Ontology-Guided Learning to Improve Communication between Groups of Agents

Mohsen Afsharchi
Department of Electrical and Computer Engineering
University of Calgary
Calgary, Canada
mafsharc@ucalgary.ca

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

Jörg Denzinger
Department of Computer Science
University of Calgary
Calgary, Canada
denzinge@cpsc.ucalgary.ca

ABSTRACT

We present a general method for agents using ontologies as part of their knowledge representation to teach each other concepts to improve their communication and thus cooperation abilities. Our method aims at getting positive and negative examples for a concept only very vaguely understood by a particular agent from the other agents. This agent then uses one of the known concept learning methods to learn the concept in question, involving the other agents again by taking votes in case of conflicts in the received knowledge. This method allows agents that are not sharing common ontologies to establish common grounds on concepts known only to some of them, if these common grounds are needed during cooperation. While the concepts learned by an agent are only compromises between the views of the other agents, the method nevertheless enhances the autonomy of agents using it substantially.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; I.2.6 [**Learning**]: Concept learning

General Terms

Algorithms, Experimentation

Keywords

ontology, concept learning, multi-agent communication

1. INTRODUCTION

The ability to communicate efficiently is one of the cornerstones of cooperation between human beings. In order to communicate efficiently, not only a common language is needed, people communicating also need to have some basic understanding of world concepts and they can use this basic understanding to teach each other additional concepts, at least up to the point where the understanding of the new concepts by the different people overlaps largely.

If we look at research in multi-agent systems concerning communication between agents, then many researchers have assumed that it is possible to not only establish a common language among agents, but also a complete common understanding of all the concepts the agents communicate about. In cooperative multi-agent systems which aim at solving a particular problem, usually all of the agents are designed by one group of developers and the problem already provides all of the concepts and the needed understanding (i.e. semantics) of them, so that the above assumption is valid. If we look at systems that have agents designed by different developers, many researchers assume that it is possible to have a common ontology for these agents and that the agent developers naturally will integrate this common ontology into their agents, thus allowing for easy communication (and understanding) among agents. But the assumption of a common ontology is often too strong or unrealistic. For many application areas, there is no agreement among developers on one ontology for the area, for many areas the potential ontologies are large, unwieldy and encompass more than a particular agent most probably will ever need and implementing complex ontologies can also easily lead to discrepancies between implementations.

Recently, the idea of having agents *learn* concepts (or languages) from other agents has been suggested as a solution to the problems above (see [3] for learning, resp. making up, a language and [11] for learning a concept). The work in [11] has focused on interactions between two agents only and single concepts, while [3] was not concerned with concepts. In this paper, we present a general method for having agents learn concepts from several other agents and our method also makes use of information from the ontologies in which the concepts to learn are integrated.

The basic ideas behind our method are 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, query 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 which of their concepts (i.e. concepts known by them) seem to fit the query which allows the learning agent to use one of the known concept learning techniques from machine learning. To help focus the negative examples, the teaching agents make use of selected relations between concepts from their

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

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. Our learning agent also uses information from the teacher agents about the paths to the concept of interest in their ontologies to prepare its own ontology for potential future concepts it might have to learn.

Our experiments with agents with ontologies representing courses and the organizational unit structure of universities show that our approach allows an agent to learn needed concepts and that what it learns is a compromise of what the other agents think about the concept in question (essentially the common ground of the agents regarding the concept). This is possible, even if the agents do not use the same features. The experiments also show that the use of ontology information allows for more accuracy in the learned concept if the number of exchanged examples is low, thus improving on the amount of information exchanged between agents.

2. BASIC DEFINITIONS

In this section, we will provide some basic definitions about ontologies and agents and we will also provide the instantiations of these concepts that we require for our methods.

2.1 Ontologies and Concepts

The usage of the term "ontology" in (computer science) literature faces problems very similar to the usage of the term "agent": there is no agreed-upon formal definition for the term, but nevertheless it is used very intensively and there are many systems out there that come with a (somehow) build-in definition of the term. Common to most usages of ontology is that it is considered a way for representing objects or concepts in a hierarchy with additional ways to define relationships between these objects/concepts.

This is very nicely reflected by Stume's formal definition (see [9]) who defines a *core ontology* as a structure $\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R)$. C and R are two disjoint sets and the elements of C are called *concept identifiers* and the elements of R are so-called *relation identifiers*. $\leq_C$ and $\leq_R$ are partial orders on C, respectively R, called *concept hierarchy* or *taxonomy* ($\leq_C$), respectively *relation hierarchy* ($\leq_R$). $\sigma : R \rightarrow C^+$ is a function providing a signature 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.

By viewing an ontology $\mathcal{O}$ as a structure on top of a set of concepts and a set of relations, this definition is able to cover a lot of the definitions of ontologies in the literature, but without providing a more precise definition of what concepts are, we are far from anything that can be used by agents to provide a basis for communication. Many works in databases and machine learning define concepts as collections of objects that share certain feature instantiations and for this work we will follow this guidance.

