A Cooperation-Based Approach For Evolution Of Service Ontologies *

Murat Şensoy and Pınar Yolum
Department of Computer Engineering
Boğaziçi University
Bebek, 34342, Istanbul,Turkey

ABSTRACT

Communication among agents requires a common vocabulary to facilitate successful information exchange. One way to achieve this is to assume the existence of a common ontology among communicating agents. However, this is a strong assumption, because agents may experience situations that result in independent evolution of their ontologies. When this is the case, agents need to form common grounds to enable communication. Accordingly, this paper proposes an approach in which agents can add new service concepts into their service ontologies and teach others services from their ontologies by exchanging service descriptions. This leads to a society of agents with different but overlapping ontologies where mutually accepted services emerge based on agents' exchange of service descriptions. Our simulations of societies show that allowing cooperative evolution of local service ontologies facilitates better representation of agents' needs. Further, through cooperation, not only more useful services emerge over time, but also ontologies of agents having similar service needs become aligned gradually.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Design, Experimentation

Keywords

Service Ontology, Semantics, E-Commerce

1. INTRODUCTION

Automatic service selection is the process of identifying an appropriate service provider for a requested service. Recent approaches to service selection advocate a social approach in which consumer agents interact with other consumer agents to identify useful service providers for their needs. The consumer interactions may consist of exchanging referrals [15] or experiences about providers [4]. While these approaches are successful in identifying service providers, they assume common service semantics. That is, these approaches assume that the agents share a service representation, such as an ontology, through which the agents can represent their service needs in the best way possible. This assumption cannot account for the fact that an agent's needs may evolve over time and new service concepts may be necessary for it to describe its evolving service needs. However, a service selection approach should be able to accommodate this, since in many e-commerce settings, individuals learn new service concepts from different sources, add them to their ontology, and further form service requests that are based on these new concepts.

The need for new services stems from several facts. One, the consumer may be in need of composed services. Individual services may be expressible by the common ontology, but if the consumer always requests multiple services together, it may prefer to express its service need compositely. Example 1 demonstrates such a composite service need. Two, the service needs of the consumer may evolve in time. Thus, the consumer may actually construct a new service description that does not exist before and start demanding this service. Example 2 demonstrates such a service need.

EXAMPLE 1. *A consumer wants to purchase a book and wants it to be delivered to an address. Assume that there is no service concept composed of both purchasing and delivering, so the consumer creates two service instances: one for purchasing and one for delivering. Then, it collects information from other agents related to these service needs. Using the collected information, it discovers that* $ProviderA$ *sells books and* $ProviderB$ *makes deliveries. Since the consumer requests both services, it is in need of a provider that does both services for itself. However, since its ontology does not have a specific concept to represent the composed service, the consumer is left to make two individual service requests.*

EXAMPLE 2. *Assume that a concept like valet parking is not known by any of the consumers in the society. A busy consumer may not want to waste her time searching for an appropriate place to park her car. Instead, she wants a new type of parking service in which an attendant parks her car on the behalf of her. The consumer has formulated a new service to express her parking needs.*

The above examples show how an agent may become aware of a new service need and why it would prefer to add it into its ontology. However, addition of a new service concept into the local ontology of a consumer does not only affect the agent's local ontology but also it gradually affects local ontologies and preferences of other agents that it interacts with.

The need for individual ontologies to evolve on their own brings in a major problem of ontology alignment. If the agents do not

*This research has been partially supported by Boğaziçi University Research Fund under grant BAP07A102 and The Scientific and Technological Research Council of Turkey by a CAREER Award under grant 105E073. This paper extends a poster at AAMAS07.

Cite as: A Cooperation-Based Approach For Evolution Of Service Ontologies, Murat Şensoy and Pınar Yolum, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. 837-844

exchange information about their ontologies frequently, they can end up with disjoint ontologies, which will obstruct their communication. Hence, if agents are allowed to add new service concepts into their ontologies, they should also be offered a mechanism with which they can notify others about their ontology changes. It is up to other agents to decide whether these changes are of interest to them or not. That is, no agent needs to keep track of all changes in others' ontologies. By cooperatively exchanging information about their service ontologies, consumers can build on the service concepts evolved by others, rather than reinventing the same services individually.

Accordingly, this paper proposes a distributed approach for the creating and dissemination of new service concepts and evolution of service ontologies in the context of service selection. If an agent cannot formulate its service need using the existing service concepts in its ontology, it first queries other agents to find a suitable service concept that corresponds to its current service need. If such a service concept does not exist, the agent can generate a new service concept to express its service need and inserts the concept into its service ontology. While using the new service concept when interacting with others, the agent first checks if the correspondent agent is aware of the concept that is used in the interaction. If the correspondent agent does not know the concept, then the agent sends a description of the concept to the correspondent agent. The correspondent agent can then add the concept into its ontology using this description and if it finds the service concept useful, it can use it in its forthcoming interactions with others. This way, the semantics of new service concepts circulate and get established in the agent society. With this approach, we investigate the following questions:

- How well can the consumers' service ontologies evolve so that the service needs of the consumers can be represented as concisely and accurately as possible?
- How much of the ontology evolutions are due to individual efforts and how much are generated by cooperation?
- How much of the learned services are useful for the consumers and how much of the useful services do they learn?

The rest of this paper is organized as follows. In the next section, we explain our insight of service ontologies. Then, we present our approach to enable consumers to agree upon the services and we propose an algorithm to add new services to ontologies. Lastly, we experimentally evaluate our approach and discuss our work with references to the literature.

