Automated Ontology Evolution in a Multi-Agent System

Mohsen Afsharchi, Behrouz H. Far
Department of Electrical & Computer Engineering
University of Calgary
Calgary, Canada
{mafsharc, far}@ucalgary.ca

Abstract

This research addresses the formation of new concepts and their corresponding ontology in a multi-agent system where individual autonomous agents try to learn new concepts by consulting several other agents. In this research individual agents create and learn their distinct conceptualization and rather than a commitment to a common ontology they use their own ontologies. In this paper multi-agent supervised learning of concepts among individual agents with diverse conceptualization and different ontologies is introduced and demonstrated through an intuitive example in which supervisors are other agents rather than a human.

1. Introduction

Over the past decade, we have seen a growing interest in Knowledge Management (KM) solutions for intellectual assets in organizations including techniques and tools for document management, workflow management, transparent capture, web conferencing, visual thinking, digital whiteboards, as well as data warehouses, decision support systems, groupwares and intranets. While most of the enabling technologies for KM have been around for several years, the ability to seamlessly integrate the components into a cohesive infrastructure evaded organizations until the advent of the Internet and semantic awareness technologies. Open Internet standard and Semantic Web technology present unprecedented opportunities for implementing knowledge management solutions, and its evolution provides a ubiquitous binding medium for integrating distributed applications, formats, and contents.

The conventional KM solutions are based on centralized architecture commonly comprised of a central knowledge repository accessible through formalized queries. Such architecture cannot be used effectively in heterogeneous computing environments with customized workflows. One of the recent attempts is to distribute the data repositories and enable communication and message passing at the data level by encapsulating the heterogeneity of legacy systems and applications within standard and interoperable wrappers. Interface description language and services, which allow data objects to be defined, located and invoked across locations and applications, are provided for the heterogeneous computing environments [15]. The most popular of such distributed paradigms are OMG's Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM), and Sun's Remote Method Invocation (RMI). These frameworks are defined for and are well suited to the 'data' level communication and sharing as opposed to 'knowledge' level. They assume a relatively stable dynamic environment that sets the common ground for communication, well defined processes and common understanding.

The core requirements for knowledge level KM solutions are [8]: (1) Incorporate the management of knowledge assets, which are distributed and belong to both people and departments; (2) allow for dynamic classification and distribution of knowledge; (3) allow for adapting to diversified contents, representation and personalized styles; (4) incorporate efficient (i.e., fast and effortless) retrieval mechanisms; and (5) facilitate interaction between distributed knowledge bases in order to support social process of knowledge management.

The KM requirements imply that the integration of knowledge and business processes requires a different architecture than the one provided by the centralized or data-level frameworks. The peer-to-peer (P2P) architecture is considered as a candidate solution. P2P technology is viewed as a way of enabling distributed control, differentiation, customization and redundancy. This paradigm represents organizational cognition as distributed processes that balance the autonomous knowledge management of individual and groups, and the coordination needed in order to exchange knowledge across different autonomous entities [5]. Furthermore, in order to have sophisticated knowledge

sharing between individuals and/or groups there must be coordination mechanisms in place. Consequently, organizations and their supporting systems must be able to allow for a degree of negotiation and adaptability in order to accommodate individual participation [15].

Although P2P is a particularly powerful paradigm when it comes to satisfying the basic requirements of knowledge level KM by allowing distribution of knowledge and diversity of contexts, it lacks the capability to account for social skills of the peers. In the current implementations of P2P based systems, peers are treated as simple computational entities with very limited social skills such as search, locate, send and get. However, in knowledge level KM solutions complex social skills such as negotiation, collaboration, coordination, learning and reasoning are indispensable. Intelligent software agents and socialability of multiagent systems (MAS) seem well suited to knowledge level KM solutions. Software agents not only can get involved in complex interactions [17] but also offer implementing sophisticated helper tools such as search, scheduling [18], negotiating [17, 20] and decision support [19, 22], knowledge capture and discovery [24, 21, 23].

In this research we are mainly targeting the multiagent learning of concepts in order to improve the knowledge sharing among agents that is necessary for efficient communication. The ontology as an implicit way of representing an agent's understanding of world should be evolved by learning new concepts and relationships through consulting several other agents. Concepts are the building blocks of the ontology of an agent. We argue that concepts can be learnt in a collaborative manner, in the same way that humans learn from each other. The basic ideas behind our supervised learning method are first to realize that there is a concept that an agent does not currently know but expects to need to know, for example, answering a query requires knowledge of an specific concept. Then the agent (learner) asks the other agents (supervisors) about this concept by providing features and their values or examples the agent thinks are associated with the concept. The queried agents provide the learner agent with positive and negative examples of what constitutes their concepts (i.e. concepts known by them and their properties). This allows the learning agent to utilize one of the known concept learning techniques to actually learn the concept. This can be done iteratively by consulting the supervisor agents again to confirm the learnt concept. To help focus on the positive/negative examples, the supervisor agents make use of selected relations among concepts from their ontologies. Conflict resolution is a key concern. The learner agent deals with the fact that the supervisor agents might not totally agree on which examples fit the concept and which not, by letting the supervisor agents vote on the examples for which it got contradictory information in the first place. Our learning agent also uses information from the supervisor agents about the paths to the concept of interest in their ontologies to prepare its own ontology for potential concepts it might want to learn later on.

