UNIVERSITY OF CALGARY

Ontology-Guided Collaborative Concept Learning in Multiagent Systems

by

Mohsen Afsharchi

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER

ENGINEERING

CALGARY, ALBERTA

APRIL, 2007

Your file Votre référence
ISBN: 978-0-494-26226-9
Our file Notre référence
ISBN: 978-0-494-26226-9

Abstract

Traditionally, communication among agents has been established based on the group commitment to a common ontology which is unfortunately often too strong or unrealistic. In the real world of communicating agents, it is preferred to enable agents to exchange information while they keep their own individual ontology. While this assumption makes agents to represent their knowledge more independently and gives them more flexibility, it adds to the complexity of communication. We believe that agents can overcome this complexity by using their learning capability. The agents can learn any concepts they don't know but want to communicate about from other agents in the multi-agent system they are working in. Our goal in this thesis is to present a general method for agents using ontologies to teach each other concepts to improve their communication and therefore, cooperation abilities. In our method a particular agent which understood a concept only ambiguously intends to learn it by receiving positive and negative examples for that concept from the other agents. Then, utilizing one of the known concept learning methods, the agent learns the concept in question. This learner agent ask the other agents again to get involved in the learning process by taking votes in case of conflicts in the received set of examples. While this method allows agents not to share common ontologies it enables agents to establish common grounds on concepts known only to some of them, if these common grounds are needed during cooperation. In fact, the learned concepts by an agent are compromise among the views of the other agents and in addition, the method improves the autonomy of agents using them significantly.

Acknowledgements

A journey is easier when you travel together. This thesis is the result of four years of work whereby I have been accompanied and supported by many people. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

The first person I would like to thank is my direct supervisor Dr. Behrouz Homayoun Far. I owe him lots of gratitude for having me shown this way of research. His intuition and insight are only matched by his genuine warmth and human compassion. Besides of being my supervisor, Behrouz was as close as a relative and a good friend to me. I am really glad that I have come to get know Dr. Far in my life.

I would like to thank my unofficial co-supervisor Dr. Jörg Denzinger who kept an eye on the progress of my work and always was available when I needed his advices. During these years I have known Dr. Denzinger as a sympathetic and principle-centered person. His overly enthusiasm and integral view on research and his mission for providing 'only high-quality work and not less', has made a deep impression on me.

I also thank the other supervisory committee members, Dr. Reda Alhajj and Dr. Diwakar Krishnamurthy for their encouragement and support.

My sincere thanks go to the Ministry of Science, Research and Technology of Islamic Republic of Iran, for granting me scholarship and study leave to carry out research towards the PhD degree. As well, I offer my thanks to IBM Corporation which has partially supported and funded my research through UIMA program.

Thanks to my many fantastic friends and fellow graduate students, especially

Keivan Kianmehr, for their support and help.

I feel a deep sense of gratitude to my family. For my mother who supported me when I was alone and taught me the good things that really matter in life. The happy memory of my late father and brother still provides a persistent inspiration for my journey in this life. I am grateful for my sister Athar and my brother Masood, for rendering me the sense and the value of grace. Thank you for being there when I was absent.

Last but not least, I want to mention my wife, Maryam Seifaee. I am very grateful for her, for her love and patience during the PhD period and for her genuine motherhood for our two little beautiful angels: Fatemeh and Saba. There is no way I can ever thank Maryam properly for everything that she has done to support me. Instead I offer every meaning I can think of for the phrase, "Without you, I would not be here right now."

This thesis is dedicated to my loving wife, Maryam

and my children Fatemeh and Saba

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This dissertation addresses a new approach to solve the semantic heterogeneity problem in multi-agent systems where individual autonomous agents learn ontology concepts from several teacher agents to better communicate and share information in the future. In this thesis we introduce a novel multi-agent concept learning where there are several teacher agents that are not committed to any common conceptualization. The result of this research shows that the learning of new concepts among individual agents with diverse conceptualizations and different background knowledge (e.g. ontology) is achievable. Although for one specific concept, there are many conceptualizations which have been formed based on agents' view points, it is assumed that these diverse conceptualizations have some overlap which are apparent in their knowledge representation (e.g. similarity in examples or features).

This work is founded on the essential ontological promiscuity of artificial intelligence: any conceptualization of the world is accommodated, and is invented (by an agent) based on its utilization [GN87]. Therefore, in order for agents with diverse ontologies or views of the world to share knowledge, they must be able to learn how to understand each other's conceptualization of the world. Previous works mostly not only assumed a common language among agents, but also a complete common understanding of all the concepts the agents communicate about.

Recently, the idea of having agents *learn* concepts (or languages) from other agents has been suggested as a solution to the problems above [Wil04], [Ste98]. In

this thesis we broaden the existing approaches in two different directions. First we assume that every agent can come with different sets of features which are used by the agent to observe the world and conceptualize it. We additionally assume that there are only some base features that are known, and respectively can be recognized by all agents with the same meaning. Second, we develop this work based on a real *multi*-agent system with real data in which a learner agent learns a concept from a *group* of agents rather than one peer agent. In this situation the learner can get confused, when the teachers are not unanimous about a concept. This thesis addresses the conflict resolution mechanism as well as learning and communicating about non-unanimous ontology concepts.

Our method of learning ontology concepts has been developed and evaluated in the course catalog ontology domain [UIl]. The result of this thesis extends our understanding of group learning of ontology concepts when a group of agents teach a concept to a learner agent. The results of the experiments conducted for this thesis have provided insight into the learning of a concept when the teachers and learner are utilizing different but overlapping set of features to characterize concepts of the world (see [AFD06b], [AF06]). Besides, as a direct consequence of multi-agent learning we see the formation of non-unanimous ontology concepts and propose a way for agents to communicate better by using them (see [AFD06a]).

1.1 Motivation

The main and very strong fact that motivated us to start this work was the huge amount of works deviating from the essential ontological promiscuity of AI. We argue

that in order to have agents communicate better and share knowledge they should be able to learn how to understand each other. Due to the fact that this issue has been generally addressed in only very few works, we followed two more specific motivations. First, we believe that one reason for agents to have diverse ontologies is the different set of features that they use to represent and conceptualize the world and another reason is the different viewpoints that agents have to organize the concepts and categorize them. This means two agents may differ in their ontologies in two dimensions: the set of features representing the concepts and the set of relations that connect concepts to each other. Therefore agents should be able to learn each other's conceptualization despite using different but overlapping sets of features as well as different sets of relations that connect concepts to each other. Our second motivation is the nature of heterogeneous *multi*-agent system: agents communicate with more than one agent, therefore they should learn how to understand the group's conceptualization of the world. The general idea of ontological diversity of agents has been addressed in very few works and we review them in Chapter 3. Feature diversity of the agents and group collaboration among agents are our motivating principles in this thesis. In fact, these motivating principles are our main contributions and therefore we discuss them in more detail in the following subsections.

1.1.1 Feature Diversity

No matter how an agent conceptualizes the world, it is important to realize that there are other conceptualizations as well. Furthermore, there need not be any correspondence between the set of features representing an object or generally a concept in one conceptualization and the set of features in another. This dispensability of

correspondence is one reason that causes agents to have diverse ontologies.

A very common model for knowledge representation is feature based representation. In this model of representation, agents use a set of features to represent the world. This set is usually called *feature vector* [Nil98]. For each feature in the vector we have its domain which is a set of values that defines possible values the feature can have. Then an object is characterized by a unique combination of the features. To characterize a concept by using feature values, we denote a concept by a set of possible values for the features. Therefore an object which is an instance of a concept is covered by the concept.

Depending on the environment that an agent is deployed in, the feature vector differs. For example, for a physical agent the sensory inputs define the feature vector. An agent using a camera or color sensors might have some features representing the color of the objects it observes. For instance, the agent might have the feature *color* and {`red`, `blue`, `white`} as the values set. As another example, for information agents that are deployed in the World Wide Web environment, the feature vector might be the set of keywords representing a document that the agent focuses on (i.e. object). Such a representation is sometimes also referred to as the *bag of words* representation [Mit97], since usually relative position of terms in the document is not captured in the resulting vector.

As a result of the above discussion, every agent may have its own set of features to represent the world. Apparently, these features can be different from one agent to another based on the sensory input of the agent, design goals or the deployment environment. In some cases one agent's conceptualization of the world is "impossible" to be understood by another agent. A famous example of this is when an agent

conceptualizes an object using its color. This conceptualization can not be understood by another agent which has no hardware to sense the colors. In other cases one agent's conceptualization of the world is "difficult" to be understood by another agent. Again an agent that can sense grayscales, can not distinguish the colors but using some transformations it is possible to understand another agent that uses color to conceptualize the world.

In this thesis we assume that there are only some base features that are known, respectively can be recognized by all agents. Also we assume that there are only some base concepts that are known to all agents by their names, 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. We believe that this feature diversity makes the whole scenario of learning *real* and in fact resembles the process of a human being's learning.

We should point out that it is relatively easy to construct scenarios in which these ideas may not be very useful. What these scenarios have in common is that the teacher agents do not only disagree on what concepts fulfill a particular communication, their ontologies and their perception of the world, i.e. their feature sets for representing the world are totally different. As a result, their attempts to teach an agent results in a total confusion. If, for example, all teacher agents associate with a particular concept totally different real concepts, then the learning agent will not be able to get useful concepts out of their information. Clearly, also a human

agent would find it impossible to learn useful information from the teachers in this situation. In this dissertation we have chosen an area where the teacher agents have some differences in their "world view", simply because there are different ways how to represent and organize the objects in the world, but where there is nevertheless a large agreement on many things.

From the point of view of knowledge representation one interesting part of ontologies are the relations that a particular ontology allows. This is also the part where we see a lot of differences between different "world views". In general, all possible relations between tuples of concepts can be used in ontologies, but usually researchers assume a small set of built-in relations (e.g. `sub-concept` and `super-concept`) and tool developers sometimes throw in the possibility to have (limited) user-defined relations (e.g. `is-similar-to`). 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 agents using ontologies over the same universe of discourse then it is very likely that either they use different relations or they organize concepts, using the same set, in different ways. In this thesis 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 the same relation identifiers referring to the same relation and vice versa, again.

1.1.2 Learning from a Group of Agents

Regarding concept learning in multi-agent environments almost all researchers have looked at one agent teaching another agent a concept [Wil04], [Ste98], [Sen02]. At first glance, learning from a group of agents instead of a single agent only seems to

add potential problems, namely the teachers might not agree on some aspects of a concept to learn so that it is up to the learning agent to decide on these aspects on its own. This has the consequence that the concept that the learner has learned is some kind of compromise between the concepts the teachers teach. Would it not be better to learn from just one teacher at a time and to make sure that the learning agent learns exactly the concept of this teacher? But what if this learner has to communicate with several other agents? Naturally, the learner could learn the necessary concepts from each of these agents one by one and then it has an ontology that has concepts like "concept X according to agent Y" and "concept X according to agent Z". As pointed out in [JvD06], it is a rather large effort to create such an ontology. But even more, it does not really solve the problem of how to communicate with all the agents at once (in a kind of broadcast or multi-cast situation). Being able to address a group of people is a necessity of human communication. The way human beings deal with the fact that listeners (or readers) might have slightly different interpretations of the concepts is often to have an idea where everyone agrees and where there is potential for misunderstandings. Then people address the potential misunderstandings by providing more details of their understanding via explanation, examples, etc.

In this thesis, we present the novel idea of *non-unanimous* ontology concepts that allows us to express different "shades" of agreements on a particular concept based on what an agent learns from a group of teacher agents. Also we provide an environment to enable agents to learn such non-unanimous concepts that represent a whole spectrum of possible definitions for a concept. The basic idea is to let the learning agent query its teachers regarding the new concept. Depending on how

teacher agents agree about the queried concept the learner agent creates boundaries for the learned concept. The *core* boundary is around the area that there is no conflict among agents regarding the objects in this area. The *periphery* boundary is the area that covers all teacher agents' viewpoints. Therefore every objects that even only one agent think belongs to the concept is in this boundary. The learner agent itself then chooses a concept definition that encompasses the core and is itself encompassed by the periphery which we call its *own* understanding. When communicating about this particular concept, the agent uses its awareness of the differences between its own definition, the core and the periphery to enhance the usage of the concept name with explicit references to objects in the periphery but not in the core that are relevant to the communication.

1.2 System Overview

As stated before, the goal of this thesis is to develop a methodology to let an agent learn new concepts and accommodate it in its ontology with the help of other agents. Given the fact that agents have not so much common knowledge, they will develop problems in working together, since the common grounds for communication are not there or are too small. To solve this problem, agents need to acquire the concepts outside of the base concepts 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.

Figure 1.1 a) shows the sequence of tasks that the learner accomplishes in the

process of ontology concept learning. The learner agent, $\mathcal{A}g_L$, initiates the learning process by querying the teacher agents about a concept, c_{goal}, that it wants to learn. 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. An example of a scenario that can lead to this realization is when $\mathcal{A}g_L$ deals with a set of objects that share certain feature values and it wants to know if other agents know more about the similarities of these objects. The query can have up to three parameters: concept identifier which is unlikely to be the same in different agents, a selection of features and the values that $\mathcal{A}g_L$ thinks are related to the concept or a set of objects that $\mathcal{A}g_L$ thinks are covered by the c_{goal}.

Lets turn our attention to the teacher agent. Its tasks are presented in Figure 1.1 b). After receiving the query the teacher agent, $\mathcal{A}g_i$, starts to answer the query by finding the best known concept. Therefore $\mathcal{A}g_i$ has first to collect all the concepts that fulfill the query into a candidate set and then it has to evaluate all these concepts to determine the concept that is, in its opinion, the best fit. According to the parameters of the query, the procedure of selecting the candidate concept may vary. But generally the teacher agent tries to find concepts that cover a good portion of the query parameters. There are many different ways how evaluation of the candidates can be performed. Each of the 3 query parts can contribute to a measure that defines what is "good", and how these contributions are combined can be realized differently.

A very important part of the methodology in this thesis is that the agents learn concepts from a set of positive and negative examples. Consequently the teacher agents should send a set of positive and negative examples to the learner in order to enable the learner agent to learn the concept c_{goal}. Since each agent $\mathcal{A}g_i$ stores for

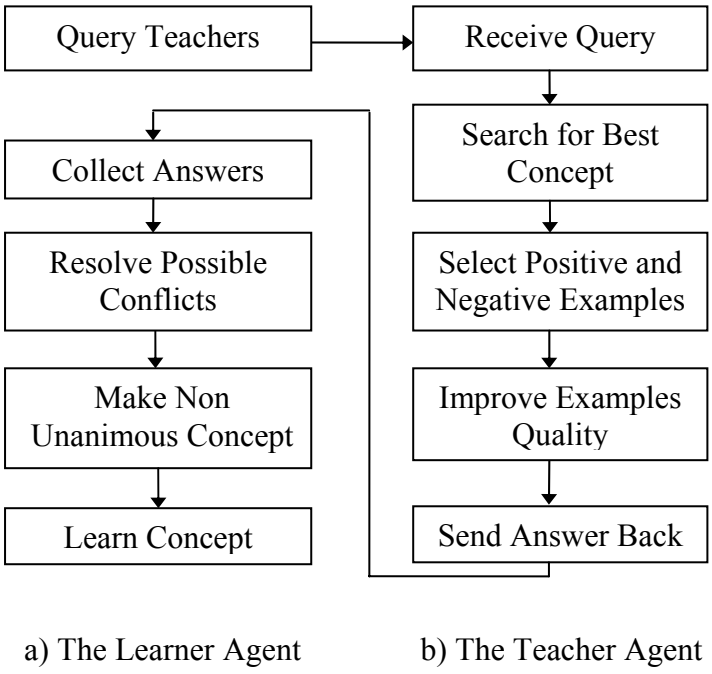

Figure 1.1: Sequence of processes in agents

each concept a set of positive examples, i.e., a set of objects covered by the concept, coming up with positive example objects for a concept known to $\mathcal{A}g_i$ is not a problem. While the more examples normally are the better, in our case we have to take into account that the more objects are selected, the more expensive the communication becomes and the more effort $\mathcal{A}g_L$ will have to spent on learning. Selecting negative examples for a concept is not as straightforward. The set of negative examples for a concept is the set of objects covered by all concepts excluding the example objects from the queried concept. This can be a very large set and usually different elements of this set provide the learner with a different quality of advice. The large set of

possible negative examples in addition to the communication cost impose on the teacher agents to intelligently select a subset of positive and negative examples to provide the learner agent with a better quality of advise. After creating the set of positive and negative examples and putting them in the reply package, the teacher agent adds some information regarding the path that leads in its ontology to the queried concept and the subtree below it and send it back to the learner.

Returning back to the learner, $\mathcal{A}g_L$ is ready to learn from the set of positive and negative examples which are collected from several teachers. 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 [Mit97]. With the set of positive and negative examples, $\mathcal{A}g_L$ has the necessary input for such a learning system, with one potential problem: conflicts among the teacher agents. Due to the differences among agents it can easily happen that the two best concepts that two teachers identified do not have much in common. There are several ways to solve this problem and these ways lead the learner agent to make a non-unanimous concept which allows the learner agent to produce different degrees of willingness to satisfy the teacher agents.

1.3 Thesis Overview

In Chapter 1, we have outlined the basic problem which this research addresses. We have delineated our research goals, and made clear what we think are our contributions.

In Chapter 2, we define the general notions that we use throughout this thesis.

We give semi-formal definitions for agent, ontology, and concepts.

In Chapter 3, we explore the literature related to "agent teaching concepts to another agent" approaches in both the traditional AI context and new applications. For each related work we highlight aspects that are not addressed by the authors.

The methodology for our novel approach to multi-agent concept learning is presented in Chapter 4. This includes both a general interaction scheme and the different realizations for each step of this general interaction scheme.

Chapter 5 contains the details for an instantiation of our abstract concepts for agents representing and communicating about university courses and how they relate to university units. This includes a high-level description for our implementation of agents, ontology construction, and our adaptation of known concept learners. In addition, we explore our definition of features in the problem domain of our instantiation.

In Chapter 6 we present some experiments that we have conducted to prove that concept learning from a group of agents with diverse ontologies is achievable. In addition we compare different realizations of each of the steps that we have proposed in our methodology.

Finally, Chapter7 contains a summery of the research presented in this thesis, and some suggestions for the direction of future work.

Chapter 2

Basic Definitions

2.1 Agents

Generally an agent is a system that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. A human agent has eyes, ears, and other organs for sensors, and hands, legs, mouth, and other body parts for effectors. A robotic agent substitutes cameras and infrared range finders for the sensors and various motors for the effectors. A software agent has encoded bit strings as its percepts and actions [RN95].

While there is no agreed-upon definition for artificial agents, most of the texts in the field agree on a general architecture. In a very general form it is defined as a mapping function from perceived sequences to actions. In this dissertation, inspired from [DE02] we assume an agent $\mathcal{A}g$ as

$$\mathcal{A}g = (Sit, Act, Dat, f_{Ag})$$

Sit is a set of situations the agent can be in which means the agent's world can be in any one of a set Sit of situations (or states). The representation of a situation naturally depends on the agent's sensory capabilities. Different agents can have different effectory capabilities. To characterize these effectory capabilities, we assume the existence of a set Act of actions, all of which can be performed by the agent we are describing. We define a function f_{Ag} as a mapping that maps each situation into an action. This function is called effectory function [GN87] and is defined as

$$f_{Ag} : Sit \times Dat \rightarrow Act$$

We further assume that the agent can be in any one of a set Dat of internal states. While there is no need for internal states in a simple case, the ability to retain information internally is extremely useful in general. Therefore there should be a memory update action in Act that maps an internal state to another internal state. This defines an agent by exploring its internal structure. There are other definitions which categorize agents based on their behaviors.

Russel and Norvig divided agents into four different types based on their behavior regarding the agents' internal states and goals [RN95]: simple reflex agents, agents that keep track of the world, goal-based agents and utility-based agents.

In a *simple reflex agent* activity at any moment is determined entirely by their environment at that moment. In these agents Dat simply includes a set of condition-action rules. Upon arrival of a new stimulus agents use a rule whose condition matches the current situation and then perform the action associated with that rule. The simple reflex agent will work only if the correct decision can be made on the basis of the current world state. In many cases, the agent may need to maintain some internal state information in order to distinguish between world states that generate the same perceptual input but nonetheless are significantly different. Here, "significantly different" means that different actions are appropriate in the two states. Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent. First, we need some information about how the world evolves independently of the agent. Second, we need some information about how the agent's own actions affect the world. These agents that *keep track of*

the world are also called *hysteric* agents [GN87].

Knowing about the current state of the environment is not always enough to decide what to do. In fact, as well as a current state description, the agent needs some sort of goal intonation, which describes situations that are desirable. The *goal-based agent* can combine this with information about the results of possible actions in order to choose actions that achieve the goal. Sometimes this will be simple, when goal satisfaction results immediately from a single action; sometimes, it will be more tricky when the agent has to consider long sequences of twists and turns to find a way to achieve the goal.

Goals alone are not really enough to generate high-quality behavior. Goals just provide a crude distinction between "happy" and "unhappy" states, whereas a more general performance measure should allow a comparison of different world states (or sequences of states) according to exactly how happy they would make the agent if they could be achieved. Because "happy" does not sound very scientific, the customary terminology is to say that if one world state is preferred to another, then it has higher utility for the agent. Utility is therefore a function that maps a state onto a real number or a value measured by an ordinal scale, which describes the associated degree of happiness. A complete specification of the utility function allows rational decisions in two kinds of cases where goals have trouble are not enough. First, when there are conflicting goals, only some of which can be achieved, the utility function specifies the appropriate trade-off. Second, when there are several goals that the agent can aim for, none of which can be achieved with certainty, *utility-based agents* provide a way in which the likelihood of success can be weighed up against the importance of the goals.

2.1.1 Agents in this Dissertation

For the agents that we are interested in, we can instantiate the general definition of this dissertation with some extensions. 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 we will define in the next 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, 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 thesis, we will concentrate on how the agent uses and manipulates just the ontology part of Dat.

The actions of an agent depends 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. We will provide more information in the following sections.

An element of Sit usually contains parts representing observations of other agents and of the environment the agent is in. In this thesis, we assume that among the observations of an agent are all messages sent 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

2.2 Ontologies

To clearly define "ontology" we should first define "conceptualization". While pure AI texts use ontology and conceptualization interchangeably [GN87], in recent years many articles from the knowledge sharing and reuse community consider an ontology as an explicit specification of a conceptualization [Gru93]. In this section we will first briefly describe what conceptualization is and then give a formal definition for ontologies. [1]

According to [GN87], the formalization of knowledge in a declarative form begins with a *conceptualization*. Generally, a conceptualization is a triple consisting of a set of concepts, a functional basis set for that set of concepts and a relation basis set. The notion of a concept used in this description is quite broad. In short, a concept can be anything about which the agent wants to know or say something. Not all knowledge-representation tasks require that we consider all the objects in the world; in many cases, only those concepts in a particular set are relevant. A function is one kind of interrelationship among the concepts. Again for a given set of concepts, in conceptualizing a portion of the world we usually emphasize some functions and

[1] Many of the solutions to the knowledge sharing, integration and reuse problems have focused on using a common ontology for different knowledge sources and a commitment of the working agents to this shared conceptualization. Inherent in all of these efforts is a changed version of the ontology definition. While Gruber [Gru93] defined ontology as "an explicit specification of conceptualization", many recent works added the word "shared" to it which changed the definition to "an explicit specification of a *shared* conceptualization". Accepting this definition makes our research insignificant. Nevertheless we argue that although using a common ontology is a method to solve the knowledge integration and agent communication problem, it is very important to notice that this is not an intrinsic part of any solution to the problem and as we will discuss later the essential promiscuity of AI emphasizes on the diversity of the ontologies of intelligent agents.

ignore others. A relation is the second kind of interrelationship among concepts. As we do with functions, in conceptualizing a portion of the world, we emphasize some relations and ignore others.

This informal definition of conceptualization is very nicely reflected in Stumme's formal definition of ontology which is as follows [Stu02]:

Definition 1. A *core ontology* is a structure

$$\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R)$$

consisting of

- two disjoint sets C and R where the elements of C are called *concept identifiers* and the elements of R are so-called *relation identifiers*,
- a partial order $\leq_C$ on C which is called *concept hierarchy* or *taxonomy*
- a function $\sigma : R \to C^+$ which is called *signature*
- a partial order $\leq_R$ on R which is called *relation hierarchy*, where 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 $|\sigma(r_1)| = |\sigma(r_2)|$ and $\pi_i(\sigma(r_1)) \leq_C \pi_i(\sigma(r_2))$.

Concept identifiers and relation identifiers are usually called concepts and relations. Almost all relations in practical use are binary. For those relations, we define their domain and their range.

Definition 2. For a relation $r \in R$ with $|\sigma(r_2)| = 2$ we define its domain and its range by $\mathrm{dom}(r) := \pi_1(\sigma(r))$ and $\mathrm{range}(r) := \pi_2(\sigma(r))$.

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. If there is no $c_3 \in C$, such that $c_1 <_C c_3 <_C c_2$, and $c_1 <_C c_2$ then c_1 is a direct subconcept of c_2 and c_2 a direct 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. In this work we will follow this example. 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 value 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 universe of discourse which is the set of all (possible) objects. In machine learning, often every subset of $\mathcal{U}$ is considered as a concept. In this thesis 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.

Apparently, the relation $\leq_C$ is supposed to be connected with how concepts are

defined. In the literature, taxonomies are often build using the subset relation, i.e. we have

$$C_i \leq_C C_j \text{ iff for all } o \in C_i \text{ we have } o \in C_j.$$

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

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 much differences 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 confusion in this area. Therefore, if we have two systems build by different developers 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 two systems. In this thesis, we will show such a usage for one relation that we have called `is-similar-to` with $\sigma(\texttt{is-similar-to}) \in C^2$.