2. DESCRIBING SERVICES

We envision a multiagent system that consists of consumer agents representing human users. Each consumer agent has access to a common meta-ontology that contains primitive concepts and properties. This ontology is static and does not contain any service concept. This ontology is public; i.e., may be downloaded from a well-defined resource. It constitutes a grounding for describing service concepts and sharing these description between the agents. A proper subset of WordNet's OWL [13] representation [12] can be selected and modified to be used as the common meta-ontology.

Each agent has a local service ontology. This ontology contains the service concepts known by the agent. Each service concept has a description. This description is made using only the concepts and the relations from the common meta-ontology. Therefore, each agent can interpret this description correctly and understand what the service does. Description of a service concept contains the properties of the service. Each property of the service is defined as a combination of an *OWL Property* and its range or value and denoted in the rest of the paper as *(OWLProperty;RangeOrValue)*. A service concept gets its semantic meaning from its properties and from nothing else. That is, semantic meaning of a concept is not related to the syntax or lexical properties of its name. This is intuitive, because services are already defined fully by their properties in real life. That is, usually properties of a service uniquely identify it. Example 3 illustrates the description of services from their properties. As a characteristic of taxonomies in ontologies, a service concept inherits the semantic meaning of its ancestors in the service ontology.

EXAMPLE 3. *Assume that the parking service already exists in the service ontology and its description has only one property;* $(hasAction; Park)$. *This means that any service that contains* $(hasAction; Park)$ *in its description is also a parking service. The valet parking service is described using only two properties;* $(hasAction; Park)$ *and* $(hasActor; Attendant)$. *The second property contains a restriction on the range of the OWL property hasActor. This restriction states that actor of the parking action must be an attendant, and not the owner of the car. If another service concept is described using exactly the same properties with the valet parking service, one can easily infer that this service is equivalent to "valet parking", even if it has another name.*

DEFINITION 1. *Let* S_A *and* S_B *be two service concepts with different names. These services are semantically equivalent if their sets of properties are equal.* ∎

DEFINITION 2. *Let* S_A *and* S_B *be two semantically different service concepts.* S_B *is a specialization of* S_A *if it has every property that* S_A *has.* ∎

While interpreting service descriptions, an agent uses ontological reasoning. For example, assume that *hotel valet parking* service is described using two properties; $(hasAction; Park)$ and *(hasActor;HotelAttendant)*. Again, assume that $HotelAttendant$ is a sub-concept of $Attendant$ in the common meta-ontology. Therefore, the agent can easily infer a new property *(hasActor;Attendant)* from the property *(hasActor;HotelAttendant)*. This implies that the description of *hotel valet parking* actually contains three properties; *(hasAction;Park), (hasActor;HotelAttendant)* and implicitly *(hasActor;Attendant)*. Therefore, this service is a specialization (subconcept) of *valet parking* service, because it has the every property of valet parking.

Each agent has a unique identifier such as a URI and a unique name-space. For example, the agent of *John Doe* has a unique identifier *http://agent.johndoe* and its name-space contains every name starting with this URI. When the agent creates a new service concept such as *Valet Parking*, it gives a name to this service concept within its name-space such as *http://agent.johndoe/valetparking*. This way, name conflicts between the service concepts that are created by different agents are prevented.

3. SIMILARITY BETWEEN SERVICES

Two service concepts can be compared to each other in terms of semantic similarity [7]. Several metrics for the semantic similarity exist. We start with Tversky's similarity metric [11]. It considers the common and different properties of two concepts in the computation of similarity and assumes that two concepts are similar as much as they have common properties. However, we think that each property does not have the same importance in the computation of similarity. Therefore, we apply a weighting scheme to the

properties of the concepts. In this work, similarity between two concepts a and b is computed using Equation 1.

$$Sim(a,b) = \frac{\sum_{p \in U(a) \cap U(b)} w_p^a}{\sum_{p \in U(a)} w_p^a + \sum_{p \in U(b) - U(a)} w_p^a} \quad (1)$$

In Equation 1, $U(a)$ is a set that contains the properties of service a including the ones inherited from its ancestor services and w_p^a is the importance of the property p for the service a. The importance of a property is determined by its level of inheritance. That is, a property is more important if it is inherited from the higher levels of the hierarchy. The intuition behind this assumption is as follows. In an ontology, if two concepts are different in terms of their most general properties (those that are inherited from higher levels in the hierarchy), these concepts will be apart in the ontology hierarchy. However, if they are different only in their most specific properties (those that are not inherited), they will be close in the ontology hierarchy. For instance, sibling concepts are different in their most specific properties. Therefore, importance of the property p for the service a (namely w_p^a) is computed as $1 + \sum_{c \in P(a)} f(c,p)$, where $P(a)$ is the set of all ancestors of a in the service ontology, and $f(c,p) = 1$ if the service concept c has the property p ($p \in U(c)$); otherwise $f(c,p) = 0$. This means that the weight of the property p will be bigger if more ancestors of the service a has it.

4. INTERACTIONS OF CONSUMERS

When a consumer agent has a service need, it seeks for information related to this need. This information can be ratings of service providers regarding this service need, referrals or identifiers of the other service consumers having the same or similar service needs and so on. Here we assume that the consumers are looking for the identifiers of other service consumers that have had the same or similar service needs in the past. When the agent gets hold of this information, it can contact the consumers to discuss potential service providers.

