Improving Agent Learning through Rule Analysis

Cristina Boicu, Gheorghe Tecuci, Mihai Boicu

Learning Agents Center, Department of Computer Science, MSN 4A5,
George Mason University, 4400 University Dr., Fairfax, VA 22030-4444, USA
{ccascava, tecuci, mboicu}@gmu.edu; http://lac.gmu.edu

Abstract. *This paper addresses the problem of improving the process by which an agent learns problem solving rules from a subject matter expert. It presents two complementary rule analysis methods that discover when a rule was learned from incomplete explanations of an example, guiding the expert to provide additional explanations. One method performs a structural analysis of the learned rule, while the other method analyzes the possible rule instantiations. Both methods have been implemented in the Disciple-RKF learning agent and have been tested both in an automatic framework and during two knowledge acquisition experiments performed with subject matter experts at the US Army War College.*

Keywords: Knowledge acquisition, machine learning, rule-based expert systems

1 Introduction

Our research addresses the problem of using learning agent technology to facilitate the development of expert or knowledge-based agents that incorporate the subject matter expertise of a human expert. Many of these systems represent expert's knowledge using if-then rules. Usually the system is built by a knowledge engineer who attempts to understand how the human expert reasons and solves problems and then encodes the acquired expertise into the system's knowledge base. This is an iterative process where the reasoning rules are successively tested and additional knowledge is elicited from the expert to improve them. One of the main reasons this approach has not had the expected impact resides in the difficulty of the mediated knowledge acquisition process, where first the knowledge engineer acquires knowledge from the human expert, and then represents the acquired knowledge into the knowledge base of the system.

As an alternative to this approach we are developing a learning agent, called Disciple, that can be taught directly by the subject matter expert how to solve problems, with limited assistance from a knowledge engineer [1]. The expert is teaching the Disciple agent to solve problems in a way that resembles how the expert would teach a student or an apprentice. For instance, the expert defines a specific problem and helps the agent to understand each reasoning step toward the solution. From each such reasoning step Disciple learns a general problem solving rule.

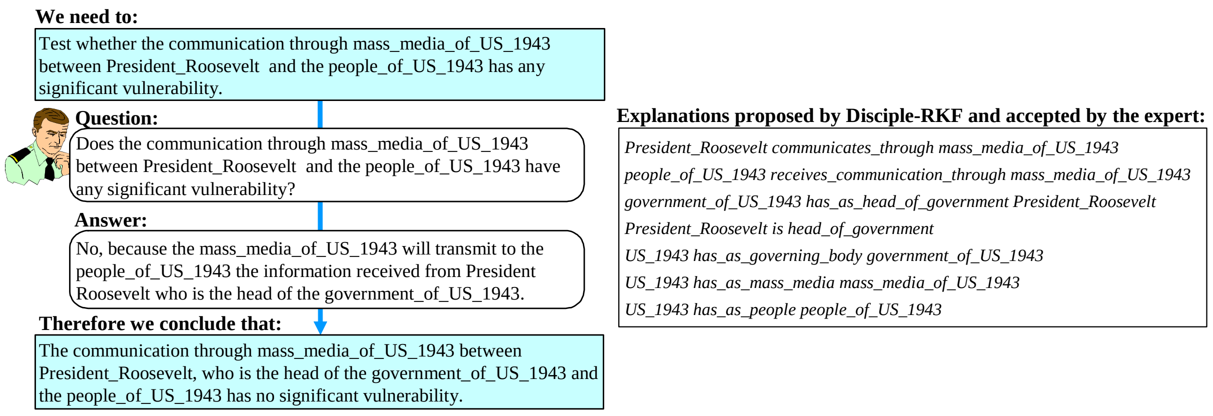
Whether the rules are defined by a knowledge engineer (as in the usual approach mentioned above), or learned by the agent (as in the Disciple approach), they are most likely incomplete and need to be refined [2]. One main reason for this situation is that, in expressing their knowledge, the subject matter experts use common sense and omit many details that are implicit in human communication.