So, in the following we assume that we have a set $\mathcal{F} = \{f_1, ..., f_n\}$ of features and for each feature f_i we have its domain $D_i = \{v_{i1}, ..., v_{im_i}\}$ that defines the possible values the feature can have. Then an object $o = ([f_1 = v_1], ..., [f_n = v_n])$ 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 databases and 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 $C_k = ([f_1 = V_1], ..., [f_n = V_n])$ where $V_i = \{v'_{i1}, ..., v'_{ij_i}\} \subseteq D_i$ (if $V_i = D_i$ then we often omit the entry for f_i). An object $o = ([f_1 = v_1], ..., [f_n = v_n])$ 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.

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

$C_i \leq_C C_j$ iff for all $o \in C_i$ 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.

From the point of view of knowledge representation the really interesting part of ontologies are the relations in R that a particular ontology allows. This is also the part where we see a lot of differences between different authors. In general, all possible relations between tuples of concepts can be used in ontologies, but usually researchers assume a small set of build-in relations and tool developers sometimes throw in the possibility to have (limited) user-defined relations. But unfortunately, different ontologies can use the same relation identifiers for different build-in relations, so that there is quite some confusion in this area. So, if we have two systems build by different people using ontologies over the same set $\mathcal{U}$ then it is very important to either identify those relations that occur in both ontologies or to find ways how the knowledge contained in the (usage of) relations in one ontology can be used in communications between the systems. In this work, we will show such a usage for one relation that we have called `is-similar-to` with $\sigma(\texttt{is-similar-to}) \in C^2$.

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 $\mathcal{A}g = (Sit, Act, Dat, f_{Ag})$. 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 $f_{Ag} : Sit \times Dat \rightarrow Act$ applied to the current situation and the current values of its internal data areas.

For the agents that we are interested in, we can instantiate this definition a little bit more. We want to focus on the knowledge representation used by agents, so we start by looking more closely at Dat. We assume that every element of Dat of an agent $\mathcal{A}g$ contains an ontology area $\mathcal{O}_{Ag}$ as defined in the previous subsection that represents the agent's view and knowledge of concepts. For the concepts in the taxonomy of $\mathcal{O}_{Ag}$ there might be additional data, beyond features, that the agent requires from time to time (see Section 3). Naturally, 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 $\mathcal{O}_{Ag}$. But in the rest of this paper, we will concentrate on how the agent uses and manipulates its ontology.

The actions of an agent obviously depend a lot on the application area the agent is designed for. We require our agents to be able to communicate with other agents using information from the agent's ontology and to manipulate this ontology based on information received from other agents and the agent's own deduction actions that include actions performing learning. Again, we will provide more information in the next section.

An element of Sit usually contains parts representing observations of other agents and of the environment the agent is in. In this paper, we assume that among the observations of an agent are all messages send by other agents since the last situation an agent was in. And we will specify the parts of f_{Ag} that deal with the relevant messages and the relevant knowledge in and around the ontology of an agent in the next section.

3. LEARNING OF NEW CONCEPTS

The goal of this research is to develop a method how an agent can learn new concepts for its ontology with the help of other agents. This naturally assumes that not all agents have the same ontology (otherwise learning would not be necessary). In fact, we additionally assume that there are only some base features $\mathcal{F}_{base} \subseteq \mathcal{F}$ that are known, respectively 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 base common knowledge, individual agents may come with additional features they can recognize and additional concepts they know. Agents might refer to the same such features and concepts by different names and they may have features and concepts that have the same name but are not the same. While all the ontologies used by the agents will use as taxonomy the subset-relation, agents may use different other relations in their ontologies and two agents cannot rely on same relation identifiers referring to the same relation and vice versa, again.

Given this setting, agents will develop problems in working together, since the common grounds for communication are not there or too small. To solve 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. Our basic idea is to have an agent *learn* required concepts (or at least a good approximation) with the *help* of the other agents. Due to the potential differences in the ontologies of agents, objects that are positive and negative examples for a concept will play the major role in teaching an agent a new concept. We will assume in the following that the identifying name of an object is a feature in $\mathcal{F}_{base}$. We do not see this as a big limitation, since it is usually not too difficult to establish a clear identification of objects. For example, if the objects are part of the environment, pointing to a particular object is sufficient to identify it.

In the following, we will first present the general interaction scheme between agents that is our method to have an agent learn a particular concept. Then we will focus in more detail on the important steps in this interaction scheme.

3.1 Our general interaction scheme

Although we want all agents to be able to learn new concepts, for explaining our interaction scheme we designate one agent, $\mathcal{A}g_L$, as the one that wants to learn a new concept and the other agents, $\mathcal{A}g_1$,...,$\mathcal{A}g_m$, will be its teachers. $\mathcal{A}g_L$ has an ontology $\mathcal{O}_L = (C_L, \leq_C, R_L, \sigma_L, \leq_{R_L})$ and knows a set of features $\mathcal{F}_L$. Analogously, $\mathcal{A}g_i$ has as ontology $\mathcal{O}_i = (C_i, \leq_C, R_i, \sigma_i, \leq_{R_i})$ 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 $\mathcal{A}g_L$. Part of Act_L are actions `QueryConcept`, `AskClassify`, `Learn`, and `Integrate`, while part of the Act_is are the actions `FindConcept`, `CreateNegEx`, `ReplyQuery`, `ClassifyEx` and `ReplyClass`; all with appropriate arguments. These actions form our interaction scheme in the following manner:

