_1 to “Intangible” in $center_1$ is $w_1 \times 2 + w_2 \times 1$. Notice that one “isa” link in Figure 1 corresponds to a pair of relationships. That is, if C_1 “isa” C_2, then C_1 has a $subClassOf$ relationship with C_2, and C_2 has a $superClassOf$ relationship with C_1. Also notice that Equation (4) is suitable for both situations, i.e., independent of whether or not the original ontology recognizes that concept. For example, if in O_1 there is no concept “Intangible”, then the distance becomes $w_1 \times 7 + w_2 \times 2$.

5.2 Compatibility Vectors

Inside the center, there is a set of compatibility vectors, one for each original ontology. A compatibility vector consists of a set of dimensions, each corresponding to one concept in the center. Therefore, all compatibility vectors have identical number of dimensions, i.e., equaling to the number of the concepts in the center. Each dimension has three sub-dimensions. The first sub-dimension tells us whether or not the original ontology understands this concept; the second sub-dimension records the concept name in the original ontology if the latter does recognize that concept; the third sub-dimension encodes the distance from the concept of the original ontology to the concept of the center. An example of compatibility vectors is shown in Figure 2.

For the first concept (“Spatial”) in the center, $Agent_1$ knows it as “Spatial” and has a concept distance of 2.7; $Agent_3$ also understands this concept, but with a different name (“Space”) and a bigger concept

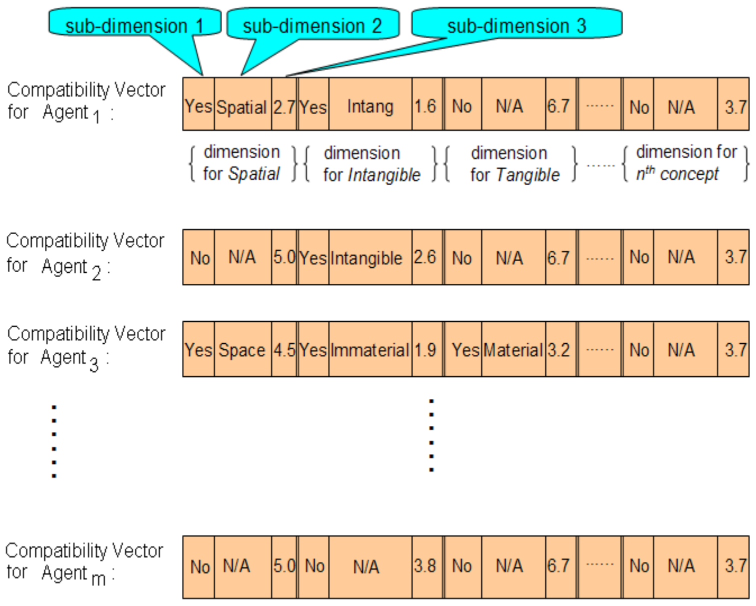
sub-dimension 1
sub-dimension 2
sub-dimension 3
Compatibility Vector for Agent 1 :
Yes Spatial 2.7 Yes Intang 1.6 No N/A 6.7 ...... No N/A 3.7
dimension for Spatial
dimension for Intangible
dimension for Tangible
dimension for nth concept
Compatibility Vector for Agent 2 :
No N/A 5.0 Yes Intangible 2.6 No N/A 6.7 ...... No N/A 3.7
Compatibility Vector for Agent 3 :
Yes Space 4.5 Yes Immaterial 1.9 Yes Material 3.2 ...... No N/A 3.7
Compatibility Vector for Agent m :
No N/A 5.0 No N/A 3.8 No N/A 6.7 ...... No N/A 3.7

