Improving Example Selection for Agents Teaching Ontology Concepts

Mohsen Afsharchi and Behrouz H. Far

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

Abstract. We present a method to improve the positive examples selection by teaching agents in a multi-agent system in which a team of agent peers teach concepts to a learning agent. The basic idea in this method is to let a teacher agent expand the features it uses to describe a concept in its ontology by additional features. This resembles the typical behavior of human teachers who describe concepts from different viewpoints in the hope that one of these viewpoints comes close to the viewpoint of a learner. The extended feature set is then used to select positive examples that together with negative examples are communicated to the learner agent. The learner uses concept learning techniques to integrate the new concept into its own ontology. An experimental evaluation shows a significant learning improvement compared to the previous approach.

1 Introduction

Knowledge sharing is an integral property of multi-agent systems (MAS). In the past, knowledge sharing among agents was usually assumed to be instantaneous and fail safe and therefore, if one agent learns something all the others have learned it. Recently many researchers have argued that if two agents have different internal knowledge representations (e.g. distinctive ontologies) the knowledge sharing is hard to be accomplished. Therefore the idea of having agents *learn* concepts from the other agents has been suggested [1, 7, 9, 11]. In our recent work, we have presented a general method for having an agent learn concepts and their features from several other agents [1].

In this method an agent (learner agent) that wants to learn a concept will query the other agents (teacher agents) about this concept by providing features (and their values) or examples that it thinks are associated with the concept. Then the teacher agents provide the learner with a set of positive and negative examples from their understanding of the concepts (i.e. concepts known by them) that seem to fit the query. These sets are further analyzed by the learner using several concept learning techniques to get a better understanding of the concept. Better understanding is equivalent to (a) identifying concepts' relevant features and (b) identifying the proper location of the concept in the concept hierarchy.

Note that the effectiveness of the learning strongly depends on the precision of the positive and negative examples that the peer agents send to the learner.

These examples should be selected in a way that covers the broad area of the concept being learned from different viewpoints. Of special importance are the positive examples that they should provide the learner with all features that discriminate the particular concept from other concepts.

When human tutors teach concepts to human learners, they usually explore and explain the concept from various viewpoints, so that the learner can select the viewpoint that fits best into his/her own view of the world. This essentially means that the teacher should investigate alternative ways to characterize a concept.

In this paper, we improve our general concept teaching/learning scenario by devising mechanisms to help teacher agents create alternative viewpoints of a concept in order to improve the selection of positive examples. More precisely, we use ideas from the area of feature selection in text classification (see [4, 12]) to have a teacher agent find all the characteristic features of a concept using the examples this teacher associates with the concept. Then the teacher ranks theses examples associated with the concept based on the characteristic features and sends the highest ranking example to the learner.

The structure of this paper is as follows. After a brief introduction of the concepts in Section 2 the learning process is described in Section 3 and a method for positive example selection using added features and example ranking is introduced in Section 4. An intuitive example using real data set is given in Section 5 followed by related works and conclusions in Section 6 and 7.

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

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 that come with a (somehow) build-in definition of the term. Common to most usages of the term ontology is that it is considered to be a way for representing concepts (or objects) in a hierarchy with additional ways of defining relationships among the concepts (or objects).

This is reflected by Stume's formal definition (see [9]) who defines a *core ontology* as a structure $\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R)$. Where 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*. $\leq_C$ is a partial order on C called *concept hierarchy* or *taxonomy* and $\leq_R$ ia a partial order on R, named *relation hierarchy*. $\sigma : R \rightarrow C^+$ 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. 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.

The Stume's definition of ontology $\mathcal{O}$ lacks the precise treatment of the conecpts, C. Many works in databases and machine learning define concepts as collections of objects that share certain *feature* instantiations. In the following we assume that we have a set of features $\mathcal{F} = \{f_1, ..., f_n\}$ 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.