1. $\mathcal{A}g_L$ determines it needs to know about a particular concept c_{goal} and performs `QueryConcept`("c_{goal}") to inform the other agents about this need.
2. Each agent $\mathcal{A}g_i$ reacts to $\mathcal{A}g_L$'s query by (see 3.3):
   (a) performing `FindConcept`("c_{goal}"), which leads to a set of candidate concepts C_i^{cand},
   (b) selecting the "best" candidate c_i out of C_i^{cand},
   (c) selecting a given number of elements out of $pex_i^{c_i}$, thus creating p_i,
   (d) performing `CreateNegEx`(c_i) to produce a given number of (good) negative examples for c_i, which we call the set n_i,
   (e) performing `ReplyQuery`($path(c_i)$,p_i,n_i).
3. $\mathcal{A}g_L$ collects the answers ($path(c_i)$,p_i,n_i) from all agents and uses a learner to learn c_{goal} from these combined examples (action `Learn`((p_1,n_1),...,(p_m,n_m))). If there are conflicts, then it resolves them with the help of the other agents using `AskClassify` (resp. `ClassifyEx` and `ReplyClass` by the other agents, see 3.4).
4. $\mathcal{A}g_L$ uses the learned c_{goal} and the collected $path(c_i)$s from the other agents to construct an ontology path C_{path} leading to c_{goal} (see 3.4) within its ontology $\mathcal{O}_L$ (action `Integrate`($path(c_1)$,...,$path(c_m)$)).

The result of this learning/teaching scheme is the description of c_{goal} in terms of $\mathcal{A}g_L$'s feature set $\mathcal{F}_L$ and an updated ontology $\mathcal{O}_L^{new} = (C_L^{new}, \leq_C, R_L, \sigma_L, \leq_{R_L})$. $\mathcal{A}g_L$ will also create a set $pex_L^{c_{goal}}$ in case another agent wants $\mathcal{A}g_L$ to teach it c_{goal}.

3.2 The initial query

In order to initiate the initial query, $\mathcal{A}g_L$ first needs to become aware that there is a concept that it needs to learn. There are a couple of scenarios that can lead to this realization. $\mathcal{A}g_L$ might observe a conversation between other agents in which an unknown concept identifier is used (or an identifier known by $\mathcal{A}g_L$, but in a way that does not make sense). Or $\mathcal{A}g_L$ might be dealing with a set of objects that share certain feature values from $\mathcal{F}_L$ and it wants to know if other agents know more about the similarities of these objects.

Based on these scenarios, we require action `QueryConcept` to have 3 parameters to be used to define c_{goal} to the other

agents:

`QueryConcept`(identifier,{$[f'_1 = V'_1]$,...,$[f'_l = V'_l]$},O_{goal}).
Here identifier is an element of C_i for some agent(s) $\mathcal{A}g_i$, $\{[f'_1 = V'_1],...,[f'_l = V'_l]\}$ is a selection of features $f'_j \in \mathcal{F}_{base}$ and the values $V'_j \subseteq D_{f'_i}$ that $\mathcal{A}g_L$ thinks are related to the concept c_{goal} and $O_{goal} \subseteq \mathcal{U}$ is a set of objects that $\mathcal{A}g_L$ thinks are covered by c_{goal}. Due to the different scenarios from above, each of the three parameters can be empty, if $\mathcal{A}g_L$ does not have any information on the parameter. Also, $\mathcal{A}g_L$ can decide to address only a subset of $\{\mathcal{A}g_1,...,\mathcal{A}g_m\}$ and then it can use as f'_is also features that are known to this subset and itself. If $\mathcal{A}g_L$ uses the identifier parameter, then there is naturally a chance that different agents use this identifier for different concepts. Again, $\mathcal{A}g_L$ might therefore address only some of the agents. Note that if the addressed agents associate different concepts with the identifier, then $\mathcal{A}g_L$ will end up with learning a subconcept of these concepts.

3.3 Answering the query

Due to our basic assumptions about what our agents are not having in common, a teacher agent $\mathcal{A}g_i$ without any additional knowledge about the ontology $\mathcal{O}_L$ of $\mathcal{A}g_L$ is rather limited in what it can put into an answer to the query by $\mathcal{A}g_L$. In fact, we are not even guaranteed that $\mathcal{A}g_i$ can really grasp what $\mathcal{A}g_L$ wants to know, since $\mathcal{A}g_L$ was already limited in its ways to express what concept c_{goal} it wants to know about. But sets of objects are something that all of our agents can communicate to each other, even if they might perceive these objects differently, and therefore we use mostly sets of objects in the answers that our teachers are creating. The answer of $\mathcal{A}g_i$ will also include information about the concept c_i it thinks $\mathcal{A}g_L$ wants to know about and how this concept is placed in $\mathcal{O}_i$, so that $\mathcal{A}g_L$ can use whatever information pieces it can understand (see Section 3.4).