Figure 2: Compatibility Vectors

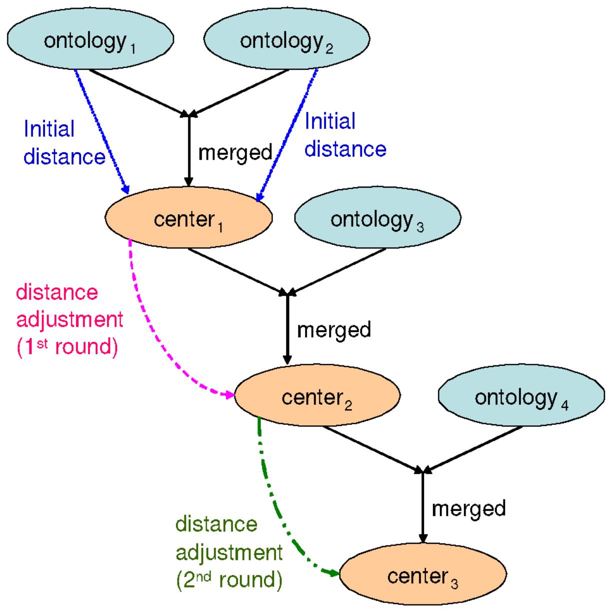

Figure 3: Dynamic Adjustment of Compatibility Vectors

distance of 4.5; neither $Agent_2$ nor $Agent_m$ recognizes concept “Spatial”, therefore, they have the same concept distance (5.0).

5.3 Dynamically Adjusting Vectors

As mentioned before, when there is only one partner, its compatibility is perfect. In the compatibility vectors stored in the center, each concept distance has a value of zero. However, with the adding of new partners into this EBC, the compatibilities for existing partners might be changed, because newly joined partners could contain ontologies with more accurate and/or complete information.

An example is shown in Figure 3, demonstrating the process of dynamic distance adjustment. After $ontology_1$ and $ontology_2$ are merged to generate $center_1$, the distance between these two original ontologies and the merged one ($center_1$) is calculated and stored in the compatibility vectors of $center_1$. Upon the joining of $ontology_3$ and the generation of $center_2$, the compatibility vector for $center_1$ in $center_2$ is calculated and integrated with the compatibility vectors for $ontology_1$ and $ontology_2$ in $center_1$; then we generate the compatibility vectors for $ontology_1$ and $ontology_2$ in $center_2$. This is explained in detail next.

For example, we have compatibility vectors in both $center_1$ and $center_2$. Now we want to update the compatibility vectors in $center_2$. Originally there are two compatibility vectors in $center_2$: one for $ontology_3$, and the other for $center_1$. The former will remain the same as is; while the latter will be replaced by several new vectors, the number of which is determined by the number of the vectors in $center_1$ (two in our example).

Remember that $center_1$ has one vector for each agent when $center_1$ is generated. Each vector in $center_1$ will be integrated with the vector for $center_1$ in $center_2$, therefore creating a new vector correspondingly in $center_2$. The following procedure describes the generation of such a new vector.

Input:

- compatibility vector v for $center_1$ in $center_2$
- compatibility vector u for $partner_i$ in $center_1$

Output:

- compatibility vector w for $partner_i$ in $center_2$

```
begin
    for each dimension $d$ in $v$
        yn = $d$'s first sub-dimension's value
        nm = $d$'s second sub-dimension's value
        dis = $d$'s third sub-dimension's value
        create a new dimension $nd$ in $w$
        if yn = “Yes”
            find in $u$ the dimension $od$ for concept nm
            yn_old = $od$'s first sub-dimension's value
            nm_old = $od$'s second sub-dimension's value
            dis_old = $od$'s third sub-dimension's value
            $nd$'s first sub-dimension = yn_old
```

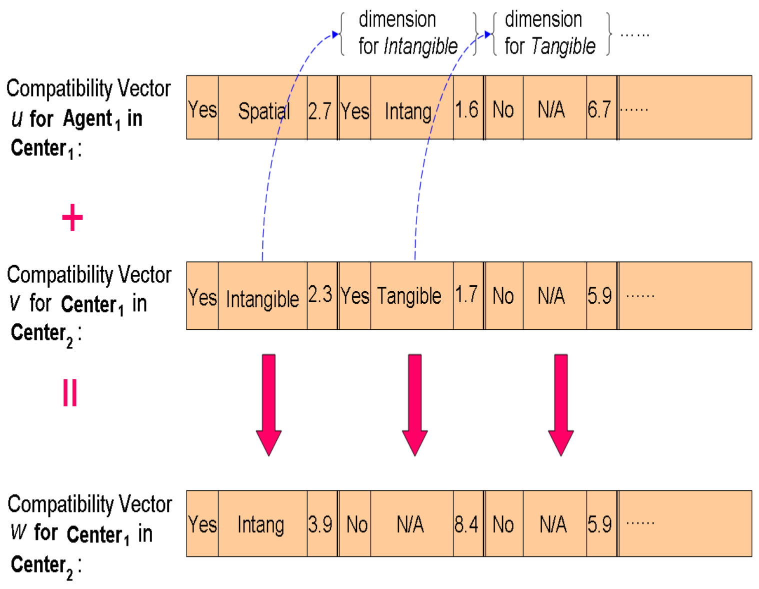
dimension for *Intangible*
dimension for *Tangible*
......
Compatibility Vector *U* for Agent₁ in Center₁:
Yes
Spatial
2.7
Yes
Intang
1.6
No
N/A
6.7
......
+
Compatibility Vector *V* for Center₁ in Center₂:
Yes
Intangible
2.3
Yes
Tangible
1.7
No
N/A
5.9
......
=
Compatibility Vector *W* for Center₁ in Center₂:
Yes
Intang
3.9
No
N/A
8.4
No
N/A
5.9
......