2. Preliminary Concepts

In this section we discuss some preliminary concepts which are playing a vital role in our work. This is necessary to clarify the concepts which we intend to use but are poorly defined elsewhere.

2.1. Ontology

Historically, ontology has referred to the philosophical study of being or what exists. In artificial intelligence (AI), ontology deals with categories we can quantify over and how those categories relate to each other. Gruber [25] refers to ontology as a formal explicit specification of a shared conceptualization. "Explicit" means that the type of concepts used and the constraints on their use are explicitly defined. "Formal" refers to the fact that ontology should be machine readable.

Inherent in definitions of ontology is a method for representing and interrelating objects or concepts into a hierarchy of concept types according to different levels of generality. In Semantic network representations, this is called a type hierarchy, taxonomic hierarchy or a subsumption hierarchy. In a frame based representation system this concept hierarchy would be represented as inheritance rules.

These conceptual taxonomies are loosely referred to as ontologies and can be useful for indexing and organizing information and for managing the resolution of conflicting defaults. While hierarchical taxonomies mostly use *is_a* and *instance_of* and *has_a* relation to connect different concept in a domain, conceptual graphs use richer vocabularies to express relation among concepts. In addition to above mentioned relations, conceptual graph enables us to express all kind of relations which is possible in domain though its formalization is a matter of research.

2.2. Agents

An agent is a computing entity with or without a "body" that has varying degrees of the following capabilities or characteristics: autonomy, reasoning, social, learning, communication and mobility [26]. A group of these interacting agents are referred to as a multi-agent system. We assume that agents in our multi-agent system are deployed in a knowledge management environment and use their learning capability to adapt to different kinds of knowledge sources and resolve differences among themselves and provide best answers for queries in the complex social knowledge management environment. Autonomy in the context of this research is as agents being independent from users during the process of concept learning.

2.3. Conceptualization

An agent, whether representing an artificially intelligent computer program or a robot has a view to its world [27]. In artificial intelligence (AI) this view of an agent has been referred to as its *conceptual modeling* or *conceptualization*. The knowledge of an agent has been represented in research through various declaration forms such as predicate logic, production rules, statistical reasoning systems, semantic networks and conceptual graphs.

No matter how we choose to conceptualize the world, it is important to realize that there are other conceptualizations as well. In some cases, changing one's conceptualization of the world can make it impossible to express certain kinds of knowledge. In other cases, changing one's conceptualization can make it more difficult to express knowledge, without necessarily making it impossible. We believe what makes one conceptualization more appropriate than another for knowledge formalization is more or less related to the problem domain. For instance, in text classification domain we usually use some kind of statistical reasoning system (e.g. Naïve Bayes) while in pattern classification in biomedical imaging it is difficult to express knowledge using these systems.

According to [27] declarative knowledge consists of a conceptualization which includes the objects or concepts presumed to exist in the world, their functions and relations and, given a conceptualization, knowledge can be formalized as sentence in a language appropriate to that conceptualization. This kind of conceptualization is very helpful when we have explicit knowledge about domain.

In case of implicit knowledge, conceptualization is also implicit and there is no explicit function and relation representing knowledge. Instead, agents use some intelligent algorithms to conceptualize a set of concepts. Considering text classification domain as an example, there are many examples (e.g. text documents) representing a concept but the knowledge which is the correlation between words in different documents is not explicit. Using some text classification mechanisms (e.g. kNN, Naïve Bayes) agents conceptualize the concepts which are hidden in examples.

3. Methodology

We assume that in a society of n agents Ag_1, …, Ag_n, where each agent managing the knowledge for a department or an organization, each agent has learnt a concept C_k and possesses positive and negative examples regarding the concept and describing features respecting it. Also each agent has utilized a learning algorithm (e.g. naïve Bayes, decision tree, etc.) to conceptualize C_k. In addition to these resources each agent has its unique ontology. Formally, we can represent each agent using four attributes:

$$Ag_i = \{L_i, \xi_i^{C_k}, f_i^{C_k}, O_i\}.$$

We believe that these features can be different from one agent to another. It is not necessary for two agents to have the same set of positive and negative examples for a certain concept or represent a set of concepts utilizing the same features. In the next subsections we discuss the rationale behind the selection of each attributes.