In order to get information related to its current service interest, the consumer interacts with its *neighbors* (see Definition 3). To selects its neighbors to contact, each consumer models other consumers in the society by keeping track of their service interests from previous interactions. To do this, each consumer keeps a table called *Service Interest Table* for maintaining the list of known service concepts and the identifiers of the consumers who have used each of these concepts in their interaction with the consumer. When the consumer needs information related to a specific service, it finds table entries belonging to the most similar service concepts in its service interest table. Similarity between service concepts is computed using their descriptions as explained in Section 3. The consumers that are referenced in these entries are selected as the neighbors of the consumer. Then, the consumer interacts with those neighbors. During these interactions, the consumer should properly express its current service need to the other party in order to get the most related information.

DEFINITION 3. *Let an agent **A** be interested in a service concept **S**. Neighbors of **A** with respect to **S** are defined as those agents that are also interested in **S** or another service type similar to **S**.* ▌

4.1 Emergence of New Service Concepts

If a consumer encounters difficulties in representing its current service need using the service concepts in its service ontology, a new service concept referring to the service need is required. This new concept may either already exist in the society but the agent may not be aware of it or the concept may be totally new to the entire society. In order to differentiate between these two cases, the consumer first creates a description of its desired service concept and assigns a unique name to it using its name-space. Example 4 demonstrates a simple case.

EXAMPLE 4. *John Doe wants to use valet parking service in a hotel during his travel. Therefore, his agent looks for the hotels that provide a valet parking service. Assume that the agent does not have a valet parking concept in its ontology. Hence, it cannot express its service need directly. As a result, it creates a description for its desired service. In this case, this description is composed of two properties;* ***(hasAction;Park)*** *and* ***(hasActor;HotelAttendant)****. Then, it chooses a name for the desired service concept; e.g.,* ***http://agent.johndoe/valetparkinginhotel****.*

After describing the desired service concept, the consumer sends a *Service Inquiry Message* to its neighbors to find out if the service concept is already known to its neighbors. This message contains the description of the desired service and the name assigned by the consumer. Upon receiving the service inquiry message, a neighbor inspects the service concepts in its ontology to find a semantic match with the desired service concept and informs the requesting consumer if there is a match or not. If a semantic match is found, then the neighbor sends the name of the matched concept in its ontology to the requesting consumer. Therefore, the consumer can add the desired service concept into its ontology with this name as described in Section 5. This way, the consumer and its neighbor address this service concept with the same name in their ontologies.

If the consumer receives different names from its neighbors for the same service inquiry message, it notes that these names are synonyms, because they refer to the same service concept. If none of the neighbors locates the desired service concept within their service ontologies, the consumer concludes that this service concept is not known by any of its neighbors. In this case, the consumer places the concept into its local ontology with the name that is stated in the service inquiry message. Note that concept names are created within the name-spaces of the consumers. That is, it is not possible for two consumers to create two different service concepts and give the same name to them. Therefore, each concept name is unique and associated with only one service concept in the agent society. By giving unique names to the new service concepts, we remove the probability of name conflicts.

At the end of the procedure above, the consumer adds the new service concept into its service ontology. During its cooperation with its neighbors, the consumer gathers important information about the service concept. The consumer shares the gathered information with its neighbors by sending a *Service Consolidation Message*. This message contains the description of the service concept, and its name in the consumer's ontology. It also contains the identifiers of the neighbors who already know the service concept and the names of this service concept within their ontologies (referred as "synonyms"). When a neighbor receives a service consolidation message, it adds the described service concept into its ontology with the referred name if its ontology does not contain the service concept yet. Furthermore, the neighbor stores the referred synonyms to remember how the same service concept is addressed by others. Therefore, the consumer and the neighbors can understand each other during their future communications regarding this service concept.

In many approaches to ontology evolution, agents do not cooperate while creating and adding new concepts into their ontologies. Therefore, they independently add the semantically equivalent concepts with different names into their ontologies. This results in the requirement of finding mappings between the concepts in different

ontologies for communicating properly. However, in the proposed approach, when a consumer generates a new service concept to represent its new service needs, it teaches this service concept to its neighbors by sharing the description of the concept or the neighbors inform the consumer about the service concept if the concept is already known by them. This leads to an interactive learning of new services. Hence, mutually understood service concepts emerge as a result of consumers' social interactions.

Moreover, this approach leads to cooperative evolution of service ontologies. When a consumer learns a useful concept from its neighbors, it can directly use it or create another concept that builds on the learned concept to describe its service needs better. Hence, more accurate concepts that describe the service needs are cooperatively and iteratively created.

4.2 Discovering Others

The approach explained above depends on the social interactions of a consumer with other consumers who have similar service interests. In this approach, when it needs to learn a new service concept, the consumer communicates with others that have used a similar service concept in their interactions. For example, if the consumer needs a new service that is a type of valet parking offered by hotels, it communicates with the consumers who are interested in valet parking services or hotel services. These consumers are determined using the service interest table of the consumer. In many cases, the agent may need to expand this table by discovering new consumers with a specific service interest. In this section, we propose a simple P2P search mechanism for this purpose.

In order to get the identifiers of the consumers with a specific service interest, the consumer generates a *Search Message*. This message contains the identifier of the message originator, the name of the service concept that represents the service interest, the desired number of search results that should be returned by the receiver and lastly a time-to-live (TTL) value to define how long the message should be forwarded. Then, using its service interest table, the consumer chooses a subset of its neighbors to whom the search message will be sent as explained in Section 4.1. Figure 1 shows a search message. In this message, the message originator states that it is looking for five consumers who are interested in *HotelValetParking* or a similar service. In the figures and tables throughout this paper, we omit the name-space prefixes for the service concept names (e.g., http://agent.johndoe/), because of space limitations.