Figure 4: Example of New Vector Generation

```
            $nd$'s second sub-dimension = nm_old
            $nd$'s third sub-dimension = dis + dis_old
        else (yn = "No")
            $nd$'s first sub-dimension = yn
            $nd$'s second sub-dimension = nm
            $nd$'s third sub-dimension = dis
        end if
    end for
end
```

Pseudocode for New Vector Generation

It is not difficult to figure out that the time complexity for the above procedure is $O(nlogn)$, because there are n dimensions in each vector, requiring n steps for the loop. Within each loop, all steps take constant time, except for the one finding dimension in u. Suppose in u the dimensions are indexed by the concept names, then a binary search is able to locate a specific dimension within $O(logn)$.

Figure 4 exemplifies how the above pseudocode works. There are two source vectors, u and v, and we traverse the second one, one dimension each time.

1. The values for the first dimension are "Yes", "Intangible", and "2.3". We then find the dimension for "Intangible" in u, and obtain ("Yes", "Intang", and "1.6"). Finally we calculate the values for the new dimension in the resultant vector w, which are "Yes", "Intang", and "3.9" (the result of 1.6 + 2.3).

2. The values for the second dimension are "Yes", "Tangible", and "1.7". After we obtain the values for dimension "Tangible" in u ("No", "N/A", and "6.7"), we figure out the values for the new dimension in w are "No", "N/A", and "8.4" (the result of 6.7 + 1.7).

3. The values for the third dimension are “No”, “N/A”, and “5.9”. We simply copy these three values into the new dimension in w.
4. This continues until we finish the traverse of all dimensions in v.

5.4 Utilities of Compatibility Vectors

5.4.1 Ontology Understanding within the EBC

The center maintains the compatibility vectors for all original ontologies; in addition, the vectors themselves contain such information as whether or not an original ontology understands a specific concept, what is the concept name in the original ontology, and so on. Therefore, if two partners would like to try to understand each other’s ontology, they can simply refer to the center and obtain the corresponding compatibility vectors. By this means, compatibility vectors help partners in their mutual understanding of ontological concepts.

5.4.2 Partner Selection from Outside the EBC

When a partner from outside this EBC requests for partner(s) to coordinate with, it would like to choose those that understand its ontology best. The requesting partner first compares its own ontology with the center, and then searches in the compatibility vectors to find all partners understanding the concept of its interest. If there is more than one candidate, the coordination request will be sent to those with good compatibilities, that is, with low concept distances. Because the compatibility vectors are stored and maintained by the center, the partners have no way to modify or manipulate the vectors. In this sense, the selection of partner(s) is objective and without bias.

5.5 Features of Compatibility Vectors

5.5.1 Correctness of Compatibility Vectors—A Precise Approach

In this section, we prove that our approach obtains correct compatibilities for partners. To record and maintain the proper compatibility of each partner inside an EBC, the key is to obtain a correct center by which to evaluate the distance from it to each original ontology, and thereby acquire the corresponding compatibility vector. When a new partner joins the EBC, instead of communicating with each existing partner, it only talks with the center. Therefore, if we can prove that the newly merged ontology is a correct new center, the correctness of compatibility vectors is guaranteed.

First, we point out that according to the merging algorithm in Section 4, each time we merge two ontologies, the resultant one will contain all information from both original ones. Next, we introduce *Lemma* 1 and *Theorem* 1.

Lemma 1. When we merge two ontologies A and B using the algorithm in Section 4, the result is the same regardless of whether we merge A into B or merge B into A.