Because the rule refinement process is a complex one, and its complexity increases with the incompleteness of the rules, one research issue is how to assure that the initial rules acquired from the expert (by the knowledge engineer or by the learning agent) are as complete as possible. This is precisely the problem addressed by this paper which presents two methods for analyzing the rules acquired from the expert, in order to detect whether they are incomplete.

The next section briefly presents the Disciple-RKF agent for military center of gravity determination, which is used in several courses at the US Army War College. It also summarizes the rule learning method of Disciple-RKF. Sections 3 and 4 present two rule analysis methods that are implemented in the rule analyzer module of Disciple-RKF. Each of these methods analyzes the rules learned by Disciple from the subject matter expert, detects when the rules are incomplete, and guides the elicitation of additional knowledge from the expert, to improve the rules. Section 5 presents several experimental evaluations of the methods that demonstrate their complementariness and usefulness. The paper is concluded with references to related and future research.

2 Disciple-RKF Learning Agent

Disciple-RKF uses task-reduction as its main problem solving paradigm. In this paradigm, a problem solving task is successively reduced to simpler tasks, the solutions of the simplest tasks are found, and these solutions are successively composed into the solution of the initial task. The knowledge base of the agent is structured into an object ontology that represents the objects from an application domain, and a set of task reduction rules and solution composition rules expressed with these objects. Disciple-RKF is a general problem solving and learning agent with no specific knowledge in its knowledge base. To develop an agent for a specific application domain, one needs to



We need to:

Test whether the communication through mass_media_of_US_1943 between President_Roosevelt and the people_of_US_1943 has any significant vulnerability.

Question:

Does the communication through mass_media_of_US_1943 between President_Roosevelt and the people_of_US_1943 have any significant vulnerability?

Answer:

No, because the mass_media_of_US_1943 will transmit to the people_of_US_1943 the information received from President Roosevelt who is the head of the government_of_US_1943.

Therefore we conclude that:

The communication through mass_media_of_US_1943 between President_Roosevelt, who is the head of the government_of_US_1943 and the people_of_US_1943 has no significant vulnerability.

Explanations proposed by Disciple-RKF and accepted by the expert:

President_Roosevelt communicates_through mass_media_of_US_1943
people_of_US_1943 receives_communication_through mass_media_of_US_1943
government_of_US_1943 has_as_head_of_government President_Roosevelt
President_Roosevelt is head_of_government
US_1943 has_as_governing_body government_of_US_1943
US_1943 has_as_mass_media mass_media_of_US_1943
US_1943 has_as_people people_of_US_1943

Figure 1: An example of a task reduction step indicated by the expert and its explanations

develop the knowledge base of Disciple-RKF. In essence, this process consists of developing the ontology for the specific application domain and of teaching Disciple how to solve problems.

In the last four years, successive versions of Disciple-RKF have been used to develop intelligent agents for center of gravity analysis that are used in several courses at the US Army War College [3]. The concept of center of gravity is fundamental to military strategy, denoting the primary source of moral or physical strength, power or resistance of a force [4]. Examples of potential centers of gravity of a state are a strong leader, its industrial capacity, or the will of its people. The most important objective of a force is to protect its own center of gravity, while attacking the center of gravity of its enemy. The knowledge base of Disciple-RKF for the center of gravity domain consists of an object ontology of 355 object concepts, 193 object feature definitions, 539 problem solving tasks, 368 task reduction rules, and 269 solution composition rules. The knowledge base also contains several hundred facts (i.e. triplets of the form "object feature value") for describing a specific war scenario. For instance, the "World War II Sicily 1943" scenario was described with 835 facts.

Each of the 368 task reduction rules from the knowledge base was learned by Disciple-RKF, as described in the following. The expert formulated an initial problem solving task, such as, "Identify and test the strategic center of gravity candidates for the Sicily_1943 scenario," and showed the agent each task reduction step required to solve it. For instance, one such task reduction step is shown in the left hand side of Figure 1.