When another consumer receives this message, it checks whether the service concept in the message is known or not. If this concept is not known by the receiver yet, it requests for the description of the service concept from the message originator and adds the concept into its service ontology as explained in Section 5. The receiving consumer processes the message as follows. First, it computes the similarity of the service concept in the message to the service concepts in its service interest table. Then, it sends the table entries belonging to the most similar concepts along with the concept names to the originator of the search message and the message originator updates its service interest table using these entries. The receiver decides the number of entries to be sent using the number defined in the search message. The receiver also updates its service interest table by adding the name of the message originator to the related service concept entry. As a result, the receiver remembers the service interest of the message originator and uses this information in the future. If TTL value of the message is greater than one, the receiver decrements the TTL value of the message and forwards the message to its neighbors that are most related to the service concept in the message.

Using these simple interactions, consumers learn service interests of each other, update their service interest tables, and use these tables in their future interactions. During the interactions of the consumers, if unknown service concepts are encountered by a consumer, the consumer requests for the description of these concepts from the consumers that have used these concepts in their interactions. Then, these concepts are added to the ontology of the consumer using the procedure in Section 5.

Received Search Message	
Originator	agent.johndoe
Service Concept	HoteValetParking
DesiredResults	5
TTL	3

Similarity to HotelValetParking	
Parking	0.4286
ValetParking	0.7143
HoteValetParking	1.0
HotelService	0.5714

Service Interest Table	
ConceptName	**Consumers**
Parking	Cons0;Cons3;Cons10;Cons12
ValetParking	Cons0;Cons16
HoteValetParking	Cons11;Cons23;Cons34
HotelService	Cons11,Cons20

Returned Search Results	
ConceptName	**Consumers**
HoteValetParking	Cons11;Cons23;Cons34
ValetParking	Cons0;Cons16

Figure 1: An example search message, service interest table of the receiver, the computed similarities, and the returned search results.

5. UPDATING SERVICE ONTOLOGIES

A new service concept is added into a service ontology using its description and ontological reasoning. In a service ontology, each service concept is described in terms of its properties. An example service ontology is shown in Figure 2. Each service concept inherits the properties of its ancestors. A property implies another one if it is a specialization of the latter. For example, in Figure 2, *HotelValetParking* concept inherits *(hasActor;Attendant)* from *ValetParking* and *(hasActor;HotelAttendant)* from *HotelService*. The latter property implies the former, because *HotelAttendant* is defined as a sub-concept of *Attendant* in the shared meta-ontology. Therefore, *HotelValetParking* can be described using only two properties; *(hasActor;HotelAttendant)* and *(hasAction;Park)*.

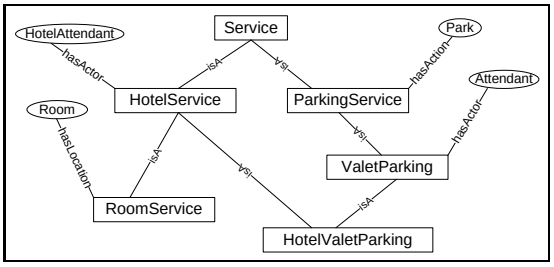

Figure 2: Service ontology of a neighbor.

In this paper, we assume that service concepts are precisely described by their properties. Hence, if the properties of two service concepts are equivalent, then they are semantically equivalent. Consider Example 4. The agent in the example describes its desired service concept using two properties; *(hasActor;HotelAttendant)* and *(hasAction;Park)*. When the neighbor referred in Figure 2 gets this description, it can trivially match the desired concept with *HotelValetParking* concept in its ontology.

An agent learns new service concepts using their descriptions. When the agent receives a description of a service concept S_{new}, it searches for a semantic match in its ontology using the approach above. If the agent could not find a semantically equal service concept in its ontology, then S_{new} should be added as a new concept into the service ontology of the agent. For this purpose, the agent

first defines the parent(s) of S_{new} in its ontology. A service concept S_P in the agent's ontology is a parent of S_{new}, if S_{new} is a specialization of S_P, but not that of S_P's any subconcept. As stated before in Section 2, S_{new} is a specialization of S_P if every property of S_P can be inferred from the properties of S_{new}. Example 5 shows how parents of a new service concept are determined.

EXAMPLE 5. *Agent of **John Doe** generates a new service concept with the name **http://agent.johndoe/restaurantvaletparking** and wants to teach this concept to the agent referred in Figure 2. It defines this concept using two properties; **(hasAction;Park)** and **(hasActor;RestaurantAttendant)**. Assume that **RestaurantAttendant** is a sub-concept of **Attendant** in the shared meta-ontology. When the agent referred in Figure 2 gets this description, it can conclude that this service is a specialization of **Service**, **ParkingService**, and **ValetParking** concepts. However, only the most specific one, namely **ValetParking**, is the parent of the new service concept. As a result, the agent adds the new concept into its ontology as a sub-concept of **ValetParking** with the referred name.*

After defining the parents of S_{new}, it is placed into the ontology as a sub-concept of these parents. Some of the properties of S_{new} are also owned by its parents. Therefore, the new concept inherits these properties from its parents. However, S_{new} may have some other properties that are not owned by its parents in the ontology. These properties are directly added to the new concept in the ontology. Example 6 shows how a new service concept is placed into the ontology.