After receiving the query from $\mathcal{A}g_L$, an $\mathcal{A}g_i$ first has to determine which of the concepts in C_i fits the query the best (i.e. what is c_i). Then it has to select positive and negative examples for this concept and finally it sends this information to $\mathcal{A}g_L$.

3.3.1 Finding the best known concept

The query from $\mathcal{A}g_L$ consists of three parts that it uses to describe what concept it wants to learn more about. Due to the differences between agents, each of these parts can 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. As a consequence, $\mathcal{A}g_i$ has first to collect 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, in its opinion, the best fit.

A concept is a candidate with respect to identifier, if its identifier is identical to the identifier in the query. A concept $c = \{[f"_1 = V"_1], ..., [f"_p = V"_p]\}$ is among the candidates due to the feature part of the query, if for all $f"_j$ with $f"_j \in \{f'_1, ..., f'_l\}$ –let us assume that $f"_j = f'_a$– we have that $V"_j \subseteq V'_a$. Finally, a concept is also a candidate, if it covers one of the objects in O_{goal}. Note that the last two conditions put all superconcepts of a concept from the candidate set also in C_i^{cand}.

There are many different ways how an evaluation of the candidates can be performed, especially if we allow for criteria coming from the application area the agents are working in. 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 identifier part does either produce a contribution or not. Every object in O_{goal} also is binary in its contribution. But this favors the more general concepts, so that some additional criterion is needed that makes the count of covered objects relative to the depth of the concept in the taxonomy tree. Finally, the feature part of the query suggests a good fit of a concept if it agrees with the feature values of many of the features used in the query. But this can also be strengthened by looking at how good the fit for a particular feature is.

We will present an example measure for how a concept fits a query in Section 4. Note that in theory different teacher agents could use different such measures. After the best concept c_i for the query is determined, an agent $\mathcal{A}g_i$ includes into its answer information on the path that leads in its taxonomy to c_i and the subtree below c_i's node (we refer to this information by $path(c_i)$).

3.3.2 Selecting positive and negative examples

Since each agent $\mathcal{A}g_i$ stores for each concept c_j in C_i a set $pex_i^{c_j}$ of positive examples for c_j, i.e. a set of objects covered by c_j, coming up with positive example objects for a concept known to $\mathcal{A}g_i$ is not a big problem. While the more examples normally are the better, in our case we have to take into account that the more objects from $pex_i^{c_i}$ are selected, the more expensive the communication becomes and the more effort $\mathcal{A}g_L$ will have to spent on learning. On the other side, less positive examples usually means a less precise result of the learning. Therefore we suggest to have the number of examples communicated to $\mathcal{A}g_L$ by each agent as a parameter of the whole system. Then selecting the appropriate number of elements for p_i is easily realized by randomly sampling $pex_i^{c_i}$.

Selecting negative examples for a concept is not as easy. Obviously, the set of negative examples nex^c for a concept c is defined as

$nex^c = \mathcal{U} - \{o | o \text{ covered by } c\}$.

This can be a very large set and usually different elements of this set provide learners with a different quality of advice. Good negative examples are examples that "nearly" are in the set covered by the concept, a kind of "near-misses" that allow to highlight the borders of a concept. The fact that our agents have ontologies allows us to do a better job in selecting negative examples than randomly selecting out of nex^{c_i} (by $\mathcal{A}g_i$). The key for this better selection is to make use of the taxonomy information $\mathcal{A}g_i$ has and the relations in R_i. The later naturally depends on what relations are available.

Let us first look at the possibilities that the *taxonomy* offers. Each superconcept of the concept c_i –that $\mathcal{A}g_i$ sees as the best concept to answer $\mathcal{A}g_L$'s query– can be used to limit the set of negative examples $nex_i^{c_i}$ that $\mathcal{A}g_i$ should consider for its answer. As a superconcept of c_i, these concepts share a lot of feature values with c_i, so that the elements in their set of positive examples that are not covered by c_i are good candidates for "near-misses". In fact, sibling concepts of c_i or its superconcepts are an even better source for negative examples since all their positive examples are not covered by c_i. Figure 1 indicates these candidates for the selection of negative examples with the help of taxonomy information

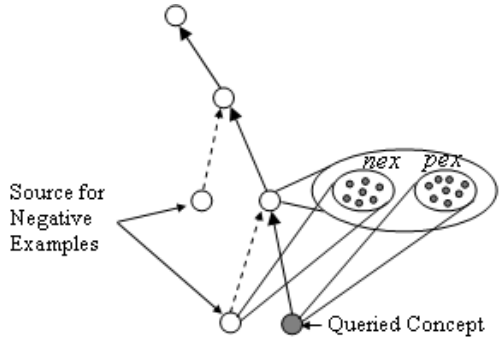

Figure 1: Negative examples using the taxonomy

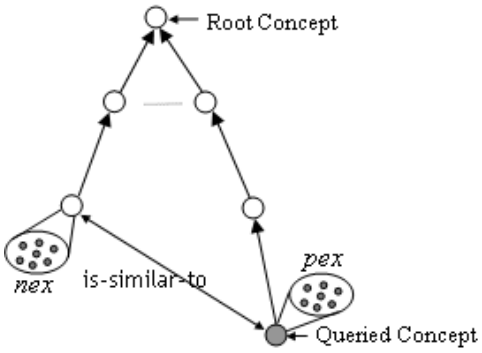

Figure 2: Negative examples using `is-similar-to`

(we use here and in the following the common graphical representation of taxonomies by trees).