The question associated with the task from the top of Figure 1 considers some relevant piece of information for solving that task. The answer identifies that piece of information and leads the expert to reduce this task to a simpler task (or, in other cases, to several simpler tasks). Thus, the question and its answer represent the expert's reason (or explanation) for performing that reduction. But they are in natural language and the expert needs to help

IF: Test whether the communication through ?O1 between ?O2 and the ?O3 has any significant vulnerability

Question: *Does the communication through ?O1 between ?O2 and the ?O3 have any significant vulnerability?*

Answer: *No, because the ?O1 will transmit to the ?O3 the information received from ?O2 who is the head of the ?O4*

THEN: The communication through ?O1 between ?O2, who is the head of the ?O4 and the ?O3 has no significant vulnerability

IF: Test whether the communication through mass media between a controlling leader and the people has any significant vulnerability
The mass media is ?O1
The controlling leader is ?O2
The people is ?O3

Explanation
?O2 communicates_through ?O1
?O3 receives_communication_through ?O1
?O4 has_as_head_of_government ?O2
?O2 is head_of_government
?O5 has_as_governing_body ?O4
?O5 has_as_mass_media ?O1
?O5 has_as_people ?O3

PUB Condition	**PLB Condition**
?O1 is mass_media	*?O1 is mass_media*
?O2 is head_of_government	*?O2 is head_of_government*
communicates_through ?O1	*communicates_through ?O1*
?O3 is people	*?O3 is people*
receives_communication_through ?O1	*receives_communication_through ?O1*
?O4 is governing_body	*?O4 is established_governing_body*
has_as_head_of_government ?O2	*has_as_head_of_government ?O2*
?O5 is force	*?O5 is single_state_force*
has_as_governing_body ?O4	*has_as_governing_body ?O4*
has_as_mass_media ?O1	*has_as_mass_media ?O1*
has_as_people ?O3	*has_as_people ?O3*

THEN: The communication through mass media between a controlling leader who is the head of the government and the people has no significant vulnerability
The mass media is ?O1
The controlling leader is ?O2
The people is ?O3
The government is ?O4

Figure 2: The rule learned from the example in Figure 1

Disciple-RKF to "understand" them. The agent will use analogical reasoning with previously learned rules, as well as general heuristics, to hypothesize the meaning of the question-answer pair. It will generate plausible explanation fragments (ordered by their plausibility), and the expert will select those that best express this meaning. The expert may also help the agent to propose the right explanation pieces by proving hints, such as pointing to a relevant object that should be part of the explanation.

Each explanation fragment proposed by the agent is a relationship (or a relationship chain) involving instances, concepts and constants from the task reduction step and from the knowledge base. For instance, the explanation fragments selected by the expert, from those proposed by the agent are shown in the right part of Figure 1. Based on the example and its explanations in Figure 1, and the knowledge from the knowledge base, the agent generated the IF-THEN rule shown in Figure 2. The rule contains an informal structure shown in the top part of the figure, and a formal structure, shown in the bottom part. The informal structure of the rule preserves the expert's natural language from the example, and is used in the agent-user communication. The formal structure of the rule is used in the internal reasoning of the agent, as explained in the following.

A specific task to be solved by the agent is matched with the task from the IF part of the rule. A successful matching results in values for the ?O1, ?O2, and ?O3 variables of the IF task, which are introduced in the condition of the rule. Then the agent matches the condition of the rule with the object ontology. If this matching is successful (i.e. the condition of the rule is satisfied in the current ontology), then it results in possible values for the other variables of the condition (i.e. ?O4 and ?O5). The values of these variables are then used to instantiate the resulting THEN task.

The above description referred to the condition of the rule. However, the rule in Figure 2 is only partially learned. Therefore, instead of a single applicability condition, it contains a plausible version space for this condition, consisting of a plausible lower bound (PLB) condition and a plausible upper bound (PUB) condition [5]. During rule refinement, these bounds will converge toward one another and toward the exact applicability condition of the rule [1]. The agent will use the partially learned rule in problem solving by using either its PLB condition (when applicable) or its PUB condition (when the PLB condition is not applicable). This will allow the agent to collaborate with the expert in solving new problems, and to refine the rule based on the correctness of the reductions generated by it (with correct reductions being used as additional positive examples of the rules, and incorrect ones as negative examples).