EXAMPLE 6. *The agent in Example 5 adds the new service concept as a subconcept of **ValetParking** and uses the referred name **http://agent.johndoe/restaurantvaletparking** as its name. This new service concept inherits **(hasAction;Park)** from its ancestors, but not **(hasActor;RestaurantAttendant)**. So this property is attached to the new service concept in the ontology.*

Lastly, we may need to modify some of the existing parent-child relationships of S_P, which is a parent of S_{new}. Let S_X be a subconcept of S_P. If S_X is also a specialization of S_{new}, by definition S_P cannot be the parent of S_X any more. Instead, S_{new} should be the new parent of S_X in the ontology. Therefore, we modify the parent-child relationships accordingly.

Consider an example scenario in which the neighbor referred in Figure 2 teaches service concepts from its ontology to an agent. This agent has only one service concept in its ontology. This concept is named *Service* and does not have any property. Figure 3 shows the initial service ontology of the agent and its change after the addition of *HotelValetParking*, *ValetParking* and *HotelService* concepts, respectively.

When the agent receives the description of *HotelValetParking* from the neighbor, it adds *HotelValetParking* as a subconcept of *Service*. Note that any service concept is trivially a specialization of *Service*, because *Service* does not have any property. In this example, the new service concept does not inherit any of its properties from its parent in the ontology, so all of its properties are directly attached to it as shown in the second sub-figure of Figure 3.

When the agent receives the description of *ValetParking* from the neighbor, it determines that *Service* is the parent of this new service concept. Hence, *ValetParking* is also added into the ontology as a subconcept of *Service*. Description of *ValetParking* contains two properties; *(hasAction;Park)* and *(hasActor;Attendant)* . These properties can be inferred from the properties of *HotelValetParking*. Therefore, *HotelValetParking* is set as a sub-concept of *ValetParking*. Note that, now *HotelValetParking* inherits *(hasAction;Park)* from its parent (see the third sub-figure of Figure 3).

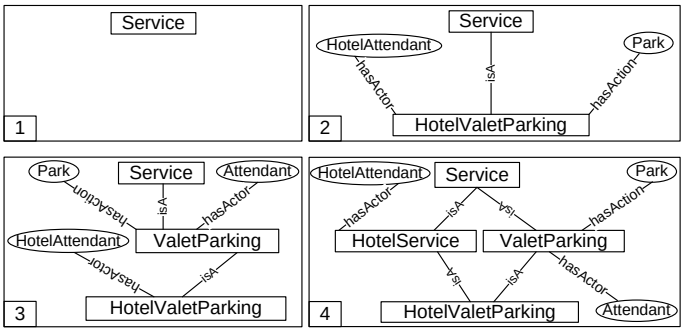

Figure 3: Service ontology of the agent; (1) initial, (2) after adding *HotelValetParking*, (3) after adding *ValetParking*, (4) After adding *HotelService* concept.

Lastly, the agent receives the description of *HotelService* from the neighbor. It has only one property; *(hasActor;HotelAttendant)*. This service concept is also added into the ontology as a subconcept of *Service*. In this time, previously added service concept *HotelValetParking* is its specialization. Therefore, *HotelValetParking* is set as a sub-concept of *HotelService* as shown in the fourth subfigure of Figure 3. Notice that now *HotelValetParking* has two parents and inherits all of its properties from those parents.

6. EXPERIMENTAL RESULTS

We have conducted simulations to evaluate our approach. In the simulations, 100 consumer agents are used. In the beginning of the simulations, each consumer agent is given identities of three randomly chosen consumers to overcome bootstrapping. Initial interactions are done with these agents. Throughout the simulations, service inquiry or search messages are sent to at most two neighbors and TTL values of the search messages are set to 1, because of the small size of the simulated agent society.

Consumer agents are implemented in Java and JENA is used as OWL reasoner [8]. Simulations are run on a computer with a 1.66 GHz Intel Core Duo CPU and 1.0 GB RAM under Windows OS. Next, we describe the simulation environment and the results of our simulations. Simulations are repeated 10 times to increase the reliability and their averages are presented.

6.1 Simulation Environment

For the simulations, we select *Food and Beverage Services* domain. In real-life, different restaurants have different service offerings, which are usually called *Food Menus*. Each service concept in this domain is described by a list of foods and beverages that are served or delivered to a consumer together for a meal. For example, *KFC Hot Wings Menu* is an instance of *Chicken Wing Menu* concept that contains a number of fried chicken wings, a bunch of fried potatoes, and a cup of drink.

To facilitate meaningful generation of service needs, we have designed roles for agents. These roles are similar to the real life roles such as student, parent, and so on. These roles define the behaviors of consumers by specifying their service needs and characteristics. For example, for dinner, the consumers playing *vegetarian* role usually demand a cup of vegetable soup, some pasta or rice, and a salad. In our simulations, there are 10 distinct roles and each agent is assigned exactly one role. Each agent has a personality ontology that contains information about the role that the agent plays. The consumer agent can reason about its service needs using this

ontology and shape its behaviors and preferences appropriately. In other words, the roles enable generation of service needs during the simulation. Roles of the consumers are set at the beginning and properties of the roles do not change during a simulation. This means that consumers playing the same role continuously have the same *service needs* during the simulations.