Note that the objects must possess all the features of the concept C_k in order to be covered by it. But it does not necessarily mean the objects should not possess other features that not exhibited in the C_k. As we stated previously the $V_i \subseteq D_i$ and that means the V_i is not necessarily equals with D_i. This naturally allows the objects to have an extended set of features resulting in a potential ability to teach the concept from a different viewpoint.

From the point of view of knowledge representation the really interesting part of ontologies are the relations R that a particular ontology allows. This is also the part where we see a lot of discrepencies among 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. Therefore, if we have two systems build by different people using ontologies over the same set $\mathcal{U}$ it is very important to either identify those relations that occur in both ontologies or to find ways 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 $\sigma($`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.

As we want to focus on the knowledge representation used by agents, so we look 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. 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 $\mathcal{O}_{Ag}$. In the rest of this paper, we will concentrate on how the agent uses and manipulates its ontology.

3 Learning Process

In this section we provide a brief discussion about the multi-agent concept learning we presented in [1]. We will see how we changed this general scheme to improve positive example selection in Section 4.

We have developed a method that demonstrates 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 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 base 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 there. 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. 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.

3.1 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:
   (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.
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} 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 Selecting Positive and Negative Examples

While supplying the learner with more examples normally are the better, in our case we have to take into account that the more objects from the positive and negative examples 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 hand, less examples usually means less precise learning result. Therefore the number of examples communicated to $\mathcal{A}g_L$ by each agent should be selected as a parameter of the whole system.

For negative examples, since every concept other than queried concept c_j can be categorized as its counter concept, the number of associated objects (which naturally are negative examples) could be potentially very high. This big volume of the negative examples makes the selection of a subset of them a

crucial task.As we elaborated negative example selection and improvement and reported the result [1], in Section 3.3 we briefly explain this problem.

For positive examples, Since each agent $\mathcal{A}g_i$ stores for each concept c_j a set $pex_i^{c_j}$ of positive examples , i.e. a set of objects covered by c_j, coming up with positive example objects for a concept known to $\mathcal{A}g_i$ does not seem to be a big problem, because selecting the appropriate number of elements for p_i could be realized by randomly sampling $pex_i^{c_i}$. But as we will show in Section 4, applying the *viewpoint* of the teacher agents to select better positive examples can make a significant improvement in the learner's effectiveness. In Section 4 we present two algorithms to extract discriminative features and ranking positive examples respectively.

3.3 Selecting Negative Examples by Ontology Guidance

Selecting negative examples for a concept is not 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 even better source for negative examples since all their positive examples are not covered by c_i.

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. 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 $f_j \notin \mathcal{F}_i$.

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.

4 Positive Example Selection by Discriminative Feature Selection and Example Ranking

Technically, there is a set of objects associated with each concept c_j for the teacher agent $\mathcal{A}g_i$ as positive examples, and if c_j is selected as an answer to a query, this set is simply available for the teacher to select positive examples and send it to the learner. Randomly selection of the positive examples is the most straightforward way which while keeps the selection process easy, does not guarantee the comprehensiveness of the positive examples. That is because the set of positive examples should cover the border of the concept as well as the body of the concept. Therefore good subset of positive examples are examples that cover the whole space of positive examples.

One very important issue here is that the selection of positive examples is the point that the teacher can exert its unique view in the teaching of a specific concept, therefore, the teacher agent should utilize some methods to reflect its viewpoint. Similar to the teaching process in human beings we used the feature describing a concept as a point that the teacher can express its viewpoint. Apart from the features in the concept definition in the ontology, there might be some other very characterizing features in the positive examples which the teacher agent can rely on in the teaching of the concept by selecting the positive examples using them. We believe that these characterizing features are the features that are more discriminatory than other features in the examples. Fortunately, there is a very close relation between the technical problem we mentioned in the previous paragraph and the teaching from different viewpoints. By selecting the subset of positive examples using more discriminative features, the teacher agent not only exerts its unique point of view, but it has a criterion to arrange the selected subset in a very comprehensive way.