2.2.1 Diverse Ontologies

As stated above, agents do not always commit a priori to a common pre-defined global ontology. This naturally assumes that not all agents have the same ontology (otherwise learning in the sense described in this thesis would not be necessary). To

clarify what we mean by "diverse ontologies" we revisit the definition of ontology. We say two agents $\mathcal{A}g_i$ and $\mathcal{A}g_j$ have different ontologies if at least one member of the tuple $\mathcal{T} =< C, R, \sigma, \leq_R >$ differs from one ontology to another. Therefore there are four reasons for ontologies to be distinctive, so we represent the ontology for $\mathcal{A}g_i$ as $\mathcal{O}_i = (C_i, \leq_C, R_i, \sigma_i, \leq_{R_i})$ and the ontology for $\mathcal{A}g_j$ as $\mathcal{O}_j = (C_j, \leq_C, R_j, \sigma_j, \leq_{R_j})$.

The first reason for ontology diversity is having different sets of concepts. C_i and C_j could be different simply because they could have different members. Agents might refer to the same concept by different names or they may have concepts that have the same name but different meanings. The later case is very likely because we assumed that the agents could use *diverse feature* sets to conceptualize the world and this allows agents to have diverse ontologies. R_i and R_j also could be different and in fact relations between concepts could be another source of distiction between ontologies. Also two ontologies could have the same set of concepts and relations but they also could differ in the way that they use relations to associate two or more concepts. This happens when two ontologyies have different σ's. We assume the same partial order $\leq_C$ for ontologies because usually researchers use `sub-concept` and `super-concept` relations to build a taxonomy. While we believe that $\leq_R$ could be different in ontologies we have no emphasis on it in this dissertation.

Chapter 3

Background

The communication among agents is an inherent characteristic of multi-agent systems. Agents committing to a common and shared ontology can communicate without any misunderstandings. Nonetheless Artificial Intelligence emphasizes on ontological diversity of intelligent agents which in fact resembles human beings. In this thesis we tackle the problem of communication among agents which have no commitment to any common ontology.

This chapter will look at the different areas that address this problem and the different proposed solutions. We begin with some discussions about ontologies and conceptualizations and the essential promiscuity of AI. We then review the proposed solutions based on concept learning and ontological evolution. Because many other works have tried to solve a similar problem in the information retrieval domain, in the final section, we present a survey of the different solutions to the problem which is called "semantic heterogeneity of knowledge sources".

3.1 Ontology, Conceptualization and the Essential Ontological Promiscuity of AI

An agent, whether representing a human being, an intelligent computer program, or a robot has a view of its world that enables it to understand it. In Artificial Intelligence (AI), this view of an agent's world has been referred to as its *conceptualization*

or ontology [GN87], [Gru91]. The knowledge of an agent has been represented through various declarative or procedural forms such as predicate logic, production rules, nonmonotonic systems, statistical reasoning systems, semantic networks, conceptual dependency and frames [RK91]. Declarative knowledge consists of a conceptualization which includes the objects or concepts presumed to exist in the world, their functions and relations [GN87]. The set of objects about which knowledge is expressed is referred to as the universe of discourse. According to [GN87], given a conceptualization (i.e. objects, functions, and relations) knowledge can be formalized as sentences in a language appropriate to conceptualization. Examples of languages used to represent knowledge include propositional calculus, predicate calculus or even natural language.

Historically, ontology has referred to the philosophical study of being, or what exist [Gru91]. In AI ontology deals with categories we can quantify over and how those categories relate to each other [RK91]. A truly global ontology specifies at a very high level what kinds of things exist and what their general properties are. Gruber [Gru91] refers to ontology as a specification of a conceptualization. *In this dissertation, ontology and conceptualization will be used interchangeably.* One distinction that will be made for this research is that an ontology includes a conceptualization, or view of the world, as well as the learning and inferencing mechanisms for manipulating knowledge about the world. Inherent in these definitions of ontology is a method for representing and interrelating the objects or concepts into a hierarchy of concept types according to different levels of generality. In a semantic network representation this is called a type hierarchy, taxonomic hierarchy or a subsumption hierarchy [Woo91]. In a frame-based representation system, this concept hierarchy

would be represented as inheritance rules.

These conceptual taxonomies, loosely referred to as ontologies, can be useful for indexing and organizing information and for managing the resolution of conflicting concepts [Woo91]. Earlier works in knowledge representation demonstrated the ability to automatically classify a structured concept with respect to a taxonomy of other concepts. Classification, according to Woods [Woo91], is the operation of assimilating a new description into a taxonomy of existing concepts by automatically linking it directly to its most specific subsumers and the most general concepts that it in turn subsumes. Stumme [Stu02] nicely reflected these properties in the formal definition which we have discussed in Chapter 2.

Every individual human being or knowledge-based program or agent creates or learns an ontology either explicitly or implicitly. According to [GN87] these ontologies are created and justified solely based on their utilization in the different tasks or domains and there is no commitment to a pre-existed ontology. This lack of commitment indicates *the essential ontological promiscuity of AI*: Any conceptualization of the world is accommodated, and agents seek those that are useful for their purpose.

Different knowledge-based programs have different underlying ontologies used to represent knowledge in their domains. The DARPA Knowledge Sharing Effort (KSE) [FFMM94] realized that various knowledge-based programs could not share knowledge because they were based on different ontologies. Researchers associated with the KSE effort sought to allow re-use of knowledge bases by creating common ontologies in order to share knowledge. Some of the results of this effort include the Knowledge Interchange Format (KIF) which is based on an extension of the first order predicate calculus, and Ontolingua [Gru93]. Gruber [Gru91] describes

an approach to using a common ontology to enable AI programs to share knowledge bases and use them as modular components. His definition of a common ontology is vocabularies of representation terms, such as classes, relations, functions, and object constants, with agreed upon definitions, in human readable text and with machine-enforceable, declarative constraints on their well-formed use. According to Gruber and Olsen [GO94], the ontology vocabulary defines the ontological commitments among agents that are agreements to use the shared vocabulary in a coherent and consistent manner. This is where the approach for sharing knowledge in this thesis diverges from the Knowledge Sharing Effort approach. We argue that not all agents share a commitment to the same ontology so we must find ways for them to share knowledge by teaching each other semantic concepts from their own point of view.

3.2 Multi-agent Learning Systems

Multi-agent learning literature mainly has sprung from historically somewhat separate communities notably reinforcement learning, dynamic programming, robotics, evolutionary computation and complex systems and is tended to the special community that it is applied [PL05]. We consider it as discipline that addresses the traditional machine learning algorithms that have been extended to work in a system of distributed, learning agents. Multi-agent learning can be defined as improving the group problem solving performance at a given set of tasks through their collective experience in the problem domain. Our multi-agent learning research deals with a system of agents that learn individually as well as collectively.

Cooperative multi-agent learning in literature has been divided into two major

categories: *team learning* and *concurrent learning* [PL05] which are named differently in [Wei99] as *centralized learning* and *decentralized learning* though their meanings are somehow the same. Team learning applies a single learner to search for behaviors for the entire team of agents. Such approaches are more along the lines of the traditional machine learning methods. Concurrent learning uses multiple concurrent learning processes and rather than learning behaviors for the entire team, concurrent learning approaches typically employ a learner for each team member, in the hope that this reduces the joint space by projecting it into separate spaces. However, the presence of multiple concurrent learners makes the environment non-stationary, which is a violation of the assumptions behind most traditional machine learning techniques. For this reason, concurrent learning requires new (or significantly modified versions of) machine learning methods.

While team and concurrent learning cover the majority of works in multi-agent learning, as we will show, we think our work can not be classified into either of these classes. That is mostly because the learning itself in our work is not MAS learning but it is the rather standard learning of a single agent. The MAS aspect of our work resides in the fact that an agent is taught by several other agents. Though there has been considerable research on agents learning concurrently or cooperatively [PLL96], [PH96] or learning by imitating a teammate's behavior [DE02] or even learning by observing an opponent's behavior [SRV00], [LASB04] there has been little work in which an agent pro-actively trains other agent. The most relevant work comes from one learner telling another agent what portions of the search space to ignore [PH96] or a teacher agent shares experience [BSSD00], problem-solving traces or even learned policies [Tan97] with another concurrent learner.

In this chapter we put the emphasis on the approaches to learning or teaching methods regarding agents training other agents. More specifically, we consider works in which agents learn or teach concepts rather than behaviors.

3.3 Approaches to Agents Training another Agent

In this section we review the paradigm of multi-agent learning when one or more agents try to teach a concept to another agent. Different works utilized different terminology (e.g. speaker and hearer, trainer and trainee) to address the learner agent and the teacher agents. To keep our terminology unified we use "learner" and "teacher" to address the agent that learns from other agents and teach another agent, respectively.

3.3.1 Learning to Share Meaning

Williams has developed DOGGIE (Distributed Ontology Gathering Group Integration Environment), a system to demonstrate how a multi-agent system can assist groups of people in locating, translating, and sharing knowledge [Wil04]. He assumes agents with diverse ontologies that may use different terms to refer to the same meaning or the same term to refer to different meanings. He argues that agents with this setting will need a method for learning and translating similar semantic concepts between diverse ontologies. Williams' work tries to propose solutions to the following three questions: 1) how do agents determine if they know the same semantic concepts?, 2) how do agents determine if their different semantic concepts actually have the same meaning?, and 3) how can agents improve their interpretation

of semantic objects by learning discriminating attributes? It also evaluates proposed methods to assess the group performance at a given collective task. Since DOGGIE is closely related to our proposed research it is important to point out the similarities and differences.

Williams assumes a *distributed collective memory* (DCM) which covers the entire set of concrete objects that exist in the world at a unique location and is accessible by any agent but is only selectively conceptualized by each agent which naturally means not every agent has every object in its conceptualization. He also defines a set of objects that can be grouped to form an abstract object as a *class* and enables agents to use inductive supervised machine learning mechanisms to learn a target function to map individual concrete objects to a particular class. Then he refers to this target function as a *concept description* of a class. He further assumes agents use a knowledge structure that can be learned using objects in the distributed collective memory.

The main goal of Williams' approach is to enable agents to locate and learn similar semantic concepts. The result of this learning is a kind of group knowledge in the form of "one agent knows that another agent knows a particular concept". To achieve this goal Williams has gone through two separate procedures: *learning similar semantic concepts* and *learning key missing descriptors*. The first procedure enables agents to locate the similar semantic concepts in acquaintance agents and to translate the similar semantic concepts. The second procedure which eventually results in better locating and translating of similar concepts, is a method to find some key features which discriminate similar concepts.

To locate a similar concept an agent queries acquaintance agents by sending them

the name of the semantic concept and pointers to some sample semantic objects in the distributed collective memory. The acquaintance agents receive the query and use their learned representations for their own semantic concepts to infer whether or not they know the same semantic concept. As stated before, agents in DOGGIE, use an inductive machine learning algorithm to learn a concept descriptor for each concept. Acquaintance agents use this concept descriptor to see if they know a similar concept or not. If an acquaintance agent knows or may know that semantic concept, it returns a sample of pointers to its corresponding semantic concept. The original querying agent receives the responses from the acquaintance agents and attempts to verify whether or not the other agents know a similar semantic concept. It does this again by using the concept descriptor it has learned before. If the original querying agent verifies the acquaintance's semantic concept, then it incorporates this applicable group knowledge into its knowledge base. This group knowledge is, in essence, "My acquaintance agent X knows my concept Y".

Williams argues that two similar semantic concepts may not have overlapping semantic objects in the distributed collective memory. If this is the case, the target function learned using supervised inductive learning for agent A's semantic concept descriptions, and agent B's target function may have different key discriminating descriptors, or attributes, in them. Then he proposes a Recursive Semantic Context Rule Learning (RSCRL) to enable agents to learn discriminating attributes which consequently improves similar concept interpretation.

Here we highlight some aspects of Williams' work and briefly introduce another realization for each aspect. In Chapter 4 we will show how our work is different from Williams' work regarding these aspects. Firstly, Williams' approach is a method for

agents to share a meaning. Agents have learned the semantic concepts and then try to learn if other agents know concepts similar to their concepts. Therefore if an agent does not know a concept nothing will happen. In DOGGIE, as a result of learning, agents gain the *meta* knowledge in the form of rules describing what semantic concepts another agent in the MAS knows. The only result of the learning that affect the understanding of the concept by any agent is the learning of some discriminating features. By contrast, agents can learn a new concept from scratch. The knowledge resulting from the learning could be a description of the new concept by a set of features and their values and from a set of positive and negative examples taught by a group of acquaintance agents.

Secondly, Williams assumes that every agent knows every feature describing objects in the distributed collective memory. Consequently in the process of learning key missing descriptors, agents learn the missing attributes (i.e. feature in our context) and it is assumed that these attributes are completely understandable by every agent. As a different approach, agents can use different features to conceptualize the world and learn the concepts. Therefore each agent could have a set of features which may have or have not an overlap with other agents' sets of features and could individualize its learning by this unique set of features.

Also in Williams' work the collaboration between agents is restricted to agent to agent communication rather than a full *multi*-agent activity. In DOGGIE one agent tries to know if *one other* agent knows a similar concept. There is no explicit hint regarding how an agent can learn a concept from a group of agents. One can argue that multi-agent collaboration can be considered as a sequence of agent to agent collaborations, nevertheless, in this case a very important question is what

will happen if there exist some contradiction between the communicating agents. As we will show in Chapter 4 our approach is a complete multi-agent collaboration where the learner agent learns a concept from *multiple* agents and in case of any conflicts there are mechanisms for conflict resolution.

In addition to the above mentioned major aspects, we should highlight some minor aspects. Williams only uses positive examples to enable agents to share a meaning, while we believe that a better approach is to use positive and negative examples to describe a concept. Also an ontology in Williams' work is a flat repository of concepts while in a more comprehensive approach an ontology could be considered as a conceptual hierarchy and utilize the relation between concepts to improve the learned concept quality.

3.3.2 The Origins of Ontologies and Communication Conventions

Steels describes an innovative evolutionary computation approach that enables a multi-agent system to converge to a common ontology (i.e. in fact ontology in this work is a shared lexicon rather than a hierarchical structure of concepts) using a language game [Ste98]. His research addresses a different, albeit important, problem of ontology consensus. That is, how can other agents converge to using a common ontology, or a shared lexicon, in what he refers to as a winner-take-all situation. To achieve this goal he proposes an adaptive systems approach to *the formation of an ontology* and *the evolution of a shared lexicon* in a group of distributed agents with only local interactions and no central control authority.

Steels first assumes that there is no central controlling agent and coherence arises in a bottom up and self-organized fashion. He also assumes that the language com-

munity is open and new agents may enter at any time. In addition he assumes that conventions are adaptive. This means new meanings may enter at any time and the group develops the appropriate lexicalizations as needed. The adaptation assumption is also true for the ontologies.

Steels models an interaction between two agents as a game and because it involves language he calls it a *language game*. This game involves a teacher and a learner and assumes that the teacher wants to identify an object to the learner given a particular context of other objects. In order to perform a communication, the teacher must conceptualize the objects so as to find a description which distinguishes the topic from the other objects in the context. This requires an ontology, which is considered as a set of distinctions. This demand for an ontology requires the agent (i.e. teacher) to create its ontology through discrimination games. A discrimination game involves an agent, an object and a set of other objects and tries to find some features that discriminate between an object and other objects. Then this set of discriminating features which are created based on some sensory channels makes the conceptualization of the agent regarding the specific object that Steels calls *topic*.

After making the ontology the teacher must find words to encode the distinctive features thus found, and transmits these words to the learner. This is a way that the agents use to find a shared lexicon for an object. Next, the learner receives the transmitted message, decodes it into one or more possible interpretations, and checks whether the interpretations are compatible with the present situation. The game succeeds if this is the case. Failure may be due to 1) missing objects in the ontology of the teacher or learner, or 2) missing or wrong linguistic conventions. In each case the agent engages in a repair action. New descriptions for objects are

created by extending the ontology, in other words by creating a new distinction or refining an existing distinction. New linguistic conventions are created by creating a new word or by adopting the word used by the teacher. Agents record the success of words and prefer words that had the most success. This causes coherence to emerge because the probability that a word is used increases if more agents adopt it. Agents also record the success of using a distinction. If a distinction is used often and has been successfully lexicalized it has a higher chance to remain in the population of possible distinctions. The coordination of ontology creation and lexicon formation in a single agent and in a multi-agent system happens by co-evolution. There is an information flow and selectionist pressure in either directions. The ontology creation produces distinctions which are lexicalized. Lexicalization is successful if the word is also used by other agents.

In a very simple statement, Steels allows agents to create their ontology by expe-riencing their environment. Then using a mechanism which he calls language game agents cooperatively evolve a common set of lexicons for the objects in their ontology. As the title of his work suggests, Steels' work is an attempt to discover what is the origin of ontologies in a multi-agent system and how agents develop the conventions to communicate about their ontologies.

As we stated before Steels research addresses the problem of ontology consensus. That is, how other agents can converge to using a common ontology, or a shared lexicon. However, as we will show in Chapter 4 our work recognizes that agents may often want to maintain their own diverse ontology but still be able to learn a new concept from different viewpoints of other agents or teach a new concept to them. This allows for each agent to maintain control over its own ontology but still be

able to improve communication with other agents. Although these agents start off with diverse ontologies, their goal is not to converge to a common one because we believe that learning ontology concepts is only one source of ontology evolution in a multi-agent system.

There is no explicit hint in Steels work showing that the agents use different sensory channels to conceptualize an object. Therefore in case of one agent not being able to discriminate an object with its set of features, it evolves its feature set toward the set which is common for all agents. For agents with different perceptions, this should pose problems which our approach does not have and we do not require for agents to evolve their feature sets to a common set of features. As we stated before an agent in our work individualizes its learning by the unique set of features it has.

Similar to Williams' work, the collaboration between agents is restricted to agent to agent communication rather than a full *multi*-agent activity. The language game described in Steels' work relies on the teacher and learner agents determining whether they use the same name to describe a single object and, obviously, there will be no conflict in this case.

The language game in Steels' work also relies on describing a *single* object but not a *class* of objects (i.e. concept). This is because Steels uses primarily evolutionary learning techniques to learn a language. As we will show, our approach uses machine learning techniques to allow for teaching a concept to an agent. Last but not least, the emphasis of Steels' work is more on language rather than concepts.

3.3.3 Sharing and Mutual Learning of a Concept

Sen proposed a general architecture for an agent teaching agent environment which he called ATA (i.e. Agent Teaching Agent) framework [Sen02]. In the ATA framework, he addresses the problem of transfer of knowledge between a teacher and a learner agent and the knowledge being transferred is a concept description. A concept description is a boolean-valued function that classifies input examples as members or non-members of the target concept. He also assumes that the teacher agent does not have access to the internal knowledge representation of the learner agent, but can observe its concept recognition abilities by providing examples and non-examples of a concept and observing the performance of the learner agent.

He claims that a teacher agent can guide the learning process of a learner agent by observing the latter's problem solving performance. This means that based on the success and failure of the learner in classifying given examples, the teacher can choose an appropriate sequence of training examples to guide the learning process of the learner. In the ATA framework, the teacher agent first acquires the target concept from its interaction with an environment, and using its learning module. This learning process produces a target concept description in the internal knowledge representation format of the teacher agent. The teacher agent also has a training module which interacts with the learner module and provides successive training and testing sets to train and evaluate the progress in learning of the learner agent. The learner learns its own concept description from the set of classified training examples provided by the teacher. It also classifies each of the unclassified test examples provided by the teacher and returns these classified instances to the teacher for

evaluation.

The ATA framework uses an iterative training procedure in which alternatively the teacher selects a set of training and testing examples. The learner trains using the training set and then classifies the testing set, the teacher observes errors made by the learner in classifying the instances in the last testing set and accordingly generates the next training and testing set. This iterative process converges when learner error falls below a given thresholds.

Sen's proposed work is a real concept learning/teaching framework in which the teacher agent uses examples to teach a concept to the learner agent . Nevertheless, his work like previously mentioned works has two major weaknesses. First Sen assumes that every agent knows every features describing concepts being learned and consequently in the proposed framework all agents must use the same feature set to learn concepts. Second, in the ATA framework the collaboration between agents is restricted to agent to agent communication rather than a full *multi*-agent collaboration.

Because the teacher agent in the ATA framework uses examples to teach the concept in an iterative way, it needs to specify procedures for selection of the initial training and testing sets, and the generation of the next test set based on the mistakes made by the learner on the current test set. In ATA, when selecting the initial training and testing instances, the goal is to select the most discriminating examples that help identify regions of the input space that do and do not belong to the target concept. When selecting the next set of training and testing instances, the goal is to first isolate the mistakes made on the previous test set, and for each of these instances, find a few neighboring points, using some of them as training data and

the rest as test data.

As we will show in Chapter 4, similar to Sen, we use the general idea of selection of most discriminating positive examples to improve the quality of the concept that is being learned, except that he used the distances between examples to select most discriminating ones and we select examples based on the most discriminating features. Besides, Sen has not explicitly mentioned the way that he selected the negative examples that he used to teach a concept. Also ATA uses a flat repository of concepts as the ontology of the agents, while ontologies of our agents are conceptual hierarchies. This allows us to enable agents to use relations between concepts to select better negative examples.

Gasser's work [WG02] regarding a framework for Mutual Online Concept Learning (MOCL) is basically a slightly different version of Sen's work. Though his emphasis is placed on instance selection, in contrast to Sen's work, there is no pre-existing concept, and hence the learners are peers rather than a teacher-learner pair.

3.3.4 ANEMONE: Concept Explication to Improve Communication

Diggelen et.al. [JvD06] developed ANEMONE which is an minimal ontology negotiation environment. ANEMONE, which was originally developed to overcome the problem of a lack of shared ontologies, proposes a layered communication protocol and enables agents to gradually build towards a semantically integrated system by establishing minimal and shared ontologies.

The upper layer of the protocol which is called Normal Communication Protocol (NCP) deals with normal agent communication, i.e. the kind of social interaction which agents normally exhibit when no ontology problems exist in the system. To

deal with ontology problems, two layers are added to the protocol: a protocol for exchanging concept definitions or Concept Definition Protocol (CDP), and Concept Explication Protocol (CEP) for teaching concepts to each other using machine learning techniques.

In the concept definition protocol, the teacher tries to convey the meaning of a concept by stating the taxonomical relations with other concepts just by sending the set of parents and children of a concept (i.e. using the subconcept and superconcept relation). If these definitions enable the hearer to derive the complete meaning of the concept, the hearer switches back to NCP. An agent considers the meaning of an acquired concept complete, if it knows the relation with every other concept in its ontology. If there is not a sufficient number of shared concepts available to convey the complete meaning, the agents switch to CEP. The purpose of CEP is to convey the meaning of a concept when no satisfactory definition of the concept in terms of other concepts can be given. ANEMONE assumes that the meaning of a concept can be conveyed to another agent by pointing to instances. The speaking agent (i.e. teacher), upon explicating a concept, communicates a number of positive and a number of negative examples of the concept. The hearer (i.e. learner) classifies these examples using the concept classifiers from its own ontology. For each concept, the agent calculates the True Positive Rate (TPR) which is the number of positively classified positive examples divided by the total number of positive examples and the True Negative Rate (TNR) which is the number of negatively classified negative examples divided by the total number of negative examples and uses this information to assess the concept relation which finally results in a mapping between concepts.

A very important point about ANEMONE is that it is not a concept learning

environment, rather it is an environment which facilitates ontology mapping between two agents in a minimal way using a layered approach. Like other related works that we reviewed before, ANEMONE does not talk about feature diversity and it is not a *multi*-agent collaboration. While ANEMONE uses positive and negative examples to assist agents to find mappings between ontology concepts, it does not describe the way of selecting positive and negative examples.

3.4 Information Integration

The main goal of this thesis is to find a method to enable the agents to communicate even if they are semantically heterogeneous. Accordingly we give the agents the ability to dynamically change their ontology by learning new concepts. There is a similar problem in the information retrieval domain which in some manners is related to our problem. Because data or information can be retrieved from many different sources, such as databases, the World Wide Web, knowledge bases, and other specific information systems, integration of heterogeneous information sources is necessary in order to answer user queries. This means that different information sources should be able to relate their understanding of data (i.e. ontology or schema) to each other. Finding the relation between concepts from the diverse information sources leads us to some different methods of mapping or merging or other ways of integration, which is somehow close to our research. In this section we explore different ways of semantic integration that we think are related to our work.

According to [She99] the heterogeneity problem can be classified into four levels: 1) the system level heterogeneity is about the physical layer of the systems

such as incompatible hardware, network communication etc, 2) the syntactical level heterogeneity refers to the different data representations or languages used, 3) the structural level heterogeneity refers to the different data models used, and 4) the semantic level heterogeneity refers to the meaning of the concepts defined. There are many technologies (for example, CORBA, DCOM, XML technologies, or other middleware software products) developed to solve the first three levels of heterogeneity, while semantic heterogeneity requires more complicated methods and satisfactory solutions have yet to emerge.

A standard way to achieve information integration is to build a global schema (schema is usually used to refer to ontology in the information retrieval domain) over all the related heterogeneous information sources, then user queries will point to this global schema. This global integrated schema approach is often used in federated databases and data warehousing, and is sometimes called "data warehousing approach" [VH01]. In general, it is very difficult to construct a global schema from individual database schemas, and even harder for other kinds of information sources. Also, if any of the existing information sources is changed, the global schema needs to be constructed all over again, this is inefficient and a waste of the computational resources.

An alternative "on-demand driven" [VH01] approach is to answer user queries and other requests on-demand by combining or joining information obtained from different sources at runtime based on the mediator architecture [Wie97], [Mel00], [AK97]. A common data model about the application domain and a common query language based on this model is required. This approach is more scalable, flexible and dynamic. Two methods are commonly used in this approach. The first one is

an "eager" paradigm, which collects all data together before answering any queries, while the second is a "lazy" approach, which postpones information collection to the query evaluation stage. Most systems existing today prefer the "lazy" approach, since it is more scalable, and easier to maintain consistency. However, at the core of both methods are algorithms for information combination or translation.

