

CONCEPTS IN ACTION: PERFORMANCE STUDY OF AGENTS LEARNING ONTOLOGY CONCEPTS FROM PEER AGENTS

Leila Safari, Mohsen Afsharchi

Department of Electrical and Computer Engineering, University of Zanjan, Zanjan, Iran
lsafari,afsharchim@znu.ac.ir

Behrouz H.Far

Department of Electrical and Computer Engineering, University of Calgary, Calgary, Canada
far@ucalgary.ca

Keywords: ontology, concept learning, multi-agent communication.

Abstract: The ability to share knowledge is a necessity for agents in order to achieve both group and individual goals. To grant this ability many researchers have assumed to not only establish a common language among agents but a complete common understanding of all the concepts the agents communicate about. But these assumptions are often too strong or unrealistic. In this paper we present a comprehensive study of performance of agents learning ontology concepts from peer agents. Our methodology allows agents that are not sharing a common ontology to establish common grounds on concepts known only to some of them, when these common grounds are needed by learning the concepts. Although the concepts learned by an agent are only compromises among the views of the other agents, the method nevertheless enhances the autonomy of agents using it substantially. The experimental evaluation shows that the learner agent performs better than or close to teacher agents when it is tested against the objects from the whole world.

1 INTRODUCTION

In many situations efficient communication is the main cornerstone of cooperation among agents. While communication does not always have to involve the use of a language, for many purposes that use of a language is a very convenient way to convey information between agents. In addition to this common language, a common semantics is also necessary for communicative agents to interact and understand each other. Ontology research community tries to address issues arisen from violation or relaxation of any of the above two requirements (Gruber 1991).

For the sake of simplicity and/or convenience many researchers in their works have assumed that it is possible to establish a common language among agents (e.g., using several variations of agent communication languages, ACL), and also the agents are provided with a complete common understanding of all the concepts they need (e.g., having a common conceptualization). In case that heterogeneity or interoperability is a requirement, many researchers assume that it is possible to have an already existing common ontology for the agents and that the agent developers

can use this common ontology when designing their agents, perhaps by calling an ontology service, thus allowing for easy communication and understanding among the agents. However, the assumption of existence of a common ontology is often too strong or unrealistic. For many application domains, there is no agreement on ontology for the domain among developers. Also for many areas existing ontologies are large, unwieldy and encompass more than what a particular agent will ever need.

A recent approach is to let the agent have their individualized ontologies and provide them with learning mechanisms to learn the concepts they need during communication. (Williams 2004) (Steels 1998). In our previous work, we have devised a methodology for having agents learn concepts from several peer agents (Afsharchi et al., 2006). The basic idea behind our method is to have an agent, that realizes that there seems to be a concept it does not currently know of but expects to need to know, so queries the other agents about this concept by providing features (and their values) or examples the agent thinks are associated with the concept. The queried agents provide the agent with positive and negative examples from their

understanding of their concepts (i.e. concepts known by them) which seem to fit the query which allows the learning agent to use one of the known concept learning techniques from machine learning to learn the concept. To help focus on the negative examples, the teaching agents make use of selected relations between concepts from their ontologies. The agent that wants to learn the concept deals with the fact that the other agents in this group might not totally agree on which examples fit the concept and which not, by letting the teaching agents vote on the examples for which it got contradictory information in the first place.

The work by Williams (Williams 2004) introduced the idea of using learning to improve the mutual understanding about a concept between two agents. In contrast to our method, Williams agents uses only a flat repository of concepts and are not involved in a *multi* agent learning. (Sen and Kar 2002) presents a method of 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. Steels (Steels 1998), like us, 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. (Diggelen et al. 2006) developed ANEMONE which is a minimal ontology negotiation environment. A very important point about ANEMONE is that it is not a concept learning environment, rather it is an environment which facilitates ontology mapping between two agents in a minimal way using a layered approach. Like other related works, ANEMONE does not talk about feature diversity and it is not a *multi*-agent collaboration.

in this paper, as an extension to (Afsharchi et al., 2006), we present a comprehensive performance study of the learner agent. We compare the classification accuracy of the learner with teacher agents against the set of all objects in the agents' world. The experiments show that the learner perform better than or close to the teachers in a multi-agent environment.

The structure of this paper is as follows: in Section 2 we give definitions for the concepts that we use throughout this paper. In Section 3 the concept learning mechanism is reviewed, Section 4 introduces our experimental domain and is followed by our experimental results.

2 BASIC DEFINITIONS

In this section, we provide a brief definition of each of the two basic concepts involved in our system which are ontologies and agents. Also we provide the instantiations of these concepts that we require for our methods.