To identify discriminative features and select examples based on them, we introduced a new action `SelectPosEx`(c_i) and replaced the section (c) of step 2 of our general interaction scheme as follows: performing `SelectPosEx`(c_i) to select a given number of good elements out of $pex_i^{c_i}$, thus creating p_i. In `SelectPosEx` we use the differences of features between the given positive examples ($pex_i^{c_k}$) and negative examples ($nex_i^{c_k}$) to calculate the *feature strength* in discrimination between the positive and negative examples. We also use feature strength to identify more discriminative features which we call them *core features*, and

denote them by $\mathcal{CF}$. Then we use $\mathcal{CF}$, to extract good positive examples from $pex_i^{c_k}$ and we call them *distinctive positive examples*(p_i).

4.1 Identifying Discriminative Features

We identify the discriminative features based on the notion called *Relief* which we borrowed the idea from [5] Using *ReliefF* which is a more robust algorithm from Relief family we developed an algorithm to identify the discriminative features. This algorithm constructs the set of core features of pex, $\mathcal{CF}$, by ranking the feature strengths among the features that are exhibited in pex and nex.

The key idea of our method , given in Algorithm 1, is to estimate the strength of features according to how well their values distinguish between examples that are near to each other. For that purpose, given a randomly selected example e_i (line 3), algorithm searches for its k nearest neighbors from the pex, called nearest hit P, and k others from the nex, called nearest miss N (line 4). It updates the strength estimation $W[F]$ for the set of all features F depending on their values for e_i, P, and N (lines 6 and 7). If e_i and majority of k examples in P are different in their values for the feature f then the the feature f separates examples in the same concept which is not desirable so we decrease the strength estimation $W[f]$. On the other hand if e_i and majority of k examples in N are different in their values for the feature f then the feature f separates a positive example from negative examples which is desirable so we increase the strength estimation $W[f]$. The k is a user definable parameter which increase the robustness of the algorithm against the noisy data. The whole process is repeated for m times, where m is also a user defined parameter.

Function $diff(f, e_i, e_j)$ calculates the difference between the values of the feature f for two instances e_i and e_j. For nominal features it is define as:

$$diff(f, e_i, e_j) = \begin{cases} 0; value(f, e_i) = value(f, e_j) \\ 1; otherwise \end{cases}$$

and for numerical features as:

$$diff(f, e_i, e_j) = \frac{|value(f, e_i) - value(f, e_j)|}{\max(f) - \min(f)}$$

The algorithm then determine ϕ as the average of $W(F)$. Obviously because the teacher is interested in the feature in pex it filters the set of features and add to the $\mathcal{CF}$ all features which are seen at least in one e in pex and $W(f) > \theta$.

4.2 Extracting Distinctive Positive Examples

We have shown how to compute feature strengths and determine ϕ so as to select a set of discriminative features for formulating the core features ($\mathcal{CF}$) of

Algorithm 1 Calculate the vector of W of estimations of the features strenght

1. set all weights $W[F] := 0.0$
2. **for** $i = 0$ to m **do**
3. Randomly select an example e_i
4. find k nearest hit examples in pex, P
5. find k nearest miss examples in nex, N
6. **for all** f in F **do**
7. $W[f] = W[f] - \sum_{j=1}^{k} diff(f, e_i, P_j)/(m \cdot k) + \sum_{j=1}^{k} diff(f, e_i, N_j)/(m \cdot k)$
8. **end for**
9. **end for**
10. $\phi \frac{1}{|F|} \sum_{i=1}^{|F|} W[f_i]$
11. **for all** f in F **do**
12. **if** $f > \phi$ and $f \in pex$ **then**
13. append f to $\mathcal{CF}$
14. **end if**
15. **end for**

the positive examples . Another important issue is, given an example, what is the criterion, in order to consider it a potential distinctive positive example?