3.1. Learning algorithm (L_i)

We use supervised inductive machine learning algorithms to learn a decision function for a concept C_k. This is the main reason of utilizing a learning algorithm. Also in our framework, a learning agent learns a concept in a supervised manner. This supervision is accomplished by teacher agents. In some cases, when the teacher agent is not expert about queried concept but it has some example regarding it, we utilize the learning algorithm L, as a part of agents' capabilities for supervision. Using L, the teacher agent can answer incoming queries regarding classification of certain examples.

3.2. Set of examples ($\xi_i^{C_k}$)

In our framework, positive and negative examples (we use 'examples' to refer to both) are transferred from teacher to the learner in order to supply the learner with examples of a concept. These examples could be any instances which the teacher agents

consider as examples of the queried concept. For instance, a positive example could be any document supporting a concept, like what we have in text classification domain. Using teacher agents' classification capability, the learner agent can decide if an example is a positive or negative instance of the learning concept, append it to its repository and learn from the set of approved examples.

3.3. Set of features ($f_i^{C_k}$)

To represent a concept, its features play a vital role. We need features to represent concepts, distinguish between them and make them amenable to machine learning algorithms. We assume that each individual agent can represent a concept using different features. We also assume that features are understandable by every agent but they are selectively used by each agent to represent different concepts.

One can claim that by assuming that features are understandable by every agent we substitute common ontology by common feature collection. We should argue that first of all agents must have a base of agreement in the lowest level in order to understand each other and what we are doing in the multi-agent concept learning is to collect the most related features from different source of knowledge to come up with a concrete and comprehensive concept. On the other hand the ontology is not just a set of concepts rather it is a complicated set of concepts and relations among them. Having a different ontology means having different set of features for concepts *and* having different hierarchy of relations for them.

3.4. Ontology (O_i)

We believe that concept learning in a multi-agent system does not make sense when agents use a common shared ontology because in this case concept learning simply can be done by copying a concept structure from one agent's repository to another. The capability of agents in defining their own individual ontology makes multi-agent concept learning meaningful.

Ontology in our framework is a conjunction of meta-concepts and fine grained concept structure. Meta-concepts are concepts which are usually not directly connected to examples. Meta-concepts are usually divided into some sub-concepts until a fine grained concept level is deduced.

Concepts in the ontological hierarchy of our framework are divided into two levels: Well-Understood concepts (WUC) and Vague-Understood concepts (VUC). For each agent, Well-Understood concepts are the concepts which are explicitly known for it and there is a node in their ontology showing that concept. Every agent is expert about its WUCs. Vague-Understood concepts are the concepts that are not explicitly mentioned in the agents' ontology. There are some examples in an agent's repository showing a VUC but the concept is not explicit for the agent.

To explain the Vague-Understood concepts we use the lattice structure which encapsulates examples and their relation with features in an ordered graph. Figure 1 shows an example of an ontology which has both WUCs and VUCs. Here the CS (Computer Science) is a WUC and is part of the agent's ontology, AI (Artificial Intelligence) and SE (Software Engineering) are implicit concepts which are not explicitly located in the agent's ontology but there are some examples with some common features which make them an implicit concept. The rationale behind WUC and VUC is that it is not essential for an agent to have every concept explicitly in its ontology, rather it is possible for any agent to have a set of examples with some common features which might not be explicitly formed as a concept in its conceptual hierarchy.

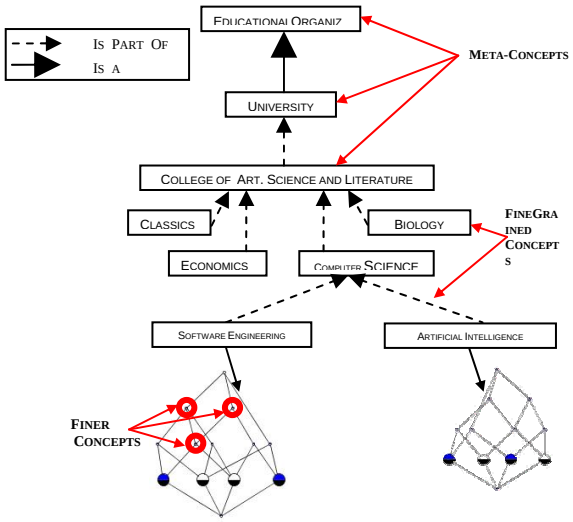

Figure 1. Conceptual hierarchy

In response to a concept query, an agent searches its ontology to find a concept, which is matched by features. In case of success, we consider the agent as expert and it returns the positive and negative examples, which are directly assigned to the WUC. In case of failure, there is no WUC for the queried concept and the agent is not expert about the concept

however the agent starts to make a partial lattice to find a VUC and sends back its comprising examples.