2.1 Ontologies and Concepts

A formal definition for ontology has been presented in (Stumme 2002) in which an *ontology* has been defined as a structure $O := (C, \leq_C, R, \sigma, \leq_R)$. C and R are two disjoint sets with members of C being called *concept identifiers* and members of R are *relation identifiers*. \leq_C is a partial order on C called *concept hierarchy* or *taxonomy* and \leq_R is a partial order on R , named *relation hierarchy*.

$\sigma : R \rightarrow C^+$ is a function providing the argument concepts for a relation such that $|\sigma(r_1)| = |\sigma(r_2)|$ for every $r_1, r_2 \in R$ with $r_1 \leq_R r_2$ and for every projection π_i ($1 \leq i \leq |\sigma(r_1)|$) of the vectors $\sigma(r_1)$ and $\sigma(r_2)$ we have $\pi_i(\sigma(r_1)) \leq_C \pi_i(\sigma(r_2))$. If $c_1 \leq_C c_2$ for $c_1, c_2 \in C$, then c_1 is called a *subconcept* of c_2 and c_2 is a *superconcept* of c_1 . Obviously, the relation \leq_C is supposed to be connected with how concepts are defined. In the literature, taxonomies are often build using the subset relation, i.e. we have

$$c_i \leq_C c_j \text{ iff for all } o \in c_i \text{ we have } o \in c_j.$$

This definition of \leq_C produces a partial order on C as defined above and we will use this definition in the following for the ontologies that our agents use.

Concepts often are seen as collections of objects that share certain *feature* instantiations. In this work, for an ontology O we assume that we have a set of features $\mathcal{F} = \{f_1, \dots, f_n\}$ and for each feature f_i we have its domain $D_i = \{v_{i1}, \dots, v_{im_i}\}$ that defines the possible values the feature can have. Then an object $o = ([f_1 = v_1], \dots, [f_n = v_n])$ is characterized by its values for each of the features (often one feature is the identifying name of an object and then each object has a unique feature combination). By \mathcal{U} we denote the set of all (possible) objects. In machine learning, often every subset of \mathcal{U} is considered as a concept. In this work we want to be able to characterize a concept by using feature values. Therefore, a *symbolic concept* c_k is denoted by $c_k([f_1 = V_1], \dots, [f_n = V_n])$ where $V_i = \{v'_{i1}, \dots, v'_{iji}\} \subseteq D_i$ (if $V_i = D_i$ then we often omit the entry for f_i). An object $o = ([f_1 = v_1], \dots, [f_n = v_n])$ is *covered* by a concept c_k , if for all i we have $v_i \in V_i$. In an ontology according to the definition above, we assign a concept identifier to each symbolic concept that we want to represent in our ontology.

2.2 Agents

A general definition that can be instantiated to most of the views of agents in literature sees an agent $\mathcal{A}g$ as a quadruple $\mathcal{A}g = (Sit, Act, Dat, f_{\mathcal{A}g})$. Sit is a set of situations the agent can be in, the representation of a situation naturally depending on the agent’s sensory capabilities, Act is the set of actions that $\mathcal{A}g$ can perform and Dat is the set of possible values that $\mathcal{A}g$ ’s internal data areas can have. In order to determine its next action, $\mathcal{A}g$ uses $f_{\mathcal{A}g} : Sit \times Dat \rightarrow Act$ applied to the current situation and the current values of its internal data areas.

As we want to focus on the knowledge representation used by agents and how this is used for communication, we have to look more closely at Dat . We assume that every element of Dat of an agent $\mathcal{A}g$ contains an ontology area $O_{\mathcal{A}g}$ that represents the agent’s view and knowledge of concepts. There might be additional data, beyond features, that the agent requires from time to time, about concepts and this data is naturally also represented in Dat . Also, there will be additional data areas representing information about the agent itself, knowledge about other agents and the world that the designer of the agent may want to be represented differently than in $O_{\mathcal{A}g}$.

3 LEARNING CONCEPTS FROM SEVERAL TEACHERS

In this section we provide a brief description of the multi-agent concept learning we presented in (Afsharchi et al., 2006). As already stated, we have developed a method that demonstrates how an agent can learn new concepts for its ontology with the help of several other agents. This assumes that not all agents have the same ontology. We additionally assume that there are only some base features $\mathcal{F}_{base} \subseteq \mathcal{F}$ that are known and can be recognized by all agents and that there are only some base symbolic concepts C_{base} that are known to all agents by name, their feature values for the base features and the objects that are covered by them. Outside of this common knowledge, individual agents may come with additional features they can recognize and additional concepts they know. Given this setting, agents will develop problems in working together, since the common grounds for communication are not always there. To come up with a solution for this problem, agents need to acquire the concepts outside of C_{base} that other agents have, at least those concepts that are needed to establish the necessary communication to work together on a given task. The basic idea is to have an agent *learn* a required

concept (or at least a good approximation of it) with the *help* of the other agents acting as teachers.