Algorithm 2 shows the mechanism that we utilized to select the set of distinctive positive examples. The key idea of our procedure , given in Algorithm 2, is to estimate the distance of every positive examples from their peers in the negative side and use this estimation to assess the distribution of the examples in the whole space of positive examples. For that purpose, for every example e_i (line 3), algorithm searches for its k nearest neighbors from the nex, called nearest miss N (line 4). To find theses nearest misses we used the similarity function that we presented in the previous section. the algorithms then updates the distance estimation $\mathcal{D}[e_i]$ for the set of all features $\mathcal{CF}$ depending on their values for e_i and each example in N using the $diff$ function(lines 5, 6 and 7). Obviously the examples with minimum value of $\mathcal{D}[e_i]$ are in the border and as the value goes higher the examples go farther from the border. In order to select more comprehensive set of positive examples, the teacher agent selects the examples that are not very close to each other assuming that the close examples do not add so much to the learner knowledge and its accuracy. The $dist$ function calculates the distance of the selected example e_j with candidate example e_i. If the value of distance is greater than θ for all selected examples, then teacher adds it to the set of selected positive example p (line 12 and 13). The value for θ is selected based on the number of examples the teacher can send to the learner and the average distance between examples.

Function $diff(f, e_i, e_j)$ is defined similar to the previous section and function $dist$ is defined based on the $diff$ function as follows:

Algorithm 2 Select the set of p of comprehensive positive examples

1. set all distances $\mathcal{D}[pex] := 0.0$
2. **for** $i = 0$ to $|pex|$ **do**
3. select an example e_i
4. find k nearest miss examples in nex, N
5. **for all** f in $\mathcal{CF}$ **do**
6. $\mathcal{D}[e_i] = \mathcal{D}[e_i] + \sum_{j=1}^{k} diff(f, e_i, N_j)/k$
7. **end for**
8. **end for**
9. $p = \emptyset$
10. **while** there is e_i in pex **do**
11. take out of pex an example e_i such that $\mathcal{D}[e_i]$ is minimum
12. **if** $dist(e_i, e_j) > \theta$ for all $e_j \in p$ **then**
13. append e_i to p
14. **end if**
15. **end while**
16. return p

$$dist(e_i, e_j) = \sum_{k=1}^{|\mathcal{CF}|} diff(f_k, e_i, e_j)$$

5 Experimental Results

We have conducted several experiments using the general setup of our multi-agent system from [1], in which 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. We changed the process of positive example selection to enable the teacher agents to reflect their specific "viewpoint" by selecting some examples that they *think* are more distinctive positive examples.

5.1 The University Units and Courses Domain

The Course catalog ontology domain has been chosen as the basic set up for our multi-agent system. (see [3]). 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 [3], the ones for Michigan from

their web site at [10]. A course file contains course identifier, course description and the prerequisites of a course. The three universities together offer 19061 courses(which naturally is the total number of examples in the system) 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.

Borrowing some ideas from the field of information retrieval, to represent the courses in terms of features, we had a little bit of preparation to do. The course description, that is the main feature in our method of learning, usually determines by which organizational units a course should be taught and the descriptions are text-based. 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.

To instantiate Algorithm 1 and 2 for our context we first used the similarity function of section 3.3 to find k nearest hit for each document example e_i. The same process is done to find k nearest miss document. To calculate feature weights, W, we needed to realize a $diff$ function which was compatible with our context. In the information retrieval domain one common scheme, to weight a keyword is known as "term frequency inverse document frequency" and originally for a keyword(i.e term) i in document j is defined as:

$$\omega_{i,j} = tf_{i,j} * \ln \frac{N}{n}$$

where $tf_{i,j}$ is the frequency of the keyword i in document j, N is the total number of documents which naturally in our context is the number of documents both in pex and nex, and n is the number of documents where keyword i occurs at least once. To make this weighting scheme works with our features, we substitute keywords by the set of keywords that we use for our system. Therefore for example in addition to "differential" and "equation" we also count the occurrences of "differential equation" in documents as a single feature. Based on this definition we instantiate the $diff$ function as follows:

$$diff(f, e_i, e_j) = |\omega_{f,e_i} - \omega_{f,e_j}|$$

5.2 Different Positive Example Selection Comparison

To show the efficiency of the learner when a concept thought from different viewpoints, we conduct some interesting experiments. For first experiment, sim-