Since all agents use the same relation $\leq_C$, all agents can use the taxonomy information to limit the pool of negative examples to choose from. But also information provided by some other relations can be used. As an example, let us look at the usage of the relation `is-similar-to` that we mentioned earlier. The motivation for `is-similar-to` is to allow to express the similarity between two concepts that are far away from each other in the taxonomy tree, but that share a lot of feature values. This makes `is-similar-to` a perfect candidate for helping with the selection of negative examples. Figure 2 shows how we can use concepts that c_i is similar to by using their positive examples as candidates for $nex_i^{c_i}$. After collecting all candidates in $nex_i^{c_i}$, we again select the given number of examples for n_i as a random sample.

Note that an `is-similar-to`-relation can be automatically computed for a given C_i and $\mathcal{F}_i$ by introducing a similarity measure sim_i^f on feature values for each feature $f \in \mathcal{F}_i$ with domain D: $sim_i^f : D \times D \rightarrow [0..1]$. We can create out of this a similarity measure $sim_i^{\mathcal{U}}$ for objects by, for example, summing up the similarities for each feature. More formally, let $o = ([f_1 = v_1], ..., [f_n = v_n])$ and $o' = ([f_1 = v_1'], ..., [f_n = v_n'])$, then

$sim_i^{\mathcal{U}} = \sum_{j=1}^{n} sim_i^{f_j}(v_j, v_j')$

where $sim_i^{f_j}(x, y) = 0$, if $f_j \neq \mathcal{F}_i$.

Out of this, we can create `is-similar-to`$_i$ between two concepts c and c', if $sim_i^{\mathcal{U}}(o, o') \geq simthreshold$ for all $o \in pex_i^c$ and $o' \in pex_i^{o'}$, with $simthreshold$ as a given parameter. While it would be better to use all objects covered by c and c', this can be impossible or at least very expensive, so that we suggest to use the examples that are already there.

3.4 Learning a new concept and path for $\mathcal{O}_L$

Learning a concept, in form of feature values, from a set of positive and negative examples is a problem that is very well researched in literature and there are many algorithms and systems available for this task. Examples are Rocchio (see [6]), k-Nearest-Neighbor (see [5]), and Naive Bayes (see [5]) in our experimental context.

With $pex^{c_{goal}} = \cup_{i=1}^{m} p_i$ and $nex^{c_{goal}} = \cup_{i=1}^{m} n_i$, we have the necessary input for such a learning system, with one potential problem: conflicts between the teacher agents. Due to the differences between agents it can easily happen that the best concepts c_i and c_j that $\mathcal{A}g_i$ and $\mathcal{A}g_j$ identified do not have much in common. 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. Obviously, such a contradiction will stop every learner. 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 $pex^{c_{goal}}$ while not including any objects in $nex^{c_{goal}}$. There are several ways how we can solve this problem and these ways allow us to produce different degrees of willingness to satisfy the teacher agents (by $\mathcal{A}g_L$).

For our system, we have chosen the following conflict resolution. After the learning component of $\mathcal{A}g_L$ has performed `Learn` and produced a more precise c_{goal}, $\mathcal{A}g_L$ will test all elements of $pex^{c_{goal}}$ and $nex^{c_{goal}}$ for correct classification by this new c_{goal}. For all the example objects that are not correctly classified, we get 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 have the majority decide how the conflict cases should be classified. Those examples that, according to the vote, are wrongly classified are fed back to the learner with priority. This is repeated until all examples are classified according to the votes. In order to avoid running in cycles we delete examples that keep coming up as misqualified from both example sets (which can be seen as imposing a partial blindness on $\mathcal{A}g_L$ to avoid the problem, but this is definitely something that we can observe in human beings also).

After having created the final description of c_{goal} in terms of $\mathcal{F}_L$, we could use the standard techniques for ontologies to integrate a single new concept into the taxonomy (see [12]). But since the goal of our whole method is to improve the communication between agents, we can use some more of the information provided by the teacher agents to *pre-structure* $\mathcal{A}g_L$'s ontology with concepts that $\mathcal{A}g_L$ will most probably have to learn if it communicates more intensely on the subject of c_{goal}.

The key information used in pre-structuring is the path information send by the teacher agents in their answers to the query. Our pre-structuring uses these paths to create *shells* for concepts that might be useful for future communications. These shells can also be used to indicate to an agent 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.