3.5. Concept formation process

Figure 2 shows the concept formation process for a learner agent. For each concept or set of concepts our agents have some objects (i.e., documents) and features (e.g. bag of words) representing them. Using rules and algorithms of Formal Concept Analysis [28], the agent builds a formal context and its corresponding concept lattice. This structure can be gradually improved when new objects and features become available. An important point here is that the formation of formal context is both automated and supervised. This means that higher-level concepts in the concept lattice are generated automatically but can explicitly be labeled to show the name of that concept by the supervisor.

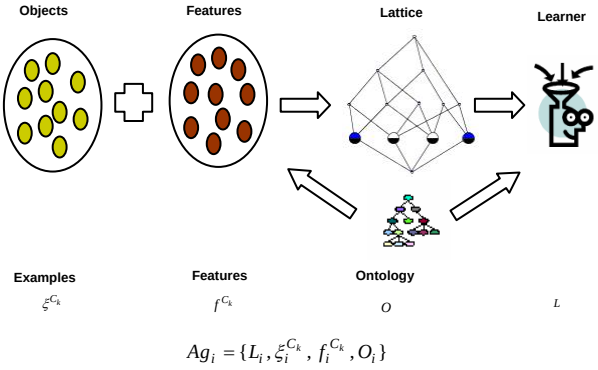

Figure 2. Concept formation process

4. Learning a Concept From Other Agents

The goal of learner agent is to learn a concept from other agents. In order to propose a solution for collaborative concept learning in a multi-agent system we should examine the three important players and mechanisms of this scenario: the learner, the supervisor, and agents' collaboration mechanism. These are explored in the following subsections.

4.1. The learner agent

For the learner agent, there are two major problems. The first problem is how the learner agent figures out that it does not know a concept. The simplest method for an agent to figure out that it does not know a concept is tracking the incoming queries. When a learner agent has failed to answer some queries, it tries to find probable coherence among previous unanswered queries. Using this coherence and the elements of incoming queries the learner agent makes a new query and submits it to teacher agent to find out which agents know about the probable concept.

The second problem is that in order to have ontological consistency, the learner agent should be able to locate a best space in its ontology and place the concept in it. In order to achieve this goal we developed a pre-structuring algorithm to enable the learner agent to locate some shell concepts, which most probably have to learn it in future. The key information used in pre-structuring is the path information, which are sent by the supervisor agents in their answers to the query. The path is a chain of concepts from the root concept to the currently concept which is being learnt. The shell concepts that are created based on these paths can also be used to indicate to an agent, the concepts it might want to learn in the future and potential queries for them. For the normal usage of the ontology, we treat shells as non-existent. Figure 3 shows the detail of our pre-structuring algorithm. We will discuss it more in the context of our experiment.

4.2. The supervisor agents

The question about supervisor agents is how they can help an agent in learning process. To answer this question we should first consider what supervisor agents have. Based on our agent model, supervisor agents possess examples regarding a certain concept and they can also get involved in conflict resolution process. They can support learning agents with positive and negative examples and also they can answer learning agents' questions regarding classification of a certain example. As stated previously, the flexible hierarchical structure of lattice lets supervisor agents to traverse their concept structure from the concept toward its examples and features or to traverse the concept structure from features to concepts. This flexibility can help supervisor agents to answer the learner agent queries based on both concepts and features.

4.3. The agents' collaboration

The following scenario describes the supervisor and learning agent and also agents' collaboration in a simple but intuitive way (see Figure 4). We suppose agents $Ag_1, \ldots, Ag_n$ have learnt C_k and have their own

examples, counter examples and features as well as a learning algorithm (e.g. Naïve Bayes, decision tree, etc.) and ontology.

```
make an array of concepts c_i from each concept chain
set lv_i to the last level of c_i.
set c_m to the array with longest chain
set lv_m to the last level of c_m
j = 0
while lv_m > 0 do
  i = 0
  for all concepts in the lv level do
    sim ← checksimilarity(c_m[lv_m], c_i[lv_i])
    if sim = 1 then
      (*concepts are similar*)
      c_m ← merge(c_m[lv_m], c_i[lv_i])
      updateweights(c_m[lv_m])
      lv_i − = 1
    else if sim = 2 then
      (*concepts are not similar at higher level *)
      levelset ← levelset + c_i[lv_i]
      lv_i − = 1
    end if
    i+ = 1
  end for
  M[j] ← c_m[lv_m]
  M[j] ← M[j]+ mergesimilars(levelset-1)
  lv_m − = 1
  j+ = 1
  levelset ← ∅
end while
```

