In Proceedings of the Twentieth National Conference on Artificial Intelligence, AAAI-05, Pittsburgh, Pennsylvania, USA, July 9-13, 2005.

Rule Refinement by Domain Experts in Complex Knowledge Bases

Cristina Boicu, Gheorghe Tecuci, Mihai Boicu

Learning Agents Center, Department of Computer Science, MSN 4A5,
George Mason University, 4400 University Dr., Fairfax, VA 22030, Phone (703) 993-4669
{ccascava, tecuci, mboicu}@gmu.edu; http://lac.gmu.edu

Introduction

We research how subject matter experts can develop knowledge-based systems that incorporate their expertise. Our approach is to develop a learning and problem solving agent, called Disciple, that an expert can teach by explaining it how to solve specific problems, and by critiquing its attempts to solve new problems (Tecuci 1998). The knowledge base of the agent is structured into an object ontology that contains a hierarchical description of the objects and features from an application domain, and a set of task reduction rules expressed with these objects.

The Disciple approach has already been applied to develop knowledge-based agents for complex military tasks such as course of action critiquing and center of gravity analysis (Tecuci et al. 2002). In this paper we present a new integrated approach to support a domain expert in refining the rules from an agent's large knowledge base.

Rule Learning and Refinement

To illustrate the rule learning and refinement process consider how a domain expert may teach a Disciple agent how to help a student select a PhD advisor. The expert formulates an initial problem solving task, such as "Determine a PhD advisor for Tom Evan." Then, using the task-reduction paradigm, the expert successively reduces this task to simpler tasks, guided by questions and answers, as illustrated by the following task reduction step:

Task: Determine whether John Smith can be a PhD advisor for Tom Evan in Artificial Intelligence.

Question: *Is John Smith likely to stay on the faculty of George Mason University for the duration of Tom Evan's dissertation?*

Answer: *Yes, because John Smith has a tenured position.*

Subtask: Determine whether John Smith would be a good PhD advisor for Tom Evan in Artificial Intelligence.

From each such task reduction step, Disciple learns a general task reduction rule. The learned rules are used by Disciple in the task reduction process, and the critiques received from the expert guides their refinement. For instance, the rule learned from the above example is applied to "Determine whether Mark White can be a PhD advisor for Tom Evan in Artificial Intelligence." However, the corresponding reduction is rejected by the expert because "Mark White is likely to move to Stanford University." Consequently, Disciple adds an except-when condition to the rule which takes the form shown in Figure 1. As illustrated in Figure 1, the rules learned by Disciple have an IF-THEN part that preserves the expert's language from the example, a main applicability condition and, optionally, several except-when conditions (which should not be satisfied to apply the rule). The conditions are partially learned, containing plausible upper and lower bounds for the acceptable values of the rule's variables. Rule's representative examples are also kept to help in its further refinement (Tecuci et al. 2002).

IF: Determine whether ?O1 can be a PhD advisor for ?O2 in ?O3

Question: *Is ?O1 likely to stay on the faculty of ?O4 for the duration of ?O2 's dissertation?*
Answer: *Yes, because ?O1 has a ?O5*

THEN: Determine whether ?O1 would be a good PhD advisor for ?O2 in ?O3

MAIN CONDITION

?O1	*is PUB (PhD_advisor)*	*PLB (PhD_advisor)*
	has_as_employer ?O4	
	has_as_position ?O5	
?O2	*is PUB (person)*	*PLB (PhD_student)*
?O3	*is PUB (research_area)*	*PLB (Artificial_Intelligence)*
?O4	*is PUB (employer)*	*PLB (university)*
?O5	*is PUB (position)*	*PLB (tenured_position)*

EXCEPT WHEN CONDITION

?O1	*is PUB (person)*	*PLB (PhD_advisor)*
	is_likely_to_move_to ?O6	
?O6	*is PUB (employer)*	*PLB (university)*

Positive Example: (?O1=John_Smith ?O2=Tom_Evan ?O3=Artificial_Intelligence ?O4=George_Mason_University ?O5=tenured_position)
Negative Example: (?O1=Mark_White ?O2=Tom_Evan ?O3=Artificial_Intelligence ?O4=George_Mason_University ?O5=tenured_position ?O6=Stanford_University)

Figure 1: Partially learned plausible version space rule

An Integrated Approach to Rule Refinement

The rule learning and refinement process in Disciple is based on the observation that it is difficult for a subject matter expert to work directly with formal rules. It is much easier for the expert to analyze specific examples, to accept them as correct, or to reject them as incorrect, and to provide explanations of his decision. Therefore, when the agent solves some task, it shows the expert the entire reasoning tree. Because the tree is generally very large, we have developed methods that highlight those task reduction steps that need special attention. These are steps generated

by using highly incomplete rules, or the plausible upper bound conditions of partially incomplete rules (see *Discovering Partially Refined Rules* below).