3.4.1 Approaches to Semantic Integration

Different applications may use terms to reflect the semantics of concepts in their domain differently, or they have different conceptualizations about different domains, or even about the same domain. Several kinds of conflicts or mismatches of terms exist. The same term or concept name might have different meanings in different conceptualizations; different terms from different conceptualizations might have the same meaning; or one term might match to several terms of another conceptualization; or one term from one conceptualization does not match any in another conceptualization exactly; or two terms with the same or similar meaning are structured differently in different conceptualizations (e.g., different paths from their respective root concepts). All these conflicts belong to the scope of the semantic heterogeneity problem in the information retrieval domain and must be solved. This can be done in several possible directions depending on the needs of particular applications.

Using a Single Centralized Global Ontology

Similar to the global schema for databases, if a single centralized global ontology is defined for the application domain, all agents or computer programs in communication use terms from this ontology. This is the most traditional way to achieve

semantic integration, but it is likely that not all the agents trust this ontology or agree with it totally, and it is very unpleasant and inflexible that agents can not define their own small ontologies according to their own points of view about the domain. Like the global schema, this approach suffers the cost of constructing and maintaining such global ontology. Gruber [Gru93] identifies many aspect of the knowledge sharing problem that are not addressed by common ontologies. Questions not addressed include how groups of people can reach consensus on common conceptualizations, and how terms can be defined outside their context of use given that agents have commitments to different tasks, representation tools, and domains. While many AI related systems such as KIF [GF92] and Ontolingua [Gru93] have established a common set of ontology description primitives, recent works which are mainly related to the Semantic Web [BLHL01] have focused on different methods for developing and utilizing common ontologies [SGH04].

Merging Source Ontologies into a Unified Ontology

In general, ontologies defined on a common domain by different applications will have a lot of overlap, so merging the source ontologies into one unified ontology before agent interactions is a natural way to fulfill semantic integration among these applications. A manual ontology merging process can be used for small ontologies, but it will be difficult for large-scale ontologies. Automatic or semi-automatic ontology merging methods are necessary. One way to find two terms to be merged can be based on spelling, linguistics or natural language processing techniques with manual validation [Hov98], [SM01]. Most of the tools developed today are based on syntactic and semantic heuristics [Hov98], [NM00] , some also use a description logic based

approach [Hov98]. This ontology merging method for integration is costly and not quite scalable. When any of the knowledge sources is modified, the merging process may need to be repeated. Also, in case of a large number of knowledge sources, a merging result may not be good at all or simply impossible to obtain.

Searching a Set of Mappings Between two Ontologies

Instead of trying to merge two source ontologies, finding a set of mappings between them is an alternative way to achieve semantic integration. A mapping can be defined as a correspondence between concept A in ontology 1 and concept B in ontology 2 which has similar or the same semantics as A. In some existing systems, human experts specify a set of mappings between two ontologies manually; others try to find a mapping between two terms totally based on lexical relations in linguistic or lexical ontologies such as WordNet [KS99], Cyc, SENSUS [Gua97]. By attaching a set of documents to each node to represent its meaning, there are some methods that adopt machine learning text classification techniques to get a similarity measure matrix of two ontologies and search for mappings based on this matrix [MDHD02], [LG01], [SPF02]. There are tools based on ontology algebra and articulation [MWK00], [MW04]. There are also some heuristic or rule based approaches being used, sometimes combined with the structure information of the taxonomy [NM03]. Two recent works published are information-flow-based ontology (i.e. taxonomy) mappings [KS02] which draws on the proven theoretical ground of information flow and channel theory, and a graph matching algorithm based on fixpoint computation called "similarity flooding" [SM02]. Most of the existing ontology mapping approaches are semi-automatic and heuristic [MDHD02]. Automatic

ontology mapping is hardly possible, since an ontology is very subjective and human intervention is always needed to give some initial suggestions to validate machine-produced mappings.

Ontology Translation

Given two ontologies, ontology translation is to translate one of the ontologies into a target ontology which uses the representation and semantics of the other ontology, sometimes with the help of an intermediate shared ontology. Ontology translation is a very difficult problem. Some tools have been developed for ontology translation, including Ontomorgh [Cha00] and OntoMerge [Dou04]. Based on a set of defined rules and transformation operators, Ontomorph [Cha00] offers syntactic rewriting and semantic rewriting to support the translation between two different knowledge representation languages.

Although many of works in the information integration domain have some common ideas with our work, they do not cognitively address the learning and evolution of one ontology. These approaches assume ontologies of different knowledge sources as static inputs to the algorithms they utilized while our approach assumes an intelligent environment in which agents can make decisions, learn and vote and consequently advise another agent in the learning process. Compared with this domain, our two main novel approaches in having agents with different set of features and *multi*-agent teaching/learning even look more interesting.

Chapter 4

Learning from a Group of Teachers: Our Approach

This chapter contains an abstract description of our approach to agents teaching other agents concepts using positive and negative examples. Throughout this chapter, we reference to our application domain to clarify abstract concepts.

First, we begin with addressing some fundamentals and give a precise definition of the problem. Second, we describe the general process of learning and introduce our general interaction scheme. Third, as a major part of the general interaction scheme we explore the process of selecting positive and negative examples and present variant realizations to provide examples with different quality. Finally we introduce non-unanimous concepts and discuss our method to enable agents to learn them and to communicate using them.

4.1 Fundamentals

We begin by addressing some fundamentals to provide a basis for discussion of our approach. These fundamentals are the problem definition, the introduction of notations and key assumptions.

4.1.1 Problem Definition

In a multi-agent environment, due to the ontological diversity of agents, individual agents may create ontologies suitable for their own problem solving needs even if they are describing the same world or domain. An agent may not want to commit to an ontology *a priori* in order to facilitate future communication and sharing of its knowledge. The agent may "selfishly" create its own ontology in order to explain concepts relevant to its own problem solving needs within its domain. To put it in a more formal way, we consider a group of n agents $\mathcal{A}g_1, ..., \mathcal{A}g_n$ which are committed to the following common knowledge:

- a subset $\mathcal{F}_{base}$ of some base features that can be recognized by all agents.
- a set of some base symbolic concepts C_{base} that are known to all agents by name, their feature values for the base features and all the objects that are covered by them.
- all the ontologies used by the agents will use as taxonomy the `subset-relation`.

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 symbolic name but are not semantically equivalent. Also they may use different relations in their ontologies and two agents cannot rely on the same relation identifiers referring to the same relation and vice versa. *Given this setting, agents will develop problems in working together, since the common grounds for communication are not there or too small.* In order to solve this problem,

these agents must learn ontology concepts outside of C_{base} and semantically interpret differing vocabularies in their ontologies rather than attempt to share a common pre-defined ontology.

Despite the fact that all agents can teach the other agents or learn from them, for explaining our problem we consider 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 is a subset of all objects in the world (i.e. $\mathcal{U}$). $\mathcal{A}g_i$ can use pex_i^c to teach c to $\mathcal{A}g_L$. Given this setting we assume a c_{goal} that the $\mathcal{A}g_L$ does not know and wants to learn from other agents.

In this thesis we propose a general method to enable the learner agent to learn c_{goal} and integrate it in its own ontology. Our general method includes proposing a general interaction scheme and different realizations of each step to achieve this goal. Also we propose a general structure for our agents which covers the action set of the agents as well as the structure of messages that the agents are communicating. 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}.

4.1.2 Notations

Because we use many notations, it is easy for a reader to confuse the symbols. Each entry in the Table 4.1 describes a symbol explicitly. Note that some symbols will be

Table 4.1: Description of notations

Notation	Description
$\mathcal{A}g_i$	*represents a teacher agent among a group of n teacher agents* $\mathcal{A}g_1 ... \mathcal{A}g_n$
$\mathcal{A}g_L$	*represents the learner agent*
$\mathcal{O}_i$	*represents the ontology for* $\mathcal{A}g_i$
$\mathcal{U}$	*is the set of all objects in the world*
$\mathcal{F}$	*is the set of all features that agents can use to conceptualize the world*
$\mathcal{F}_i$	*is the unique set of features for agent* $\mathcal{A}g_i$
$\mathcal{F}_{base}$	*is the set of base features that are known and can be recognized by all agents*
$\mathcal{C}_{base}$	*is the set of base concepts that are known all agents*
c_{goal}	*is the goal concept that is being learned by the learner agent* $\mathcal{A}g_L$
c	*represents a concept*
o	*represents an object*
pex_i^c	*is the set of positive examples for concept c in agent* $\mathcal{A}g_i$
nex_i^c	*is the set of negative examples for concept c in agent* $\mathcal{A}g_i$
p_i	*is the set of positive examples* $\mathcal{A}g_i$ *sends to the Learner agent* $\mathcal{A}g_L$
n_i	*is the set of negative examples* $\mathcal{A}g_i$ *sends to the Learner agent* $\mathcal{A}g_L$
c_{core}	*represent the core part of a non-unanimous concept*
$c_{periphery}$	*represents the periphery part of a non-unanimous concept*
c_{own}	*represents the agent's own definition boundary in a non-unanimous concept*
c_{nu}	*represents a non-unanimous concept*

introduced later in this chapter.

4.1.3 Key Assumptions

Our work makes several key assumptions that are much weaker than the assumptions other works have made.

- Ontologies of agents are diverse. Agents may have different sets of concepts and relations and also may arrange the concepts differently.

- Agents use different sets of features to conceptualize the world. Among these

features there is a subset of some base features that can be recognized by all agents.

- Agents know a set of some base concepts by name and their feature values for the base features and the objects that are covered by them.
- Agents can send or receive example objects regarding the concept which is being learned.
- Agents use example objects as part of their knowledge structure and can learn from examples.
- The identity of example objects in the world are accessible to all the agents.

4.2 The General Interaction Scheme

Our basic idea is to have an agent which *learns* required concepts (or at least an approximation) with the *help* of the other agents. Due to the potential differences in the ontologies of agents and in the available features, objects that are positive and negative examples for a concept will play the major role in teaching an agent a new concept.

In this section, we present the general interaction scheme among agents that is the core of our methodology to have an agent learn a particular concept. Then in the following sections we will concentrate in more detail on the important steps of this interaction scheme and discuss possible realizations for them.

Part of Act_L are actions `QueryConcept` to query other agents regarding a concept, `AskClassify` to ask other agents to classify an example concept, `Learn` to learn a

concept from a set of positive and negative examples, and `Integrate` to integrate a newly learned concept in its ontology.

Also part of the Act_is are the actions `FindConcept` to search the ontology to find the best matching concept with the query and make a candidate set of concepts, `SelectBestConcept` to select the best matching concept out of the candidate set, `SelectPosEx` to select a set of positive examples regarding a queried concept, `CreateNegEx` to create a set of negative example regarding a queried concept, `ReplyQuery` to reply to a query, `ClassifyEx` to reply to a query about an example and `ReplyClass` to send the result back to the learner. All of the actions both in the learner and teachers come 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) performing `SelectBestConcept` to select the "best" candidate c_i out of C_i^{cand},

   (c) performing `SelectPosEx` to select 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`($path(c_1)$,...,$path(c_m)$)).

As stated before 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 $\mathcal{O}_L^{new} = (C_L^{new}, \leq_C, R_L, \sigma_L, \leq_{R_L})$. Also $\mathcal{A}g_L$ creates a set $pex_L^{c_{goal}}$ in case another agent wants $\mathcal{A}g_L$ to teach it c_{goal}.

This general interaction scheme is the fundamental part for our methodology that allows agents to learn new concepts out of C_{base}. In fact, this chapter is dedicated to give a detailed description of each step of this general interaction scheme. In the following sections we highlight our preliminary realizations for each step. Then we introduce different realizations for some steps to improve the quality of c_{goal}. This improvement will be accomplished by changing the teachers' positive example selection mechanism to select more comprehensive positive examples and also by changing the negative example creation mechanism to create more discriminative negative examples. We will also propose a new definition for concepts which allows the learner agent to communicate with other agents even if agents are not unanimous about it.

4.3 The Initial Query

In order to submit a 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:

$$\texttt{QueryConcept}(identifier,\{[f'_1 = V'_1],...,[f'_l = V'_l]\},O_{goal}).$$

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 their 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 (if possible). We will present the different parameters making the query in the context of our application example

in Chapter 5 Section 5.1.2 and Section 5.1.3.

Due to our basic assumptions about what our agents do not have 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 $\mathcal{A}g_L$ is 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 sets of objects in the answers that 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 4.7).

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 $\mathcal{A}g_L$.

4.4 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 $V_a" \subseteq V_j"$. This means the value (i.e. $V_a"$) for any feature (i.e. f'_a) of the query should be a subset of values of at least one feature in the concept (i.e. $f_j"$). 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 three query parameters can contribute to a measure that defines what is the "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 (with respect to $\leq_C$) 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 Chapter 6. Note that in theory different teacher agents could use different measures.

After the best concept c_i for the query is determined, an agent $\mathcal{A}g_i$ includes into its answer information about the path that leads in its taxonomy to c_i and the

subtree below c_i's node (we refer to this information by $path(c_i)$). In addition to this information, $\mathcal{A}g_i$ must select a set of positive examples and create a set of negative examples to include into its answer package.

4.5 Selecting Positive and Negative Examples

In this section we present our proposed methods for selecting positive and creating negative examples. Before giving the detailed description of our approach we should discuss some issues regarding the examples and their distribution in the space of all possible objects.

Figure 4.1 shows a pictorial interpretation of the distribution of example objects for a symbolic concept c_j in the space of all possible objects, i.e. $\mathcal{U}$. In most cases the number of negative examples is much greater than the positive examples and that is because every concept other than c_j could be a possible source concept for the negative examples. The excessive number of objects which is covered by these concepts makes the set of negative examples very large.

Figure 4.1 also indicates that the positive and negative examples are distributed both inside the border of c_j and outside. For instance, e_1 represents a positive example in the core of the concept and e_2 represents another one in the border of the concept. e_1 is an instance of c_j in the central part and can clearly represent it, but e_2 is closer to the negative examples and can help in clarifying the concept border with other concepts. While both examples are inside the border of c_j they provide $\mathcal{A}g_L$ with a different quality of advice. We believe that a good method of positive example selection for a teacher agent should guarantee to include a fair

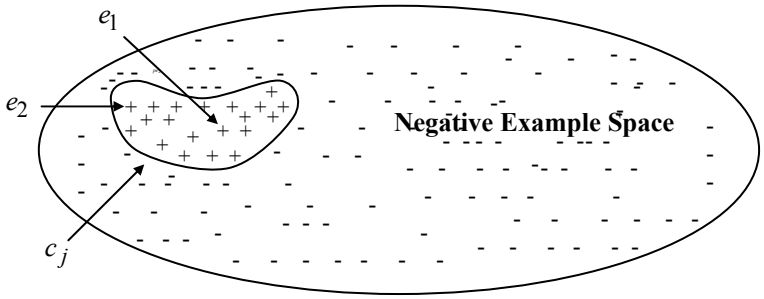

Figure 4.1: Flat representation of example space

distribution of examples both from the border and the core of the concept in the reply package and, in fact, this makes the teacher's answer more comprehensive. For the set of negative examples we need examples that highlight the border of the concept. As Figure 4.1 shows the set of negative examples is very large and this makes the selection of examples a very tricky task. To select the negative examples we can use some information that the agent's ontology might offer us. This information allows us to first concentrate on the concepts that are a possible source for negative examples and then select among the examples that are covered by them.

In the following subsections we present our basic selection methods which are simple and easy to implement. Later we present two rather complex methods that, in fact improve the quality of selected examples and consequently improve the quality of a learned concept for $\mathcal{A}g_L$.

4.5.1 Random Selection of Positive 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 i.e. a set of objects covered by c_j, coming up with positive example objects for a

concept known to $\mathcal{A}g_i$ can be rather straightforward. While providing $\mathcal{A}g_L$ with more examples normally produces better results, 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 spend on learning. On the other side, less positive examples usually means a less precise result of the learning on the $\mathcal{A}g_L$ side. 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 can be most easily realized by randomly sampling $pex_i^{c_i}$.

4.5.2 Using the Taxonomy to Select Negative Examples

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

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

As already stated, 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. Because random selection of negative examples does not guarantee the selection of these "good" examples we need to develop a mechanism that includes such near-miss examples in the set of negative examples.The fact that our agents have ontologies allows us to do a better job in selecting negative examples than randomly selecting out of nex_i^c (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 and then in Section 4.6.1 we will see how we can select even better negative examples using relations in R_i. 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 4.2 indicates these candidates for the selection of negative examples with the help of taxonomy information. Because siblings share many features, the taxonomy assures that the near siblings are good concepts to select negative examples from. Therefore to select these concepts we use examples from siblings of c_i or its superconcepts or even super-superconcepts.

The above mentioned mechanism gives us some concepts that can provide $\mathcal{A}g_i$ with a sufficient number of negative examples in $nex_i^{c_i}$. Nevertheless if the number of negative examples exceeds the system setting of number of examples, then $\mathcal{A}g_i$ selects the appropriate number of elements for n_i simply by randomly sampling $nex_i^{c_i}$.

4.6 Improving the Quality of Advice

In this section we present two alternative realizations for positive and negative example selection. Using these methods we try to improve the comprehensiveness of examples to provide the learner with a better quality of advice. For positive examples we emphasize on the comprehensiveness, therefore we use a new method that selects

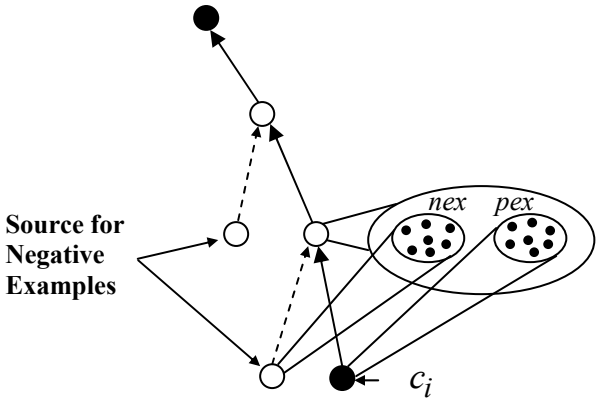

Figure 4.2: Negative examples using the taxonomy

examples that cover the example space. For negative examples we use a relation that we believe is common to many different ontologies to produce more discriminative examples.

4.6.1 Negative Example Selection Using an `is-similar-to` Relation

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. While the taxonomy assures that the near siblings are good concepts to select negative examples from, it does not guarantee that the concepts in the other parts of the tree are not "near-miss". It is very likely that information provided by some other relations can help in the negative example selection process. As an example, let us look at the usage of a 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

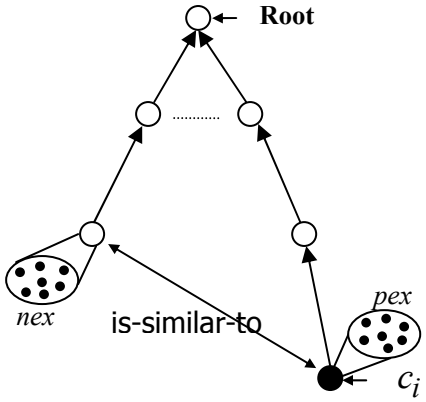

Figure 4.3: Negative examples using the taxonomy and `is-similar-to` relation

negative examples. Figure 4.3 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, therefore we suggest to use the

examples that are already available.

4.6.2 Positive Example Selection by Discriminative Feature Selection and Example Ranking

Random selection of the positive examples is the most straightforward way which, while keeping the selection process simple, does not guarantee the comprehensiveness of the set of selected positive examples, p_i. That is because the random selection of positive examples does not assure to cover the whole space of positive examples. In this section we propose a different realization for positive example selection.

To improve the positive example selection method we looked at the problem from a different point of view. The selection of positive examples is the point that the teacher can exert its unique vision in the teaching of a specific concept, therefore, the teacher agent should utilize some methods to reflect its viewpoint. It is a very common process for human beings that teach to reflect a viewpoint by emphasizing on some features that reflect that viewpoint more. Getting the idea from this general phenomenon, we used the features describing a concept as the points that the teacher wants to express its viewpoint about. Apart from the features in the concept definition in the ontology, there might be some other additional 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 of random selection of positive examples 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 developed a different realization for `SelectPosEx`(c_i). In the new realization 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 *core features*, and denote them by $\mathcal{CF}$. Then we use $\mathcal{CF}$, to extract good positive examples from $pex_i^{c_k}$ that we call *distinctive positive examples*, p_i. Without loss of generality in the following subsections we use pex and nex instead of $pex_i^{c_k}$ and $nex_i^{c_k}$ to keep our discussion simple.

Identifying Discriminative Features

We identify the discriminative features based on the notion called *Relief* which we borrowed from [RiK03]. Using *ReliefF* which is a more robust algorithm from the Relief family we developed an algorithm to identify the discriminative features. This algorithm constructs a set of core features of pex which is denoted by $\mathcal{CF}$, by ranking the feature strengths among the features that are exhibited in pex and nex.

The key idea of this 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 on the two sides of the border. For that purpose, given a randomly selected example e_i (line 3), the algorithm searches for the k most similar neighbors from pex which we call the nearest hits, i.e. H, and k most similar neighbors from

Algorithm 1 Calculates the vector of W of estimations of the features strength

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 to e_i in pex : $H = \{H_1, ..., H_k\}$
5. find k nearest miss examples to e_i in nex : $M = \{M_1, ..., M_k\}$
6. **for all** f in F **do**
7. $W[f] = W[f] - \sum_{j=1}^{k} \text{diff}(f, e_i, H_j)/(m \cdot k) + \sum_{j=1}^{k} \text{diff}(f, e_i, M_j)/(m \cdot k)$
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**

nex which we call nearest misses, i.e. M (line 4). Then it updates the strength estimation $W[F]$ for the set of all features that have been seen in F, depending on their values for e_i, each element of H, and each element of M (lines 6 and 7). If e_i and an example in H are different in their values for the feature f then 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 an example in M are different in their values for the feature f then the feature f separates a positive example from negative examples which is desirable and therefore the algorithm increases the strength estimation $W[f]$. The k is a user definable parameter which increases 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 $\text{diff}(f, e_i, e_j)$ calculates the difference between the values of the feature f for two instances e_i and e_j. We

define nominal features as:

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

and numerical features as:

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

The algorithm then determines ϕ as the average of $W(F)$. The ϕ is a threshold that allows us to select the features which have strength above the average. Because the teacher is interested in the features in pex it filters the set of features and add to the $\mathcal{CF}$ all features which have been seen in at least in one example in pex and $W(f) > \phi$.[1]

Extracting Distinctive Positive Examples

We have shown how to compute feature strengths and how to determine ϕ so as to select a set of discriminative features for formulating the core features ($\mathcal{CF}$) of the positive examples. Another important issue is, given an example, what is the criterion, in order to consider it a potential distinctive positive example?

The key idea behind this method, given in Algorithm 2, is to estimate the distance of every positive example from its peers in the negative side and use this estimation to assess the distribution of the examples in the whole space of positive examples. For

[1] We keep the basic assumptions of the original algorithm unchanged. These assumptions includes division by $(m.k)$ in line 7 and definition of ϕ in line 10.

Algorithm 2 Selects the set p of comprehensive positive examples

1. set all distances $\mathcal{D}[pex] := 0.0$
2. **for all** e_i in pex **do**
3. find k nearest miss examples to e_i in nex : $M = \{M_1, ..., M_k\}$
4. **for all** f in $\mathcal{CF}$ **do**
5. $\mathcal{D}[e_i] = \mathcal{D}[e_i] + \sum_{j=1}^{k} \text{diff}(f, e_i, M_j)/k$
6. **end for**
7. **end for**
8. $p = \emptyset$
9. **while** there is e_i in pex **do**
10. take out of pex an example e_i such that $\mathcal{D}[e_i]$ is minimum
11. **if** $\text{dist}(e_i, e_j) > \theta$ for all $e_j \in p$ **then**
12. append e_i to p
13. **end if**
14. **end while**
15. return p

that purpose and using a similarity function (see Section 4.6.1), for every example e_i (line 3), the algorithm searches for its k nearest neighbors from the nex, i.e. M(line 4). The algorithm then updates the distance estimation $\mathcal{D}[e_i]$ for the set of all features in $\mathcal{CF}$ depending on their values for e_i and each example in M using the diff function (lines 5, 6 and 7). Apparently the examples with lesser values for $\mathcal{D}[e_i]$ are in the border and as the value increases the example moves more away from the border (i.e. into the "inside" of the concept). In order to select a 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 a candidate example e_i. If the value of distance is greater than θ for all selected examples, then the teacher adds it to the set of selected positive example

p_i (line 12 and 13). The value for θ can be simply the average distance between examples. This value can be adjusted based on the system parameter that defines the number of the examples that can be communicated to the learner. In fact to select more examples we should decrease the value for θ and to select less examples we should increase it. Function diff(f, e_i, e_j) is defined similar to Algorithm 1 and function dist is defined based on the diff function as follows:

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

4.7 Learning and Integration of New Concepts

Learning a concept, in the 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. 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 among agents it is possible that concepts c_i and c_j that $\mathcal{A}g_i$ and $\mathcal{A}g_j$ identified do not have much in common. Using a conflict resolution mechanism, $\mathcal{A}g_L$ should create a kind of *compromise* concept c_{goal} as result. After having created the final description of c_{goal} in terms of $\mathcal{F}_L$, $\mathcal{A}g_L$ should also use some techniques to integrate the single and newly learned concept into its taxonomy.

In this section we first describe our conflict detection and resolution mechanisms. Then we present our concept learners which act as target functions to map individual example objects to a particular concept. We also describe a simple integration

algorithm in which we 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}.