Figure 3. Pre-structuring algorithm

1. An Ag_l has been queried by some other agents about concept C_k and failed to answer it. Finally it determines that it does not know the concept and should learn it.
2. Agent Ag_l queries supervisor agents for similar concepts by sending them the name of the concept or by sending some features which it believes is related to the concept.
3. The supervisor agents receive the query and use their learned representations for their own conceptual hierarchy to infer whether or not they know the same semantic concept. In other words, the supervisor agents attempt to interpret the semantic objects based on their own ontology.
4. The supervisor agents reply to the querying agent with a
   - "Yes, I know that semantic concept"
   - "I may know that semantic concept" or
   - "No, I don't know that concept".

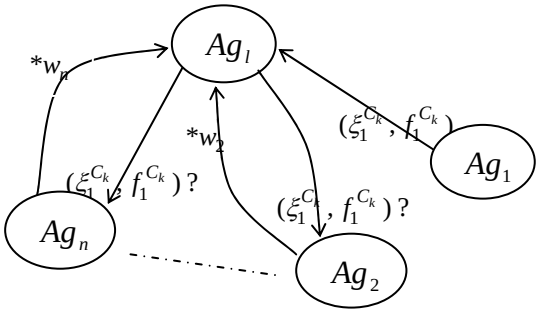

Figure 4. Agents' collaboration mechanism

The agent Ag_l makes a virtual team of teacher agents comprising agents Ag_1, …, Ag_n, which have enough knowledge regarding concept C_k.

5. The learner agent Ag_l queries team members asking their examples ($\xi_i^{C_k}$) regarding concept C_k and features ($f_i^{C_k}$) which they have considered to learn it. Then it receives their responses.
6. For each teacher agent, learner agent attempts to verify whether or not the other agents classify its examples as C_k.
7. Finally after resolving conflicts, the learner agent starts to learn from training examples.

5. An Example Application

To evaluate our approach to multi-agent concept learning we set up an experiment with a group of four agents. Following our agent model and collaboration schema, every agent is equipped with an ontology, which was the course catalog of one university. We considered each educational organization (e.g. departments, programs, etc.) as a concept, which potentially can be learnt by the learner agent. To represent a concept we chose two features: name and description, where both of them are in the plain text format and in order to be processed, they should be mapped to the vector space model. This, eventually, cause the features to be further broken down to words. To define a concept by its comprising objects we assigned a set of courses (as objects) to each concept. Each course has been represented by three features: name, description and prerequisite. Again all of these features were in the plain text format and must be handled in the same way as the concepts. In order to find a concept to answer a query, we enabled each agent to index the features of the concept and searched its ontology to find the best matching concepts with the query. In the example level, the supervisor agents

indexed the features to find the top *n* courses supporting the concept. The *n* is the maximum number of the positive and negative examples which is supposed to be sent to the learner by the supervisors. Also each agent could communicate with other agents and send and receive KQML based messages.

This section discusses how we evaluated our approach using the system we developed and the results for the experiments.

5.1. Problem characteristics

We used the course catalog ontology domain [29]. The course catalog domain describes courses at Cornell university, the university of Washington and the university of Michigan (see Table 1). Courses (as the objects that form concepts) are organized into programs and schools, then into departments and centers within each college. We used the entire data set for two universities from [29] and constructed the other set of ontology and constructed the data instances from the university of Michigan site [30]. We developed our ontology by OWL [31] and used the JENA [32] as an application program interface to OWL to manage the incoming queries. We ran our experiment of concept learning in groups of 2, 3 and 4 agents to see how different agents' knowledge can change the ontology of the learner agent.

We chose the cosine similarity method as amalgamation function in our similarity measure function. The Rocchio [33], kNN [34] and Naive Bayes [34] were used to evaluate the classification accuracy and because there were no significant difference between them we only report the evaluation by Naive Bayes. The domain characteristics are shown in Table 1.

Ontologies	Cornell	Michigan	Washington
#concepts	176	174	166
#non-leaf Concepts	27	21	25
depth	4	4	4
#objects in ontology	4360	7744	6957
Max # objects at a leaf	161	293	214

Table 1. Domain characteristics

5.3. Learning from a group: An example

In this experiment we determined how our system would construct the concept and corresponding hierarchy (shell concepts) when the hierarchies that are coming from different agents are different.