3.1 The General Interaction Scheme

In the following, $\mathcal{A}g_L$ refers to the agent that wants to learn a new concept and the other agents, $\mathcal{A}g_1, \dots, \mathcal{A}g_m$, will be its teachers. $\mathcal{A}g_L$ has an ontology $O_L = (C_L, \leq_C, R_L, \sigma_L, \leq_{R_L})$ and knows a set of features \mathcal{F}_L . Analogously, $\mathcal{A}g_i$ has as ontology $O_i = (C_i, \leq_C, R_i, \sigma_i, \leq_{R_i})$ and knows a set of features \mathcal{F}_i . For a concept c known to the agent $\mathcal{A}g_i$, this agent has in its data areas a set $pex_i^c \subseteq \mathcal{U}$ of positive examples for c that it can use to teach c to another agent. Parts of Act_L are actions `QueryConcept`, `AskClassify`, `Learn`, and `Integrate`, while part of the Act_i s are the actions `FindConcept`, `CreateNegEx`, `ReplyQuery`, `ClassifyEx` and `ReplyClass`. For teaching $\mathcal{A}g_L$ a new concept c_{goal} , we have as general interaction scheme:

After becoming aware that there is a concept that it needs to learn, $\mathcal{A}g_L$ performs the action:

$$\text{QueryConcept}(\text{identifier}, \{[f'_1 = V'_1], \dots, [f'_i = V'_i] = V'_i\}, O_{goal}).$$

The three parameters of `QueryConcept` allow for three different ways to identify to the teachers what $\mathcal{A}g_L$ is interested in. The parameter `identifier` allows $\mathcal{A}g_L$ to refer to a concept name it observed from other agents, which means that `identifier` is an element of C_i for some agent(s) $\mathcal{A}g_i$. By $\{[f'_1 = V'_1], \dots, [f'_i = V'_i]\}$, $\mathcal{A}g_L$ can use a selection of features $f'_j \in \mathcal{F}_{base}$ and the values $V'_j \subseteq D_{f'_j}$ that $\mathcal{A}g_L$ thinks are related to the concept c_{goal} . Finally, $O_{goal} \subseteq \mathcal{U}$ is a set of objects that $\mathcal{A}g_L$ thinks are covered by c_{goal} .

Each $\mathcal{A}g_i$ then reacts to $\mathcal{A}g_L$ ’s query by performing:

$$\text{FindConcept}(\text{identifier}, \{[f'_1 = V'_1], \dots, [f'_i = V'_i]\}, O_{goal}).$$

Naturally, each of the parameters can already point to different concepts that an agent $\mathcal{A}g_i$ knows of. In fact, if $\mathcal{A}g_L$ provides several objects in O_{goal} , they might be classified by $\mathcal{A}g_i$ into several of its concepts. So, $\mathcal{A}g_i$ first collects all the concepts that fulfill the query into a candidate set C_i^{cand} and then it has to evaluate all these concepts to determine the concept that is the best fit. So, the output of `FindConcept` is a set of candidate concepts C_i^{cand} . To select the “best” candidate c_i out of C_i^{cand} , there are many different ways how an evaluation of the candidates can be performed. Each of the 3 query parts can contribute to a measure that defines what is “best”, but how these contributions are combined can be realized differently.

The number of examples communicated to \mathcal{A}_{GL} by each agent is a parameter of our system. So, in the next step, each teacher selects the given number of elements out of the set of positive examples, $pe_x^{c_i}$, for the best candidate concept c_i and we call this set p_i . Again, there are many possible ways how this selection process can be done, so far we used random sampling of $pe_x^{c_i}$. By then performing `CreateNegEx(c_i)`, the teacher agents produce a given number of (good) negative examples for c_i , which produces the set n_i . Since every concept c_j other than c_i (and its subconcepts) can be categorized as a counter concept, the number of objects associated with these c_j s (which naturally are negative examples) is often very high. We used both taxonomy information (siblings of c_i) and a relation `is-similar-to`(similar concepts to c_i) to select the concepts from which we randomly selected examples as elements for n_i .