4.7.1 Learning of a New Concept Using a Concept Learner

As stated before, in the setting of this thesis $\mathcal{A}g_L$ can learn a concept from scratch. This means the structure of an ontology in the agents is not static and we can not define a multi-class learner to learn the whole ontology. That is because if $\mathcal{A}g_L$ uses one multi-class learner it should *re-learn* the whole ontology when it learns a new concept. To avoid this we used a binary class learning formulation. This enables us to have a binary class concept classifier for every concept that $\mathcal{A}g_L$ has learned. Note that even the teacher agents should have a set of binary class concept classifier for every concept they know. They use these classifiers to classify objects that $\mathcal{A}g_L$ sends to them to resolve conflicts.

As we defined in Chapter 2 a symbolic concept is an abstract phenomenon that is mapped to a set of example objects. We define a concept classifier as a binary function $\mathcal{T}_{c_j}$ that maps an example object e_i to a particular concept c_j or not. More formally we define $\mathcal{T}_{c_j}$ as the following:

$$\mathcal{T}_{c_j}(e_i) = \begin{cases} 1 & \text{if } e_i \text{ is an instance of } c_j \\ 0 & \text{otherwise} \end{cases}$$

Using pex^{c_j} and nex^{c_j} and a supervised inductive machine learning algorithm the learner agent can learn an approximation target function $\mathcal{H}_{c_j}$ such that $\mathcal{H}_{c_j} \approx \mathcal{T}_{c_j}$ for every concept c_j respectively. Analogously using $pex^{c_{goal}}$ and $nex^{c_{goal}}$, $\mathcal{A}g_L$ learns an approximation target function $\mathcal{H}_{c_{goal}}$.

Throughout this thesis we use two standard learning methods as our concept learners. Each method represents a different machine learning approach: generative modelling using a naive Bayes [Mit97] learner and the Rocchio algorithm [Roc71].

A naive Bayes learner is a simple probabilistic learner based on the Bayes' theorem with strong (i.e. naive) independence assumptions. A more descriptive term for the underlying probability model would be the independent feature model. Depending on the precise nature of the probability model, naive Bayes learners can be trained very efficiently in a supervised learning setting. In spite of their naive design and apparently over-simplified assumptions, naive Bayes classifiers often work much better in many complex real-world situations than might be expected [Mit97].

The basic idea of the Rocchio algorithm is to construct a prototype vector to represent the examples for each concept and compare a subsequent example with it to classify that example. Using positive and negative examples describing the same concept and a summation formula, the Rocchio algorithm computes a prototype vector as the centroid vector of the concept. To determine whether an example belongs to a concept or not, the similarity between the example and the prototype vector is measured using the cosine product. We will describe these learners in the context of our experimental evaluation in Chapter 5 in more detail.

4.7.2 Conflict Resolution

Learning from a group of agents is a very conflict prone process compared to just learning from one agent. It can easily happen that the best concepts c_i and c_j that $\mathcal{A}g_i$ and $\mathcal{A}g_j$ identified are not the same. The worst case can be that an example that $\mathcal{A}g_i$ sent as being positive for c_{goal} is considered as a negative one by $\mathcal{A}g_j$. In

most cases such a contradiction will stop the 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 $nex^{c_{goal}}$. In order to solve these problems the learner agent must detect the conflicts and resolve them.

As we stated in Section 4.7.1, using positive and negative examples and a machine learning algorithm, for every concept c_j, we approximate the concept classifier with a target function $\mathcal{H}_{c_j}$. This concept classifier is assumed to classify an example object e_i regarding c_j. Needless to say, we expect that $\mathcal{H}_{c_{goal}}$ classifies $e_i \in pex^{c_{goal}}$ as a positive and $e_i \in nex^{c_{goal}}$ as a negative example. We consider these truly classified examples as consistent examples. Due to the fact that $\mathcal{H}_{c_{goal}}$ is trained by mostly *consistent* examples, we consider the *inconsistent* examples as possible conflicts. Therefore after the learning component of $\mathcal{A}g_L$ has performed `Learn` and produced $\mathcal{H}_{c_{goal}}$, $\mathcal{A}g_L$ will test all elements of $pex^{c_{goal}}$ and $nex^{c_{goal}}$ for correct classification by this new c_{goal}. We consider all the example objects that are not correctly classified, as possible conflicts.

To resolve conflicts, we go back to the teacher agents and ask them to classify each conflicting example according to the c_i they used to produce their examples. We then treat the answers as votes and include all positive examples for which a majority of the teachers voted, while requiring the exclusion of all negative examples for which a majority voted. This produces some kind of compromise concept that might appeal to most of the teachers (although it might not be identical to any of the c_is).

Apparently, there are other conflict resolution methods. If the learner wants to

be very strict and sure that what c_{goal} produces is a subset of each c_i, it will only accept positive and negative examples for which the vote was unanimous. On the other side of the spectrum of possible methods is to accept every object as positive example for which at least one teacher says this is a positive example (and to adjust the negative examples accordingly). This might result in a very generic concept, but there can be situations where this is what a user might want.

As discussed above, each possible conflict resolution method will come up with some concept for the learner's ontology according to our definition of a concept from Chapter 2 Section 2.2. The fact that each of these outcome concepts makes sense points to a general problem of the definition of concepts in ontologies, at least if we think about our general goal, namely allowing an agent to communicate with other agents without having the use of a common ontology. Therefore we will rethink our concept definition to allow for the use of the information the learner has collected with regard to the opinions of the teachers about objects in and outside of c_{goal} in Section 4.8.

4.7.3 Pre-Structuring of the Learner's Ontology

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 [WPB03]). But since the goal of our whole method is to improve the communication among 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 sent 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.

Algorithm 3 Simple pre-structuring algorithm

1. Strip all concepts $c_j \in path(c_i) \bigwedge c_j \notin C_{base}$ of all features $f \notin F_{base}$
2. Merge similar concepts in $path(c_i)$
3. Merge similar concepts in all $path(c_i)$ of different agents to *shellpool*
4. Make a set P of all alternative paths by $c_j \in$ *shellpool* using $\leq_C$
5. Integrate the longest path of P in $\mathcal{O}_L$

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. Algorithm 3 shows the simple pre-structuring algorithm that we used. 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 similar 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 $\mathcal{O}_L$.

4.8 Non-unanimous Concept Ontologies

One of the basic assumptions of our work is that different agents will often have at least slightly different definitions for a concept, due to the reasons mentioned before. This reflects well what we observe among human beings. If all communications only involve two agents/persons, this fact would not produce a lot of problems, since an agent could learn the definition of a concept of every other agent it communicates with and use the appropriate definition in each communication (this might be expensive, but is possible within the known definitions and methods). But the moment an agent has to communicate with two or more other agents at the same time, there is a serious problem: what if these agents differ in their definitions of the concepts the communication is about?

The solution used by human beings is to be aware of what (most) agents agree on and to provide additional information to clarify aspects that parts of the audience might misunderstand. Note that this does not mean that an agent is unclear about its own definition of a concept. We believe that for the individual work of an agent having a precise definition of a concept is important. The ability to be aware of potential misunderstandings when communicating with other agents is the goal that we are trying to achieve using non-unanimous concepts. Later in Section 4.8.2 we will propose a method that realizes how to obtain this ability for an agent.

Below, we will first introduce an extension of the definition of a concept from Chapter 2 Section 2.2 and subsequent extensions to the ontology definition that allow us to represent the potential aspects of misunderstanding in communicating about a concept. Then we will show how to create such extended concepts and

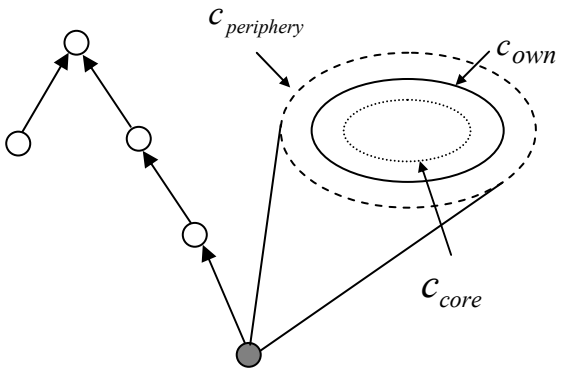

Figure 4.4: A Non-unanimous Ontology Concept

ontologies in a very straightforward manner, if the group of teacher agents covers well the possible differences in understanding a concept. Finally, we present a way how to make use of such extended concepts in communications to a group of agents.

4.8.1 Non-unanimous Concepts

In order to allow to express the range of possible misunderstandings about a concept, instead of representing a concept by one feature set, we use three such feature sets, which means that we essentially use three concepts:

$$c = (c_{core}, c_{own}, c_{periphery}).$$

These three "normal" concepts provide us with two boundaries and the agent's own definition of the particular concept (represented by c_{own}).

Figure 4.4 gives a graphical representation of the trio of "old" concepts that we use to represent a *non-unanimous concept*. The inner boundary c_{core} is intended to provide the agent with a concept definition that represents all objects for which there is no doubt among all agents that they belong into the concept, so c_{core} covers the

core of the concept. The outer boundary $c_{periphery}$ covers all objects that ever might be considered to belong to the concept, which means that all objects not covered by $c_{periphery}$ for sure are not in the concept c. Hence, essentially $c_{periphery}$ defines the extend of the *periphery* of the concept.

Figure 4.4 also visualizes an obvious requirement for the three concepts that make up a non-unanimous concept, namely that $c_{core} \leq_C c_{own} \leq_C c_{periphery}$ (remember that $\leq_C$ essentially is the subset relation). Note that there is no requirement that c_{core}, c_{own} and $c_{periphery}$ are different from each other. In fact, if there is no possibility for misunderstandings, as for example for a concept in C_{base}, then we have $c_{core} = c_{own} = c_{periphery}$.

If we want to use non-unanimous concepts within ontologies, then most of what we defined in Chapter 2 does not have to be changed. The only potential problem is $\leq_C$, since there is always the chance that the peripheries of two concepts might overlap (given that the objects in $c_{periphery}$ - c_{core} are somewhat questionable with regard to really representing the concept and the objects in $c_{periphery}$ - c_{own} are not covered by the concept in the point of view of the agent). If we have to provide the equivalent of $\leq_C$ for non-unanimous concepts, then we will use the relation $\leq_{C_{nu}}$, which we define as

$$(c_{core_1}, c_{own_1}, c_{periphery_1}) \leq_{C_{nu}} (c_{core_2}, c_{own_2}, c_{periphery_2}),$$
$$\text{iff for all } o \in c_{periphery_1} \text{ we have } o \in c_{periphery_2}.$$

Finally, we should mention that all the three concepts c_{core}, c_{own} and $c_{periphery}$ of some $c_{j_{nu}}$ naturally have a presentation as feature value sets according to our preliminary definition. For a feature f_i this means that we have now three value sets, namely $V_{i_{core}}$, $V_{i_{own}}$ and $V_{i_{periphery}}$, with $V_{i_{core}} \subseteq V_{i_{own}} \subseteq V_{i_{periphery}} \subseteq D_i$ where

D_i is the set of possible values the feature can have. So, associated with potential misunderstandings in communication will be certain feature values for some or all the features that an agent uses in its ontology.

4.8.2 Learning Non-unanimous Concepts

Now that we have defined non-unanimous concepts and how they fit into the ontology of an agent, the next question is how do we create such a non-unanimous concept for an agent. Fortunately, the answer is very straightforward: by learning the non-unanimous concept using our method from Section 4.2. Without non-unanimous concepts, we have to choose one of the three conflict resolution methods that we described in Section 4.7.2 and our suggested method was to use the concept that could be learned from the positive and negative examples on which a majority of the teachers agreed on. If we represent concepts by the sets of the objects covered by them, then Figure 4.5 visualizes the three different possible outcomes of the concept learning process for the three conflict resolution methods for three agents $\mathcal{A}g_1$, $\mathcal{A}g_2$ and $\mathcal{A}g_3$ and the candidates c_1, c_2, and c_3 that these agents selected.

If we assume that the set of teachers that $\mathcal{A}g_L$ uses to learn a concept represents well the different understandings that a group of agents has about a concept, then the three different conflict resolution strategies seem to be a perfect way to learn the 3 concepts $c_{core_{goal}}$, $c_{own_{goal}}$, and $c_{periphery_{goal}}$ that we need to define a non-unanimous concept. We use what the learner produces out of the examples all the teachers agree on as $c_{core_{goal}}$ and the result of the concept learning when using everything as positive example that is suggested by at least one teacher as a positive example is $c_{periphery_{goal}}$.

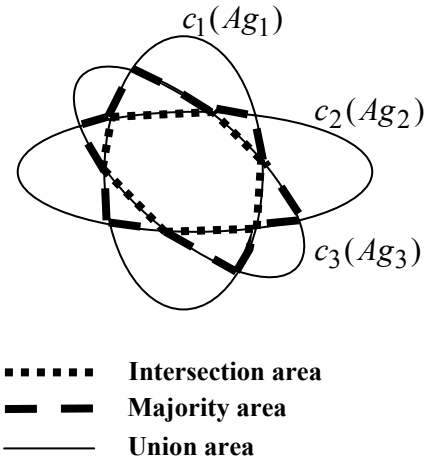

Figure 4.5: Visualization of Conflict Resolution

In defining $c_{own_{goal}}$ we essentially can use any concept that lies between $c_{core_{goal}}$ and $c_{periphery_{goal}}$ (including using one of these concepts). Since $c_{own_{goal}}$ is intended to express the agent's personal belief in what the concept really is, we still think that using some kind of compromise concept between the two extremes is a good idea and using the result of what a majority of the teachers agrees on is a good compromise. Given the usage of non-unanimous concepts that we target, it makes sense to create a set of positive examples for each of the 3 concepts $c_{core_{goal}}$, $c_{own_{goal}}$, and $c_{periphery_{goal}}$. We will call these example sets $pex^{c_{core_{goal}}}$, $pex^{c_{own_{goal}}}$, and $pex^{c_{periphery_{goal}}}$.

4.8.3 Using Non-unanimous Concepts in Group Communications

Before we present our general method for using non-unanimous concepts in communicating with groups of agents, let us look at an example that demonstrates the problem we are solving. This example stems from introductory Math classes and essentially represents the question: Is zero a natural number? The concept of nat-

ural numbers is introduced in nearly every elementary Math course and while there is absolutely no disagreement between mathematicians with regard to 1, 2, 3 and so on being natural numbers (and -1, -2, -3 and so on not being natural numbers), we often are told that 0 is a natural number in introductory courses of set theory, while introductory courses in number theory usually are not seeing 0 as a natural number. Using non-unanimous concepts, we can model this fact by having c_{core} as the set of number objects 1, 2, 3 and so on and $c_{periphery}$ as the set of number objects 0, 1, 2, 3 and so on.

An agent $\mathcal{A}g$ (which could be a computer science student) that has to learn about natural numbers will make its own decision about zero. But regardless of this decision, what if this agent has to communicate with a large group of mathematicians (including both number and set theoreticians) about something related to natural numbers? If $\mathcal{A}g$ has decided to favor the view of 0 being a natural number (i.e. $c_{own} = c_{periphery}$), then by using the concept identifier "natural number" and explicitly stating that whatever he says holds true for 0 (which is the sole element of $pex^{c_{own}}$ - $pex^{c_{core}}$) $\mathcal{A}g$ can be sure that his communication is understood by everyone (although some might see some redundancy in this communication). On the other side, if $\mathcal{A}g$ decided to have $c_{own} = c_{core}$ and wants to express that something does not hold for all natural numbers, then it should again use the concept identifier "natural number" in its communication and explicitly state that this something also does not hold for 0 (since now 0 is the sole element of $pex^{c_{periphery}}$ - $pex^{c_{own}}$).

More precisely, assuming that after learning a concept, learner and the teachers establish a common concept identifier, using non-unanimous concepts to communicate with groups is based on enhancing the usage of the concept identifier with

additional objects to convey to the listeners the agent's understanding of the concept. If we communicate positively about a concept, these additional objects are taken from $pex^{c_{own}}$ - $pex^{c_{core}}$ (and from $pex^{c_{periphery}}$ - $pex^{c_{own}}$ if we want to refer to the universe without the concept). Note that the obvious solution of simply referring to all objects that are in c_{own} but not c_{core} by features and their values is not open to us, since we assume that different agents have different sets of features.

While it is possible to simply use the whole set $pex^{c_{own}}$ - $pex^{c_{core}}$ in the communication, we have to assume that this might be a large set (in fact, there might be infinitely many objects in c_{own} that are not in c_{core} and using the stored positive examples is already just an approximation). Communicating all these examples might be too costly for the agent and therefore we assume that the first step of this agent is to determine a maximal number $exmax$ of objects that it is willing to use in the communication. $exmax$ might be a constant parameter or it could be adjusted based on the success of earlier communications with a group of agents.

In order to select the $exmax$ examples, we naturally want these examples to cover the difference between c_{own} and c_{core} as best as possible. Again, there are different methods how this can be achieved. A simple, but in our opinion good enough, method is the following:

$$\text{Let } FC = \cup_{i=1}^{n}\{(f_i, v_i) | v_i \in V_{i_{own}} - V_{i_{core}}\}.$$

For each element o in $pex^{c_{own}}$ - $pex^{c_{core}}$ compute:

$$cover(o) = |\{(f,v) \in FC | v \text{ is a value of feature } f \text{ for } o\}|$$

and select one of the elements, let's say o', with the highest $cover$ value. Then remove all the (feature, feature value) pairs of o' from FC and repeat this selection

$exmax - 1$ times.

Chapter 5

An Example Application

In this chapter we describe how we implemented an instantiation of our concepts in the form of a multi-agent system. We used this system as a prototype to investigate our novel approach to concept learning among agents with diverse ontologies.

5.1 Problem Domain

To evaluate our concepts from Chapter 4, we have chosen the course catalog ontology domain (see [UIl]) which is a very popular domain in ontology research. 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. As stated before, our teacher agents will be agents that each represents one of these three universities in the following called $\mathcal{A}g_C$, $\mathcal{A}g_W$, $\mathcal{A}g_M$. The course files (and unit structure) for Cornell and Washington were taken from[UIl], the ones for Michigan from their web site at [UMi]. The three universities together offer 19061 courses and each university's ontology has at least 166 concepts on top of their courses. Table 5.1 shows the characteristics of our domain that are of interest [MDHD02].

Table 5.1: Domain characteristics

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

5.1.1 Concepts in Domain

As stated before, ontology of each agent is formed according to the university units that it is representing. Clearly the most general concept is the university itself that has all other units as its subconcepts. On the other hand the more specific concepts are programs that each department offers or in case of not having any official program, a department itself can be a specific concept.

As Figure 5.1 shows `Biological_Anthropology` is a program in department of Anthropology which makes it a more specific concept. Department of Anthropology and higher units are more general concepts. By Learning of `Biological_Anthropology` concept the learner agent provides someone in the University of Michigan with suggestions for how this concept should be characterized.

5.1.2 Objects in Domain

As stated in Chapter 4, associated with each concept there is set of positive examples representing it. These sets of objects in our domain are consisting of files describing courses offered in each program. A course file contains a course identifier, a course description and the prerequisites of a course. What we really need is the

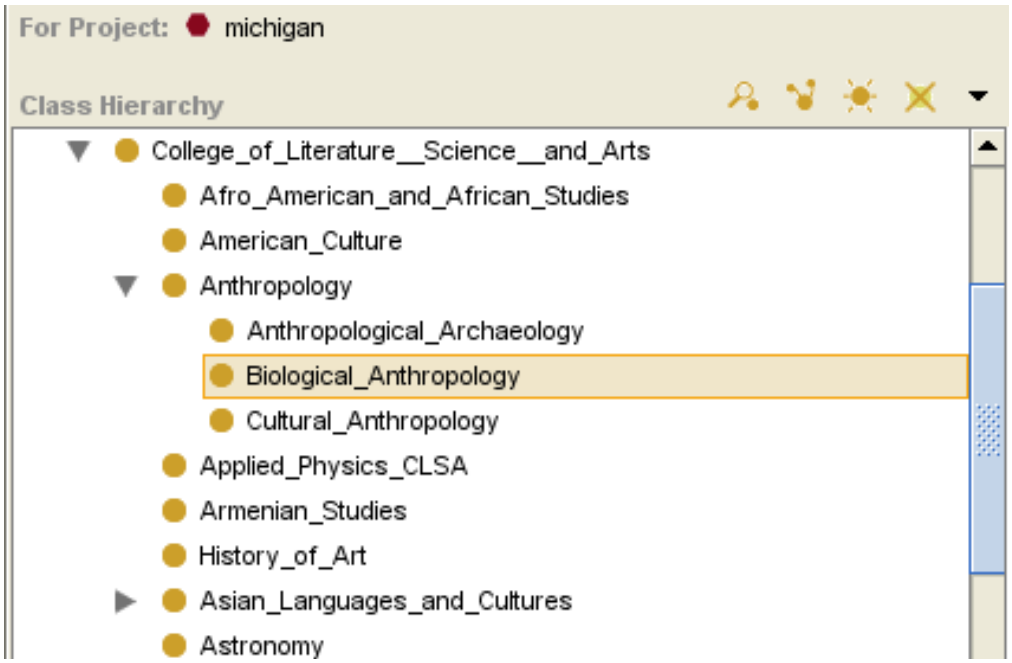

Figure 5.1: Specific concept VS more general concepts in $\mathcal{A}g_M$

course description part which based on it we form our features. Figure 5.2 shows two example objects for `Biological_Anthropology` concept. Red, blue and black colors are showing the course identifier, course prerequisite and course description respectively.

One important assumption in this thesis is that the example objects in the world are *must* be accessible to all the agents. That is because the teacher agents are teaching concepts using objects. We defined a uniform way of locating example objects in our proof of concept application. Based on our Uniform Object Locator (i.e. UOL) scheme each agent points to the objects using a combination of IP address and the uniform path of the course description file.

ANTHRBIO_450_ Molecular_Anthropology.

At least one anthropology or biology course. (3). (Excl). (BS). May not be repeated for credit.

The course encompasses the theory and methodology of molecular genetic data collection, as well as the analysis of molecular genetic data. Molecular genetic data is used to illustrate the basic principals of population genetics and molecular evolution. Class projects involve analyses of molecular data.

ANTHRBIO_562_Human_Nature.

Consent of instructor required (Prerequisites enforced at registration). ANTHRBIO 467. (2). (Excl). (BS).

An advanced seminar in evolutionary psychology. Topics covered include: sexual selection, mating systems theory, parental investment, reciprocity, morality, and religion.

Figure 5.2: Two example objects representing `Biological_Anthropology` concept

For instance, UOLs for the example objects in Figure 5.2 are:

192.168.213.1/michigan/Literature_Science_and_Arts/Anthropology/Biological_Anthropology/ANTHRBIO_450_Molecular_Anthropology

192.168.213.1/michigan/Literature_Science_and_Arts/Anthropology/Biological_Anthropology/ANTHRBIO_562_Human_Nature

The teacher agents use these UOLs to point to the objects that they like to teach a concept with. The learner agent also use these UOLs to locate the objects and learn from them or resolve any possible conflicts. Additionally these UOLs can be used to query the teacher agents when the learner agent asks about a concept using its representing objects.

5.1.3 Feature Preparation for Example Representation

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. Features describing a course are its identifier f_{idnt} which is a string, and its prerequisites f_{prereq} which is 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 third feature that we are interested in, is f_{descr} which is a simple text-based course description. The course description usually determines by which organizational units a course should be taught and a simple definition of the course. Textual information in this feature uniquely represent a course and give us the ability to conduct a successful learning process.

Defining concepts based on objects that consist of natural language texts is not easy, but an area of quite a lot of interests and practical applications. One way of representing text documents in information retrieval and text classification is the *bag-of-words* technique in which text documents are parsed into a single-word terms vector(i.e. keywords) with the term frequencies in each component. Unfortunately this scheme is not general enough to meet our needs in representing our example objects. That is because the *bag-of-words* technique represents documents in the form of single-word terms vectors and it is not guaranteed that these keywords will be present in every example objects. This means that we may not be able to represent a set of examples for a concept using a set of relevant features. As we defined in Chapter 2 Section 2.2, we refer to a concept as a set of objects with a set of common features and the same value or set of values for those features. Therefore

if we consider each term as a feature then its appearance in one example object and absence in another one may make it impossible to represent the set of examples using a consistent set of features as we defined in Chapter 2. In fact we need a mechanism in which the example objects are represented with a set of features that have the appropriate values to uniquely identify a concept.

One other way of defining features for such texts is to group them to look for particular words in the texts or word combinations [Sah96], [PS03]. Using this method we combine some keywords (i.e. terms) to make a new feature. For example, feature $f_{\texttt{picture,photo,figure}}$: text $\rightarrow$ Boolean is a feature which is made by combining `picture`, `photo` and `figure` and it is true for a text t, if *either* one of the keywords occurs in t. This word combination helps us to overcome the above mentioned problem and allows us to come up with a fixed set of features that represents every objects of a particular concept for each agent.