Figure 5 shows the example where the concepts and their hierarchy are represented for the queried concept "greek" from three agents. While the hierarchy from *cornell* and *washington* are close to each other the hierarchy from *michigan* is different and has two more specific concept for "greek". Table 2 shows the formation of conceptual hierarchy from bottom level (*level 1*) to top level (*level 4*), in the learner agent when it learns the concept from one, two and three agents, respectively. The assigned numbers for levels show the weight of features in the result hierarchy. The higher the number the higher the chance of feature in order be in the result hierarchy. The first row shows the formation of hierarchy when one teacher agent is around (*cornell*) and the outcome of learning is quite similar to the teacher agent's hierarchy. Adding the *michigan* to the teachers (i.e. second row), changes the hierarchy in the learner. In order to make the hierarchy the learner assumes the longer hierarchy as a better answer to the query and considers it as target hierarchy. This is due to the fact that a longer hierarchy most probably shows more specific answer to a query and shows the expertise of the sender. The learner agent then starts to integrate hierarchies from other agents to the target hierarchy by step by step checking of similarities between concepts and merging them.

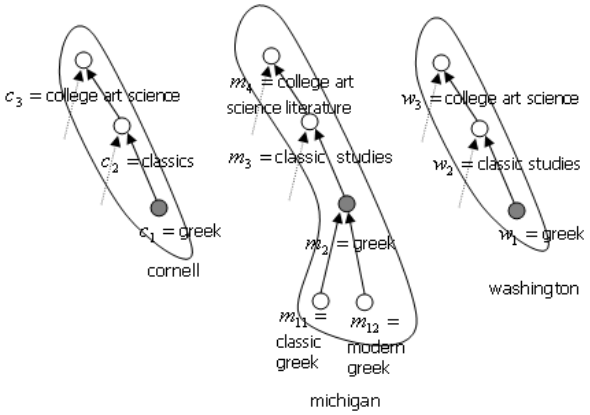

Figure 5. Teachers' Hierarchy for Concept 'greek'

Based on pre-structuring algorithm in Figure 3, the learner first checks the similarity between the c_1 and m_{11} and m_{12}. They are not similar and the learner agent makes new concepts l_{11} and l_{12} in the last level and sends the c_1 to upper level to further processing. In the next step c_1 and m_3 are checked and because they are similar they are merged and considered as c_{goal}. This process is repeated for the upper levels and l_3 and l_4 are produced. The third agent (*washington*) does not make a significant change in the hierarchy (i.e. third row)

because to some extent it is similar to the hierarchy from *cornell*. Figure 6 shows a pictorial interpretation for the Table 2. Figure 6 a), b) and c) show three stages of learning when one, two and three teacher agents are around respectively.

# Agents	Hierarchy Formation
1	*level 1:* greek: 67 *level 2:* classic: 67 *level 3:* college: 67 science: 67 art: 67
2	*level 1:* classic: 75 greek: 75 modern: 11 greek: 11 *level 2:* greek: 156 *level 3:* classic: 153 studies: 86 *level 4:* college: 153 literature: 86 art: 153 science: 153
3	*level 1:* classic: 75 greek: 75 modern: 11 greek: 11 *level 2:* greek: 156 *level 3:* classic: 175 studies: 86 *level 4:* college: 175 literature: 86 art: 175 science: 176

Table 2. Tabular Results for Pre-structuring

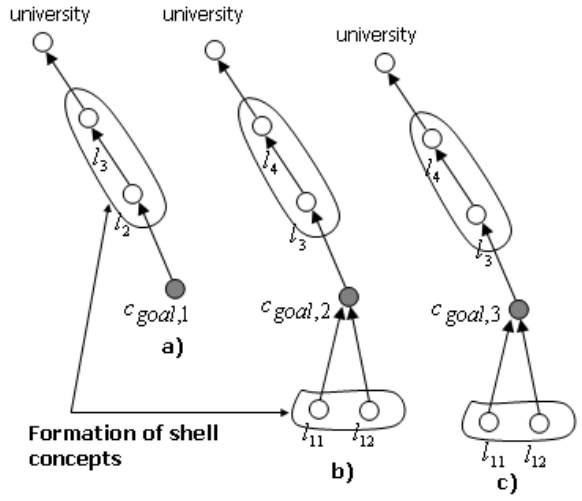

Figure 6. Goal and Shell Concept Formation for Table 2

The weight that the learner assigns to the teachers has a direct affect in the similarity measure value. Here we also use this weight in the formation of the concept's name. For example, the assigned number to "classic" is 175 and it shows that the level of confidence of the learner in the naming of the formed concept as "classic" is higher than naming it by "classic studies". The learner in our experiment used a threshold for weights in order to justify whether to include a feature (e.g. word) in the name of concept or not.