`ReplyQuery(c_i, p_i, n_i)` is the last action performed by a teacher agent before the initiative goes back to the learner. It sends the result back to the learner. \mathcal{A}_{GL} collects the answers (c_i, p_i, n_i) from all teachers, $pe_x^{c_{goal}} = \cup_{i=1}^m p_i$ and $nex^{c_{goal}} = \cup_{i=1}^m n_i$, and then uses a concept learner to learn c_{goal} from the combined examples (action `Learn($(p_1, n_1), \dots, (p_m, n_m)$)`). Naturally, the concept learner only uses features and their values from \mathcal{F}_L .

Learning from a group of agents is a very conflict prone process compared to just learning from one agent. It can easily happen that the best concepts c_i and c_j that \mathcal{A}_{g_i} and \mathcal{A}_{g_j} identified are not the same. The worst case can be that an example that \mathcal{A}_{g_i} sent as being positive for c_{goal} \mathcal{A}_{g_j} sent as a negative one. But we can also have more indirect conflicts where a learning algorithm simply cannot come up with a concept description that covers all objects in $pe_x^{c_{goal}}$ while not including any objects in $nex^{c_{goal}}$. There are several methods how we can solve this problem and these methods represent different degrees of willingness to satisfy the teacher agents (by \mathcal{A}_{GL}).

For our system, we have chosen the following conflict resolution to produce c_{goal} for \mathcal{A}_{GL} . After the learning component of \mathcal{A}_{GL} has performed `Learn` and produced a more precise c_{goal} , \mathcal{A}_{GL} will test all elements of $pe_x^{c_{goal}}$ and $nex^{c_{goal}}$ for correct classification by this new c_{goal} . For all the example objects that are not correctly classified, we go back to the teacher agents and ask them to classify these examples according to the c_i they used to produce their examples. We then treat the answers as votes and include all positive examples for which a majority of the teachers voted, while requiring the exclusion of all negative examples for which a majority voted. This produces some kind of compromise concept that might appeal

to most of the teachers (although it might not be identical to any of the c_i).

The result of this learning/teaching scheme is the description of c_{goal} in terms of \mathcal{A}_{GL} 's feature set \mathcal{F}_L and an updated ontology $O_L^{new} = (C_L^{new}, \leq_C, R_L, \sigma_L, \leq_{R_L})$.

4 EXPERIMENTAL RESULTS

To study the performance of the learner agent, we have chosen the course catalog ontology domain (see (Illinois Archive)). In the following, we will first introduce this domain and present an example of formation of a concept in the learner agent. Then we will use 3 different concepts to evaluate the performance of the learner agent regarding these newly learned concepts.

4.1 The University Units and Courses Domain

The university units and courses domain consists of files describing the courses offered by Cornell University, the University of Washington and the University of Michigan, together with ontologies for each of the three universities describing their organizational structure (see (Illinois Archive) and (Michigan Units)). In our examples, each of the three universities is represented by an agent (\mathcal{A}_{g_C} , \mathcal{A}_{g_W} , \mathcal{A}_{g_M}) and these agents are acting on the one hand side as the teachers to an agent \mathcal{A}_{GL} and on the other side as the agents \mathcal{A}_{GL} communicates with afterwards.

The objects of this domain are the course files that consist of a course identifier, a plain text course description and the prerequisites of a course. All in all, there are 19061 courses among the three universities and each university's ontology has at least 166 concepts on top of their courses. To create features and their values fulfilling the definitions from Section 2.1 we used or-combinations of key words as single features (with differing key word sets for the different agents to create a situation where the features used by the agents are different) and a feature was fulfilled, if one of the words from the or-combination occurred in the course description (Afsharchi et al., 2006). This view of features corresponds nicely with the learning method we chose for \mathcal{A}_{GL} , namely the method of (Koller and Sahami, 1997). So, the feature for the key word subset $\{picture, photo, figure\}$ ($f_{picture, photo, figure}$) is true for a text t , if either *picture* or *photo* or *figure* occur in t .

The information for the three universities does not come with the appropriate feature values (at least not

directly), instead we only know the taxonomy and for each unit what courses belong to it. In order to create the ontologies O_C , O_W and O_M for the agents \mathcal{A}_{g_C} , \mathcal{A}_{g_W} and \mathcal{A}_{g_M} we used the learning method of (Koller and Sahami, 1997) to create feature-based descriptions of each concept (as described in Section 2.1). Naturally, the examples for the learning algorithm were only taken from the courses of the particular university and we used all the courses as examples to achieve a perfect fit. To cover exactly the courses of a particular unit, we also had to adjust the initial key word sets of the agents, but we were able to keep these sets different between the agents. Also, since no examples from other universities were used for a particular agent, this agent really reflects just the view of this university, so that differences in the understanding of a concept (i.e. what should be taught by what unit) between the universities are preserved.