The length and the complexity of the rule refinement process depend of how well the rule was learned in the first place. Many times the rule is learned from an incomplete set of explanations of an example. As a consequence, the plausible version space of the rule will be larger, and the rule refinement process will be more complex. It would be very useful if the agent could determine whether the set of explanations is incomplete, and could guide the expert to provide additional explanations [6]. The next sections present two rule analysis methods that allow the agent to accomplish this task.

3 The Structural Analysis Method

The variables from the IF task of the rule are called input variables because they are instantiated when the rule is invoked. The other variables of the rule are called the output variables. The output variables are instantiated by the agent during the problem solving process with specific values that satisfy the rule's applicability condition, as described in the previous section. Notice in Figure 2 that the formal structure of the rule includes also generalizations of the explanation pieces of the example from which the rule was learned (see Figure 1). There are two types of explanations: IS explanations that specify the concept covering the instances of a variable (e.g. "?O1 is head_of_government"), and BETWEEN explanations that specify a path of features between two variables (e.g. "O4 has_as_head_of_government ?O2"). The IS explanations restrict the possible values of a variable to the instances of a concept. The BETWEEN explanations link the output variables to the input variables, and thus limit

Table 1: The Structural Analysis Method

```
Let R be a reasoning rule;
StructuralAnalysis(R)
Set_of_Variables linkedVariables ← IfTaskParameters(R)
Set_of_Variables outputVariables ← QuestionParameters(R) ∪
    AnswerParameters(R) ∪ ThenTasksParameters(R)
Set_of_Sets_of_Variables interlinkedVars ← ∅;
for each explanation E ∈ RuleExplanations(R) do
  for each fragment F ∈ ExplanationFragments(E) do
    fragmentSet ← fragmentParameters (F)
    for each set S ∈ interlinkedVars do
      if (FragmentParameters(F) ∩ S ≠ ∅) then
        fragmentSet ← fragmentSet ∪ S
        interlinkedVars ← interlinkedVars \ {S}
    if (fragmentSet ∩ linkedVariables ≠ ∅)
      then linkedVariables ← linkedVariables ∪ fragmentSet
      else
        if (CONSIDER_IS_EXPLANATION & fragment is an
              IS_explanation)
        then linkedVariables ← linkedVariables ∪ fragmentSet
        else interlinkedVars ← interlinkedVars ∪ {fragmentSet}
outputVariables ← outputVariables \ linkedVariables
return outputVariables
```

the possible values of the output variables. Moreover, such an explanation can contain several explanation pieces or fragments. For instance, the explanation "?O1 has_as_opposing_force ?O2 is_opposed_to ?O3" contains two fragments: "?O1 has_as_opposing_force ?O2" and "?O2 is_opposed_to ?O3."

In a well-formed rule, the output variables need to be linked through explanation pieces to some of the input variables of the rule and/or be constrained using IS explanations. The structural analysis method determines which of the rule's output variables are not constrained based on the existing explanations. This method is presented in Table 1.

The linked variables are initially the variables from the IF task. The output variables are initially all the other variables of the rule. The method maintains a set with all the sets of variables which are inter-linked based on the rule's explanations. For each fragment of each rule's explanation, the method computes the sets of variables linked based on this fragment. If such a set has common elements with the linked variables, then it is added to the linked variables of the rule.

The method uses also a flag which is true when IS explanations are used to consider a variable as being constrained. In such a case the variables from the IS explanations are also added to the linked variables.

The method returns all the output variables which are not linked through explanation fragments to the input variables of the rule or are not constrained by IS explanations. Based on this result, the agent can ask the expert to provide additional explanations related to the objects from the example (see Figure 1) that correspond to the unconstrained variables.

Table 2: The External Analysis Method

```
Let R be a reasoning rule;
inputValues - the instantiated values corresponding to the inputVariables;
kb - the knowledge base in which the IF task of the rule is instantiated;
threshold - the number of instantiations allowed for a rule;
ExternalAnalysis (R, inputValues, kb, threshold)
nbSolutions ← 0 //keeps the number of solutions
   //generated by the rule R in kb
varInstantiations ← ∅ //keeps a list with the instances in
   //the kb of each variable of the rule R
for each variable V of the rule R do
   varInstantiations[V] ← ∅
for each instantiation I of the rule R in kb having the given inputValues do
   for each variable V of the rule R do
      add each instance of variable V in kb to
           varInstantiations[V]
   nbSolutions++
if (nbSolutions ≥ threshold) then
   unboundedVariables ← ∅
   maxNbInstantiations ← 0
   for each variable V of the rule R do
      if (varInstantiations[V].size > maxNbInstantiations)
         maxNbInstantiations ← varInstantiations[V].size
         clear unboundedVariables and add the variable V
      else
       if (varInstantiations[V].size = maxNbInstantiations)
           add variable V to unboundedVariables
return unboundedVariables
```