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. A very important question here is how we can form a set $\mathcal{K}$ of key words knowing that the bag of words for objects representing a concept usually is very large. A major characteristic, or difficulty in the systems that use textual documents is the high dimensionality of the key word space. This unique set of key words that occurs in documents (i.e. objects) can be tens or hundreds of thousands of terms for even a moderate-sized object collection. While random selection of words is a simple solution it is not intelligent. Therefore for every particular concept learning process and in a pre-processing phase in every teacher agent, we use some techniques from the information retrieval domain

```
computer   26.0
system   19.0
design   18.0
science   14.0
performance   13.0
model   12.0
theory   12.0
parallel   12.0
algorithm   9.0
technology   9.0
language   9.0
logic   9.0
analysis   8.0
program   8.0
structure   6.0
synthesis   6.0
knowledge   6.0
development   6.0
process   5.0
formal   6.0
project   5.0
information   5.0
digital   5.0
software   5.0
control   5.0
complexity   5.0
circuit   5.0
data   4.0
```

a) Computer Science

```
latin   111.381
greek   72.726
roman   36.194
literature   25.484
classic   20.557
modern   16.049
epic   15.437
attic   11.454
drama   11.339
ancient   10.341
antique   10.341
painting   9.774
culture   8.803
sculpture   8.028
medieval   7.917
religion   7.809
odyssey   7.768
orator   7.406
prose   7.379
tragedy   7.325
horace   7.100
pagan   6.528
aristotelian   6.448
democritus   6.448
aesthetics   6.357
cicero   6.357
herodotus   5.671
tradition   5.597
```

b) Greek

Figure 5.3: Keywords for `Computer Science` and `Greek` using DF and $\mathcal{X}^2$ statistics

to reduce the key words dimensionality. Our automatic key word selection methods include the removal of non-informative words according to a set of objects statistics, and the construction of new features which combine lower level key words into higher level orthogonal dimensions. We utilized two different key word selection methods: Document Frequency (DF) and $\mathcal{X}^2$ statistics (see [YP97]) to select key words that are basically creating the feature sets for our agents regarding every concept which is being learned. Figure 5.3 part a) shows the output of our key word selection module

for the concept `Computer Science` using the DF method while part b) shows some key words as a part of the output for the concept `Greek` using $\mathcal{X}^2$ statistics, both for $\mathcal{A}g_M$. The numbers show the appropriate priority number that different methods produce for each key word.

To make our description complete, let us assume that the key words in Figure 5.3 part a) are the set $\mathcal{K}$ for one agent. Then among features representing example objects for `Computer Science` we have:

$$f_{\texttt{design,computer,system}},\ f_{\texttt{logic,circuit}},\ f_{\texttt{data,model,theory}}$$

$$f_{\texttt{knowledge,control}},\ f_{\texttt{program,process,formal}},\ f_{\texttt{language,parallel,project}}$$

These features are utilized to represent *every* example for `Computer Science` and in case of appearance of only one key word of each feature in one example its value is true and in case of absence of all key words of the feature, the value of the feature is false.

As mentioned above, we use some statistical methods to automatically produce the set $\mathcal{K}$ of keywords for the teacher agents. This is possible because for any concepts, there are many associated objects that the teacher agents can use as the input for our statistical analysis. Nevertheless this is not possible for the learner agent. That is because $\mathcal{A}g_L$ does not have any associated objects for a concept which has not been learned yet. It should be noted that the set of features describing the world for an agent is completely subjective. In fact an agent either can be equipped with some sensors that defines the origins of the features it uses to conceptualize the world or can be directly supplied by the set features from the user and based on the environment it is deployed. In our context and in order to supply $\mathcal{A}g_L$ with the set

$\mathcal{K}$ of keywords we used the key word sets of the teacher agents to make a unique but overlapping key word set for the learner. We have chosen a selection of key words from the teacher agents to enable $\mathcal{A}g_L$ to have a reasonable overlap with the teachers regarding the features. Then we added some unique key words that were related to the concept that was being learned to define a unique feature set for the learner.

5.2 Ontology Construction

In our system, an ontology is a hierarchical structure that in conjunction with the set of examples and the concept learner helps agents to conceptualize the world. To enable our agents to work with ontologies we had some preparations to do. First we had to construct the ontology and express it in a standard ontology language. Then we had to develop an application programming interface to use it. There are various different ontology languages available for representing ontology information on the Semantic Web. The most expressive one of these languages is OWL [owl], which has different flavors for different complexities. To have the OWL format of our ontology we had a long way to go. First the information of universities which were the taxonomies of units in an unstructured form has been created in simple textual form. In the second step we used Protege [pro] to make our ontologies. Protege is a platform that provides a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protege implements a rich set of knowledge modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Fortunately Protege ontologies can be exported into a variety of formats including OWL which was

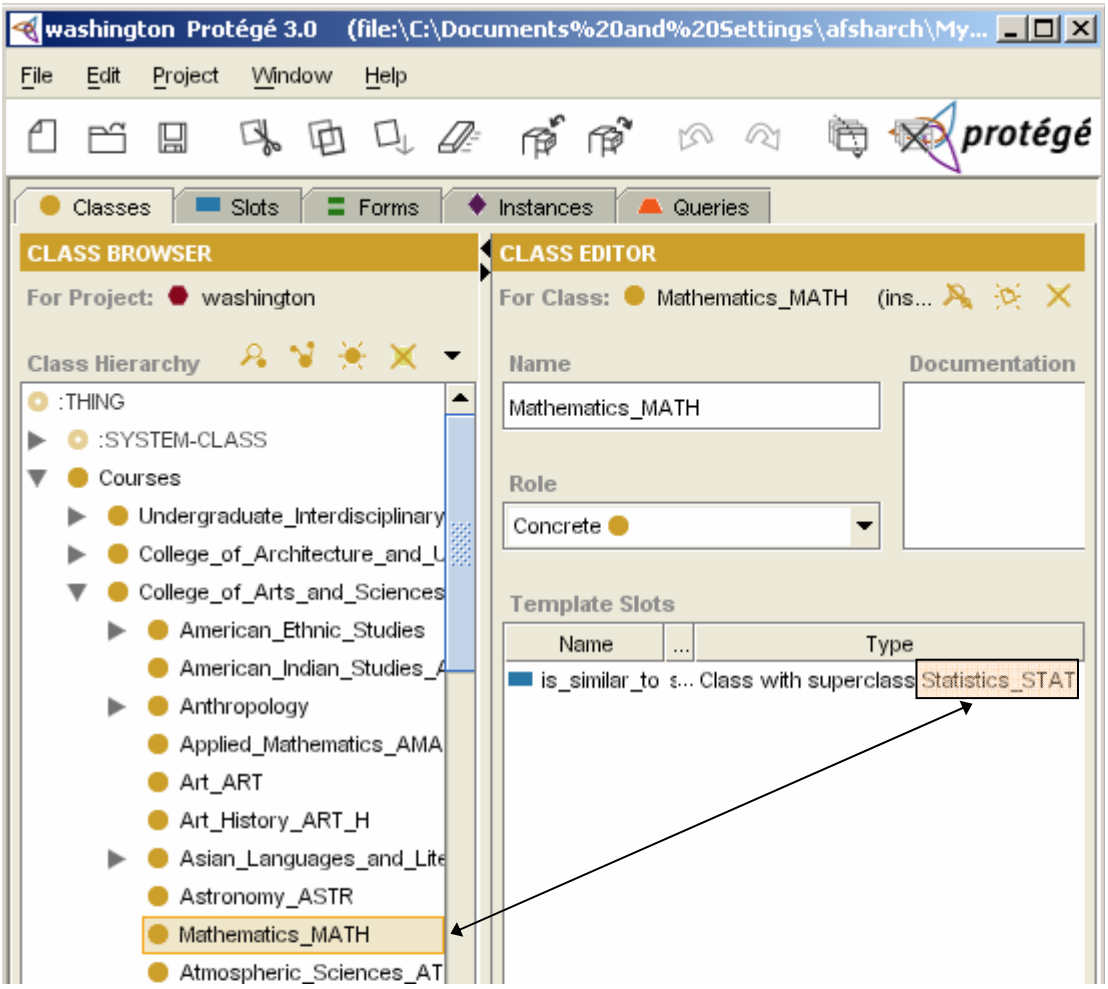

Figure 5.4: A snapshot from Protege showing a part of ontology for $\mathcal{A}g_W$

our target format. Figure 5.4 shows a snapshot from Protege showing a part of the ontology for the University of Washington. Figure 5.5 shows a part of the ontology of the University of Washington in OWL format for the graphical representation of Figure 5.4. As stated in Chapter 4 Section 4.6.1 we use information that is provided by some specific relations to improve the quality of our negative examples. We used `is-similar-to` to show how this relation improves the good coverage of negative examples. To show this relation in our ontologies we used a computation-intensive

```
- <owl:Class rdf:ID="Computer_Science_and_Engineering_CSE">
    <rdfs:subClassOf rdf:resource="#College_of_Engineering"/>
  </owl:Class>
- <owl:Class rdf:ID="Art_ART">
    <rdfs:subClassOf rdf:resource="#College_of_Arts_and_Sciences"/>
  </owl:Class>
- <owl:FunctionalProperty rdf:ID="is_similar_to">
    <rdfs:domain rdf:resource="#Mathematics_MATH"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <protege:allowedParent rdf:resource="#Statistics_STAT"/>
    <rdfs:range rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </owl:FunctionalProperty>
</rdf:RDF>
```

Figure 5.5: OWL format for ontology

method to find out the level of similarity between two concept. Using the similarity function of Section 4.6.1 and for every agent, we first compared every concept in the ontology with each other. Then we ranked the results for each concept and selected top five concepts as the similar concepts. Then using the graphical user interface of Protege we setup the similar concepts to each concept. Finally after constructing ontologies we exported the ontologies to OWL format which has been utilized by the ontology handler component of our agents.

To use ontologies the agents should be able to navigate through the ontology to find concepts according to queries and add new concepts or integrate new paths to it. We used Jena [jen] to satisfy these requirements. The Jena ontology API aims to provide a consistent programming interface to the ontology application developers, independent of which ontology language is being utilized. Although Jena satisfies many of our requirements on navigating of ontologies it does not address our requirements when it comes to search. We need a search mechanism that navigates

all over the concepts in the ontology to find a specific concept based on its features. In order to achieve this capability we extended Jena to support our specific needs of searching.

5.3 Concept Learners

As we stated before throughout this thesis we use two standard learning methods as our concept learners: naive Bayes [Mit97] and Rocchio [SB91]. In this section we describe how we adopted these two algorithms for our instantiation.

5.4 Naive Bayes Concept Learner

In this section we present the naive Bayes learner in the context of our experimental domain. We based our naive Bayes learner on [Mit97] and modified it to achieve our specific requirements. We first modified the algorithm to learn according to the specific key word set of the learner agent (i.e. $\mathcal{K}_{\mathcal{A}g_L}$). That means the algorithm ignores the key words that are not present in the $\mathcal{K}_{\mathcal{A}g_L}$. Second we changed the definition of features from just a key word to a combination of key words and consequently we changed the method of calculation of class conditional probabilities. Finally we modified the algorithm to learn two classes: $fit_{c_{goal}}$ to show how an example object belongs to c_{goal} and $\neg fit_{c_{goal}}$ otherwise.

The modified version of the naive Bayes algorithm that we present in Algorithm 4 applies to the following general setting. Consider an example space X consisting of all possible examples describing a concept c_{goal}, so we have $X = pex \cup nex$.[1] We are

[1] To keep our presentation simple we denote $pex^{c_{goal}}$ and $nex^{c_{goal}}$ by pex and nex respectively.

given training examples of some unknown target function $\mathcal{T}_{c_{goal}}$, which can take one value from two possible values. The task is to learn a target function $\mathcal{H}_{c_{goal}}$ from these training examples to classify subsequent example objects and $\mathcal{H}_{c_j} \approx \mathcal{T}_{c_j}$. For illustration, we will consider the target function classifying course descriptors which are in form of small documents(i.e. strings) as interesting (i.e. instance of concept) or uninteresting (i.e. not instance of concept) to the learner agent, using the target values $fit_{c_{goal}}$ and $\neg fit_{c_{goal}}$ to indicate these two classes.

Algorithm 4 Modified naive Bayes algorithms for learning c_{goal}

1. $T \leftarrow$ A vector of collection of all key words that occur in $X = pex \cup nex$ and are in $\mathcal{K}_{\mathcal{A}g_L}$
2. $P(fit_{c_{goal}}) \leftarrow \frac{|pex|}{|X|}$
3. $S_p \leftarrow$ a single vector created by key words occurring in pex that are in $\mathcal{K}_{\mathcal{A}g_L}$
4. $n_p \leftarrow$ total number of distinct key words in S_p
5. **for all** f_k that are of interest to $\mathcal{A}g_L$ **do**
6. $\quad n_k \leftarrow$ sum of the number of times each key word in f_k occurs in S_p
7. $\quad P(f_k|fit_{c_{goal}}) = \frac{n_k+1}{n_p+|T|}$
8. **end for**
9. $P(\neg fit_{c_{goal}}) \leftarrow \frac{|nex|}{|X|}$
10. $S_n \leftarrow$ a single vector created by key words occurring in nex that are in $\mathcal{K}_{\mathcal{A}g_L}$
11. $n_n \leftarrow$ total number of distinct key words in S_n
12. **for all** f_k that are of interest to $\mathcal{A}g_L$ **do**
13. $\quad n_k \leftarrow$ sum of the number of times each key word in f_k occurs in S_n
14. $\quad P(f_k|\neg fit_{c_{goal}}) = \frac{n_k+1}{n_n+|T|}$
15. **end for**

To calculate the class for a subsequent example object, we require estimates for the probability terms $P(fit_{c_{goal}})$ and $P(f_k|fit_{c_{goal}})$. The first of these can easily be estimated based on the fraction of each class in the training examples (line 2). Because we setup the teacher agents to send an equal number of positive and negative examples these prior probabilities are 0.5. As usual, estimating the class conditional probabilities is more problematic because we must estimate one such probability

term for each feature f_k that we have used to represent our examples.

Fortunately, we can make a reasonable assumption that reduces the number of probabilities that must be estimated. In particular, we shall assume the probability of encountering a specific feature f_k is independent of another feature f_j (although there might be some dependency because of the same key words occurring in several features, to keep the basic assumption of naive Bayes unchanged, we consider them independent). To complete our learning algorithm, we must still choose a method for estimating the probability terms. We change the method from [Mit97] to compute $P(f_k|c_{goal})$ as follows:

$$P(f_k|fit_{c_{goal}}) = \frac{n_k + 1}{n_p + |T|}$$

Where n_k is the sum of the number of times each key word in f_k occurs in S_p and n_p is the total number of distinct key words in S_p. We repeat the same process with appropriate sets to calculate $P(\neg fit_{c_{goal}})$ and $P(f_k|\neg fit_{c_{goal}})$. Further details can be found in [Mit97].

We use Algorithm 5 to classify subsequent example objects based on our learning from Algorithm 4. $\mathcal{H}_{NB}$ indicates if e is an instance of c_{goal}.

To illustrate the algorithm let us assume that the learner agent uses the features in Table 5.2 to represent the concept `Computer Science` as our c_{goal}. As we stated before because we use the same number of positive and negative examples, the probability values for $P(fit_{c_{goal}})$ and $P(\neg fit_{c_{goal}})$ are equally 0.5.

Now let us assume that $\mathcal{A}g_L$ wants to classify an example that has {`power`, `program`, `logic`} present in it. These key words enable features 2, 3, and 5. Based

Algorithm 5 Modified naive Bayes algorithms for classifying an example e_i

1. $E \leftarrow$ collect all features based on key words that occur in e and are in $\mathcal{K}_{\mathcal{A}g_L}$
2. $$a = [P(fit_{c_{goal}}) \prod_{f_k \in E} P(f_k | fit_{c_{goal}})]$$
3. $$b = [P(\neg fit_{c_{goal}}) \prod_{f_k \in E} P(f_k | \neg fit_{c_{goal}})]$$
4. $$\mathcal{H}_{NB}(e) = \begin{cases} fit_{c_{goal}} & \text{if } a > b \\ \neg fit_{c_{goal}} & \text{othewise} \end{cases}$$

on our algorithm we compute the following probabilities based on the features that have been enabled.

$$a = 0.5 \times 0.21 \times 0.33 \times 0.28 = 0.0097$$

$$b = 0.5 \times 0.28 \times 0.26 \times 0.37 = 0.0134$$

The result indicates that the example does not belong to the concept `Computer Science`.

5.5 Rocchio Concept Learner

This concept learner is based on the relevance feedback algorithm originally proposed by Rocchio [Roc71] and adapted for the vector space retrieval model [SB91]. Similar to naive Bayes we need to make some changes to adapt the standard Rocchio algorithm to work with our definition of features. Like other text categorization algorithms, Rocchio considers one key word as a feature and uses the TF/IDF (i.e. term frequency/inverse document frequency) key word weighting method to come up with a value for that feature. This weighting method is a statistical measure used

Table 5.2: Example features and their probabilities

	features	$P(f_k \mid fit_{c_{goal}})$	$P(f_k \mid \neg fit_{c_{goal}})$
1	$f_{\texttt{design,computer,system}}$	0.42	0.31
2	$f_{\texttt{logic,circuit}}$	0.28	0.37
3	$f_{\texttt{program,process,formal}}$	0.33	0.26
4	$f_{\texttt{data,model,theory}}$	0.47	0.39
5	$f_{\texttt{power,control}}$	0.21	0.28

to evaluate how important a key word is to an example object (i.e. a document in the information retrieval domain) in a collection or positive and negative examples. The importance increases proportionally to the number of times a key word appears in the document but is offset by the frequency of the word in the collection. The *term frequency* in the given example is simply the number of times a given key word appears in that example. In our context, the term frequency is the number of times that each key word appears in f_{descr}. The *inverse document frequency* is a measure of the general importance of the key word. It is the logarithm of the number of all examples divided by the number of examples containing the key word. For our modified version of Rocchio we first computed the TF/IDF for each key word that was in $\mathcal{K}_{\mathcal{A}g_L}$. Then to come up with the weight for the feature f_k we simply sum up the TF/IDF values for each key word which was presented in the f_k. To see how concept learner works, let us assume each example as a vector $\overrightarrow{e_i}$ which is created by all features f_k. Then learning is achieved by combining example vectors in to a prototype vector $\overrightarrow{w_{c_{goal}}}$. First, both the example vectors of the positive examples as well as those of the negative examples are summed up as follows:

$$\overrightarrow{w_{c_{goal}}} = \frac{1}{|pex|} \sum_{\overrightarrow{e_i} \in pex} \overrightarrow{e_i} - \beta \frac{1}{|nex|} \sum_{\overrightarrow{e_i} \in nex} \overrightarrow{e_i}$$

Rocchio requires that negative elements of the vector $\overrightarrow{w_{c_{goal}}}$ are set to 0. β is a parameter that adjusts the relative impact of positive and negative training examples. The optimal values are likely to be task-dependent and the performance of the resulting classifies strongly depends on a good choice of β.

To classify a new example $\overrightarrow{e}$, we compute the cosine between $\overrightarrow{w_{c_{goal}}}$ and $\overrightarrow{e}$. Using an appropriate threshold θ on the cosine leads to a binary classification rule which in fact is our interest.

$$\mathcal{H}_{Rocchio}(\overrightarrow{e}) = \begin{cases} fit_{c_{goal}} & \text{if } \cos(\overrightarrow{w_{c_{goal}}}, \overrightarrow{e}) > \theta \\ \neg fit_{c_{goal}} & \text{otherwise} \end{cases}$$

The Rocchio algorithm does not provide a means to compute a good threshold. One solution is to obtain a threshold via hold-out testing [SB91].

5.6 Agent Architecture

Figure 5.6 shows the components that are essential to instantiate our concepts. It should be noted that the general architecture of an agent capable of teaching and learning new concepts should have some other components. For instance agents use planners to plan their activities or preceptors to sense their environment. To keep our discussion focused we concentrate on the components that are of interest for our instantiation. Some of the essential components that we describe in this section

can be embedded in the general architecture of an agent to enable it to teach/learn concepts. Some others define some functionalities that can be added to some pre-existed component of an agent. For example agents must have a *decision making* component and the functionalities of our *decision maker* can be added to the decision making component of such a general architecture.

As stated before we want all agents to be able to learn and teach concepts. Therefore we designed our agents to play the role of a learner as well as a teacher by encapsulating the agents' behavior into seven different packages. These packages implement different actions that form the *Act* set of our agents. In this section we first describe these actions then throughout the description of the responsibilities of each components and their interactions we present the instantiation of *Sit* and *Dat* in the context of our implementation.

5.6.1 Actions for Agents

As discussed in Chapter 2 Section 2.1, *Act* is the set of possible actions that an agent can perform. Agents in this work, by performing an action or a sequence of actions may modify their understanding of the world or provide other agents with different types of helps. Also agents can act as a learner as well as a teacher depending on the different situation they are in. Therefore the actions we describe here are common to all agents. It should be mentioned that here we only discuss the actions that are essential to instantiate our concepts.

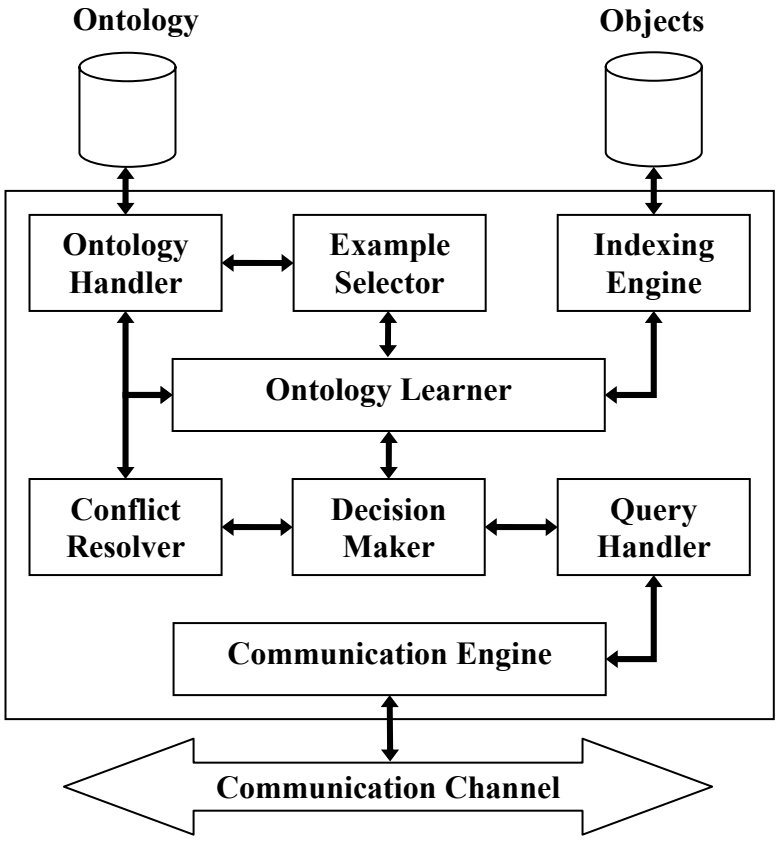

Figure 5.6: The essential components for the agent teaching/learning concepts

`QueryConcept`$(identifier,\{[f'_1 = V'_1],...,[f'_l = V'_l]\},O_{goal})$

The learner agent uses this action to query other agents about the concept that it is intended to learn. The learner agent could query teachers by the identifier of the goal concept, some features and their values which $\mathcal{A}g_L$ thinks are related to the goal concept or some example objects that again $\mathcal{A}g_L$ thinks are covered by the goal concept.

`FindConcept`(c_{goal})

The teacher agents use this action to find a set of candidate concepts that are satisfied by the query elements. The only parameter for this action is c_{goal} which is the uniform representation of the parameters of the query.

`SelectBestConcept`(c_{cand})

The teacher agents use this action to select the best concept from the set of candidate concepts that action `FindConcept` has made.

`SelectPosEx`(c_j)

The teacher agents use this action to *randomly* select a given number of positive examples for c_j. Each teacher agent $\mathcal{A}g_i$ stores a set of positive examples for each concepts c_j in its ontology. Due to the communication cost $\mathcal{A}g_i$ does send only a selected set of positive examples to the learner agent.

As a different realization, instead of selecting the positive example objects in a random manner for c_j, this action allows $\mathcal{A}g_i$ to select more distinctive examples based on the discriminative features that it considers for c_j.

`CreateNegEx`(c_j)

The teacher agents perform this action to produce a given number of negative examples for the concept c_j. This action uses taxonomical structure of the ontology to select negative examples from the sibling concepts of c_j.

AS an alternative realization the teacher agents utilize information provided by a specific relation(i.e. `is-similar-to`) to create more informative negative examples.

`ReplyQuery`($path(c_j), p_i, n_i$)

The teacher agents use this action to send the answer package back to $\mathcal{A}g_L$. The answer contains the sets of selected positive and negative examples, p_i, n_i respectively and the information on the path that leads in its taxonomy to c_j and the subtree below c_j's tree (i.e. $path(c_j)$).

`Learn`((p_1,n_1),...,(p_m,n_m)))

The Learner agent uses this action to learn c_{goal}. This action collects the sets of positive and negative examples and learns the concept in the feature-value form.

`AskClassify`(o_j)

In case of any conflicting examples o_j, using this action $\mathcal{A}g_L$ asks the teacher agents to classify the o_j regarding the c_{goal} and vote for, or against it.

`ClassifyEx`(o_j)

The teacher agents use this action to classify the requested example object, o_j regarding the concept that is being learned.

`ReplyClass(answer)`

The teacher agents use this action to reply the result of the classification of the requested example object to the learner.

`Integrate`($path(c_1)$,...,$path(c_m)$)

This action is utilized by the learner agent to integrate the proposed *paths*(i.e. subtrees) in its ontology. This action first merges the paths $path(c_1)$,...,$path(c_m)$ to one subtree and then integrates it in the agents ontology to come up with a new ontology.

5.6.2 Components and their Responsibilities

To interact with each other, agents use a *communication engine*. The communication engine enables agents to send messages directly to any other agent using TCP sockets. To avoid the complexity of designing a specific directory agent, agents are required to keep track of the current IP address of other agents. We assume that the agents keep a record of IP addresses of other agents using a setup file during the agent's initialization process. Agents use XML based messages to communicate. For each message we defined four different fields: type, sender, receiver and content. Table 5.3 shows a simple description of four message types that we have defined in our system related to the learning process. By processing these messages the agents change their situation and prepare to do different actions. The sender and receiver fields are IP addresses of the sender and receiver respectively. Based on the different message types the content could be another different XML document. For example when $\mathcal{A}g_L$ queries about a concept by sending the `ask_about_a_concept` message, the content consists of a concept identifier, a list of features or a list of objects as we described in Chapter 4 Section 4.3.