6. Related Works

Most works in literature assume that agents use a common ontology [14]. The works by Williams [1] introduced the idea of using learning to improve the mutual understanding about a concept between two agents. In contrast to our method, Williams uses only a flat repository of concepts, not a real ontology. The learning is used to have only two agents develop a common feature description about a particular concept assuming that the agents share the same perception of objects (i.e.\ use the same features). No negative examples are used and there is naturally no pre-structuring and no need to deal with conflicts between teachers. [3] presents a method how one agent can train another agent to recognize a concept by providing selected positive training examples. The multi-agent dimension is not addressed and no usage of ontologies is made. Similar to our work, Steels [2] allows for differences in ontologies of agents and wants to minimize these differences for concepts that are of interest to some of his agents. In contrast to us, his agents do not use learning to allow for teaching a concept to an agent, his approach suggests to have the agents cooperatively learn (evolve) a common set of features (using what he calls naming games) and then use a common method for individual agents to create their ontologies by experiencing their environment. For agents with different perceptions, this must pose problems which our approach does not have. But we do not allow for agents to change their feature sets. Also, the emphasis of Steels' work is more on language than concepts.

Nowadays, there is an increasing interest in the use of peer-to-peer and multi-agent concepts in KM, mainly motivated by the fact that KM domains involve an inherent distribution of resources, problem solving capabilities and responsibilities [4,5]. That is, the integrity of the existing organizational structures and the autonomy of participants must be maintained, which calls for an autonomous and distributed

representation of KM systems [6]. The use of shared representational ontologies has been questioned in [7] and a distributed architecture based on explicitly distributed ontologies has been proposed in [8]. We have discussed enabling technologies and the research trends from Web-based centralized KM to the distributed agent mediated knowledge management in [10]. Other projects that address these aspects are: COMMA [4], FRODO [9] and EDAMOK [8].

To achieve ontology-based semantic integration, two agents must find a way to share the semantics of the terms in their ontologies, this can be done in several possible directions: (1) using a single centralized global ontology- a single centralized global ontology is defined for the application domain- and all agents or computer programs in communication use terms from this ontology; (2) merging source ontologies into a unified ontology: ontologies defined on a common domain by different applications have lots of overlap, therefore merging the source ontologies into one unified ontology before agent interactions is a way to fulfill semantic integration [11]; (3) searching a set of mappings (or matches) between two ontologies: instead of trying to merge two source ontologies, finding a set of mapping rules between them is an alternative way to achieve semantic integration [11,12]; (4) runtime ontology resolution: for a multiagent system, agents are often from different heterogeneous environments, it is impractical to restrict all agents to use a single ontology or to have ontology merging, matching, and translation services available prior to the deployment of the agent system. A better way is to resolve semantic differences when they arise during run-time interaction [13].

7. Conclusion

We presented a general concept for improving communication between agents that use ontologies as part of their knowledge representation and that not only have different ontologies but even different features sets that they can recognize. Similar to the behavior of humans, the key idea is to have agents that know a needed concept (but that might not agree totally on all details) teach an interested agent the concept by providing positive and negative examples for the concept that the learning agent feeds into a concept learner. Conflicts are resolved by voting/elimination of examples. We provided an example that illustrates how our method comes up with a concept (and its integration into the agent's ontology) that represents the common ground between the teachers.

8. References

[1] A.B.Williams, "Learning to Share Meaning in a Multi-Agent System", Journal of Autonomous Agents and Multi-Agent Systems, Vol. 8, No. 2, 165-193, March 2004.

[2] L. Steels, "The Origins of Ontologies and Communication Conventions in Multi-agent Systems," J. Autonomous Agents and Multi-Agent Systems, vol. 1, no. 2, pp. 169-194, Kluwer, 1998.

[3] S. Sen, P.P. Kar, "Sharing a concept" , AAAI Tech Report SS-02-02, Stanford, 2002.

[4] F.Gandon, R.. Dieng, Corby, O., Giboin, A., "A Multiagent System to Support Exploiting an XML-based Corporate Memory," Proc. PAKM'00, Basel, 2000.

[5] A. Susarla, D. Liu, and A. Winston, "Peer-to-Peer Knowledge Management," Handbook of Knowledge Management, vol. 2, ch. 39, 2002.

[6] V. Dignum, "Using Agents to Support Knowledge Sharing," Proc. Workshop on Autonomy, Delegation and Control, AAMAS'03, Melbourne, 2003.

[7] J. Wang, and L.Gasser, "Mutual Online Concept Learning for Multiple Agents," Proc. AAMAS'02, Bologna, Italy, pp. 362-369, 2002.

[8] M. Bonifacio, P. Bouquet, A. Manzardo, "A Distributed Intelligence Paradigm for Knowledge Management, Bringing Knowledge to the Business Process," Proc. AAAI Spring Symposium, Technical Report SS-00-03, AAAI Press, 2000.