Proof by induction:

1. Base Case: Both A and B contain two concepts, i.e., besides one common built-in root, “Thing”, A contains C_1 and B contains C_2.
   If we merge A into B according to the Top-Level Merging Procedure in Section 4, “Thing” in A is considered equivalent with “Thing” in B; then C_1 is compared with all the direct children of the root in B, in this case C_2, to determine where to put C_1 in B. This is based on the relocate function inside the Top-Level Merging Procedure. On the contrary, if we merge B into A, “Thing” in B is considered equivalent with “Thing” in A; then C_2 is compared with C_1 to determine where to put C_2 in A. Obviously, we obtain the same merged ontology in both cases.

2. Induction: Assume that Lemma 1 holds for all cases where the numbers of concepts contained in A and B are less than (i+1) and (j+1), respectively. Now consider the case where A and B contain (i+1) and (j+1) concepts, respectively.
   Suppose the superClass set of the $(i+1)^{th}$ concept in A, C_{i+1}, is $P_A(C_{i+1})$, and suppose the location

of P$_A(C_{i+1})$ in merged ontology M is P$_M(C_{i+1})$. The position of C_{i+1} in M is determined by the relationships between C_{i+1} and all the direct children of P$_M(C_{i+1})$. From the inductive hypothesis we know that P$_M(C_{i+1})$ is identical no matter whether we merge A into B or merge B into A. Therefore, the position of C_{i+1} in M will also be the same in both situations. That is, C_{i+1} , the (i+1)th concept in A, will be put into the same position in M in both merging orders. Similarly, the (j+1)th concept in B will also be put into the same position in M in both merging orders. So in the case where A and B contain (i+1) and (j+1) concepts, respectively, we still have the same resultant ontology regardless of the merging order taken.

Theorem 1. The final result of merging a number of ontologies is identical no matter by which order the original ontologies are merged using the algorithm in Section 4.

Proof by induction:

1. Base Case: There are two ontologies to be merged.
   According to Lemma 1, when we merge two ontologies A and B, the result is the same no matter whether we merge A into B, or merge B into A.

2. Induction: Assume that Theorem 1 holds for all cases where the number of ontologies to be merged is less than (n+1). Now consider the case where we merge (n+1) ontologies.
   Let the indexes of these ontologies be: 1, 2, ..., (n+1). Consider two arbitrary orders by which we merge these (n+1) ontologies: $order_1$ and $order_2$. Suppose the last indexes in $order_1$ and $order_2$ are i and j, respectively.

   - If i equals j, then the first n indexes in $order_1$ and $order_2$ are the same, just in different orders. We merge the first n ontologies to get Merged$_n$. According to the inductive hypothesis, Merged$_n$ in $order_1$ is identical with Merged$_n$ in $order_2$. Then we merge Merged$_n$ with the last ontology in both $order_1$ and $order_2$, and we will get the same result.
   - If i does not equal j, we mutate the first n indexes in $order_1$ and make the nth index be j; then mutate the first n indexes in $order_2$ and make the nth index be i. Now the first (n-1) indexes in $order_1$ and $order_2$ are in common (possibly in different orders), and the last two are (j, i) and (i, j), respectively. Notice that this kind of mutation will not affect the merging result of the first n ontologies according to our inductive hypothesis. We then merge the first (n-1) ontologies to get Merged$_{n-1}$. According to the hypothesis, Merged$_{n-1}$ in $order_1$ is identical with Merged$_{n-1}$ in $order_2$. Finally we merge Merged$_{n-1}$ with the last two ontologies in both $order_1$ and $order_2$, and we will get the same result.

5.5.2 Complexity of Compatibility Vectors—An Efficient Approach

- The time complexity of establishing an EBC, along with the achievement of a mutual understanding of ontological concepts, is on the order of $O(mn^2)$, with n the number of the concepts in the center, and m the number of original ontologies. The process of creating an EBC is the one to generate a merged center. For the ontology merging, $O(mn^2)$ is needed, because we need to merge m ontologies, and each merging procedure takes time $O(n^2)$ as described in Section 4.

- In order to dynamically update the compatibility vectors during the formation of an EBC, extra time will be spent. According to the previous analysis in Section 5.3, $O(nlogn)$ is needed for updating one partner, so the extra time for all partners is $O(mnlogn)$. Therefore, the total time complexity of establishing an EBC becomes $O(mn^2 + mnlogn)$, which is still on the order of $O(mn^2)$.