After the expert selects a task reduction step (example) to analyze, the agent interacts with him to critique the example and then updates the rule accordingly. Next the agent guides the expert to analyze the other task reduction steps of the current reasoning tree that were generated by the same rule. This leads to the further refinement of the rule, while keeping the expert focused on a manageable task (see *Refinement of the Rule's Conditions* below).

Often the refinement of a rule requires the verification of the rule's previous examples which may not be natural or even possible to do at that time. We have developed a *Lazy Rule Refinement* method to handle such cases (see section below). In the following we will summarize some of the methods included in our integrated approach.

Discovering Partially Refined Rules. In expressing their knowledge, domain experts use common sense and omit details that are implicit in human communication. Therefore an agent will learn rules from partially explained examples. To alleviate this problem we have developed two complementary rule analysis methods that guide the expert to provide more complete explanations. One method performs a structural analysis of the rule, checking if the values of its output variables (i.e. the variables from the question, answer and THEN tasks, such as ?O5 in Figure 1) can be obtained from the values of the variables of the IF task. The other method determines if there are too many instances of the rule and which are the under-constrained variables. Detailed experimental results for these methods are described in (Boicu et al. 2005).

Refinement of the Rule's Conditions. As shown in Figure 1, a rule has a complex structure. The rule is applied when its main condition is satisfied and none of its except-when conditions are satisfied. When the expert rejects a reduction generated by a rule and provides explanations of why the reduction is wrong, the agent must refine the conditions of the rule. One strategy is to specialize the main condition to no longer cover that reduction. Another strategy is to generalize one of the except-when conditions to cover that reduction. Yet another strategy is to create a new except-when condition. If the types of explanations elicited from the expert do not allow the agent to choose between competing strategies, the agent uses a lazy refinement method, postponing the decision until more cases are analyzed by the expert.

Lazy Rule Refinement. After analyzing a task reduction step generated by the agent, the expert may decide to update it. If the expert adds an explanation to the task reduction step then the agent refines the corresponding rule with a generalization of that explanation. However, if the expert deletes an explanation, or changes one of the rule's tasks, the question, or the answer, then it is not clear whether the rule should be updated, or a new rule should be learned from the modified example. All depends on whether the modification makes sense for the previous examples from which the rule was learned. However, many of these examples may not be accessible in the context of the current problem and situation. To deal with such a case, we have developed a lazy rule refinement method which is briefly described in the following. The agent creates a new version of the rule corresponding to the modified example, but it also keeps the old, unchanged version, linked to this new version. In order to avoid the generation of very similar solutions by different versions of a rule, the agent generates first the solutions corresponding to the newest version. Solutions corresponding to the older versions are considered only if they are different from those generated by the more recent versions of the rule. If in a future refinement session the expert confirms an example generated by a previous version of the rule, then this becomes an independent rule and is removed from the linked list. On the other hand, if the examples generated by the previous versions are rejected, they will be used to specialize the conditions of these rule versions. When the conditions of these previous rule versions become empty, the rules are deleted from the knowledge base. This lazy refinement method allows the modification of a learned rule, or the learning of a closely related rule, without requiring the expert to perform an analysis of the rule's representative examples at the time of the modification. Instead, this analysis is postponed until the agent applies the rule in problem solving.

Experimentation. Preliminary versions of the methods discussed in this paper have been implemented in the Disciple-RKF system, and have been successfully evaluated in several knowledge acquisition experiments performed at the US Army War College in the context of the center of gravity analysis domain (Tecuci et al. 2004). New experiments are planned for Spring 2005.

Acknowledgements. This research was sponsored by several US Government agencies, including DARPA, AFRL, AFOSR, and US Army War College.

References

Boicu, C.; Tecuci, G.; and Boicu, M. 2005. Improving Agent Learning through Rule Analysis. To appear in *Proceedings of the 2005 International Conference on Artificial Intelligence, IC-AI'2005*, Los Alamitos, California: IEEE Computer Society.

Tecuci, G. 1998. *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies*. London, England: Academic Press.

Tecuci, G.; Boicu, M.; Marcu, D.; Stanescu, B.; Boicu, C.; and Comello, J. 2002. Training and Using Disciple Agents: A Case Study in the Military Center of Gravity Analysis Domain. *AI Magazine*. 23(4): 51–68.

Tecuci, G.; Boicu, M.; Marcu, D.; Stanescu, B.; Boicu, C.; and Barbulescu, M. 2004. Parallel Knowledge Base Development by Subject Matter Experts. In *Proc. of the 14th Int. Conference on Knowledge Engineering and Knowledge Management* (EKAW 2004). Springer-Verlag.