Our agent architecture is based on ontologies and their evolution. An ontology, the agent's understandings of the world, is part of the *Dat* of an agent. The agents search their ontology to find a concept regarding a query or manipulate it to integrate a new concept into it. For instance when an agent is confronted with a situation in which it is asked about a concept, it searches its ontology (i.e. *Dat*) and if it finds a concept it performs a set of actions in respect to that situation. It should be noted that some actions, such as `Integrate`, change the *Dat*.

Table 5.3: Description of message types

Message type	Description
`ask_about_a_concept`	*the learner agent uses this message to initiate a learning session*
`reply_query_result`	*using this message the teacher agents send the answer package regarding a concept to the learner*
`ask_to_classify`	*the learner agent uses this message to resolve possible conflicting examples*
`reply_classification_result`	*the teacher agent uses this message to vote about a conflicting example*

The *query handler* implements `QueryConcept`, `ReplyQuery` and `ReplyClass` actions and is responsible to properly format the outgoing queries and incoming answers. It also converts the incoming queries into a unified form and sends them to the agent's decision making engine. Generally this package coordinates agent's communication with other agents.

The *decision maker* is a very important component that makes most of the decisions throughout the life time of an agent and in fact it is responsible for coordinating the learning process through our general interaction scheme. This component monitors the environment to find out if there is any concept that an agent should learn. For instance an agent can observe the communication channel to see if there are any concepts that is its of interest. Prioritizing the shell concepts to be learned in the future is another responsibility of decision maker. This component also decides which agents should be contacted for different tasks. The decision maker keeps a record of the behaviors of other agents to decide which agent is knowledgeable about which concept.

The *ontology learner* implements the action `Learn`. As stated before, we developed two different learning algorithms to provide our system with different ap-

proaches to learning. Naive Bayes [Mit97] and Rocchio [SB91] have been implemented respectively. According to our definition of features we had to change the basic algorithms of these learners to deal with the features that are not just one key word. We will discuss them in Section 5.3. This component also forwards any conflicting examples to the conflict resolver asking for conflict resolution. This *conflict resolver* which implements the action `AskClassify` starts a conflict resolution session.

As we stated in Chapter 4 Section 4.7.1 the concept learner $\mathcal{H}_{c_j}$ can classify an example object e_i regarding the concept c_j in the agent's ontology. Therefore this component is responsible for deciding about incoming queries regarding classification of an object which again, is forwarded to the conflict resolver. In order to decide about the conflicting objects that come to the agent to be voted, the concept learner initiates the action `ClassifyEx`.

The *ontology handler* is developed around Jena [jen] which is a Java framework for building Semantic Web applications (see Section 5.2). This component implements the actions`FindConcept`, `SelectBestConcept` and `Integrate` actions. By cooperating with the *example selector*, the ontology handler helps the concept learner to select positive and negative examples to answer the incoming queries or feed the interpretation functions. The example selector implements every action that the agent utilizes to select or create examples or improve their quality.

Chapter 6

Experiments

We have now laid out our methodology, and considered the practical issues involved in its implementation as well as the implementation of our proof-of-concept prototype. The discussion in this chapter focuses on the experiments we have conducted with this prototype. These experiments illustrate all aspect of our approach to multi-agent learning of concepts. Also, they evaluate the efficiency of different realizations of some actions that we have defined in our methodology.

To illustrate and evaluate our approach, first we set up the learner agent to learn some concepts, namely: `Computer Science`, `Mathematics`, `Greek`, `Linguistics`, `Japanese`, `Chinese`, `Chemistry`, `German` and `Physics`. As we mentioned in Chapter 5 each of these concepts are representing a unit or program in the taxonomy of a university. Therefore by learning a concept, the learner agent can provide a university with suggestions for how a unit concerned with that concept should be characterized. We build our illustration and evaluation around these nine concepts and in the following sections we present some results either for each of them individually or report some average results for a group of them. Also it should be noted that for each of the experiments that we have done, our agents used their full ontologies even if we report only some on parts of them.

In Section 6.1 we illustrate the formation of a new concept in the learner agent. We will show how a concept forms in $\mathcal{A}g_L$ with regard to its key words. This experiment demonstrates how different agents influence the formation of a new concept in

the learner agent.

In Section 6.2 we assess the performance of the learner agent regarding a newly learned concept. We train $\mathcal{A}g_L$ with different percentages of examples of a learned concept to see how it classifies example objects in the set of all objects in the world. We also compare the performance of the learner with the teachers and in Section 6.3 we compare the performance of different concept learners.

In Section 6.4 we compare the performance of the different realizations of selecting positive and negative examples. We will show how these different methods influence the performance of $\mathcal{A}g_L$.

We have proposed three different conflict resolution strategies. In Section 6.5 we demonstrate the behavior of the learner agent when it utilizes these strategies to learn a concept. Finally in Section 6.6 we present some examples of our proposed non-unanimous concepts and illustrate our approach to learning of such a concept using a case study.

6.1 Concept Formation

To provide a better picture of how our method works, first we take a look at the formation of concepts in $\mathcal{A}g_L$ when it learns the concept from different numbers of agents. By this experiment, we aim to present a qualitative view of the formation of a new concept in the learner. This qualitative view reveals how a new concept forms in the form of the features that the learner agent recognizes. In this experiment we observe the formation of a new concept when different teacher agents get involved in the learning process. Therefore we can see the influence of a particular teacher agent

in the learning process. For this experiment we fixed the teacher agents to randomly select the positive examples and use the taxonomy to select the negative examples. Also, in this experiment the learner agent uses the majority voting to resolve the conflicts and when there are two teacher agents just one vote puts an example in the majority side.

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

6.1.1 Learning of Concept "Greek"

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 by 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 in Chapter 4, using an identifier does not make much sense here). Let us further assume that the relevant concepts in C_{base} are C_{base} = {`University`} and the relevant features in $\mathcal{F}_{base}$ are created using the key set:

$$\mathcal{K}_{base} = \{\texttt{class, course, program, literature, modern, attic, classic, culture, graduate, seminar, grammar, drama, greek, prose}\}.$$

Apparently, 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

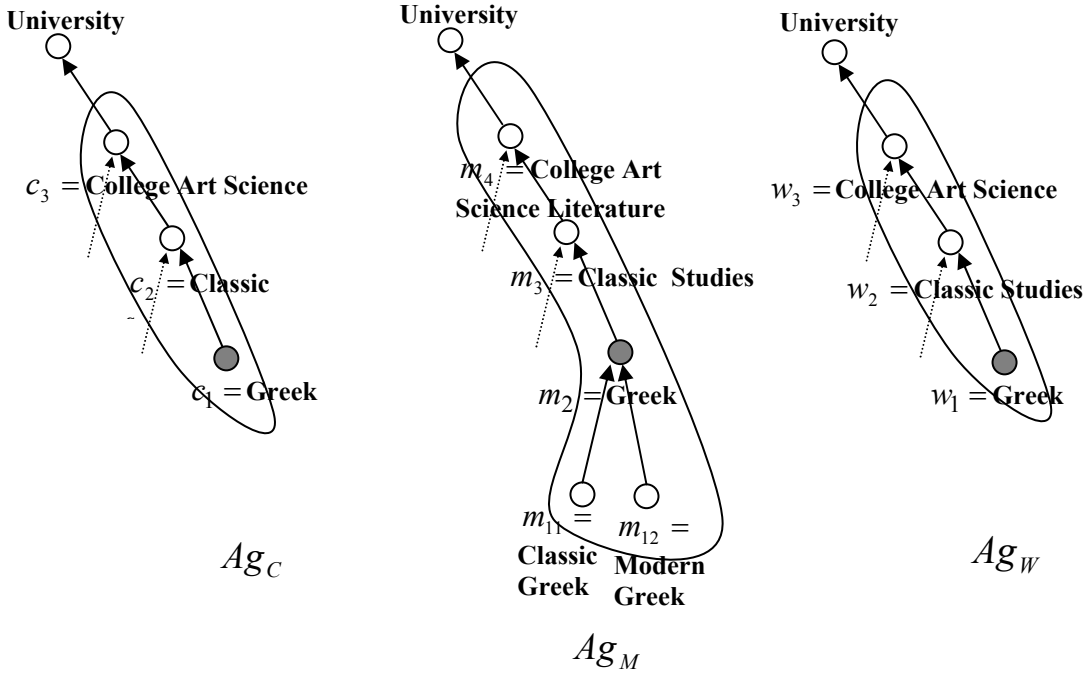

Figure 6.1: Relevant taxonomy paths of teachers for `Greek`

agents. In our example, the relevant key sets for the features used in the agents' ontologies are:

$$\mathcal{K}_C = \{\texttt{prose}\},$$

$$\mathcal{K}_W = \{\texttt{tragedy, orator, antique}\} \text{ and}$$

$$\mathcal{K}_M = \{\texttt{ modern, epic, classic, odyssey, ancient, aristotelian}\}.$$

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. We used our technique from Chapter 5 Section 5.1.3 to come up with these relevant key words. To make things interesting, we give $\mathcal{A}g_L$ as key set:

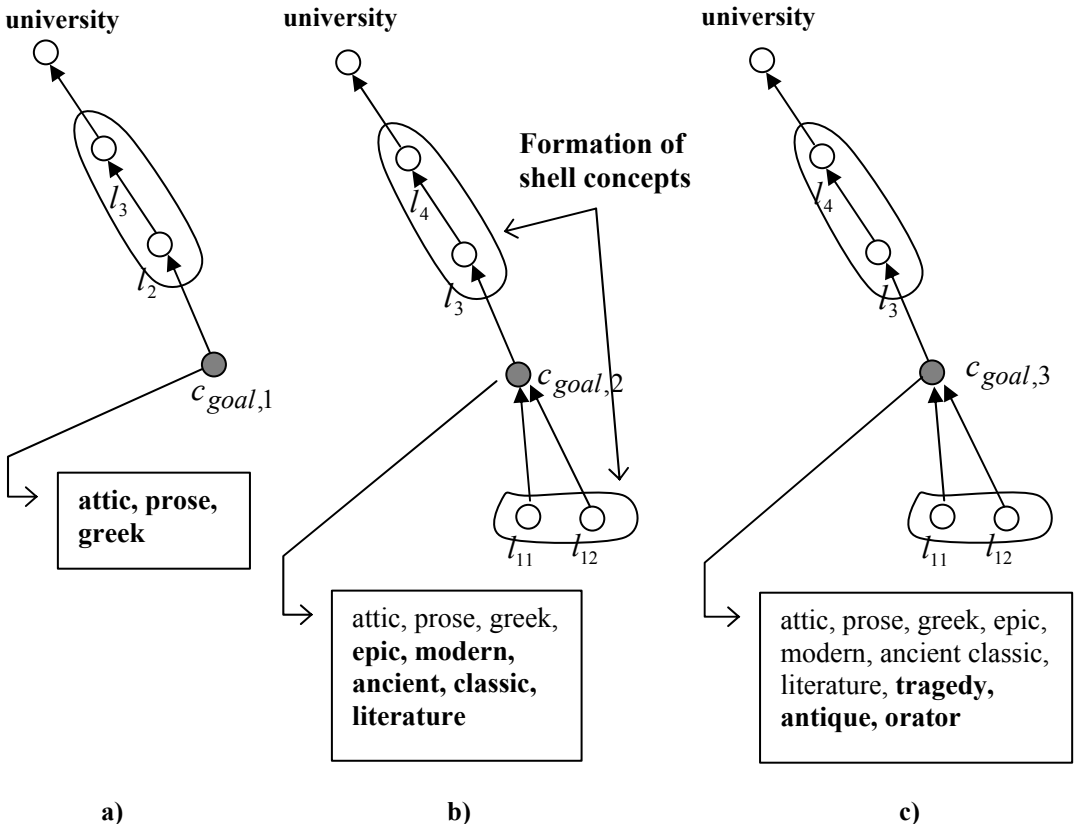

Figure 6.2: Formation of the learned concept `Greek` regarding key words when different agents participate in teaching. a) $\mathcal{A}g_C$ as teacher b) $\mathcal{A}g_C$ and $\mathcal{A}g_M$ are teachers c) $\mathcal{A}g_C$, $\mathcal{A}g_M$ and $\mathcal{A}g_W$ are teachers

$$\mathcal{K}_L = \{\texttt{epic, antique, ancient, tragedy, orator}\}.$$

Figure 6.1 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

Table 6.1: Usage of key words in features

key word	$\mathcal{A}g_C$	$\mathcal{A}g_M$	$\mathcal{A}g_W$	$\mathcal{A}g_L(C)$	$\mathcal{A}g_L(C,M)$	$\mathcal{A}g_L(C,M,W)$
`literature`	6	19	6	6	12	15
`greek`	8	55	9	8	26	33
`attic`	3	8	12	3	11	22
`prose`	1	0	0	1	1	1
`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

by c and multiplies it by 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 Chapter 5 Section 5.3. 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 6.1 shows, (a) to (c) refer to the concepts learned according to Figure 6.2 (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 word 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 6.2 shows the new taxonomy paths created by pre-structuring. 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. Although there are too many features identifying them to represent them completely here, we show some of the keywords making features. The bold words represent the keywords that teachers add to the learner keywords.

6.1.2 Learning of Concept "`Computer Science`"

To provide a better understanding of how different viewpoints of agents can affect the formation of c_{goal}, we conducted another experiment in which the learner is assumed to provide some suggestions about how a program concerned with `Computer Science` should be characterized. Unlike `Greek`, the teacher agents have not so

much overlap characterizing `Computer Science`. For instance, $\mathcal{A}g_M$ organizes it as an engineering discipline and as a joint program with electrical engineering. $\mathcal{A}g_W$ also considers `Computer Science` as an engineering discipline but independent from electrical engineering and as a joint program with computer engineering. In $\mathcal{A}g_C$ `Computer Science` is a pure science program in the science faculty.

$\mathcal{A}g_L$ starts the learning process by submitting a query consisting of {`computer, science, program, system`} as key words. We keep C_{base} the same as Section 6.1.1 and assume that a part of relevant features in $\mathcal{F}_{base}$ are created using the key set:

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

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

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

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

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

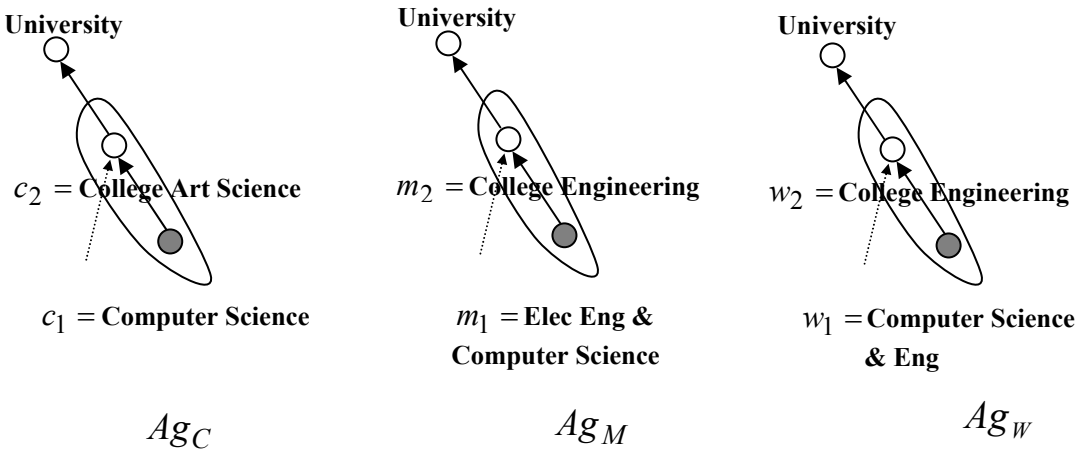

Figure 6.3: Relevant taxonomy paths of teachers for `Computer Science`

We also keep the measures used by the system and their weights unchanged from the previous section. Figure 6.3 shows the paths in the taxonomies of the teachers that our system selected as the best concepts for the query. Table 6.2 again provides an idea on how the teachers can affect what $\mathcal{A}g_L$ learns regarding the key word set accumulation which leads to the formation of c_{goal}.

The process of learning of `Computer Science` demonstrates how having agents with different viewpoints can influence key word set formation in $\mathcal{A}g_L$. As table 6.2 shows when $\mathcal{A}g_L$ learns from three agents it drastically decreases the number of key words from $\mathcal{A}g_M$ in the final description of c_{goal}. To make this impact clear let's take a closer look at Table 6.2. {`signal, circuit, power, plasma`} are four key words from $\mathcal{A}g_M$ which usually are not frequent in computer science course descriptions. In fact, these key words are frequent in electrical engineering course descriptions and their appearance in the key word set of $\mathcal{A}g_M$ shows the mixed nature of the concept for this agent (i.e. electrical engineering and computer science). As column

Table 6.2: Usage of key words in features for `Computer Science`

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

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

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

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

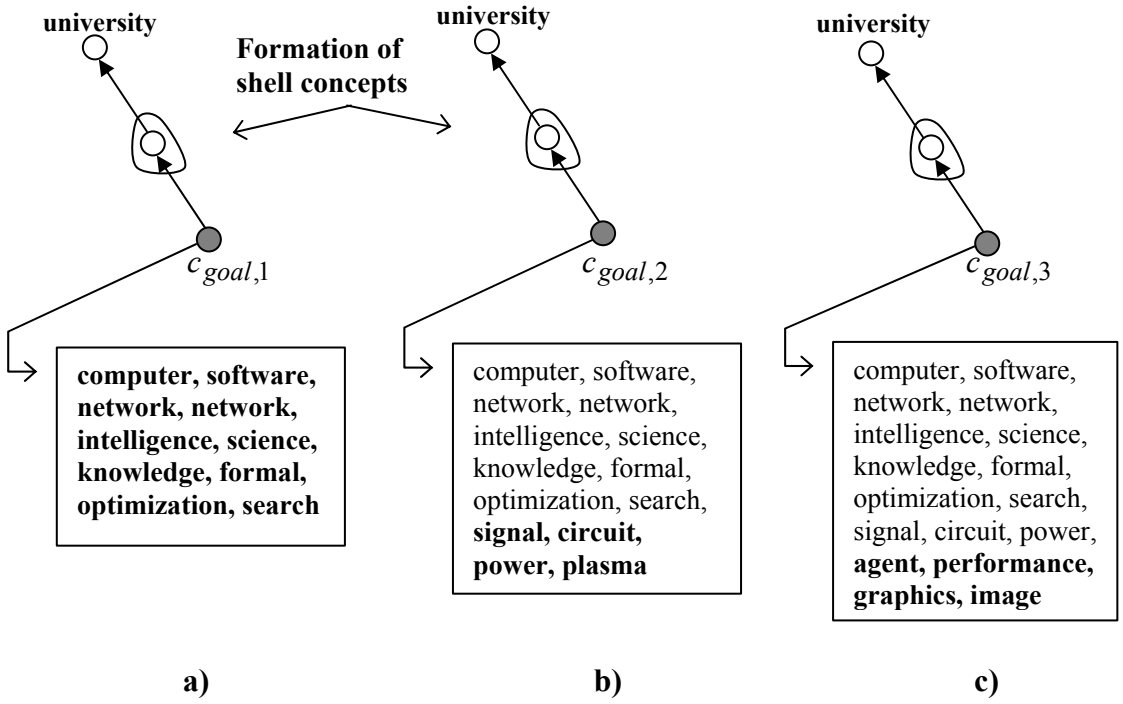

Figure 6.4: Formation of `Computer Science` regarding key words when different agents participate in teaching. a) $\mathcal{A}g_C$ as teacher b) $\mathcal{A}g_C$ and $\mathcal{A}g_M$ are teachers c) $\mathcal{A}g_C$, $\mathcal{A}g_M$ and $\mathcal{A}g_W$ are teachers

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

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

and with all agents as teachers we have:

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

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

Figure 6.4 shows the new taxonomy paths created by pre-structuring and the formation of key words for the concept `Computer Science`. Although the paths from different agents seem smooth and easy to integrate they are not similar. As stated before, $\mathcal{A}g_M$ and $\mathcal{A}g_W$ consider `Computer Science` as a sub-concept of engineering while $\mathcal{A}g_C$ considers it as a sub-concept of science. Our integration algorithm produced a shell concept with characteristics of `Engineering` as the superconcept of `Computer Science`.

6.2 Concepts in Action

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

In this experiment we fixed the teacher agents to select the positive and negative examples based on our improved version of example selection that we discussed in Chapter 4 Section 4.6. Also we fixed the learner agent to use the majority voting scheme to resolve the conflicts.

As we discussed in Chapter 4 Section 4.7.2, according to the different conflict resolution mechanisms that $\mathcal{A}g_L$ chooses, it has different sets of examples to learn from. These different sets shown in Figure 4.5 using different boundaries, affect the quality of the learned concept. By accepting the union boundary, the learner

Table 6.3: Truly classified examples for concepts `Mathematics`, `Computer Science` and `Greek`

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

accepts the positive examples which the group of agents are non-unanimous about, naturally this set has a maximum number of positive examples compared with other boundaries. The majority area has the positive examples that the majority of agents have agreed upon. While the number of positive examples in this area is less than the union area, it is more than the number of examples in the intersection area. These different sets of positive examples directly affect the quality of the learned concept and consequently the performance of the learner agent. Nevertheless, it is not guaranteed that the performance of the concept with the union boundary is better than the intersection boundary. A common scenario is that due to the fact that different teachers have different viewpoints, accepting every positive example (which makes the example set bigger) as a part of the learned concept (i.e union area) scatters the viewpoint of the learner and declines its performance. As we stated before, the most popular way of relying on a group decision is to follow the

majority of votes. Therefore, for this experiment, we have chosen the set of examples from the majority area to be learned by $\mathcal{A}g_L$.

Using the same set up as Section 6.1, we enabled $\mathcal{A}g_L$ to learn three different concepts `Greek`, `Computer Science` and `Mathematics`. We allowed $\mathcal{A}g_L$ to use the examples from the *majority boundary* as the representative examples of the concept. Then we trained the learner using different percentages of positive examples in this area (i.e. n% column). These percentages show us the classification accuracy of $\mathcal{A}g_L$ when the learner does not utilize the maximum number of available examples from the teachers. That is the case when due to the communication cost, the teachers could not send every possible example that they possess to the learner. Table 6.3 shows the classification results of the learner for the three different concepts. In fact this table shows the number of truly classified examples both for positive examples and negative examples in two separate columns. We should mention that, when we consider the area that a majority of agents agreed upon as the boundary for the concept in the learner, every other examples will be tested as the negative examples by $\mathcal{A}g_L$ in the testing process. This includes the positive examples that are in the union area and are *not* in the majority area. For instance and for concept `Mathematics`, the majority set has 501 positive examples and the other objects (i.e 19061-501=18560) could be considered as negative examples for it.

One interesting preliminary result, that in fact we expected, was the significant increase of truly classified examples when the concept is mostly unanimous. For example the programs `Mathematics` and `Greek` have more common courses than `Computer Science` among three different universities (which also is very true among other universities). As Table 6.3 shows the accuracy result for `Mathematics` is much

better than `Computer Science`. The last row of Table 6.3 shows the performance of the learner when it is trained by the whole set of examples it possess for each concept. For instance, the second and third columns show that $\mathcal{A}g_L$ classified 497 positive and 17324 negative examples out of 501 positive and 18560 negative examples. Therefore $\mathcal{A}g_L$ classified 93% ((497+17324)/(501+18560)) of objects correctly for `Mathematics` while this accuracy is 81% ((429+15104)/(505+18556)) for `Computer Science` and 90% ((170+16991)/(171+18890)) for `Greek`. There is a small "dip" in `Greek` when $\mathcal{A}g_L$ is trained by 70% of examples when the accuracy jumps to 91% and then comes back to 90%. Despite this "dip" the learner shows a consistent behavior classifying positive examples. We conclude that having agents with close viewpoints helps the learner to have a concrete understanding of a concept which naturally leads to a learner with better performance.

In this experiment we also compare the performance of the learner with the teacher agents. To compare the performance of $\mathcal{A}g_L$ with the teacher agents we had to compare the classification capability of $\mathcal{A}g_L$ with $\mathcal{A}g_W$, $\mathcal{A}g_C$, and $\mathcal{A}g_M$ respectively. As we discussed in Chapter 4, we assume that the teacher agents have learned the concepts in their ontology before they start to teach a concept to the learner. This learning has been achieved using some supervised inductive learning mechanisms and using the example objects that in each agent are associated with every concept in its ontology. Therefore we are supposed to simply compare the classification efficiency of $\mathcal{A}g_L$ with $\mathcal{A}g_W$, $\mathcal{A}g_C$, and $\mathcal{A}g_M$.

Nevertheless we can not guarantee that $\mathcal{A}g_L$ learns a concept using the same number of examples as each teacher agent and as we mentioned earlier the more examples are provided to the agent the better a classifier it can learn. This possibility

causes an unbalanced situation in which $\mathcal{A}g_L$ and other agents can not be compared. To overcome this problem, we have to prepare a fair situation in which the learner agent classification efficiency could be compared with each teacher agent. Therefore we selected a fragment of positive examples in $\mathcal{A}g_L$ which is quantitatively equal with the number of positive examples in each teacher agent to train $\mathcal{A}g_L$ with the same number of examples that the teacher agents utilized to learn the concept *before*.