- For partner selection, the time complexity is $O(n^2)$, because we only need to compare the ontology from the requesting partner with the center.

5.6 Comparison to Other Vector-Based Approaches

Vector-based approach is widely adopted in ontology matching and text classification area. Some well known systems are summarized in the following.

In (Soh, 2002) and (Soh, 2003), description vectors are introduced and adopted in the conceptual learning during ontology matching. A description vector consists of a list of word-frequency pairs. For each word found in all the experience cases describing the same concept, the agent manages to find the word's frequency, and therefore learns the different significance of the words that describe an ontology concept. The vector fields are then fed into an inductive learner that parses the input vectors into a decision tree, which deterministically allocates each example into a semantically unique branch. Finally, these branches are traversed to arrive at a set of rules.

(Lacher and Groh, 2001) emploies machine learning techniques for text categorization. Their methodology is: to calculate a representative feature vector for each concept node in an ontology; then to measure similarity of two of those class vectors by a simple cosine measure. The representative feature vector for one concept node is calculated as a modified Rocchio centroid vector. By this means, the representative vector for a concept node represents an average of all documents assigned to that concept node. The feature vectors are exctracted from the documents and weighted. That is, a word-count feature vector is created and the features are weighted with a TF/IDF weighting scheme (Term Frequency/InverseDocument Frequency). In essence, vectors are computed from the instance data.

(Williams, 2004) tackles the issue of sharing meaning in a multiagent system through ontology learning. The author describes how agents learn representations of their own ontologies using a machine learning algorithm and then seek to locate and/or translate semantic concepts by using examples of their concepts to query each other. In this paper, a semantic concept comprises a group of semantic objects that define each token, i.e., word and HTML tag from the Web page, as a boolean feature. The entire collection of Web pages that were categorized by a user's bookmark hierarchy is tokenized to find a vocabulary of unique tokens. This vocabulary is then used to represent a Web page by a vector of ones and zeroes corresponding to the presence or absence of a token in a Web page.

Our compatibility vector system is quite different from the systems mentioned above.

- Our system is based on ontology schema (structure) alone, aiming to avoid the difficulty in getting enough/good quality instance data from real-world ontologies. Ontology schemas usually have a lot more varieties than instance data, we are therefore dealing with a more challenging problem than those algorithms that make use of instance as well.
- Our focus is on the comparison between original ontologies and the merged center ontology. By considering both concept names and concept relationships, we aim to include more complete semantics for concepts of interest.
- The information contained in our vector system is in great details. This will facilitate the mutual understanding among agents, by simply referring to the center ontology for the corresponding vectors.

6 Experiment Results

6.1 Test Ontologies

We take ten **real-world** ontologies, created and maintained by professionals, as our test ontologies:

1. **terror:** http://www.mindswap.org/2003/owl/swint/terrorism
2. **travel:** http://opales.ina.fr/public/eon2003/Travel-OilEdExportRDFS.rdfs
3. **tour:** http://homepages.cwi.nl/∼troncy/DOE/eon2003/Tourism-OilEdExportRDFS.rdfs
4. **space:** http://212.119.9.180/Ontologies/0.3/space.owl
5. **priv:** http://www.daml.org/services/owl-s/security/privacy.owl
6. **ops:** http://moguntia.ucd.ie/owl/Operations.owl
7. **obj:** http://www.flacp.fujitsulabs.com/tce/ontologies/2004/03/object.owl

Table 1: Characteristics of Test Ontologies

Features	**terror**	**travel**	**tour**	**space**	**priv**	**ops**	**obj**	**swap**	**mgm**	**gfo**
Max Depth of Ontology	5	7	6	8	5	8	8	7	9	11
Number of Concepts	27	51	53	90	26	91	38	61	72	127
Number of Relationships	41	47	48	158	38	139	70	87	109	162
super/subClassOf Relationships	29	36	33	115	31	110	57	64	75	117
Percentage of *super/subClassOf*	70%	77%	68%	73%	81%	79%	82%	73%	69%	72%