Each $path(c_i)$ from an agent $\mathcal{A}g_i$ contains the path in $\mathcal{O}_i$ leading to c_i and the taxonomy tree below c_i in $\mathcal{O}_i$ using the features from $\mathcal{F}_i$ for characterization. We first strip all concepts in $path(c_i)$ that are not in C_{base} of all features

that are not in $\mathcal{F}_{base}$. This creates what we call a concept shell. We then merge shells that are identical in their feature descriptions, both within one $path(c_i)$ and between paths from different agents. Then we combine the shells from all the teachers into one path for $\mathcal{O}_L$ (if there are concept shells that are not comparable with respect to $\leq_C$, then we select the longest path from a concept in C_{base} to c_{goal} that can be formed using the concept shells available; for concept shells smaller than c_{goal} with respect to $\leq_C$ we just create the subtree) and integrate this path into $\mathcal{O}_L$.

4. AN EXAMPLE APPLICATION

Before we present an example for a, in our opinion, very successful application of our ideas from the last section, let us point out that it is relatively easy to construct scenarios in which these ideas may not be very useful. These scenarios have in common that the teacher agents do not only disagree on what concepts fulfill a particular query, their ontologies and their perception of the world, i.e. their $\mathcal{F}_i$'s, are totally different. As a result, their attempts to teach an agent result in total confusion.

If, for example, all teacher agents associate with a particular concept identifier totally different real concepts, then the learning agent will not be able to get useful concepts out of their information. If $\mathcal{F}_{base}$ and C_{base} are empty or nearly empty and $\mathcal{F}_i \cap \mathcal{F}_j = \emptyset$ (or near empty) for all pairs of agents ($\mathcal{A}g_i$,$\mathcal{A}g_j$) (including $\mathcal{A}g_L$), then there is simply no common ground for a successful learning experience. Obviously also a human agent would find it impossible to learn useful information from the teachers in this situation.

With the following application we have chosen an area where the teacher agents have some differences in their "world view", simply because there are different ways how to organize the objects in the world, but where there is nevertheless a large agreement on many things.

4.1 The University Units and Courses Domain

For evaluating our method, we have chosen the course catalog ontology domain (see [2]). The set of objects $\mathcal{U}$ consists of files describing the courses offered by Cornell University, the University of Washington and the University of Michigan. The domain is additionally structured according to the university units of these universities, which creates different ontologies for each of them. In fact, our teacher agents will be agents that each represents one of these 3 universities ($\mathcal{A}g_C$, $\mathcal{A}g_W$, $\mathcal{A}g_M$). The course files (and unit structure) for Cornell and Washington were taken from [2], the ones for Michigan from their web site at [10]. A course file contains course identifier, course description and the prerequisites of a course. Note that this is not a small domain. The three universities together offer 19061 courses and each university's ontology has at least 166 concepts on top of their courses. For each of the following examples, our agents used their full ontologies even if we report only on parts of them.

To represent the courses in terms of features, we had a little bit of preparation to do, borrowing ideas from the field of information retrieval. Obvious features describing a course are its identifier f_{idnt}, a string, and its prerequisites f_{prereq}, a set of course identifiers. Different universities use different systems to create identifiers, so that these features are not really of any help for our purpose. The course description usually determines by which organizational units a course should be taught and the descriptions are text-based.

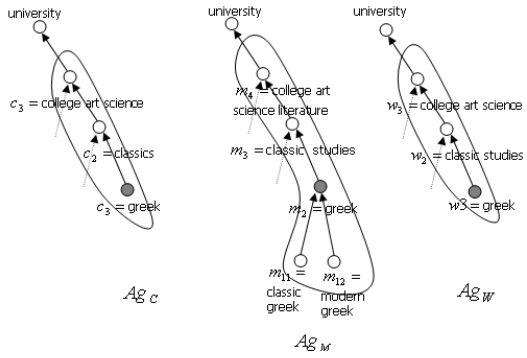

Figure 3: Relevant taxonomy paths of teachers

Defining concepts based on objects that consist of natural language texts is not easy, but an area of quite a lot of interest and practical applications. One way of defining features for such texts to group them is to look for particular words in the texts or word combinations (see [4]). Unfortunately, there is a lot of substitutivity in these word combinations, so that we need features that allow us to express this substitutivity. For example, feature $f_{\texttt{picture,photo,figure}}$: text $\rightarrow$ Boolean is true for a text t, if either `picture` or `photo` or `figure` occurs in t. For our application domain, it is not clear what substitutivities should be considered (just synonyms are not what we are looking for here), so that we base our features for the course descriptions on what we call a set $\mathcal{K}$ of key words. Then we have a feature for each possible subset of $\mathcal{K}$ (excluding the empty set) as described above. Different key sets create different feature sets.

4.2 An example query

To provide a better picture of how our method works, we take a look at a particular query, the relevant parts of the ontologies of the three teacher agents and what an agent can learn from these teachers with regard to this query. Let us assume that the learning agent is supposed to provide someone at a university with suggestions for how a unit concerned with `Greek` should be characterized. This learning agent would pose a query based on providing a key set out of its own key set of words, in our example this query key set would be {`greek, program, attic, literature`} (as stated above, using an identifier does not make much sense here and we are not using example objects for this example due to lack of space). Let us further assume that the relevant concepts in C_{base} are C_{base} = {university} and the relevant concepts in $\mathcal{F}_{base}$ are created using the key set $\mathcal{K}_{base}$ = {`class, course, program, literature, modern, attic, classic, culture, prose, graduate, seminar, grammar, drama, greek`}.