4.2 Concept Formation

To provide a better picture of how our method works, first we take a look at the formation of concepts in \mathcal{A}_{g_L} when it learns the concept from different numbers of agents. By this experiment, we aim to present a quantitative view of the formation of a new concept in the learner. This quantitative view reveals how a new concept forms in the form of the features that the learner agent recognizes. For this experiment we fixed the teacher agents to randomly select the positive examples and use the ontology (as described in 3.1) to select the negative examples. Also, we used the majority voting to resolve the conflicts.

For a particular query, we observe the relevant parts of the ontologies of the three teacher agents and study what an agent can learn from these teachers with regard to that query. The accumulation of different key words in the learner agent for a particular concept shows how the learner agent shapes its understanding of that concept.

4.2.1 Learning of Concept“Computer Science”

Let us assume that the learning agent is supposed to provide someone at a university with suggestions for how a unit concerned with Computer Science should be characterized. Unlike many other concepts (e.g. `mathematics`), the teacher agents have not so much overlap characterizing Computer Science. For instance, \mathcal{A}_{g_M} organizes it as an engineering discipline and as a joint program with electrical engineering. \mathcal{A}_{g_W} also considers Computer Science as an engineering discipline but independent from electrical engineering and as a joint program with com-

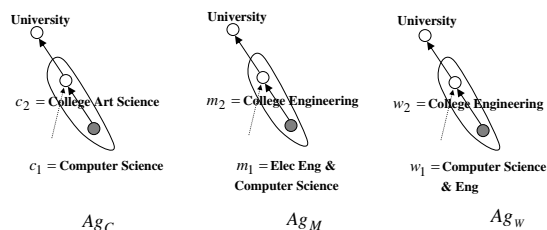


Figure 1: Relevant taxonomy paths of teachers for Computer Science

puter engineering. In \mathcal{A}_{g_C} Computer Science is a pure science program in the science faculty.

\mathcal{A}_{g_L} starts the learning process by submitting a query consisting of {`computer`, `science`, `program`, `system`} as key words. We assume the relevant concepts in C_{base} are $C_{base} = \{\text{University}\}$ and also assume that a part of relevant features in \mathcal{F}_{base} are created using the key set:

$\mathcal{K}_{base} = \{\text{class, course, prerequisite, program, system, design, computer, science, software, data, design, logic, theory, analysis, digital, language, parallel, algorithm, network, intelligent, plasma, artificial, process}\}$

In this example a part of relevant key words for the features used in the agents’ ontologies are:

$\mathcal{K}_C = \{\text{knowledge, optimization, formal, search}\}$,
 $\mathcal{K}_M = \{\text{power, circuit, signal, frequency, transition}\}$ and
 $\mathcal{K}_W = \{\text{complexity, information, performance, model, graphics, agent, image}\}$.

We give \mathcal{A}_{g_L} as key set:

$\mathcal{K}_L = \{\text{artificial, performance, agent, formal, circuit, image, optimization}\}$.

Figure 1 shows the paths in the taxonomies of the teachers that our system selected as the best concepts for the query. We used the measure from (Afsharchi 2007) to select the best matching concept. Table 1 provides an idea on how the teachers can affect what \mathcal{A}_{g_L} learns regarding the key word set accumulation which leads to the formation of c_{goal} .

The process of learning of Computer Science demonstrates how having agents with different viewpoints can influence key word set formation in \mathcal{A}_{g_L} . As table 1 shows when \mathcal{A}_{g_L} learns from three agents it drastically decreases the number of key words from \mathcal{A}_{g_M} in the final description of c_{goal} . To make this impact clear let’s take a closer look at Table 1. {`signal`, `circuit`, `power`, `plasma`} are four key words from \mathcal{A}_{g_M} which usually are not frequent in computer science course descriptions. In fact, these key words are frequent in electrical engineering course descriptions and their appearance in the key word set of \mathcal{A}_{g_M} shows the mixed nature of the concept for this agent (i.e. electrical engineering and computer science). As column ($\mathcal{A}_{g_L}(C, M)$) shows these four key words make a significant contribution in the formation of c_{goal} when \mathcal{A}_{g_C} and \mathcal{A}_{g_M} are