Table 6.4: Comparison of the performance of $\mathcal{A}g_L$ and $\mathcal{A}g_M$

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

Table 6.5: Comparision of the performance of $\mathcal{A}g_L$ and $\mathcal{A}g_W$

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

Table 6.4, 6.5 and 6.6 show the results of comparisons of $\mathcal{A}g_L$ with $\mathcal{A}g_M$, $\mathcal{A}g_W$ and $\mathcal{A}g_C$ respectively. The second column in every table shows the number of truly classified examples, both positive and negative, out of 19061 test examples (i.e. objects in $\mathcal{U}$) by $\mathcal{A}g_L$. The third column shows the number of truly classified

Table 6.6: Comparision of the performance of $\mathcal{A}g_L$ and $\mathcal{A}g_C$

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

examples by the teacher agent and finally the forth row shows the percentage of examples that $\mathcal{A}g_L$ has been trained with, to produce this result. For instance the first row of Table 6.4 shows that $\mathcal{A}g_L$ has truly classified 15859 examples out of 19061 when it is trained by 53% of the whole set of its positive examples for `Mathematics`. The third column indicates that 15886 example objects are truly classified by $\mathcal{A}g_M$. The last column indicates that the number of associated examples with concept `Mathematics` in $\mathcal{A}g_M$ is 53% of examples in $\mathcal{A}g_L$. A very interesting outcome of this experiment is that $\mathcal{A}g_L$ in the most cases has a better performance than the teachers regarding the learned concept.This emphasizes on the fact that $\mathcal{A}g_L$ learns the *compromise* concept and its learning reflects a mutual viewpoint of agents. Therefore it will perform better when it tests against the objects from the whole world.

As an example concept, if we look at the `Computer Science` we see that $\mathcal{A}g_L$ is doing better compared to the other agents. For instance its accuracy is 68% (12962/19061) when it is trained by 43% of the training examples(see table 6.4). Clearly this is a better performance than $\mathcal{A}g_M$ which has classified 57.7% (11009/19061) truly. These results also confirm our preliminary result which we discussed in Sec-

tion 6.1.2 regarding the formation of `Computer Science`. Here we see that $\mathcal{A}g_L$ classifies 10.3% better than $\mathcal{A}g_M$ while this margin is 3.9% for $\mathcal{A}g_W$ and 3.6% for $\mathcal{A}g_C$. We believe that this is because the viewpoint of the learner is closer to $\mathcal{A}g_W$ and $\mathcal{A}g_C$ and as we showed in Section 6.1.2 the compromise concept in $\mathcal{A}g_L$ does not have so much of the characteristics of `Computer Science` from $\mathcal{A}g_M$. Therefore $\mathcal{A}g_L$ is doing *much* better in classifying objects from $\mathcal{U}$.

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

6.3 Comparison of Concept Learners

To compare the performance of the two concept learners that we are using we used the same setup as in Section 6.2. We allowed the learner agent to create its features using only combinations of two or three words. The substitutivities are made in a random manner. For the Rocchio algorithm we set β to 0.25 based on the recommendation from [BSA94]. Other assumptions were kept unchanged from the previous section. Figure 6.5, 6.6 and 6.7 show the comparison of the performance of our naive Bayes and Rocchio learner for the same three concepts of the previous section. Clearly the naive Bayes concept learner outperforms the Rocchio in all three example concepts. Therefore throughout the experiments in this thesis we use the naive Bayes as the

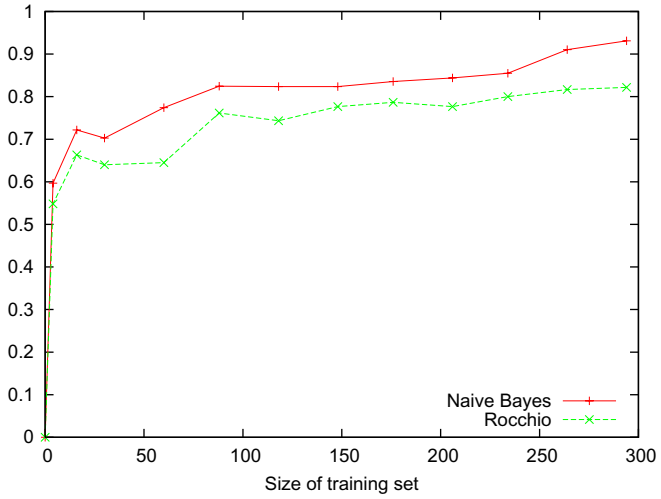

Figure 6.5: Comparison of concept learners for `Mathematics`

default concept learner for $\mathcal{A}g_L$.

6.4 Better Concepts Using Improved Example Selection

An important question to evaluate our method 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. One way to better utilize the communication channel is to send a set of examples that conveys more information to $\mathcal{A}g_L$. In this section we present the experiments that we have conducted to see how such improved examples influence the quality of the learned concept.

As stated in Chapter 4 Section 4.6, we proposed two alternative instantiations for positive and negative example selections. In this section we compare these instanti-

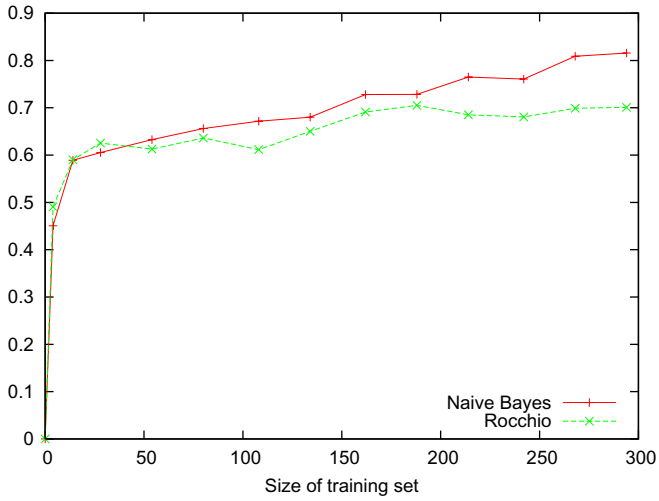

Figure 6.6: Comparison of concept learners for `Computer Science`

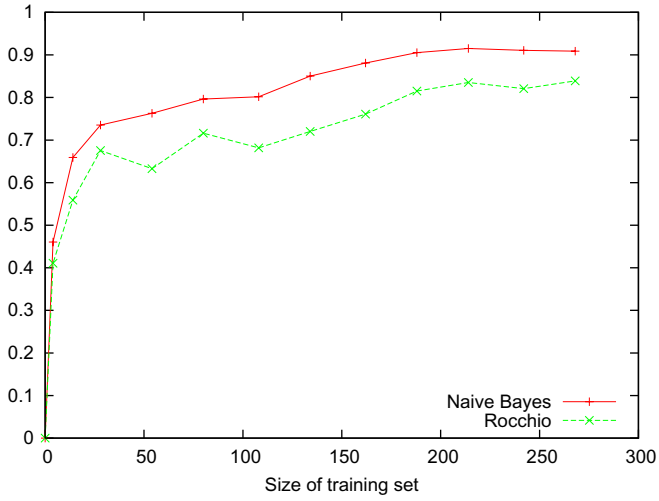

Figure 6.7: Comparison of concept learners for `Greek`

ations. We introduced the `is-similar-to` relation to select negative examples. We compared the performance of $\mathcal{A}g_L$ with the situation in which the negative examples are selected only using the taxonomy. To compare different instantiation of positive example selection, we fixed the negative example selection on selection using the `is-similar-to` relation. Then we compared the proposed methods and observed the superiority of positive examples selection using discriminative features over the random selection.

6.4.1 Using more Than the Taxonomy

Using the general setup of our multi-agent system, we conducted an experiment in which $\mathcal{A}g_L$ posed queries regarding a concept. We enabled the teachers to create and send two different sets of negative examples: the first set was created just by using `subconcept` and `superconcept` relations (i.e. the taxonomy) and for the second set in addition to the information from the taxonomy we used the information provided by the `is-similar-to` relation. We enabled agents to select positive examples according to our instantiation from Chapter 4 Section 4.6.2. Then we observed the quality of c_{goal} created by the learner to see how better the concept works in classifying the whole objects in $\mathcal{U}$. We repeated the experiment for nine different concepts and we present the result for two concepts individually and the average result for the whole nine concepts.

Each data point in Figure 6.8 represents the average value of 5 runs, since the selection of negative examples is performed randomly by the teachers out of the sets they consider.

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. Similar to our experiments of Section 6.2, every course classified under `Mathematics` by the *majority* of the three teachers was expected to be a positive example, all others a negative one. Figure 6.8 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`, and `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).

Using the setup of Section 6.1.1 we repeated the experiment for the concept `Greek`. $\mathcal{A}g_C$ has in its ontology that `Greek` is-similar-to `Classical Civilization`, `Classical Art and Archaeology` and `Romance Studies`, while $\mathcal{A}g_M$ has it similar to `Classical Archaeology` and `Near Eastern Studies` , and $\mathcal{A}g_W$ to classics and `Romance Language and Literature`. As Figure 6.9 shows, once more we see a significant improvement for the range between 50 and 100 training examples. There is a "dip" when we use 100 training examples which means the performance of

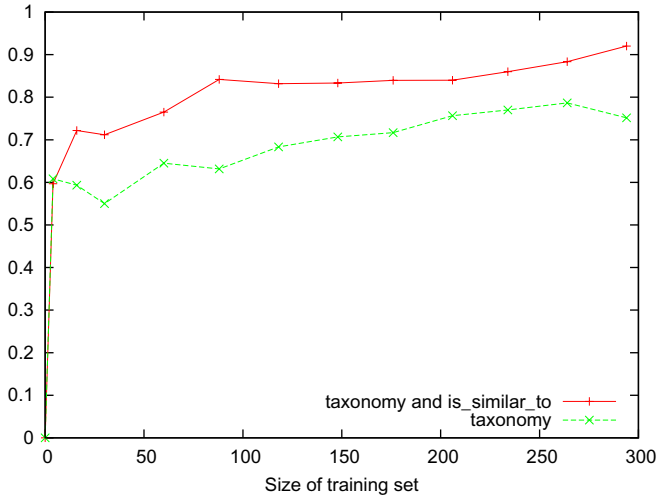

Figure 6.8: Comparison of negative example selection mechanisms for `Mathematics`

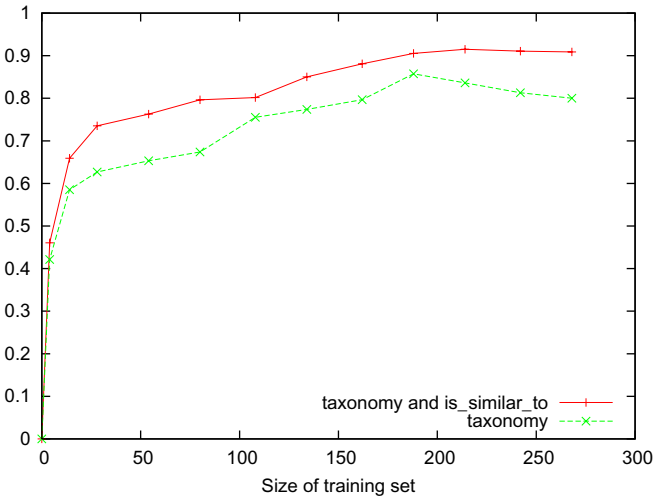

Figure 6.9: Comparison of negative example selection mechanisms for `Greek`

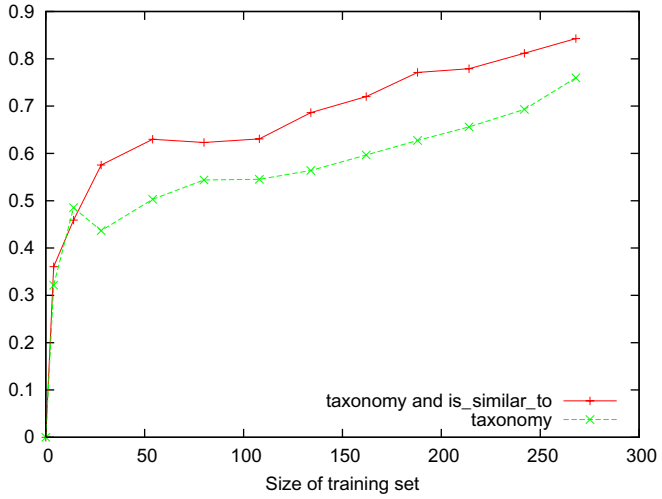

Figure 6.10: Comparison of negative example selection mechanisms for nine concepts

the learner has been affected by potential conflicts. Figure 6.10 shows the average result for nine concepts that we tested and demonstrates the superiority of selecting negative examples both from taxonomy and the `is-similar-to` relation.

Table 6.7, 6.8 and 6.9 shows the tabular representation of result for Mathematics and Greek and the average for nine concepts.

Table 6.7: Tabular representation of comparison of negative example selection mechanisms for `Mathematics`

n%	Taxonomy and `is-similar-to`	Taxonomy
10	0.702	0.552
20	0.773	0.629
30	0.824	0.625
40	0.820	0.630
50	0.823	0.667
60	0.835	0.679
70	0.844	0.691
80	0.855	0.715
90	0.910	0.797
100	0.931	0.756

Table 6.8: Tabular representation of comparison of negative example selection mechanisms for `Greek`

n%	Taxonomy and `is-similar-to`	Taxonomy
10	0.736	0.625
20	0.770	0.649
30	0.792	0.657
40	0.830	0.730
50	0.861	0.737
60	0.890	0.760
70	0.912	0.835
80	0.908	0.812
90	0.902	0.803
100	0.904	0.796

Table 6.9: Tabular representation of comparison of negative example selection mechanisms for nine concepts

n%	Taxonomy and `is-similar-to`	Taxonomy
10	0.572	0.425
20	0.618	0.519
30	0.614	0.543
40	0.648	0.551
50	0.686	0.560
60	0.729	0.609
70	0.740	0.615
80	0.768	0.632
90	0.806	0.680
100	0.839	0.756

6.4.2 Using more Distinctive Positive Examples

In this section, we evaluate the effectiveness of our method in selection of better positive examples. Based on our algorithms from Chapter 4 Section 4.6.2 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. Similar to the previous section we present the result of our experiments for `Greek` and `Mathematics` and the average result for nine concepts.

Table 6.10: 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 6.11: 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

To instantiate Algorithm 1 and 2 for our context we first used our similarity function (see Chapter 4 Section 4.6.1) to find k nearest hits for each document example e_i. The same process is accomplished to find k nearest miss documents.

To calculate feature weights, W, we needed to realize a diff function which was compatible with our context. As we discussed in Chapter 5 Section 5.1.3 we build our features in the form of key word combinations. In fact to instantiate the diff function we are interested in the boolean value of each feature. Therefore for a feature $f_{\mathtt{a,b,c}}$ we define

Table 6.12: 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

$$\text{diff}(f_{\texttt{a,b,c}}, e_i, e_j) = \begin{cases} 0; \text{value}(f_{\texttt{a,b,c}}, e_i) = \text{value}(f_{\texttt{a,b,c}}, e_j) = true \\ 0; \text{value}(f_{\texttt{a,b,c}}, e_i) = \text{value}(f_{\texttt{a,b,c}}, e_j) = false \\ 1; \text{otherwise} \end{cases}$$

For the first experiment, we assumed that the learning agent is supposed to learn the concept `Greek` (see Table 6.10). Based on the assumptions from Section 6.1.1, we enabled our agents to apply Algorithm 1 to come up with the core features representing the unique viewpoint of each agent. The subset of the core feature key set which is not common with $\mathcal{K}_{base}$, for each agents were as follows:

$$\mathcal{CF}_C = \{\texttt{democritus, religion, english, herodotus, medieval}\}$$
$$\mathcal{CF}_W = \{\texttt{tragedy, orator, antique, myths, archeology}\}$$
$$\mathcal{CF}_M = \{\texttt{modern, epic, classic, odyssey, ancient, aristotelian}\}.$$

Then we changed the agents to extract positive examples according to $\mathcal{CF}$ and using Algorithm 2. In the learner side and in order to evaluate the efficiency of our method in selecting distinctive positive examples, we first trained the learner with

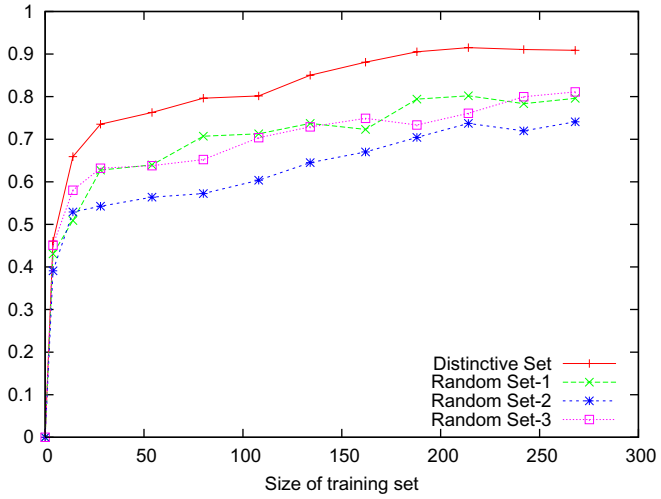

Figure 6.11: Graphical representation of comparison of positive example selection mechanisms for `Greek`

the set of distinctive positive examples and test it against the set of all example objects in $\mathcal{U}$. The second column in Table 6.10 shows the percentage of the truly classified examples by $\mathcal{A}g_L$ when it is trained by distinctive positive examples. To see how our method of selecting better positive examples improves the performance of the learner, we repeated the learning process three times and for each trial we asked teachers to select a random set of positive examples out of the associated positive examples.

As Table 6.10 shows, we see a significant improvement of 9.3 point difference (i.e. 90.4%-81.1%) in $\mathcal{A}g_L$ when it is trained by distinctive positive examples. This improvement emphasizes on the fact that by selecting better positive examples, the teacher agents help the learner to learn more accurate compromise concepts which consequently improves the future communication.

We repeated the experiment for the concept `Mathematics`. Again the result shows

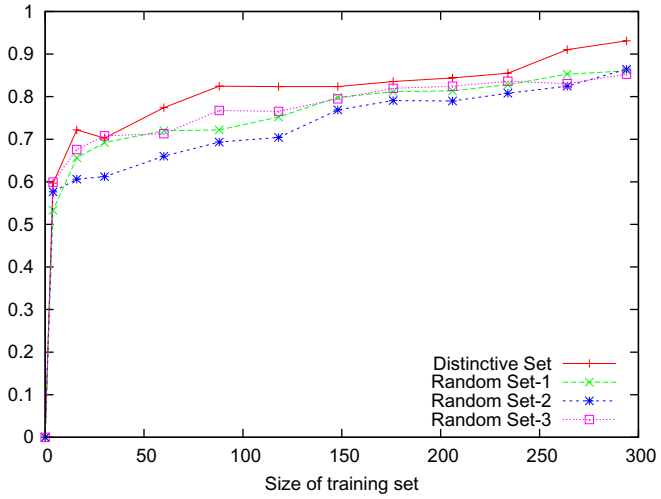

Figure 6.12: Graphical representation of comparison of positive example selection mechanisms for `Mathematics`

an improvement (i.e. 93.1%-86.0% = 7.1%) in performance of $\mathcal{A}g_L$ (see Table 6.11). In addition to `Mathematics` and `Greek` we repeated our experiment for seven other concepts. Table 6.12 shows the average result for nine concepts. The average result also confirms our preliminary results of `Greek` and `Mathematics`.

Figure 6.12, 6.11 and 6.13 show the graphical representation of comparison of different positive example selection mechanisms for `Mathematics` and `Greek` and average result for nine concept.

6.5 Evaluation of Conflict Resolution Strategies

As already stated, each possible conflict resolution method will come up with some concept for the learner's ontology. Based on which mechanism $\mathcal{A}g_L$ uses to learn a concept its behavior varies when it communicates with other agents. To observe

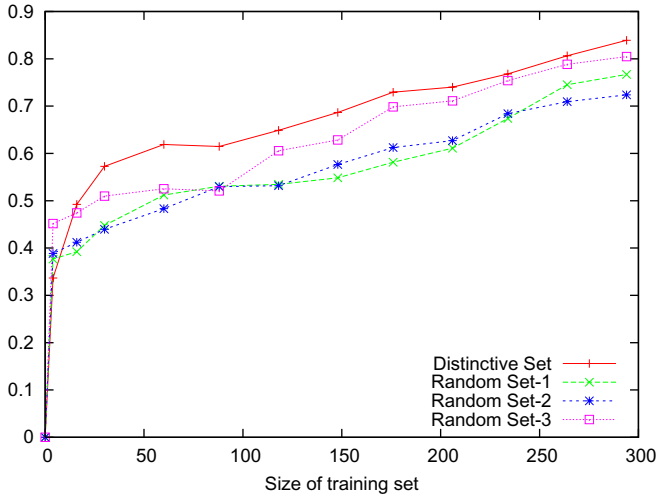

Figure 6.13: Graphical representation of comparison of positive example selection mechanisms for nine concepts

this behavior we allowed $\mathcal{A}g_L$ to learn a specific concept using the different conflict resolution strategies and then we monitored its classification accuracy when it is presented with examples from different boundaries of the non-unanimous concept that it has learned. For instance, this experiment shows how a learner behaves when it is trained by the set of "all agree" examples (i.e. which naturally has been produced by the "all agree" conflict resolution mechanism) how it would classify the examples from $c_{periphery}$ area. This will show us the behavior of the learner when it is learned a concept using a specific conflict resolution strategy and communicate with other agents which have different perceptions of the same concept.

More specifically to evaluate each strategy we allowed $\mathcal{A}g_L$ to be trained by the positive examples regarding that strategy for a specific concept. Then we tested it against c_{core}, c_{own}, and $c_{periphery}$ respectively to see how these strategies affect the learner's capability of classifying examples related to the learned concept.

Strategy	c_{core}	c_{own}	$c_{periphery}$
All agree	427/434	485/501	518/541
Majority agree	434/434	497/501	527/541
One agree	434/434	498/501	531/541

Table 6.13: Conflict resolution evaluation for `Mathematics`

Table 6.13 shows the evaluation results for `Mathematics`. The first row shows the truly classified positive examples out of the maximum number of positive examples in c_{core}, c_{own} and $c_{periphery}$ when it is trained by the examples that "all" agents agreed upon. The second row shows the same statistics when $\mathcal{A}g_L$ is trained by the examples that "majority" agents agreed upon and finally the third row shows the result when $\mathcal{A}g_L$ is trained by the examples that only "one" agent consider them as an example of concept.

As it is expected, the number of false negatives increases when $\mathcal{A}g_L$ classifies examples $c_{periphery}$. For instance when $\mathcal{A}g_L$ is trained by "all agree" examples it misclassified 16 positive examples from $c_{periphery}$. This confirms our expectation because $c_{periphery}$ contains *every possible* positive examples regarding `Mathematics` and $\mathcal{A}g_L$ which is trained by the "all agree" set might not accept all of these examples. As we move from $c_{periphery}$ toward c_{core} false negatives decreases and that is because the positive examples are in the core of the concepts and the teachers are unanimous about them. Concept `Mathematics` is a somehow unanimous concept and in most cases $\mathcal{A}g_M$, $\mathcal{A}g_C$ and $\mathcal{A}g_W$ have the same points of view to the examples. On the other hand, `Computer Science` is rather a scattered concept and as we showed in Section 6.1.2 the teacher agents have very different viewpoints to it. This is

Strategy	c_{core}	c_{own}	$c_{periphery}$
All agree	180/188	417/505	468/565
Majority agree	184/188	429/505	479/565
One agree	184/188	442/505	488/565

Table 6.14: Conflict resolution evaluation for `Computer Science`

nicely reflected in Table 6.14 where we present the result of the performance of $\mathcal{A}g_L$ respecting different conflict resolution strategies in the formation of `Computer Science`. Unlike `Mathematics` in which we don't see any false negatives in the c_{core} column in the second and third row, here we see 4 false negatives when $\mathcal{A}g_L$ is trained by "one agree" and tested against c_{core}. Similar to `Mathematics` we see the number of false negatives decreases when we broaden the border of accepted examples (i.e. moving from "all agree" to "majority agree" and "one agree"). For instance $\mathcal{A}g_L$ produced 97(565 - 468), 86(565 - 479), and 77(565 - 488) false negatives out of $c_{periphery}$ when it is trained by the different outcomes of the conflict resolution strategies. For the same reason that we presented for `Mathematics` the number of false negatives decreases when we move from $c_{periphery}$ toward c_{core}.

As it is expected, $\mathcal{A}g_L$ has a better performance classifying the examples from c_{core} and when the boundary stretched to c_{own} and $c_{periphery}$ the performance decreases. On the other hand the "one agree" strategy behaves better than "majority agree" and "all agree". It is also expected because of the increase in the number of the training examples in "one agree" strategy.

Unit	c_{core}	c_{own}	$c_{periphery}$
Mathematics	434	501	541
Computer Science	188	505	565
Linguistics	203	283	346

Table 6.15: Some positive example statistics

6.6 Non-unanimous Concepts: a Case Study

Borrowing information from our last experiment, in Table 6.15 we present some statistics on three concepts, namely the number of objects (i.e. courses) that are in the three concepts within a non-unanimous concept. As it is expected the difference between the core and the periphery for `Computer Science` is rather large, and even for `Linguistics` there is still quite a number of courses for which the three universities differ in their categorization. Nevertheless this is not true for the concept `Mathematics` which is a more unanimous concept. Table 6.16 presents some example objects (i.e. courses) belonging to `Mathematics` and `Linguistics` in different boundaries.

For instance regarding `Mathematics`, all universities agree that the courses with titles `Calculus I` and `Mathematical Logic` are in c_{core}. But, for example, a course with the name `Combinatorial Theory` was not classified into Mathematics by $\mathcal{A}g_C$, while `Model Theory` did not find the approval by $\mathcal{A}g_W$. On the other hand, `Pre Calculus` is just approved by $\mathcal{A}g_W$. This shows that being able to deal with differences in opinion between agents is something that agents need to be able to do.