Table 1. Comparison of positive example selection mechanisms for `Greek`

n%	Distinctive Set	Random Set1	Random Set2	Random Set3
10	0.736	0.627	0.542	0.631
20	0.770	0.639	0.563	0.637
30	0.792	0.707	0.572	0.652
40	0.830	0.712	0.603	0.703
50	0.861	0.737	0.645	0.729
60	0.890	0.722	0.669	0.748
70	0.912	0.794	0.704	0.733
80	0.908	0.802	0.737	0.761
90	0.902	0.783	0.719	0.799
100	0.904	0.796	0.741	0.811

Table 2. Comparison of positive example selection mechanisms for `Mathematics`

n%	Distinctive Set	Random Set1	Random Set2	Random Set3
10	0.702	0.691	0.611	0.707
20	0.773	0.719	0.658	0.712
30	0.824	0.721	0.693	0.767
40	0.820	0.752	0.704	0.765
50	0.823	0.798	0.768	0.794
60	0.835	0.811	0.790	0.819
70	0.844	0.813	0.789	0.824
80	0.855	0.828	0.808	0.836
90	0.910	0.852	0.824	0.830
100	0.931	0.860	0.842	0.852

ilar to our experiment in [1], we assumed 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`}. We also further assumed 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`}.

Based on the above mentioned assumptions, we enabled our agents to apply Algorithm 1 to come up with the core features representing the unique viewpoint of each agent. A subset of the core feature key set which is not common with $\mathcal{K}_{base}$, for each agents were as follows:
$\mathcal{CF}_C$ ={`democritus,religion,english,herodotus,medieval`}
$\mathcal{CF}_W$ = {`tragedy,orator,antique,myths,archeology`}
$\mathcal{CF}_M$ = {`modern,epic,classic,odyssey, ancient,aristotelian,aeneid`}.
Then we let the agents to extract positive examples using $\mathcal{CF}$ and Algorithm 2 and send them to the learner. In the learner side and in order to evaluate the efficiency of our method in selecting distinctive positive examples, we first

Table 3. Comparison of positive example selection mechanisms for nine concepts

n%	Distinctive Set	Random Set1	Random Set2	Random Set3
10	0.572	0.447	0.439	0.509
20	0.618	0.511	0.483	0.525
30	0.614	0.530	0.529	0.520
40	0.648	0.534	0.531	0.605
50	0.686	0.548	0.576	0.628
60	0.729	0.581	0.612	0.698
70	0.740	0.611	0.627	0.711
80	0.768	0.674	0.684	0.754
90	0.806	0.745	0.709	0.788
100	0.839	0.767	0.723	0.804