[9] Abecker A., van Elst, L., "Ontologies for Knowledge Management," in Handbook on Ontologies, S. Staab and R. Studer (eds.), Ch. 22, pp. 435-454, Springer, 2003.

[10] M. Afsharchi, B.H. Far, "Knowledge Orchestration Agency: Knowledge Management Using Intelligent Software Agents," Decision Support in Agent Mediated Environments, Chapter 3, pp. 71-89, G. Phillips-Wren and L. Jain (Edts.), IOS Press B.V., 2005.

[11] N.F. Noy, M.A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment." Proc. 17th National Conf. on Artificial Intelligence (AAAI-2000), Austin, TX, 2000.

[12] A. Doan, J. Madhavan, P. Domingos, A. Halevy, "Ontology Matching: A Machine Learning Approach," Handbook on Ontologies in Information Systems, S. Staab and R. Studer (eds.), pp. 397-416, Springer, 2004.

[13] F. Wiesman, N. Roos, P. Vogt, "Automatic Ontology Mapping for Agent Communication. Technical Report, MERIT, 2001.

[14] R. Neches, R. Fikes, T. Finin, T. Gruber, R Patil, T. Senator, , Swartout, W., "Enabling Technology for Knowledge Sharing," AI magazine, vol. 12, no. 3, pp. 36-56, 1991.

[15] V. Dignum, "A model for organizational interaction: based on agents, founded in logic," SIKS Dissertation, series No. 2004-1.

[16] M. Klusch, S. Lodi, and G. Moro, "Agent-based distributed data mining: The kdec scheme," AgentLink, Springer Lecture Notes in Computer Science, vol. 2586, 2003.

[17] B.H. Far, W. Wei, and M. Afsharchi, "A Unified View of Software Agents Interactions," Transactions of Institute of Electronics, Information and Communication IEICE, Vol. E87-D, No. 4, pp. 896-907, April 2004.

[18] J. Collins, C. Bilot, M. Tsvetovat, M. Gini, and B. Mobasher, "Plan Execution by Contracting in a Multi-Agent Environment," Workshop on Agents for Electronic Commerce and Managing the Internet-Enabled Supply Chain, Agents'99, Seattle, May 1999.

[19] B.H. Far and G. Ruhe, “Prescriptive Decision Support based on Software Agent Interaction,” Decision Support Systems in Agent-Based Intelligent Environment, This volume.

[20] S. Bussmann, N.R. Jennings, M. Wooldridge, “Re-use of Interaction Protocols for Agent-Based Control Applications,” Springer Lecture Notes in Computer Science, vol. 2585, pp. 73 – 87, 2003.

[21] J. Ferber, O. Gutknecht, F. Michel, “From Agents to Organizations: An Organizational View of Multi-agent Systems,” Springer Lecture Notes in Computer Science, vol. 2935, pp. 214 – 230, 2003.

[22] N. Houari, B.H. Far, “An Intelligent Project lifecycle Data Mart-Based Decision Support System,” IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2004), pp. 727-730, May 2004.

[23] M. Montoni, et al., “Knowledge Acquisition and Communities in Practice: An Approach to Convert Individual Knowledge into Multi-organizational Knowledge,” Springer Lecture Notes in Computer Science, vol. 3096, pp. 110 – 121, 2004.

[24] A. Tiwana, and B. Ramesh, “Integrating Knowledge on the Web,” IEEE Internet Computing, vol. 5, no. 3, pp. 32-39, May 2001.

[25] T. Gruber. “The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases.” Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, Cambridge, MA, pages 601-602. Morgan Kaufmann, 1991

[26] M. Wooldridge, Intelligent agents in G. Weiss, (ed.), “Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence”, MIT Press: Cambridge, MA, pp. 28–77, 1999

[27] M. Genesereth and N. Nilsson, logical foundations of artificial intelligence, Morgan Kauffman publishers, Palo Alto, CA 1987

[28] R.Wille “Concept lattices and conceptual knowledge systems.” Computers & Mathematics with Applications, 23, 493-515, (1992)

[29] Illinois Semantic Integration Archive.http://anhai.cs.uiuc.edu/archive/, as seen on Jan 30, 2005.

[30] University of Michigan academic units. http://www.umich.edu/units.html, as seen on Jan 30, 2005.

[31] OWL - Web Ontology Language. http://www.w3.org/TR/owl-features

[32] Jena – A Semantic Web Framework for Java. http://jena.sourceforge.net/

[33] J.J. Rocchio, “Relevance feedback in information retrieval”,in The SMART Retrieval System - Experiments in Automatic Document Processing, Prentice Hall, 1971, pp. 313--323.

[34] T.M. Mitchell, “Machine Learning”, McGraw-Hill, 1997.