Table 1: Usage of key words in features for Computer Science

key word	\mathcal{A}_{g_C}	\mathcal{A}_{g_M}	\mathcal{A}_{g_W}	$\mathcal{A}_{g_L(C)}$	$\mathcal{A}_{g_L(C,M)}$	$\mathcal{A}_{g_L(C,M,W)}$
computer	52	75	71	48	75	135
software	10	19	25	9	18	36
network	12	11	14	11	17	39
intelligent	8	3	6	8	10	14
science	17	19	16	17	28	40
knowledge	25	0	0	22	19	18
optimize	13	0	0	7	6	6
formal	11	0	0	4	4	3
search	4	0	0	2	2	2
signal	0	24	0	0	22	7
circuit	0	14	0	0	12	5
power	0	12	0	0	11	4
plasma	0	4	0	0	2	0
agent	0	0	2	0	0	1
performance	0	0	14	0	0	11
graphics	0	0	9	0	0	6
image	0	0	6	0	0	3

teachers. Because our conflict resolution policy allows every example object to be included in the final description of c_{goal} when we have two agents as teachers, in this case, \mathcal{A}_{g_L} accepts every example from the teacher agents. Adding \mathcal{A}_{g_W} to the teachers makes a drastic change in the formation of c_{goal} regarding these keywords from \mathcal{A}_{g_M} . The number of appearances of keywords changes to 7, 5, 4, and 0 from 22, 12, 11, and 2. Due to the fact that \mathcal{A}_{g_C} and \mathcal{A}_{g_W} have close viewpoints on Computer Science, they naturally reject many of the examples which are reflecting electrical engineering aspect of the concepts from \mathcal{A}_{g_M} .

With just \mathcal{A}_{g_C} as teacher, among the features enabled in c_{goal} we have:

$f_{\text{parallel, search, optimize}}$ and $f_{\text{formal, language,}}$

with \mathcal{A}_{g_C} and \mathcal{A}_{g_M} we have:

$f_{\text{circuit, design, system, artificial, intelligent, agent}}$
and $f_{\text{signal, plasma,}}$

and with all agents as teachers we have:

$f_{\text{performance, analysis}}$ and $f_{\text{digital, image, process.}}$

The important point here is that after learning c_{goal} from three agents, \mathcal{A}_{g_L} does not establish $f_{\text{signal, plasma}}$ as an enabled feature. That is because $\{\text{plasma}\}$ is not a key word in the final description of c_{goal} in column ($\mathcal{A}_{g_L(C, M, W)}$).

4.3 Concepts in Action

It is very important to assess the performance of the learner agent regarding a newly learned concept. As

stated before our main concern in this paper is to have agents learn concepts to improve communication. Needless to say, to communicate about a concept, an agent must distinguish an instance of the concept (i.e. an object) from other instances. Based on this fact, we conducted an experiment to see how \mathcal{A}_{g_L} classifies objects in \mathcal{U} (i.e. the set of all possible objects) when it learns a new concept.

As we stated before, the most popular way of relying on a group decision is to follow the majority of votes. Therefore, for this experiment, we have chosen the set of examples from the majority area to be learned by \mathcal{A}_{g_L} .

Using the same set up as Section 4.2, we enabled \mathcal{A}_{g_L} to learn three different concepts Greek, Computer Science and Mathematics. We allowed \mathcal{A}_{g_L} to use the most popular way of relying on a group decision which is to follow the majority of votes as the representative examples of the concept. Then we trained the learner using different percentages of positive examples in this area (i.e. Table 2 n% column). These percentages show us the classification accuracy of \mathcal{A}_{g_L} when the learner does not utilize the maximum number of available examples from the teachers. That is the case when due to the communication cost, the teachers could not send every possible example that they possess to the learner. Table 2 shows the classification results of the learner for the three different concepts. In fact this table shows the number of truly classified examples both for positive examples and negative examples in two separate columns. We should mention that, when we consider the examples that a majority of agents agreed upon as the boundary for the concept in the learner, every other examples will be tested as the negative examples by \mathcal{A}_{g_L} in the testing process. For instance and for concept Mathematics, the majority set has 501 positive examples and the other objects (i.e. 19061-501=18560) could be considered as negative examples for it.