trained the learner with the set of distinctive examples and test it against the set of all 1016 positive examples associated with concept `greek` in the world of three agents. Table 1 shows the percentage of the true classification that the learner did. To see how our method improves the efficiency of whole process of learning we also trained the learner with three different random set(e.g. Random Set 1,2,3) of positive examples(which obviously associated with concept which is being learned). As Table 1 shows the major improvement in the classification where the accuracy of Distinctive Set (90.4 %) is 9.3% more than the best random set(Random set 3 - 81.1%) . The experiment is repeated n times, where n is the percentage of the positive examples used in training.
We repeated the experiment for the concept `mathematics` with query key set: {`mathematics,program,science,calculus`} and the key set $\mathcal{K}_{base}$ = {`class, course, program, science, calculus, mathematics, school, graduate, seminar,systems,number, solution`}. A subset of core features for each agents were as follows:
$\mathcal{CF}_C$= {`vector,elementary,statistics,geometry,function,proof`}
$\mathcal{CF}_W$ = {`equation,logic,linear,fourier,integral `}
$\mathcal{CF}_M$ = {`algebra,matrices,graph,theorem,dynamics,logarithm `}.
Table 2 shows the major improvement(93.1%-86.0% =7.1%) in the classification when we test the learner over 2117 positive examples associated with concept `mathematics` in the world of three agents. In addition to `mathematics` and `greek` we repeated our experiment for seven other concepts: `computer science, linguistics, german, japanese, chemistry, physics` and `chinese`. Table 3 shows the average result for nine concepts. Again we see a significant improvement in classification accuracy of 3.5%(83.9% - 80.4%).

6 Related Works

Most works in the multi-agent concept learning did not focus on the quality of the positive and negative examples. The Williams's work [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. Also there is no concentration on the quality of examples. [7] presents a method how one agent can train another agent to recognize a concept by providing selected positive training examples. while the multi-agent dimension is not addressed and no usage of ontologies is made, the quality of examples also not addressed.

Researchers have studied various aspects of feature selection. Different feature selection methods can be broadly categorized into the wrapper model [4] and the filter model [8, 6]. The wrapper model uses the predictive accuracy of a predetermined learning algorithm to determine the goodness of the selected subsets. The filter model separates feature selection from classifier learning and selects feature subsets that are independent of any learning algorithm. It relies on various measures of the general characteristics of the training data such as distance, information, dependency, and consistency. According to the availability of class labels, there are feature selection methods for supervised learning [5].

7 Conclusion

We presented a method to improve the process of positive example selection by teaching agent in multi-agent systems that a group of agents try to teach a concept to a learning agent. We established our method base on the reflection of the viewpoints of the teacher agents. Similar to the behavior of human beings, the teacher agents express their viewpoints with the features that they think are more discriminatory. Then they use these features to extract more distinctive positive examples which naturally characterize the queried concept better. We found this method very useful in monotonous distribution over whole positive example space. Our experimental results revealed the improvement of the learner effectiveness using this new method of positive example selection. As a future work we will expand the method for better selection of negative examples and we will analyze the behavior of the selected subset of the negative examples over the whole space of negative examples.

References

1. M. Afsharchi, B.H. Far, J. Denzinger: Ontology-Guided Learning to Improve Communication between Groups of Agents, Proc. AAMAS-06 (in press), 2006. http://www.enel.ucalgary.ca/ afsharch/aamas06.pdf
2. R. Kohavi and G.H. John. Wrappers for feature subset selection. Artificial Intelligence, 97(1-2): 273324, 1997.
3. Illinois Semantic Integration Archive. http:// anhai.cs.uiuc.edu/archive/, as seen on Jan 30, 2005.
4. D. Koller, M. Sahami: Hierarchically Classifying Documents Using Very Few Words, Proc. ICML-97, 1997, pp. 170–178.
5. M. Robnik-Sikonja, I. Kononenko: Theoretical and Empirical Analysis of ReliefF and RReliefF. Machine Learning Journal, Volume 53, 2003, pp. 23–69.

6. M.A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In Proceedings of the Seventeenth International Conference on Machine Learning, pages 359366, 2000.
7. S. Sen, P.P. Kar: Sharing a concept, AAAI Tech Report SS-02-02, Stanford, 2002.
8. H. Liu, H. Motoda, and L. Yu. Feature selection with selective sampling. In Proceedings of the Nineteenth International Conference on Machine Learning, pages 395402, 2002.
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. Y. Yang, Y., J.P. Pedersen: A Comparative Study on Feature Selection in Text Categorization. Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), 1997, pp412-420.