4 The External Analysis Method

The external analysis method examines the rule in the context of problem solving. If the rule is generating too many solutions, then the rule may be under-constrained. We consider a threshold value for the number of acceptable rule instances. This threshold value depends on the application domain, and can be statistically determined.

The external analysis method is presented in Table 2. It computes the number of solutions generated by the rule in the knowledge base. If the number of solutions is greater than the threshold value, the method computes the variables with the highest number of instances in the kb. These are the variables which are not well constrained.

An extension of this method takes into account not only the instances generated for the current application of the rule (based on the given instantiated IF task), but also all the other applications of the rule in the current kb.

5 Experimental Results

The two rule analysis methods presented in the previous sections are complementary, one analyzing the internal structure of the rule, and the other analyzing the instances of the rule in a given knowledge base. Indeed, it is possible for a rule that has all the output variables constrained to still have too many instances. And vice-versa, a rule that does not generate a lot of solutions in a particular context can have an incomplete structure because of missing explanations. For this reason, both methods are implemented in the Rule Analyzer module of Disciple-RKF which combines their results, and is able to guide the expert to provide additional explanations related to certain objects of the example from which the rule was learned.

Several evaluations of the Rule Analyzer module have been performed, as described in the following. Usability evaluations have been performed by military experts, as part of the Military Applications of Artificial Intelligence course, taught in Spring 2002 and in Spring 2003, at the US Army War College [3]. The military experts, who did not have any prior knowledge engineering experience, were introduced to the Disciple approach. Then they taught Disciple how to determine the centers of gravity of the opposing forces from several war scenarios. During these experiments the Rule Analyzer module assisted the subject matter experts in the rule learning process by issuing warnings when rules were learned from incomplete explanations. Figures 3 and 4 present the assessments of the subject matter experts after using the Rule Analyzer module, in Spring 2002 and Spring 2003, respectively. There is a difference in the scale used in these two experiments. In 2002 we used a nominal scale, while in 2003 we adopted a numerical, equal interval scale from -2 to 2. Nevertheless, both results indicate that the subject matter experts found this module useful.

We have also evaluated the Rule Analyzer module in an automatic framework, in order to determine: 1) the percent of cases in which each method signaled incomplete rules to the user; 2) the percent of cases in which the methods made correct suggestions for refinement; 3) the significant factors for each method; and 4) the appropriate values for the parameters used in these rule analysis methods.

The automatic testing has implemented the following method:

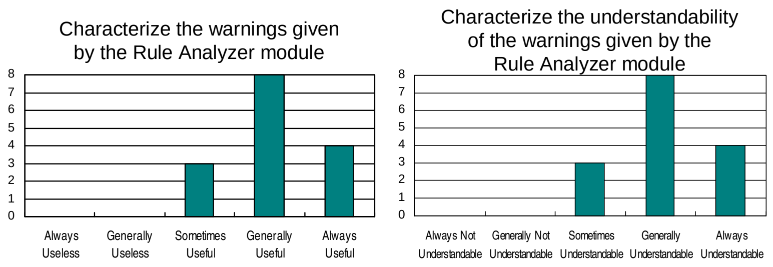


Figure 3: Results from MAAI 2002 experiment

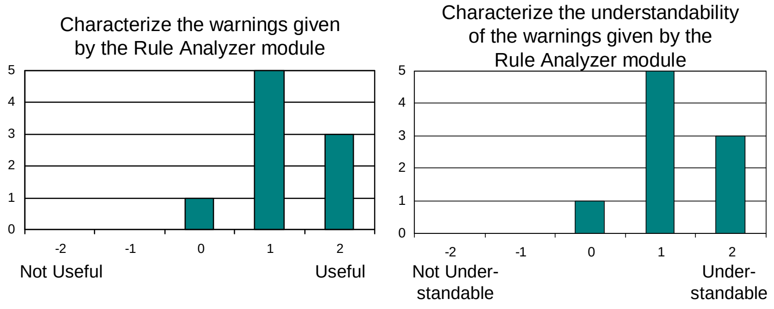


Figure 4: Results from MAAI 2003 experiment

```
• For each knowledge-base KB
    o For each applicable rule R
        1. Delete a percentage from the initial set of explanations (25%, 50%, 75%, and 100%, respectively),
           in all possible combinations
        2. For each combination relearn the rule R with the remaining explanations
        3. Call each method of the Rule Analyzer module and store the result
```

The experiments have been repeated for different values of the parameters used in the methods. We have used two-factor ANOVA (Analysis of Variance) tests at 95% confidence level, to determine the significant factors and the appropriate values for the parameters used in the methods.

In order to evaluate the Structural Analysis method we have considered two factors: the use of IS explanations and the percentage of deleted explanations from each rule (which indicates the completeness level of the rule). The goals were: 1) to determine whether to use the IS explanations, or to ignore them, when analyzing if the rule's variables are well constrained, and 2) to determine how the rule's completeness level influences the results of the method. We have computed the percentage of cases in which the method discovered problems in the rules, after the rules were modified in the automatic test by deleting a percentage of their explanations, and the results are shown in the top graph from Figure 5. For instance, when 25% explanations were deleted from the rule, the method discovered missing explanations in 20% cases using IS explanations, versus 35% discovered cases when IS explanations were ignored. We have also tested the precision of the method in both versions (with and without IS explanations), by computing the percent of cases in which the suggestions made by the method were correct (when the variables signaled by the method were indeed the variables with problems, i.e. they were contained in the explanations deleted from the rule), and also when the suggestions were incorrect. These results are presented in the bottom graph from Figure 5. One can notice that in almost all the cases in which the method identified a problem in the rule, it also made the correct suggestions for correction.

Then we have performed a two-factor ANOVA testing the use of IS explanations versus the percentage of deleted explanations from each rule, and we have obtained that both these factors are significant. *The unexpected result of this experiment is that the method identifies better the problems in the rules when IS explanations are not used.* This indicates that even if a variable may be constrained with an IS explanation, this may not be sufficient, and the variables may still need to be linked with other variables through explanations of type BETWEEN. The results also show that as more explanations are missing, the method identifies more problems in the rules, as expected.

For the evaluation of the External Analysis method we have considered two factors: the threshold value for the number of rule's applications and the percentage of deleted explanations from each rule. We have computed the percentage of cases in which the method discovered problems in the rule for various percentages of deleted explanations, using a threshold of 3 and 10 rule instances, respectively. For instance, when 25% explanations were deleted from the rule, the method discovered missing explanations in 21% cases using a threshold of 3 rule applications, versus 17% discovered cases for a threshold of 10 rule applications, as shown in the top graph of Figure