For the simulations, we extended W3C's food ontology [13] and we use the resulting ontology as the shared meta-ontology. This ontology has more than 200 concepts, 1020 individuals and 60 properties. In the beginning of simulations, each consumer agent has the same service ontology. This ontology contains only 10 shared service concepts such as pizza service, salad service and so on. Each of these service concepts contains only one type of food. For example, an instance of pizza service contains an instance of pizza and nothing else. However, each service need that is imposed by the roles is composed of several food types. Hence, none of these shared service concepts is enough to represent a service need on its own. Therefore, in the beginning of the simulations, a consumer having a service need must request several services together to satisfy its service need. For example, in the beginning, the consumer playing *vegetarian* role must demand several services together such as soup service, pasta or rice service and salad service, because there is not a service concept that contains a soup, rice or pasta, and a salad together. This may result in problems in real-life. For example, you want to have some soup and pasta. However, there is not a food service that contains soup and pasta together. Instead, there is a soup service offered by a restaurant and a pasta service that is offered by another restaurant. Hence, you have to make two different orders from two different restaurants. If the pasta arrives much earlier than the soup, you should either eat pasta before the soup or wait for the soup and let the pasta get cold.

Table 1 shows the service concepts and the roles that use these concepts in the beginning of the simulations. During the simulation, each consumer tries to express its service need as concisely as possible (i.e., using a single concept). To do so, at each epoch, the consumer tries to create a new service concept with a small probability (0.1). A new service concept is created by either combining two existing service concepts as in Example 1 or adding a new property to an existing service concept as in Example 2. If the service concept to be created is learned from others, the consumer uses the learned service concept rather than recreating it.

Table 1: Service concepts and roles using them in the 0^{th} epoch.

Service Concept Name	Roles Using The Service Concept
SoupService	Role5, Role4, Role2, Role0
DessertService	Role8, Role7
AlcoholicBeverageService	Role8, Role5, Role4, Role0
NonAlcoholicBeverageService	Role9, Role8, Role7, Role6
PizzaService	Role7, Role6, Role2, Role0
PastaService	Role9, Role6, Role2
RiceService	Role4, Role3, Role2, Role1
SaladService	Role5, Role4, Role3, Role0
SeaFoodService	Role6, Role5, Role4, Role2, Role1
MeatService	Role8, Role6, Role5, Role3, Role1, Role0

6.2 Simulation Results

With the above setup, we first investigate whether the consumers have actually been successful in formulating their service needs concisely as desired. Table 1 show the initial configuration of the environment, where consumers use around four service concepts to describe their service needs. However, in our simulations, this number sharply decreases over time and on the average it becomes 1 in the 15^{th} epoch. That is, after this point, new service concepts emerge so that every consumer can and does use only one service concept to describe its service need. The first plot in Figure 4 shows that starting from the $15th$ epoch, the ratio of new service concepts used by a consumer reaches 1; meaning that any service concept used by a consumer to describe its service need is a new service concept (i.e., did not exist in the starting service ontology).

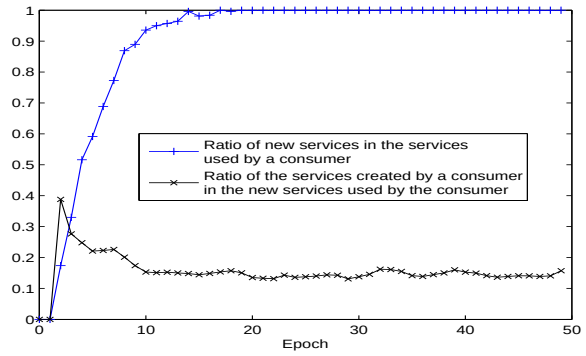

Figure 4: Average ratio of adoption and creation of new service concepts by the consumers.

For example, the consumers playing $Role7$ cooperatively create a new service concept before 15^{th} epoch and use only the instances of this new service concept to describe their needs thereafter. This new service concept contains a dessert, an alcohol-free beverage, and a pizza. This confirms that our approach can enable consumers to create new and more expressive service descriptions to satisfy their service needs.

Next, we study whether consumers create accurate service concepts based only on their experiences or whether they have benefited from interactions with others. This is a crucial point, since the proposed approach is meant to help agents evolve their ontologies *cooperatively*. The second plot in Figure 4 shows the ratio of the service concepts created by a consumer to the new service concepts used by the consumer. In the beginning, this ratio is 0.4, which means that on the average 40% of the new service concepts used by a consumer are created by the consumer. On the other hand, this ratio rapidly decreases and approaches 0.1, which means that only 10% of the new service concepts used by a consumer is created by the consumer. The remaining service concepts are created by other consumers and adopted by this consumer. This proves that there is a powerful cooperation between the consumers. They teach each other new service concepts and furthermore the taught service concepts are adopted and used to describe service needs of the consumers in the society. As stated before, this situation leads to the cooperative creation of new service concepts, because new service concepts are created using the learned ones.

Another interesting question is how well each agent becomes aware of the service concepts created by other agents. To study this point, we make a distinction between *unique* and *replica* service concepts. A service concept is unique at the time it is created if nobody in the system has created another service concept with the same meaning. A service concept is a replica if an equivalent service concept has been previously created by another agent in the system before (but not known by the creator of the service concept).