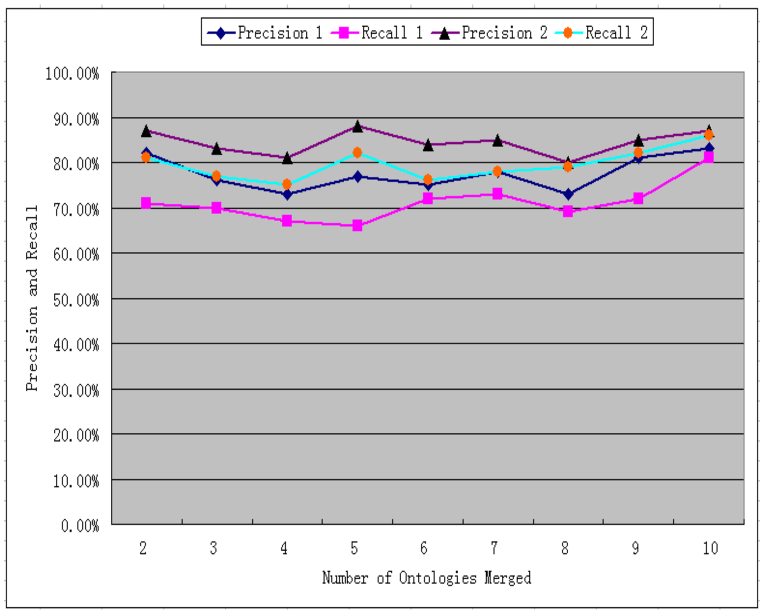

Figure 5: Ontology Merging Results

8. **swap:** http://svn.mindswap.org/pychinko/pychinko/allogtests/mindswapRealized.rdf
9. **mgm:** http://ontologies.isx.com/onts/2005/02/isxbusinessmgmtont.owl
10. **gfo:** http://www.onto-med.de/ontologies/gfo.owl

These ten ontologies are all in "Business" domain, specified by OWL. Their characteristics are summarized in Table 1.

6.2 Experiments on Merging Algorithm

We randomly pick up an order to merge these ten ontologies, and we compare the results from our merging algorithm with those from a manual matching by three ontology experts. We then evaluate on both *Precision* and *Recall* measures, and plot the results in Figure 5. "Precision 1" and "Recall 1" are for the original algorithm in (Huang et al., 2005), they range from 73% to 83%, and 66% to 81%, respectively. "Precision 2" and "Recall 2" are for the extended version in this chapter, range from 80% to 88%, and 75% to 86%, respectively. After the extensions discussed in Section 4.3, *Precision* and *Recall* have an average increase of 6.9% and 8.3%, respectively.

6.3 Experiments on Compatibility Vectors

We first fix one original ontology (randomly chosen) as the one from the coordination-requesting partner, and then simulate an EBC out of the remaining nine ontologies. In our first setting, the requesting ontology interacts with a randomly chosen ontology; while in our second setting, this interaction always happens with the ontology with the best compatibility (according to concept distances calculated). We switch the fixed

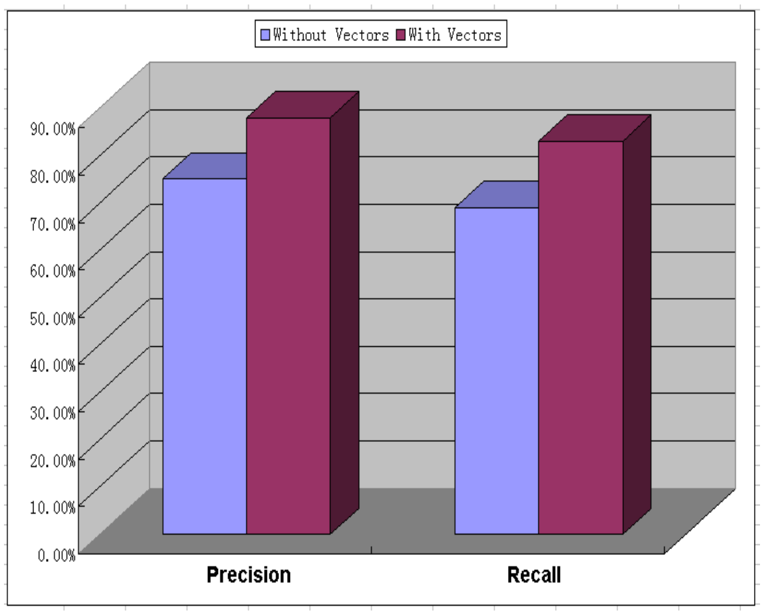

Figure 6: Utility of Vectors

ontology from the first one to the tenth one, calculate the average values of *Precision* and *Recall* for all settings, and then plot the results in Figure 6. It is clear that, after adopting our compatibility vectors, both measures have been improved. Therefore, in cases where sufficient resources are not available and only a certain number of partners can be chosen for coordination, our approach increases the efficiency by choosing suitable partners.