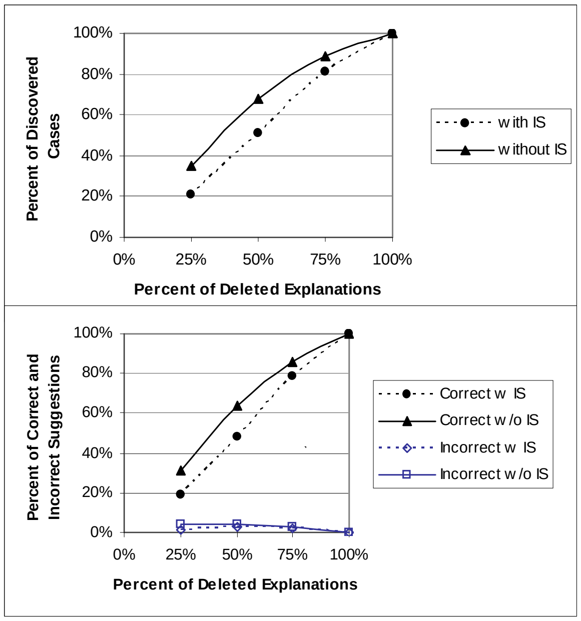


Figure 5: The Structural Analysis Method Results

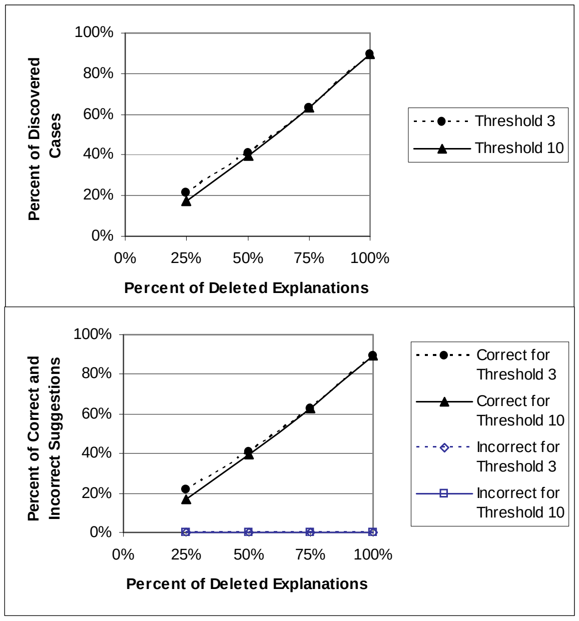


Figure 6: The External Analysis Method Results

6. We have also tested the precision of this method, by computing the percent of cases in which the method made correct and incorrect suggestions, as shown in the bottom graph of Figure 6. We have chosen two different threshold values, a smaller one, and a larger one, to see how the results differ. We have again noticed that when the method discovered problems in a rule it also made the correct suggestions.

We have also compared the two rule analysis methods, in order to decide whether it is good to use both of them in the Rule Analyzer module. For various percentages of deleted explanations from each rule, we have computed: 1) the percent of cases in which only the first method signaled problems in the rule (the line marked with "Structural Method Only" in Figure 7); 2) the percent of cases in which only the second method signaled problems in the rule (the line marked with "External Method Only" in Figure 7); and 3) the percent of cases in which both methods signaled problems in the rule (these cases are not included in the previous two and are marked by the line "Both Methods" in Figure 7). The top line marked with "Total" shows the global percentage of cases discovered by the Rule Analyzer module using both methods. We have performed this experiment on several rule samples, and then we have used two-factor ANOVA with replication, which indicated that both factors are significant. A graphic of the percentage of discovered cases obtained for one of the rule samples is shown in Figure 7. We can observe that a significant percentage of cases are identified only by one of the methods, proving that each method has an important contribution. Also, another aspect observed is that when the rule is almost learned (only 25% of explanation missing), the two methods discover different problems. This indicates that the Rule Analyzer performs better when both methods are used to identify problems in the rule.

The results of the automatic experiments confirmed the subjective evaluation of the subject matter experts. One interesting aspect is that the methods reported very few false positives, as shown in Figures 5 and 6 by the Incorrect lines. Therefore, the users are not annoyed by the suggestions made by the module.