Obviously, different universities can use different key sets of words to express the area a course belongs to, so that this creates additional features for the different agents. In our example, the relevant key sets for the features used in the agents' ontologies are $\mathcal{K}_C$ = {`democritus`}, $\mathcal{K}_W$ = {`tragedy, orator, antique`} and $\mathcal{K}_M$ = {`modern, epic, classic, odyssey, ancient, aristotelian`}. By relevant we mean those key words that really occur in the features that these agents use to characterize the concepts that are related to the query. To make things interesting, we give $\mathcal{A}g_L$ as key set $\mathcal{K}_L$ = {`epic, antique, ancient, tragedy,`

Table 1: Usage of key words in features

key word	$\mathcal{A}g_C$	$\mathcal{A}g_M$	$\mathcal{A}g_W$	a)	b)	c)
`literature`	6	19	6	6	12	15
`greek`	8	55	9	7	26	33
`attic`	3	8	12	3	11	22
`epic`	0	3	0	0	3	3
`antique`	0	0	1	0	0	1
`ancient`	0	6	0	0	4	4
`tragedy`	0	0	3	0	0	3
`orator`	0	0	1	0	0	1

`orator`}.

Figure 3 presents the paths in the taxonomies of the teachers that our system selected as the best concepts for the query. The measure used by the system was calculated as $w_{ident} \times P_{ident}(c) + w_{feat} \times M_{feat}(c) + w_{obj} \times M_{obj}(c)$. w_{ident}, w_{feat} and w_{obj} are weight parameters. In our experiments we used $w_{ident} = w_{feat} = w_{obj} = 1$. $P_{ident}(c)$ is 1, if the concept c's identifier is equal to the identifier in the query and 0 else. The submeasure $M_{obj}(c)$ counts the number of objects from the query covered by c and multiplies it with the length of the path to the concept in the taxonomy. This is then divided by the product of the number of objects in the query and the maximal length of a path in the taxonomy. The submeasure $M_{feat}(c)$ makes use of the key set in the query. For every feature that is true for c, we check if it is formed solely by words from the key set. $M_{feat}(c)$ is the number of these features divided by the number of all features that are true for c.

The set of relevant features for this example is still too big to be easily presented. We used a learner based on Naive Bayes (as described in [4]). This learner can, in fact, use the key sets directly and creates the features implicitely. To provide an idea on how the teachers influence what $\mathcal{A}g_L$ learns, let us look at how many features include some of the key words. As Table 1 shows (a) to c) refer to the concepts learned according to Figure 4, where a) is learned from $\mathcal{A}g_C$, b) from $\mathcal{A}g_C$ and $\mathcal{A}g_M$ and c) from all agents), the difference between the features the different agents know results in rather different numbers for the usage of some key words to make up for words that an agent cannot use in its features. This is especially obvious in the base key set, as the first three examples show.

As the above already suggests, the learned concept c_{goal} is different for different agents used as teachers. For example, with just $\mathcal{A}g_C$ as teacher, among the features enabled in c_{goal} we have $f_{\texttt{attic,prose,greek}}$, which is not enabled by any other agent and also not enabled in the learned concept for the other two scenarios. In the concept learned from $\mathcal{A}g_C$ and $\mathcal{A}g_M$ together, we have the feature using the words `classic, greek, epic, modern, ancient` and `literature` enabled, which is not enabled in the other two scenarios. Finally, learning from all three teachers results in a feature using `greek, tragedy, antique, orator` and `attic` enabled, which is, again, not used in cases a) and b). Remember that an enabled feature means that each course covered by the concept that uses the feature contains at least one word from the set of words associated with the feature!

Figure 4 shows the new taxonomy paths created by prestructuring. The teachers agree on the need for two superconcepts for c_{goal} and $\mathcal{A}g_M$ introduced two subconcepts that are added if $\mathcal{A}g_M$ is one of the teachers. Naturally, we can not associate meaningful names with these concepts