One interesting preliminary result, that in fact we expected, was the significant increase of truly classified examples when the concept is mostly unanimous. For example the programs Mathematics and Greek have more common courses than Computer Science among three different universities (which also is very true among other universities). As Table 2 shows the accuracy result for Mathematics is much better than Computer Science. The last row of Table 2 shows the performance of the learner when it is trained by the whole set of examples it possess for each concept. For instance, the second and third columns show that \mathcal{A}_{g_L} classified 497 positive and 17324 negative examples out of 501 positive and 18560 negative examples respectively. Therefore \mathcal{A}_{g_L}

Table 2: Truly classified examples for concepts Mathematics, Computer Science and Greek

n%	Mathematics			Computer Science			Greek		
	Positive Out of 501	Negative Out of 18560	%	Positive Out of 505	Negative Out of 18556	%	Positive Out of 171	Negative Out of 18890	%
10	400	13125	70	324	11375	61	128	13906	74
20	458	14157	77	339	11934	64	134	14551	76
30	460	15290	82	346	12307	66	142	14958	79
40	472	15267	82	354	12494	67	154	15681	83
50	481	15358	83	367	13053	70	159	16254	86
60	485	15501	83	380	13426	72	163	16808	89
70	484	15691	84	391	13613	73	165	17193	91
80	489	15886	85	399	14172	76	167	17005	90
90	495	16765	90	423	14731	79	170	16995	90
100	497	17324	93	429	15104	81	170	16991	90

classified 93% $((497+17324)/(501+18560))$ of objects correctly for Mathematics while this accuracy is 81% $((429+15104)/(505+18556))$ for Computer Science and 90% $((170+16991)/(171+18890))$ for Greek. There is a small “dip” in Greek when \mathcal{A}_{g_L} is trained by 70% of examples when the accuracy jumps to 91% and then comes back to 90%. Despite this “dip” the learner shows a consistent behavior classifying positive examples. We conclude that having agents with close viewpoints helps the learner to have a concrete understanding of a concept which naturally leads to a learner with better performance.

In this experiment we also compare the performance of the learner with the teacher agents. To compare the performance of \mathcal{A}_{g_L} with the teacher agents we had to compare the classification capability of \mathcal{A}_{g_L} with \mathcal{A}_{g_W} , \mathcal{A}_{g_C} , and \mathcal{A}_{g_M} respectively. As we discussed earlier, we assume that the teacher agents have learned the concepts in their ontology before they start to teach a concept to the learner. This learning has been achieved using some supervised inductive learning mechanisms and using the example objects that in each agent are associated with every concept in its ontology. Therefore we are supposed to simply compare the classification efficiency of \mathcal{A}_{g_L} with \mathcal{A}_{g_W} , \mathcal{A}_{g_C} , and \mathcal{A}_{g_M} .

Nevertheless we can not guarantee that \mathcal{A}_{g_L} learns a concept using the same number of examples as each teacher agent and, obviously, the more examples are provided to the agent the better a classifier it can learn. This possibility causes an unbalanced situation in which \mathcal{A}_{g_L} and other agents can not be compared. To overcome this problem, we have to prepare a fair situation in which the learner agent classification efficiency could be compared with each teacher agent. Therefore we selected a fragment of positive examples in \mathcal{A}_{g_L} which is quantitatively equal to the number of positive examples in each teacher agent to

train \mathcal{A}_{g_L} with the same number of examples that the teacher agents utilized to learn the concept before.

Table 3: Comparison of the performance of \mathcal{A}_{g_L} and \mathcal{A}_{g_M}

concepts	Truly classified examples by \mathcal{A}_{g_L}	%	Truly classified examples by \mathcal{A}_{g_M}	%	% of example from \mathcal{A}_{g_L}
Mathematics	15859	83.2%	15886	83.3%	53%
Computer Science	12962	68.0%	11009	57.7%	43%
Greek	17011	89.2%	16719	87.7%	68%

Table 4: Comparison of the performance of \mathcal{A}_{g_L} and \mathcal{A}_{g_W}

concepts	Truly classified examples by \mathcal{A}_{g_L}	%	Truly classified examples by \mathcal{A}_{g_W}	%	% of example from \mathcal{A}_{g_L}
Mathematics	15780	82.7%	15756	82.6%	44%
Computer Science	12533	65.7%	11788	61.8%	28%
Greek	15492	81.2%	15121	79.3%	35%

Table 3, 4 and 5 show the results of comparisons of \mathcal{A}_{g_L} with \mathcal{A}_{g_M} , \mathcal{A}_{g_W} and \mathcal{A}_{g_C} respectively. The second column in each table shows the number of truly classified examples, both positive and negative, out of 19061 test examples (i.e. objects in \mathcal{U}) by \mathcal{A}_{g_L} . The third column shows the number of truly classified examples by the teacher agent and finally the fourth row shows the percentage of examples that \mathcal{A}_{g_L} has been trained with, to produce this result. For