6 Conclusions and Future Research

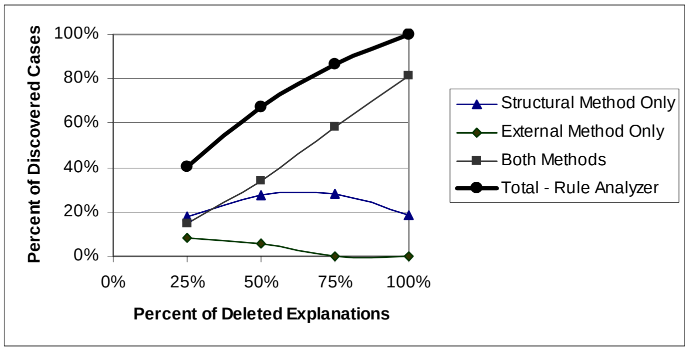


Figure 7: Comparative Analysis of the Methods

There are several tools that analyze a knowledge base to detect problems in the rules of a system [7, 8, 9]. For instance, the VALENS (VALid Engineering Support) tool for the validation and verification of the rules in a knowledge base can be used both during and after the construction of a knowledge base and focuses on the logical verification of knowledge-based systems [7]. The techniques used to verify a knowledge base are: meta-rules, an inference engine to verify hypotheses posed by meta-rules (proof-by-processing) and meta information (provided by the user). The tool creates a document in which all the detected invalid rules (combinations) are reported, and each fault is classified and explained. However, this tool requires the user to provide additional meta-information and meta-rules, which makes it less usable by a subject mater expert. The Rule Analyzer module presented in this paper is integrated naturally in the interaction between the system and the expert, signaling potential problems based on specific examples without confusing the expert with the general structure of the rule.

Pazzani and Brunk propose an approach which combines empirical learning and explanation-based learning to detect errors in rule-based expert systems and suggest revisions to the rules [8]. Their knowledge refinement tool, KR-FOCL, partially automates the task of identifying the rules responsible for errors. However, KR-FOCL does not explain to the expert why it recommends a specific revision, assuming the expert will realize that the revision will improve the system's performance. Additionally, the expert must approve the suggested revisions and assign the blame to a specific rule, after KR-FOCL has localized blame to a small number of rules. KR-FOCL was tested only on small examples with few errors. However, as the size of the rule base and the number of errors increase, it may be necessary to organize and order the possible revisions based on their effect on the accuracy of the system. Moreover, it would be useful to dynamically determine the impact of a revision on the knowledge base, because the user must run the system again in order to determine the impact of a modification.

We plan to extend the Rule Analyzer module by dynamically adjusting the rule applications threshold used in the External Analysis method, based on the current ontology structure, and by learning different threshold values for various classes of rules rather than for the entire KB. The process presented in this paper improves the agent learning, triggering the learning of more accurate rules. It also improves the agent's problem solving efficiency, by generating and testing fewer solutions.

Acknowledgments: This research was sponsored by several US Government agencies, including the Defense Advanced Research Projects Agency, the Air Force Research Laboratory, the Air Force Office of Scientific Research, and the US Army War College.

References

[1] Gheorghe Tecuci, Mihai Boicu, Dorin Marcu, Bogdan Stanescu, Cristina Boicu, Jerome Comello. 2002. Training and Using Disciple Agents: A Case Study in the Military Center of Gravity Analysis Domain. *AI Magazine* 23(4), pp. 51–68. AAAI Press, Menlo Park, California.

[2] Mihai Boicu. 2002. *Modeling and Learning with Incomplete Knowledge.* PhD Thesis in Information Technology, Learning Agents Laboratory, School of Information Technology and Engineering, George Mason University.

[3] Gheorghe Tecuci, Mihai Boicu, Dorin Marcu, Bogdan Stanescu, Cristina Boicu, Marcel Barbulescu. 2004. Parallel Knowledge Base Development by Subject Matter Experts. In *Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management* (EKAW 2004). Springer-Verlag.

[4] Joe Strange. 1996. Centers of Gravity & Critical Vulnerabilities: Building on the Clausewitzian Foundation So That We Can All Speak the Same Language. Quantico, VA: Marine Corps University.

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

[6] Cristina Boicu, Gheorghe Tecuci, Mihai Boicu. 2005. Rule Refinement by Domain Experts in Complex Knowledge Bases. To appear in *Proceedings of the Twentieth National Conference of Artificial Intelligence* (AAAI-2005). July 9-13, 2005, Pittsburgh, Pennsylvania.

[7] Silvie Spreeuwenberg, Rik Gerrits, Margherita Boekenoogen. 2000. VALENS: A Knowledge Based Tool to Validate and Verify an Aion Knowledge Base. In *Proceedings of the 14th European Conference on Artificial Intelligence*. IOS Press.

[8] Michael Pazzani and Clifford Brunk. 1991. Detecting and correcting errors in rule-based systems: an integration of empirical and explanation-based learning. *Knowledge Acquisition* (3), pp. 157-173.

[9] Dirk Ourston and Raymond J. Mooney. 1994. Theory Refinement Combining Analytical and Empirical Methods. *Artificial Intelligence* (66), pp. 311-344.