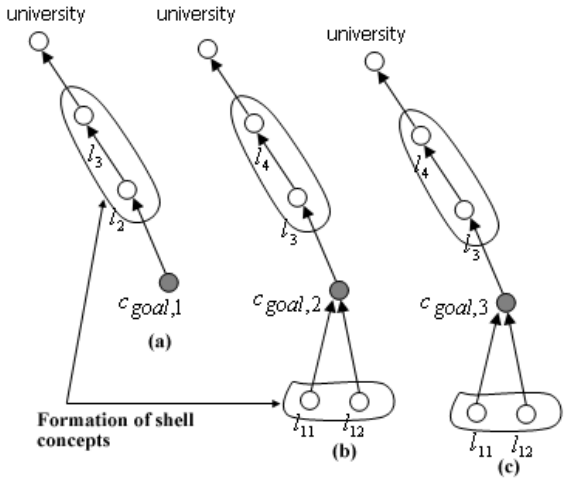

Figure 4: Query results

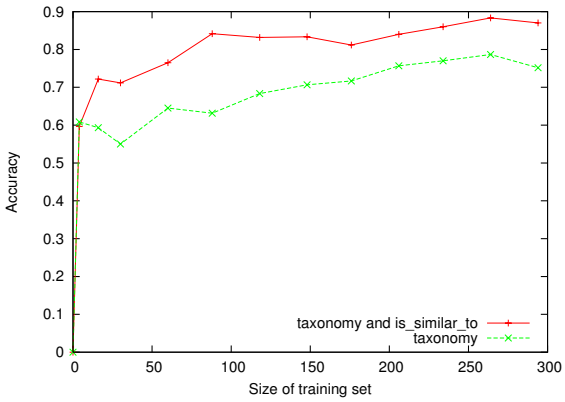

Figure 5: Accuracy results for Mathematics

and, again, there are too many features identifying them to represent them here.

4.3 Using more than the taxonomy

An important question to evaluate our concept is how efficient it is to have an agent learn a new concept. Clearly, the more examples are provided to this agent the better it can learn. But more examples mean more communication overhead, more learning effort and more chances for confusion between the different teachers. We tested several queries in our example domain and present here a typical example. The query is consisting of the key word set {`mathematics, program, arithmetic`} as the feature part and one Mathematics course from each of the three teachers in the example part of the query. Obviously, this query should result in learning the concept "Mathematics Department". Each data point in Figure 5 represents the average value of 5 runs, since the selection of positive and negative examples is performed randomly by the teachers out of the sets they consider (see Section 3.3.2).

Since there is not really a total agreement on what courses constitute Mathematics (beyond a certain strong core, naturally) there is no possibility to achieve 100 percent accuracy. In fact, it can be already debated what accuracy

means in our context, since there is no clearly defined concept $\mathcal{A}g_L$ is supposed to learn. For our experiments, every course classified under Mathematics by one of the 3 teachers was expected to be a positive example, all others a negative one. Naturally, this is not how our approach classifies, but, as Figure 5 shows, we come rather near to that. Figure 5 also shows that using the `is-similar-to`-relation greatly enhances accuracy (especially for small numbers of exchanged examples), due to providing well focused negative examples. $\mathcal{A}g_C$ has in its ontology that Mathematics is similar to theoretical-and-applied-mechanics, computer-science, operations-research-and-industrial-engineering, while $\mathcal{A}g_M$ has it similar to statistics, geological-sciences and astronomy, and $\mathcal{A}g_W$ to statistics, aeronautics-and-astronautics and earth-and-space-sciences. Especially for the range between 50 and 100 training examples, which means 17 to 34 examples per teacher agent, the usage of `is-similar-to` clearly pays off. Note that the "dips" in accuracy are not only due to presenting an average here, more examples also means that more conflicts between teachers occur (remember that using the test set as we do means that a conflict always results in a potential misclassification).

5. RELATED WORK

As already mentioned, most works in literature assume that agents use a common ontology (see, for example, [1]). The works by Williams (see [11]) 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. No negative examples are used and there is naturally no pre-structuring and no need to deal with conflicts between teachers. [7] 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.

Steels (see [8]), 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, 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.

6. CONCLUSION AND FUTURE WORK

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. We also showed in experiments that the use of ontologies allows for a more targeted selection of negative examples resulting in a good understanding of a concept after relatively few exchanged examples.

This work represents a proof-of-concept, not more. We have indicated various parts of our method where alternative realizations of these parts are possible and future work should explore these possibilities. Especially the conflict handling allows for many different ways that could create alternative levels of inclusion of examples. For example, we could have agents vote on every communicated example, not only the clearly problematic ones. Also, selection, resp. filtering, of the available teachers is an interesting future topic. There is also the need for more explorations with the current application, for example by allowing for learning of new features as in [8], and for instantiating our method for other applications.

7. REFERENCES

[1] T. Finin, Y. Labrou, J. Mayfied: KQML as an Agent Communication Language, in J. Bradshaw (Ed.), Software Agents, MIT Press, 1997.

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

[3] K-C. Jim, C.L. Giles: Talking Helps: Evolving Communicating Agents for the Predator-Prey Pursuit Problem, Artificial Life 6(3), 2000, pp. 237–254.

[4] D. Koller, M. Sahami: Hierarchically Classifying Documents Using Very Few Words, Proc. ICML-97, 1997, pp. 170–178.

[5] T.M. Mitchell: Machine Learning, McGraw-Hill, 1997.

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

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

[8] 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.

[9] G. Stumme: Using Ontologies and Formal Concept Analysis for Organizing Business Knowledge, in J. Becker, R. Knackstedt (Eds.): Wissensmanagement mit Referenzmodellen – Konzepte für die Anwendungssystem- und Organisationsgestaltung, Physica, 2002, pp. 163–174.

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

[11] A.B. Williams: Learning to Share Meaning in a Multi Agent System, Autonomous Agents and Multi Agent Systems 8(2), 2004, pp. 165–193.

[12] A.B. Williams, A. Padmanabhan, M.B. Blake: Local Consensus Ontologies for B2B-Oriented Service Discovery, Proc. AAMAS-03, 2003.