7 Conclusion

E-business is having a significant impact on the evolution of the Internet. In order to meet the new demands of highly dynamic environments and open e-markets, there has been a growing need for e-businesses to coordinate their activities. The first step towards this coordination is for e-business partners to understand each other's service description. Although using ontologies can aid in this understanding, independently designed ontologies usually have heterogeneous semantics, resulting in each partner having its own unique semantics. To tackle this emerging challenge, we present an ontology compatibility vector system as a method to evaluate and maintain ontology compatibilities, thereby handling the problem of how to choose partners with good compatibilities. We not only prove that our approach is both precise and efficient, but also show promising results experimentally.

Some future work is envisioned here: (1) how to handle the vulnerability issue inherent in the centralized solution that our current approach uses, (2) how to update compatibility vectors when existing partners modify their corresponding ontologies, (3) what kind of mechanism is suitable if we simultaneously consider qualities of both ontologies and services, and (4) we plan to design a GUI for our algorithm, and we believe that such a front-end would make it easier for a practitioner to use.

References

Bilgin, A. and Singh, M. (San Diego, CA, July 2004). A daml-based repository for qos-aware semantic web service selection. In *Proceedings of IEEE International Conference on Web Services (ICWS 04).*

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

Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., and Halevy, A. (Springer-Verlag, New York, NY, USA, 2003). Learning to match ontologies on the semantic web. *The VLDB Journal*, 12(4):303–319.

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

Honavar, V., Andorf, C., Caragea, D., Silvescu, A., Reinoso-Castillo, J., and Dobbs, D. (Seattle, WA, August 2001). Ontology-driven information extraction and knowledge acquisition from heterogeneous, distributed biological data sources. In *Proceedings of the IJCAI-2001 Workshop on Knowledge Discovery from Heterogeneous, Distributed, Autonomous, Dynamic Data and Knowledge Sources*.

Huang, J., Zavala, R., Mendoza, B., and Huhns, M. (Berlin, Germany, September 2005). Ontology reconciling agent ontologies for web service applications. In *Proceedings of Multiagent System Technologies: Third German Conference (MATES 05)*.

Kalepu, S., Krishnaswamy, S., and Loke, S. (San Diego, CA, July 2004). Reputation = f(user ranking, compliance, verity). In *Proceedings of IEEE International Conference on Web Services (ICWS 04)*.

Lacher, M. and Groh, G. (Key West, FL, May 2001). Facilitating the exchange of explicit knowledge through ontology mappings. In *Proceedings of 14th international FLAIRS conference*.

Madhavan, J., Bernstein, P., and Rahm, E. (Roma, Italy, 2001). Generic schema matching with cupid. In *Proceedings of the Twenty-seventh VLDB Conference*.

Melnik, S., Garcia-Molina, H., and Rahm, E. (San Jose, CA, 2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the Eighteenth International Conference on Data Engineering (ICDE 02)*.

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

Singh, M. and Huhns, M., editors (2005). *Service-Oriented Computing - Semantics, Processes, Agents, 1st edn.* Wiley, Chichester, England Press.

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

Soh, L.-K. (Melbourne, Australia, 2003). Collaborative understanding of distributed ontologies in a multiagent framework: Design and experiments. In *Proceedings of the Third International Workshop on Ontologies in Agent Systems (OAS 03)*.

Song, J., Zhang, W., Xiao, W., Li, G., and Xu, Z. (Hong Kong, China, March 2005). Ontology-based information retrieval model for the semantic web. In *Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*.

Tijerino, Y., Embley, D., Lonsdale, D., Ding, Y., and Nagy, G. (2005). Towards ontology generation from tables. *World Wide Web: Internet and Web Information Systems*, 8(3):261–285.

Williams, A. (Kluwer Academic Publishers, The Netherlands, 2004). Learning to share meaning in a multiagent system. *Autonomous Agents and Multi-Agent Systems*, 8(2):165–193.

Zhou, C., Chia, L., and Lee, B. (San Diego, CA, July 2004). Daml-qos ontology for web services. In *Proceedings of IEEE International Conference on Web Services (ICWS 04)*.