Border	Linguistics	Mathematics
c_{core}	Topics in Linguistics Historical Linguistics Introduction to Linguistics Sound Patterns Linguistic Typology	Calculus I Mathematical Logic Algebra I Linear Algebra Mathematical Models
c_{own}	Introduction to Sociolinguistics Language and Mind Philosophy and Linguistics Ethnolinguistics Anthropological Linguistics	History of Mathematics Combinatorial Theory Model Theory Mathematics of Finance Mathematical Applications
$c_{periphery}$	Mathematical Linguistics Language and Gender Interactive Discourse Language North America Languages South Asia	Mathematical Biology Applied Mathematics III Pre Calculus Actuarial Science Topics in Modern Mathematics

Table 6.16: Examples of courses(objects) in c_{core}, c_{own}, and $c_{periphery}$ for Mathematics and Linguistics

6.6.1 Group Communication: An Example

As already stated, we were not so surprised to see a lot of objects in $c_{periphery}$ that are not in c_{core} for the concept `Computer Science`. This is due to the fact that already the taxonomies of the three agents were rather different with regard to where they put the `Computer Science` concept. For example, $\mathcal{A}g_M$ runs the `Computer Science` program as a program within the Engineering faculty as does $\mathcal{A}g_W$. In contrast to this, $\mathcal{A}g_C$ has the Department of Computer Science in its Science faculty. This also means that $\mathcal{A}g_L$ favors for its c_{own} (for `Computer Science`) the more engineering view, but, in contrast with what we see from time to time with real human agents, it is aware of the potential for misunderstandings due to using non-unanimous concepts.

Consider the following communication problem: $\mathcal{A}g_L$ as representant of a new university that learned their structure from $\mathcal{A}g_M$, $\mathcal{A}g_W$, and $\mathcal{A}g_C$ is giving a talk on its university in front of agents from many North-American universities. One of the points of the talk is to tell the other agents that at $\mathcal{A}g_L$'s university, all courses in the `Computer Science` program are taught by real professors (no instructors).

Table 6.17 shows some example objects belonging to `Computer Science` in different boundaries. Due to learning from the three teachers (that we assume to represent the extremes with regard to opinions on what is part of a `Computer Science` program, simply because they are the places that are providing the data), $\mathcal{A}g_L$ has `Computer Science` as a non-unanimous concept. For example, in its c_{core} for `Computer Science` it has courses with names like `Computer Programming I`, `Design and Analysis of Algorithms II`, `Computer Networks`, or `Computer Architecture` that have course descriptions that all teacher agents classify into `Computer Science`.

Some of the courses that only one of the teachers voted for as belonging to `Computer Science` have the names `Reliable Computing Systems`, `Computational Molecular Biology`, `Computers and Society`, or `Computational Tools for Finance`.

Border	`Computer Science`

c_{core}	`Computer Programming I`
	`Design and Analysis of Algorithms II`
	`Computer Science Research Seminar`
	`Introduction to Artificial Intelligence`
	`Computer Networks`
	`Introduction to Computer Organization`
	`Computer Architecture`
	`Foundations of Computer Science`
	`Interactive Computer Graphics`

c_{own}	`Applied Logic`
	`Theory of Computing`
	`Computer System Performance`
	`Computer Game Design and Development`
	`Parallel Computing`
	`Computational Geometry`
	`Introduction to Formal Models in Computer Science`

$c_{periphery}$	`Computational Molecular Biology`
	`Computational Tools and Methods for Finance`
	`Computers and Society`
	`Intelligent Transportation Systems`
	`Introduction to Logic Design`
	`Reliable Computing Systems`

Table 6.17: Examples of courses(objects) in c_{core}, c_{own}, and $c_{periphery}$ for `Computer Science`

Among the 193 courses that are not in c_{core} but in c_{own} for `Computer Science` of $\mathcal{A}g_L$ are, for example, courses named `Theory of Computing`, `Introduction to`

`Formal Models`, `Applied Logic`, `Parallel Computing`, or `Computer Game Design and Development`. If we would set the parameter *exmax* of Chapter 4 Section 4.8.3 to 1, then $\mathcal{A}g_L$ would enhance its communication about no instructors in Computer Science by using something like:

In Computer Science including ***`Theory of Computing`****, we are not using any instructors.*

This is because we have the following values for the *cover* function: *cover*(`Theory of Computing`) = 5, *cover*(`Introduction to Formal Models`) = 4, *cover*(`Applied Logic`) = 3, *cover*(`Parallel Computing`) = 3, *cover*(`Computer Game Design and Development`) = 2, *cover*(`Computer System Performance`) = 2, and *cover*(`Computational Geometry`) = 2.

Naturally, *exmax* = 1 still leaves room for misunderstandings (in fact, without stating all 193 courses there is always room for misunderstandings), but for *exmax* = 3 we first add `Parallel Computing` to the communication about `Computer Science`. The *cover* values for `Introduction to Formal Models` and `Applied Logic` are greatly reduced in the next rounds, since they are using the same features as `Theory of Computing`. We then get the following communication:

In Computer Science including ***`Theory of Computing`*** *and* ***`Parallel Computing`*** *we are not using any instructors.*

In the last round the *cover* value for `Computer System Performance` is reduced due to the fact that it uses the same features as `Parallel Computing`. Therefore we add `Computer Game Design and Development` and `Computational Geometry` to

the communication about `Computer Science`. The communication should promote to the following:

In Computer Science including `Theory of Computing`, `Parallel Computing`, `Computer Game Design and Development` *and* `Computational Geometry` *we are not using any instructors.*

Chapter 7

Conclusion and Future work

This chapter draws conclusion on the research presented in this dissertation, and provides possible suggestions for future work. At the start of our research, learning of ontology concepts among a group of agents with diverse ontology were largely constrained by two factors. First, all of the approaches assumed that agents are committed to a common set of features and conceptualized the world by these features. This assumption at most, leaves agents diversity of conceptualization of world at the ontology level in which agents use the same set of features to represent the ontology concepts. Second, the collaboration among agents to learn a concept from each other is restricted to one to one collaboration. This model of collaboration simply eliminate the critical problem of conflict resolution which is non-evitable in group collaboration.

We set out to propose a methodology that enable researchers to extend their idea to have an agent that learns a concept from group of agents utilizing not necessarily the same set of features to conceptualize the world. To do this we uses techniques from a number of areas. In particular, we borrowed from machine learning, in which researchers have already developed a plenty of techniques for learning in different contexts.

Section 7.1 briefly summaries our methodology, the main contribution of this research and the prototype system that we developed as a proof of concept. Then in Section 7.3 we provide some suggestions for possible extensions of this work.

7.1 Summery

As already stated, we have developed a method that demonstrates how an agent can learn new concepts for its ontology with the help of several other agents. This assumes that not all agents have the same ontology. We additionally assumed that there are only some base features that are known and can be recognized by all agents and that there are only some base symbolic concepts that are known to all agents by name, their feature values for the base features and the objects that are covered by them. Outside of this common knowledge, individual agents may come with additional features they can recognize and additional concepts they know. Given this setting, agents will develop problems in working together, since the common grounds for communication are not always there. To come up with a solution for this problem, agents need to acquire the concepts outside of the set of base concepts that other agents have, at least those concepts that are needed to establish the necessary communication to work together on a given task.

We also assumed that the learner agent has an ontology and knows a set of specific set of features. Analogously, each teacher agent has an ontology, and also knows a specific set of features. For any concept known to the a teacher agent, this agent has in its data areas a set of positive examples that it can use to teach the concept to another agent. Part of the action set of the learner agent are actions that: query other agents regarding a concept, ask other agents to classify a concept, learn a concept from a set of positive and negative examples , and integrate a newly leaned concept in its ontology. Also part of the action set of the teacher agents are the actions that: search the ontology to find the best matching concept with the

query, create a set of negative example regarding a queried concept, reply to a query, and classify an example and send the result back to the learner.

We proposed a general interaction scheme in which we generalized the interaction among agents regarding the learning process as the following. After becoming aware that there is a concept that it needs to learn, The learner agent poses a query. The learner agent uses three parameters to have three different ways to identify to the teachers what the learner is interested in. The first one is the parameter identifier which allows the learner to refer to a concept name it observed from other agents, which means that identifier is a known concept for some agent(s). The second way is the combination of some features and their values which enable the learner to use a selection of features and their values that thinks are related to the goal concept. The third way is the set of some objects that the learner thinks are covered by the goal concept.

Each teacher agent then reacts to the learner's query by finding the best concept that covers the queried concept. Naturally, already each of the parameters can point to different concepts that a teacher agent knows of. In fact, if the learner provides several objects in the set of queried objects, they might be classified by the teacher into several of its concepts. So, the teacher first collects all the concepts that fulfill the query into a candidate set and then it has to evaluate all these concepts to determine the concept that is the best fit. To select the "best" candidate out of many candidate concepts, there are many different ways how an evaluation of the candidates can be performed. Each of the three query parts can contribute to a measure that defines what is "best", but how these contributions are combined can be realized differently.

While supplying the learner with more examples normally produces better results, in our case we have to take into account that the more objects are selected as positive and negative examples the more expensive the communication becomes and the more effort the learner 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 the learner by each agent is a parameter of our system and reflects the wishes of a user. So, in the next step, each teacher selects the given number of elements out of the set of positive examples, for the best candidate concept. There are many possible ways how this selection process can be done. We proposed two different realizations. The first one was random sampling of the set of positive examples. In the second method we established our selection based 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. After selecting positive examples, the teacher agents produce a given number of (good) negative examples for the best concept. Since every concept other than the best candidate concept (and its subconcepts) can be categorized as a counter concept, the number of objects associated with these concepts is often very high. This big volume of possible negative examples makes the selection of a subset of them a crucial task. The best negative examples are objects that "nearly" are in the set covered by the best candidate, 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 just randomly selecting out of all candidate objects.

To show this more accurately we used both taxonomy information (siblings of the best candidate concept) and a relation `is-similar-to` to select the concepts from which we randomly selected examples.

The last action which is performed by a teacher agent before the initiative goes back to the learner is to reply to the learner which sends the result back to the learner. part of the reply package is the path information that contains the path in the ontology of the teacher leading to the candidate concept and the taxonomy tree below the concept. Then the learner collects the answers from all teachers and then uses a concept learner to learn the goal concept from the combined set of examples. Naturally, the concept learner only uses features and their values from its own set of features. In case of conflicts between the teacher agents, the learner employs one of several methods to resolve these conflicts. For example, the learner agent goes back to the teacher agents and ask them to classify these examples according to the best candidate concept they used to produce their examples. The learner then treat the answers as votes and include all positive examples for which a majority of the teachers voted, while requiring the exclusion of all negative examples for which a majority voted. This produces some kind of compromise concept that might appeal to most of the teachers. As the final step of our scheme, the learner uses the learned goal concept and the collected paths from the other agents to construct a new ontology path leading to the goal concept within its ontology. The key information used in this integration with pre-structuring is the path information send by the teacher agents in their answers to the query. The 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. The result of this learning/teaching scheme is the description of goal concept in terms of the learner's feature set and an updated ontology.

We also extended our preliminary definition of a concept in an ontology that allows an agent to simultaneously communicate with a group of agents that might have different understandings of some concepts. We also provided a way to learn such non-unanimous concepts by using a method for learning concepts from a group of teachers. The general idea of non-unanimous concepts is to use the teachers to identify the core of a concept everyone agrees on and what else at least some of the teachers think belongs into the concept. The learning agent also decides what belongs to the concept for itself and whenever it needs to communicate with a group of other agents and needs to be precise it makes use of these three concept aspects by providing additional example objects for what might be misunderstood.

7.2 Contributions

In this section, we reconsider each of three areas of our contributions in light of what has been presented in this dissertation.

7.2.1 Agents with Diverse Set of Features

We set out to develop a methodology for agents teaching one other agent concept in which agents not only may not want to commit to an ontology *a priori* but also not use the same set of features to represent the ontology concepts. We assumed that there are only some base features that are known, respectively can be recognized by

all agents. Outside of this common set of features, individual agents may come with additional features they can recognize. Our proof-of-concept prototype confirmed that agents teaching concepts to one other is achievable when they are not using a common set of features and this preserve the essential ontological promiscuity of AI.

7.2.2 A Group of Agents Teaching One Agent

Early on in our dissertation we mentioned that, at first appearance, learning from a group of agents instead of a single agent only seems to add potential problems, namely the teachers might not agree on some aspects of a concept to learn so that it is up to the learning agent to decide on these aspects on its own. But being able to address a group of agents is a necessity of multi-agent communication. In our thesis we addressed the group collaboration to teach a new concept to an agent. Despite the arisen problems, we proposed a general interaction scheme to resolve the conflicts among agents and to come up with a new multi-boundary concept.

7.2.3 Non-unanimous Concepts

We proposed the novel idea of non-unanimous ontology concepts that allows us to express different "shades" of agreements on a particular concept based on what an agent learns from a group of teacher agents. Also we provided an environment to enable agents to learn such non-unanimous concepts that represent a whole spectrum of possible definitions for a concept. We defined three different boundaries for every non-unanimous concept. The core boundary is the area that all agents view are consistent about it and there is no conflict among agents regarding the objects in this area. The periphery boundary is the area that covers all teacher agents' viewpoints.

The learner agent itself then chooses a concept definition that encompasses the core and is itself encompassed by the periphery which we called it own boundary.

7.3 Future Directions

This section focuses on future directions for the framework that has been presented in this thesis. There are many opportunities for further research in this challenging area. The following are the suggested research area.

7.3.1 Learning Relations

As stated in Chapter 2, one really interesting part of ontologies are the relations 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 user-defined relations. A possible future direction will be the extension of our proposed methodology to enable agents to *learn relations*. Here there are two sub-points:

- *learning rule sets to allow an agent to learn relations from other agents:* The solution should address ideas on what rule learners might be useful (i.e. inductive learners), what changes to the scheme for concepts might be necessary, and how to address conflicts between teachers.

- *finding an analogue to non-unanimous concepts for relations:* Obviously agents can be non-unanimous about a relation. The solution would start with going back to the basics, namely what a relation is supposed to be and try to propose

a definition about what a core and periphery relation might be. The solution also might associate some probabilities to the relations.

7.3.2 Ontology Reorganization

A potential interesting research area will be the use of our method and its ability to deal with different feature sets to deal with the loss of the ability to perceive one or several features by rearranging (re-learning) the parts of the ontology that are unclear due to the feature loss. This is obviously not a multi-agent topic anymore, but is something that there is obviously potential for more exploration.

7.3.3 Applying Concepts

To see the strength of our methodology we suggest a research direction in which our general ideas are applied to a real product application. A potential application would be a distributed text based document management system that is partly managed by IBM Unstructured Information Management Architecture. UIMA is an open, industrial-strength, scalable and extensible platform for creating, integrating and deploying unstructured information management solutions from combinations of semantic analysis and search components. While UIMA can manage the local ontology for an agent our proposed methodology can help agents to evolve their ontologies collaboratively.

Bibliography

[AF06] Mohsen Afsharchi and Behrouz H. Far. Improving example selection for agents teaching ontology concepts. In *CIA*, volume 4149 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 2006.

[AFD06a] Mohsen Afsharchi, Behrouz H. Far, and Jörg Denzinger. Learning non-unanimous ontology concepts to communicate with groups of agents. In *Proceedings of The IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-06)*, pages 211–217, 2006.

[AFD06b] Mohsen Afsharchi, Behrouz H. Far, and Jörg Denzinger. Ontology-guided learning to improve communication between groups of agents. In *Proceedings of the Fifth International Conference on Autonomous Agents and Multi-agent Systems (AAMAS06)*, pages 923–930, 2006.

[AK97] Naveen Ashish and Craig A. Knoblock. Semi-automatic wrapper generation for internet information sources. In *Proceedings of International Conference on Cooperative Information Systems*, pages 160–169, 1997.

[BLHL01] Tim. Berners-Lee, James. Hendler, and Ora. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[BSA94] Chris Buckley, Gerard Salton, and James Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the 17th annual international conference on Research and development*

in information retrieval(SIGIR94), pages 292–300. Springer-Verlag New York, Inc., 1994.

[BSSD00] Dipyaman Banerjee, Sabyasachi Saha, Sandip Sen, and Prithviraj Dasgupta. Learning mutual trust. In *Working Notes of AGENTS-00 Workshop on Deception, Fraud and Trust in Agent Societies*, pages 9–14. 2000.

[Cha00] Hans Chalupsky. Ontomorph: A translation system for symbolic knowledge. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*, San Francisco, CA, 2000. Morgan Kaufmann.

[DE02] Jörg Denzinger and Sean Ennis. Being the new guy in an experienced team: enhancing training on the job. In *Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems(AAMAS02)*, pages 1246–1253. ACM, 2002.

[Dou04] Dejing Dou. *Ontology translation by ontology merging and automated reasoning*. PhD thesis, New Haven, CT, USA, 2004.

[FFMM94] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. KQML as an Agent Communication Language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA, 1994. ACM Press.

[GF92] Michael. R. Genesereth and Richard. E. Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-92-1, Com-

puter Science Department, Stanford University, Stanford, CA, USA, June 1992.

[GN87] Michael. R. Genesereth and Nils. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kauffman Publishers. Inc, Palo Alto, CA, 1987.

[GO94] Thomas R. Gruber and Gregory R. Olsen. An ontology for engineering mathematics. In *Proceedings of KR'94: Principles of Knowledge Representation and Reasoning*, pages 258–269. Morgan Kaufmann, San Francisco, California, 1994.

[Gru91] Thomas R. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In *Proceedings of the Second Iniernanonal Conference on Principles of Knowledge Representation and Reasoning*, pages 601–602, 1991.

[Gru93] Thomas R. Gruber. Translation approach to portable ontology specifications. *Knowledge Acquisition*, 2:l99–220, 1993.

[Gua97] Nicola. Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. *Lecture Notes in Computer Science*, 1299:139–170, 1997.

[Hov98] Eduard. Hovy. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In *Proceedings of the 1st. International Conference on Language Resourcesand Evaluation (LREC)*, pages 535–542, 1998.

[jen] Jena a semantic web framework for java . `http://jena.sourceforge.net/`. As seen on Sep 20, 2006.

[JvD06] Frank Dignum Rogier M. van Eijk John-Jules Ch. Meyer Jurriaan van Diggelen, Robbert-Jan Beun. ANEMONE: An effective minimal ontology negotiation environment. pages 899–906, 2006.

[KS99] Atanas. Kiryakov and Kiril Iv. Simov. Ontologically supported semantic matching. In *Proceedings of the Nordic Conference on Computational Linguistics(NoDaLiDa'99).*, 1999.

[KS02] Yanis Kalfoglou and Marco Schorlemmer. Information flow based ontology mapping. In *Proceedings of the 1st International Conference on Ontologies, Databases and Application of Semantics (ODBASE'02), Irvine, CA, USA*, pages 1132–1151, 2002.

[LASB04] Agapito Ledezma, Ricardo Aler, Araceli Sanchís, and Daniel Borrajo. Predicting opponent actions by observation. In *Proceedings of RoboCup04*, volume 3276 of *Lecture Notes in Computer Science*, pages 286–296. Springer, 2004.

[LG01] Martin S. Lacher and Georg Groh. Facilitating the exchange of explicit knowledge through ontology mappings. In *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference, Florida, USA*, pages 305–309. AAAI Press, 2001.

[MDHD02] Jayant Madhavan, Pedro Domingos, Alon Y. Halevy, and AnHai Doan.

Learning to map between ontologies on the semantic web. In *Proceedings of the World-Wide Web Conference (WWW-2002)*, pages 662–673, 2002.

[Mel00] Sergey Melnik. Declarative mediation in distributed systems. In *Proceedings of International Conference on Conceptual Modeling (ER'00)*, pages 66–79, 2000.

[Mit97] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[MW04] Prasenjit Mitra and Gio Wiederhold. An ontology-composition algebra. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 93–116. Springer, 2004.

[MWK00] Prasenjit Mitra, Gio Wiederhold, and Martin L. Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Proceedings of 7th International Conference on Extending Database Technology EDBT 2000*, volume 1777 of *Lecture Notes in Computer Science*, pages 86–100. Springer, 2000.

[Nil98] Nils. J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kauffman Publishers. Inc, 1998.

[NM00] Natalya Noy and Mark A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of Twelfth Conference on on Innovative Applications of Artificial Intelligence (AAAI/IAAI-2000)*, pages 450–455, 2000.

[NM03] Natalya F. Noy and Mark A. Musen. The prompt suite: interactive tools for ontology merging and mapping. *International Journal of Human and Computer Studies*, 59(6):983–1024, 2003.

[owl] Owl - web ontology language. `http://www.w3.org/TR/owl-features/`. As seen on Sep 20, 2006.

[PH96] Foster J. Provost and Daniel N. Hennessy. Scaling up: Distributed machine learning with cooperation. pages 74–79, 1996.

[PL05] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[PLL96] Nagendra M.V. Prasad, Victor Lesser, and Susan Lander. Learning organizational roles in a heterogeneous multiagent system. In *Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, pages 72–77, Stanford University, CA, 1996.

[pro] Protege an ontology editor and knowledge-base framework. `http://protege.stanford.edu/`. As seen on Sep 20, 2006.

[PS03] Fuchun Peng and Dale Schuurmans. Combining naive bayes and n-gram language models for text classification. In *Proceedings of The 25th European Conference on Information Retrieval Research (ECIR03)*, 2003.

[RiK03] Marko Robnik-ikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning*, 53(1-2):23–69, 2003.

[RK91] Elaine. Rich and Kevin. Knight. *Artificial Intelligence*. McGraw Hill, 1991.

[RN95] Stuart Russell and Peter Norvig. *Artificial Intelligence: a Modern Approach*. Prentice-Hall, 1995.

[Roc71] J.J. Rocchio. Relevance feedback in information retrieval. In Gerald Salton, editor, *The SMART Retrieval System Experiments in Automatic Document Processing*, International Handbooks on Information Systems, page Chapter 14. Prentice Hall, 1971.

[Sah96] Mehran Sahami. Learning limited dependence Bayesian classifiers. In *Proceedings of Second International Conference on Knowledge Discovery in Databases*, 1996.

[SB91] Gerard Salton and Chris Buckley. Automatic text structuring and retrieval: Experiments in automatic encyclopedia searching. In *Proceedings of the 14th Annual International Conference on Research and Development in Information Retrieval (SIGIR91)*, pages 21–30, 1991.

[Sen02] Sandip. Sen. Sharing a concept. In *the Working Notes of the AAAI-02 Spring Symposium on Collaborative Learning Agents (AAAI Tech Report SS-02-02)*, 2002.

[SGH04] Larry M. Stephens, Aurovinda K. Gangam, and Michael N. Huhns. Constructing consensus ontologies for the semantic web: A conceptual approach. *World Wide Web*, 7(4):421–442, 2004.

[She99] Amit. P. Sheth. Changing focus on interoperability in information systems: from system, syntax, structure to semantics. pages 5 – 30, 1999.

[SM01] Gerd Stumme and Alexander Maedche. FCA-MERGE: Bottom-Up merging of ontologies. In *Proceedings of the seventeenth International Conference on Artificial Intelligence (IJCAI-01)*, pages 225–234, San Francisco, CA, 2001. Morgan Kaufmann Publishers, Inc.

[SM02] Erhard. Rahm Sergey Melnik, Hector. Garcia-Molina. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th International Conference on Data Engineering (ICDE '02)*, pages 117–129, Washington, DC, USA, 2002. IEEE Computer Society.

[SPF02] Yun Peng Sushama Prasad and Tim Finin. A tool for mapping between two ontologies using explicit information. In *Proceedings of Workshop on Ontologies and Agent Systems, AAMAS02*, 2002.

[SRV00] Peter Stone, Patrick Riley, and Manuela M. Veloso. Defining and using ideal teammate and opponent agent models. In *Proceedings of Twelfth Conference on on Innovative Applications of Artificial Intelligence (AAAI/IAAI-2000)*, pages 1040–1045, 2000.

[Ste98] Luc. Steels. The origins of ontologies and communication conventions in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 1(2):169–194, 1998.

[Stu02] Gerd Stumme. Using ontologies and formal concept analysis for organizing business knowledge. In *Wissensmanagement mit Referenzmodellen - Konzepte fr die Anwendungssystem- und Organisationsgestaltung*, pages 163–174. 2002.

[Tan97] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative learning. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 487–494. Morgan Kaufmann, San Francisco, CA, USA, 1997.

[UIl] Illinois semantic integration archive.. `http://anhai.cs.uiuc.edu/archive/`. As seen on Sep 20, 2006.

[UMi] University of michigan academic units. `http://www.umich.edu/units.php`. As seen on Sep 20, 2006.

[VH01] Richard Vdovjak and Geert-Jan Houben. RDF-based architecture for semantic integration of heterogeneous information sources. In *Proceedings of the International Workshop on Information Integration on the Web*, pages 51–57, 2001.

[Wei99] Gerhard Weiß, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1999.

[WG02] Jun. Wang and Les. Gasser. Mutual online concept learning for multiple agents. In *Proceedings of the First Iniernanonal Conference on Autonomous Agents and Multi-Agent Systems (AAMAS02)*, pages 362–369, 2002.

[Wie97] Gio Wiederhold. Mediators in the architecture of future information systems. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 185–196. Morgan Kaufmann, San Francisco, CA, USA, 1997.

[Wil04] Andrew. B. Williams. Learning to share meaning in a multi-agent system. *Autonomous Agents and Multi-Agent Systems*, 8(2):165–193, 2004.

[Woo91] William. A. Woods. Understanding subsumption and taxonomy: A framework for progress. In J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 45–94. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[WPB03] Andrew Williams, Anand Padmanabhan, and Brian M. Blake. Local consensus ontologies for b2b-oriented service composition. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS03)*, pages 647–654, 2003.

[YP97] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of 14th International Conference on Machine Learning (ICML-97)*, pages 412–420, 1997.