Figure 5 shows plots related to this question. The first plot is the number of new service concepts created through the simulations. After the 15^{th} epoch, there are 92 new service concepts in the environment. This number includes the unique service concepts as

well as the replicas. The second plot, on the other hand, shows only the number of unique service concepts, which is equal to 75. Put together, these numbers reveal that 82% of the new concepts in the environment are unique and only 18% of them are replicas. This result confirms that consumers become aware of the useful service concepts instead of reinventing them by creating replicas.

Next, we study how much of the service concepts are learned by an agent on the average and whether the agent misses out important service concepts in the society. The third plot in Figure 5 is the average number of unique service concepts known by a consumer. Consumers do not have the knowledge of whether a service concept is unique or replica, because each consumer has a limited knowledge of the environment. If the consumer knows two service concepts with the same meaning, it considers the one that is learned earlier as unique and the other service as replica (synonym). The dashed line in the figure shows that the average number of unique service concepts known by a consumer stabilizes at 11 after the 15^{th} epoch. This shows that the consumers know only a small portion of the service concepts in the environment.

The last result shown in the figure is the average number of known unique service concepts that are useful for the consumer. In this case, we count the unique service concepts known by a consumer only if these service concepts are useful for the consumer in defining its service needs. For example, for the consumers playing $Role7$, a service concept containing only a pizza and an alcohol-free beverage is useful, so we count this service concept for the consumers playing $Role7$. However, we do not count this service concept for the consumers playing $Role2$, because these consumers do not demand alcohol-free beverages with pizza (see Table 1). Figure shows that the number of useful unique service concepts for a consumer is around 8 out of the 11 unique service concepts it knows. These two observations show that (1) consumers learn a small proportion of the service concepts in the environment, and (2) this small proportion is enough for agents to represent their service needs. Hence, they learn only enough to express their needs, and no more.

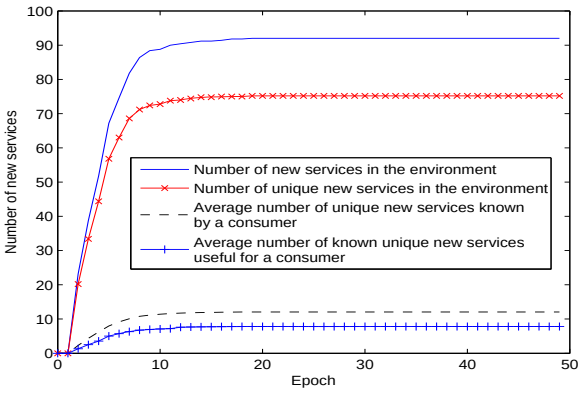

Figure 5: Number of new service concepts, unique new service concepts, known service concepts, and useful known service concepts for a consumer.

A concrete conclusion of Figure 5 is that most of the service concepts learned by consumers are useful. An immediate question is how much of the useful and frequently used service concepts are known by a consumer. This is important to find out since knowing useful service concepts that are not used in communication is rarely of value. Figure 6 plots the average ratio of *useful* unique concepts known by a consumer to the *useful* unique concepts in the environment. To calculate this, we weight every service concept in the environment by considering whether the service concept is useful for describing the consumer's service needs as well as how frequent the service concept is used. The frequency of service concepts is calculated by counting the number of occurrences of the service concepts in communications of all the consumers. As Figure 6 shows the ratio of known useful service concepts dramatically increases and approaches 0.9, which means that each consumer knows most of the useful unique service concepts, which are frequently used by the consumers in the society.

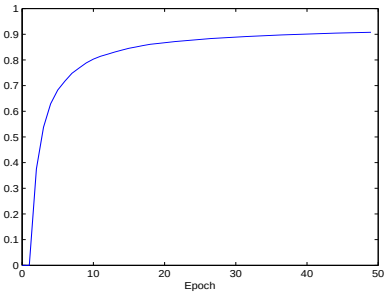

Figure 6: Average ratio of known useful service concepts.

7. DISCUSSION

This paper proposes a novel approach for cooperative evolution of service ontologies. Our simulations show that consumer ontologies evolve considerably so that at the end of the simulations each consumer is able to represent its service need as concisely as possible. The evolutions are not only due to individual effort but mostly a result of cooperation. That is, most of the service concepts used by each consumer are devised by others in the society. Furthermore, the consumers learn a small portion of the service concepts that emerge in the society, making sure that most of the learned service concepts are useful for them in representing their service needs. Finally, consumers learn most of the useful, frequently used service concepts in the society. Thus, through cooperation, agents can have different but evolving ontologies, yet they can communicate with those that are similar to them to represent their needs.

7.1 Extensions To The Proposed Approach

Main contribution of our research is the techniques that are proposed in Section 4 for the cooperative evolution of service ontologies in a P2P setting. In this paper, we assume that service concepts can be described by the agents using a *shared meta-ontology* that is composed of fundamental concepts and properties. This ontology should carefully be designed by domain experts so that it can be used to describe any new service concept in the domain. This may require considerable amount of human effort and may not be possible in some settings. The proposed techniques in this paper can also be used together with other concept description models. However, each concept description model has its own advantages and disadvantages over the description model that is used in this paper. For example, in the concept learning literature, a concept is described using its instances (positive examples) and its non-instances (negative examples) [9, 2]. Although this concept description model does not requite a shared meta-ontology, it requires instances to be shared among agents (e.g., through a public directory). We can easily integrate this example-based description model into the proposed framework.