Table 5: Comparison of the performance of \mathcal{A}_{g_L} and \mathcal{A}_{g_C}

concepts	Truly classified examples by \mathcal{A}_{g_L}	classified examples by \mathcal{A}_{g_C}	Truly classified examples by \mathcal{A}_{g_L}	classified examples by \mathcal{A}_{g_C}	% of example from \mathcal{A}_{g_L}
Mathematics	15163	79.5%	15086	79.1%	26%
Computer Science	12805	67.1%	12112	63.5%	39%
Greek	14894	78.1%	14597	76.5%	24%

instance the first row of Table 3 shows that \mathcal{A}_{g_L} has truly classified 15859 examples out of 19061 when it is trained by 53% of the whole set of its positive examples for *Mathematics*. The third column indicates that 15886 example objects are truly classified by \mathcal{A}_{g_M} . The last column indicates that the number of associated examples with concept *Mathematics* in \mathcal{A}_{g_M} is 53% of examples in \mathcal{A}_{g_L} . A very interesting outcome of this experiment is that \mathcal{A}_{g_L} in the most cases has a better performance than the teachers regarding to the learned concept. This emphasizes on the fact that \mathcal{A}_{g_L} learns the *compromise* concept and its learning reflects a mutual viewpoint of agents. Therefore it will perform better when it tests against the objects from the whole world.

As an example concept, if we look at the *Computer Science* we see that \mathcal{A}_{g_L} is doing better compared to the other agents. For instance its accuracy is 68% (12962/19061) when it is trained by 43% of the training examples (see table 3). Clearly this is a better performance than \mathcal{A}_{g_M} which has classified 57.7% (11009/19061) truly. These results also confirm our preliminary result which we discussed in Section 4.2.1 regarding the formation of *Computer Science*. Here we see that \mathcal{A}_{g_L} classifies 10.3% better than \mathcal{A}_{g_M} while this margin is 3.9% for \mathcal{A}_{g_W} and 3.6% for \mathcal{A}_{g_C} . We believe that this is because the viewpoint of the learner is closer to \mathcal{A}_{g_W} and \mathcal{A}_{g_C} and as we showed in Section 4.2.1 the compromise concept in \mathcal{A}_{g_L} does not have too much of the characteristics of *Computer Science* from \mathcal{A}_{g_M} . Therefore \mathcal{A}_{g_L} is doing *much* better in classifying objects from \mathcal{U} .

The story is different for *Mathematics*. The performance of the \mathcal{A}_{g_L} is worse than \mathcal{A}_{g_M} (i.e. 83.2% vs 83.3%) but it is better than \mathcal{A}_{g_W} (i.e. 82.7% vs 82.6%) and \mathcal{A}_{g_C} (i.e. 79.5% vs 79.1%). This observation shows that the performance of the learner is close to the performance of other agents and that is because *Mathematics* is more unanimous than *Computer Science* which makes the viewpoints close to each

other.

REFERENCES

- M. Afsharchi, B.H. Far, J. Denzinger(2006). Ontology-Guided Learning to Improve Communication between Groups of Agents. In *Proc. AAMAS 2006, Hakodate*, pp. 923–930.
- Illinois Semantic Integration Archive. <http://anhai.cs.uiuc.edu/archive/>, as seen on Jan 30, 2005..
- D. Koller, M. Sahami (1997). Hierarchically Classifying Documents Using Very Few Words. In *Proc. ICML-97, 1997*, pp. 170–178.
- S. Sen, P.P. Kar Sharing a concept. In *AAAI Tech Report SS-02-02, 2002 Stanford*
- G. Stumme: Using Ontologies and Formal Concept Analysis for Organizing Business Knowledge, In *Wissensmanagement mit Referenzmodellen Konzepte für die Anwendungssystem- und Organisationsgestaltung, Physica, 2002*, pp. 163–174.
- University of Michigan academic units. <http://www.umich.edu/units.html>, as seen on Jan 30, 2005.
- A.B. Williams. Learning to Share Meaning in a Multi Agent System, In *Autonomous Agents and Multi Agent Systems 8(2), 2004*, pp. 165–193.
- L. Steels. The Origins of Ontologies and Communication Conventions In *Multi-Agent Systems, Autonomous Agents and Multi-Agent Systems 1(2), 1998*, pp. 169–194.
- Mohsen Afsharchi: Ontology Guided Collaborative Concept Learning in Multi-agent Systems *PhD Dissertation, Department of Electrical and Computer Engineering, University of Calgary, 2007*.
- J. van Diggelen, R. Beun, F. Dignum, R. M. van Eijk, J. Ch. Meyer ANEMONE: An effective minimal ontology negotiation environment. pages 899–906, 2006.
- T.R. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 601–602, 1991.