For example, in Web Services domain, we can describe *Book Selling* service concept using its positive and negative examples. In

this case, UDDI [3] entries related to book selling concept (e.g., Amazon book selling service) can be used as the positive examples and UDDI entries that are not related to book selling (e.g., New York Times newspaper selling service) can be used as the negative examples. When an agent gets this description, it can learn the *Book Selling* concept using the examples in the description as explained in [9, 2]. Furthermore, it can compute the semantic similarity or equivalence between two service concepts using these examples and machine learning techniques as described by Doan *et. al.* [5]. Doan *et. al.* also offer a method for testing subsumption between two concepts using examples. This is equivalent to testing whether a concept is a specialization of another concept or not. Therefore, a service concept can easily be placed into an ontology using its example-based description, using a method similar to the method in Section 5. Using the approaches above, we can easily replace our concept description model with the example-based concept description model. Similarly, we can switch between different concept description models to use our framework in a broader range of real-life applications.

7.2 Related Work

The necessity for individual ontologies to evolve on their own results in a major problem of ontology alignment. There are substantial amount of research related to ontology alignment and reconciliation in the literature [10, 14, 2, 1, 5]. Similarity-based approaches for ontology alignment are powerful and flexible enough for aligning ontologies expressed in languages like OWL. In these approaches, similarity between the concepts from two different ontologies is computed and the concepts that are similar to each other are mapped to align these two ontologies. Some other approaches use syntactic or lexical properties of concepts' names in addition to semantic similarity metrics. They use string matching algorithms or lexical databases like WordNet. We believe that semantic similarity is much more important than syntactic similarity, because similarity or dissimilarity between the names of concepts may be highly misleading.

Afsharchi *et. al.* [2] use an instance based approach for learning concepts. In that setting, when an agent confronts an unknown concept, it chooses teacher agents among its neighborhood and these teachers teach the concept to the agent by providing positive and negative examples of the concept. Then, the agent uses a machine learning approach to learn the properties of the new concept. This approach is similar to the approach of Sen and Kar [9] in the sense that agents teach each other concepts by providing examples. This approach is not practical for cases where there is not a sufficient number of instances shared by the agents.

Williams [14] proposes a methodology and algorithms for improving the mutual understanding of two agents. In this approach, agents develop a common feature description of a particular concept using knowledge sharing and machine learning techniques in a peer-to-peer setting. So, they gradually arrive at consensus on the concepts and they develop mappings between the concepts in their ontologies. However, Williams' approach does not support cooperative ontology evolution.

Aberer *et. al.* [1] propose an approach for the global semantic agreements. They assume that mappings between two different ontologies are already made by skilled human experts. These mappings are exchanged by the agents and global semantic agreements are reached using the properties of the exchanged mappings. Laera *et. al.* [6] use argumentation over concept mappings to reach global semantic agreements. Similar to Aberer *et. al.*, in Laera *et. al.*'s approach, mappings between concepts are assumed to be made beforehand by a mapping engine. Each mapping has a confidence value for different agents. By using argumentation theory together with these mappings, an agreement over heterogeneous ontologies are reached dynamically.

Our work distinguishes from the literature in several ways. In the other approaches, ontologies of agents evolve independently. Each agent creates new concepts on its own and adds them into its ontology. This results in highly different ontologies. Then, the mentioned approaches are used to align these ontologies. In our approach, ontologies evolve cooperatively. That is, a consumer learns new service concepts from its neighbors and furthermore creates new ones using the learned service concepts. As a result, not only more useful service concepts emerge over time, but also service ontologies of the consumers having similar service needs become aligned over time. Additionally, our approach has a proactive nature. In our approach, a consumer prohibits its future communication problems by informing its neighbors about the created service concepts before using them. As future work, we plan to study the evolution of service ontologies when business constraints are added; e.g., in the context of service composition.

8. REFERENCES

[1] K. Aberer, P. Cudre-Mauroux, and M. Hauswirth. Start making sense: The chatty web approach for global semantic agreements. *Journal of Web Semantics*, 1(1):89–114, 2003.

[2] M. Afsharchi, B. Far, and J. Denzinger. Ontology guided learning to improve communication among groups of agents. In *Proceedings of AAMAS'06*, pages 923–930, 2006.

[3] L. Clement, A. Hately, C. von Riegen, and T. Rogers. UDDI version 3.0.2, October 2004.

[4] M. Şensoy and P. Yolum. A context-aware approach for service selection using ontologies. In *Proceedings of AAMAS'06*, pages 931–938, 2006.

[5] A. Doan, J. Madhaven, R. Dhamankar, P. Domingos, and A. Helevy. Learning to match ontologies on the semantic web. *VLDB Journal*, pages 303–319, 2003.

[6] L. Laera, I. Blacoe, V. Tamma, T. Payne, J. Euzenat, and T. Bench-Capon. Argumentation over ontology correspondences in MAS. In *Proceedings of AAMAS'07*, pages 1285–1292, 2007.

[7] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304, 1998.

[8] B. McBride. Jena: A semantic Web toolkit. *IEEE Internet Computing*, 6(6):55–59, 2002.

[9] S. Sen and P. Kar. Sharing a concept. In *Working Notes of the AAAI-02 Spring Symposium*, pages 55–60, 2002.

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

[11] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

[12] M. van Assem, A. Gangemi, and G. Schreiber. RDF/OWL representation of WordNet, June 2006.

[13] W3C. OWL: Web ontology language guide, February 2004.

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

[15] P. Yolum and M. P. Singh. Engineering self-organizing referral networks for trustworthy service selection. *IEEE Transactions on Systems, Man, and Cybernetics*, A35(3):396–407, 2005.