The Use of Simulated Experts in Evaluating Knowledge Acquisition

P. Compton, P. Preston, B. Kang
School of Computer Science and Engineering
University of New South Wales
Sydney 205, Australia
email compton@spectrum.cs.unsw.oz.au

Abstract

Evaluation of knowledge acquisition methods remains an important goal; however, evaluation of actual knowledge acquisition is difficult because of the unavailability of experts for adequately controlled studies. This paper proposes the use of simulated experts, i.e., other knowledge based systems as sources of expertise in assessing knowledge acquisition tools. A simulated expert is not as creative or wise as a human expert, but it readily allows for controlled experiments. This method has been used to assess a knowledge acquisition methodology, Ripple Down Rules at various levels of expertise and shows that redundancy is not a major problem with RDR.

Introduction

Evaluation of knowledge acquisition (KA) methods remains an important goal. Many KA methods have been proposed and many tools have been developed. However, the critical issue for any developer of knowledge based systems (KBS) is to select the best KA technique for the task in hand. This means that papers describing methods need to provide convincing evidence of the particular advantage of the method over other methods and clear identification of the problems and weaknesses of a method. Unless this clear evidence is provided it is very hard to be sure whether or not to believe the author, who with the best will in the world, is mainly concerned to highlight the advantages he or she believes are provided by the new method they are proposing. As an example, it seems there are still very few case studies of maintenance problems with KBS, e.g. (Bachant and McDermott 1984; Compton, Horn et al. 1989) . The problem for KA researchers is that they need to demonstrate results on actual KA from experts. Obviously it is expensive to use experts for other than real applications and they are not readily available for controlled studies. The closest to a controlled scientific evaluation of KA so far seems to be Shaw's study of different experts using KSSO (Shaw 1988) . However the aim of this study was to investigate variability in how experts provide knowledge rather than to evaluate a KA method. The study suggested that experts organised their knowledge of the same domain quite differently from each other and the same expert was likely to vary his or her knowledge organisation on repeat experiments.

Clearly any study using experts needs to take into account the variability between experts as well as the difficulty of repeat experiments on the same experts, whereby they become more expert at contributing to a KBS. These are standard problems in empirical science, but are major stumbling blocks in KA because of the cost and unavailability of experts. Experts by definition are people whose expertise is scarce and valuable.

One solution to this problem is to pick tasks for which many people have significant expertise so that experts are readily available. Little work appears to have been done on this approach and in our own experience it is very difficult to identify suitable tasks. Another approach is simply to report on how methods have been used on a wide range of real world systems. This is useful but less than ideal as it only applies to established methods not new research and it is very difficulty to

quantify and compare. New methods are difficult as one must first convince an organisation of the advantages of using a hitherto untested approach. It normally happens because the developer is part of the organisation, which hardly provides a good controlled environment. However, clearly some standards on how application work should be reported to make comparison possible is desirable.

The major attempts to evaluate KA to date are the Sisyphus projects (Linster 1992; Gaines and Musen 1994) . In these projects a sufficient paper specification of a problem has been provided so that a KBS solution could be implemented without further information being required. These studies have been very valuable because they have resulted in papers describing the development of a variety of solutions to the same problem. The basis for comparison has been informal but very interesting, even resulting in joint papers where authors contrast their methods (Fensel and Poeck 1994) However, these papers necessarily have no information on actual KA. All the relevant knowledge was already in a paper specification. What the studies are concerned with is identifying and perhaps building problem solving methods suitable for the described problem, and developing an appropriate domain model and perhaps even a KA tool suited to the problem. They are not concerned with the further process of actually acquiring the knowledge to go into the knowledge base. For a system fully specified on paper in a single small document, acquisition of the knowledge is trivial once the problem solving method and domain model have been developed. However, this does not seem to be the case with real KBS projects.

This paper proposes the notion of using another KBS as a simulated expert from which knowledge can be acquired. The KA method or tool is used to acquire knowledge from the simulated expert and to build a new KBS which should have the same competence as the KBS from which the simulated expert is derived. Instead of asking a human expert what the reasons are for reaching a particular conclusion, etc., one asks the simulated expert whose source of expertise is a previously built expert system for the domain. The obvious advantage of such an approach is that endless repeat experiments are possible and the experimenter has complete control over all the variables.

A weakness of this approach is that a data model is already given or will be very easily derived, whereas with a human expert this may be more difficult. We mean here the data model required for communicating with the user and/or acquiring the data about a particular case to be processed. We are not concerned with further abstraction that may appear attractive and may be useful internally in the KBS. Perhaps a new data model will be developed, but there is an already implemented model and the chances are that the new system will use an identical model. However, this does not seem a major lack as the development of an appropriate data model is itself a major concern of conventional software engineering and knowledge engineering seems to offer little to this except that the development of the data model is integrated into the overall knowledge engineering process.

The interesting question of deciding on a problem solving method still remains. The simulated expert KBS of course has a specific problem solving method, but this is not necessarily apparent, nor need it be reproduced in the new KBS. The key issue in using the simulated expert, is what type of knowledge it provides. The knowledge provided by the simulated expert essentially comes from its explanation component. The explanation component may provide a way of browsing the system or it may provide explanations that differ from its reasoning in reaching a conclusion, but most likely it is going to provide some sort of activation or rule trace. This further implies a set of cases to exercise the simulated expert KBS. However, if such a KBS exists, the chances are suitable cases can be made available. Because of the likely use of cases to exercise the simulated expert, this approach relates to machine learning evaluation. In machine learning evaluation extensive use has been made of date bases of cases. The performance of different methods has been able to be compared by their performance on learning from these databases. Some of these databases are used in the studies described below. The crucial difference between KA and ML

evaluation is that ML uses the raw data of the cases to derive a KBS, whereas for KA evaluation the simulated expert's explanation of its conclusion is used to build the new KBS. ML is concerned with identifying important features from data. KA is concerned with organising knowledge about the important features provided by the expert. Clearly different styles of KBS and different explanation facilities are going to provide quite different evaluations of different strengths and weaknesses of various KA systems. Also, the evaluation may use some or all of the knowledge provided by the simulated expert to provide different levels of expertise.

The major weakness apparent in this approach to evaluation is that the simulated expert has no meta-knowledge. It can't report that it thinks it has told the knowledge engineer everything that is important and can't reorganise its knowledge presentation to suit the desires of the knowledge engineer etc. However, these are also, at least partially, weakness of human experts, so it is probably reasonable to use a simulated expert that has no ability in this regard. However, KA methods that rely heavily on the meta-knowledge abilities of the expert will have problems with this approach to evaluation.

Experimental Studies

Aim

This paper describes the application of a simulated expert to evaluating the Ripple Down Rule (RDR) methodology. A frequent question raised with respect to RDR is the level of redundancy in the KBS and the importance of the order in which cases are presented. This question is explored with respect to three different domains and three different simulated expert KBS for each domain, three different levels of expertise and a number of different orderings of the data presented. The three different domains are the Tic-Tac-Toe and Chess End Game problems from the UC Irvine data repository and the Garvan thyroid diagnosis problem, also included in the Irvine repository, but here based on a larger data set. The KBSs used for the simulated expert were built by induction from the same data sets using the C4.5, Induct and the RDR version of Induct machine learning algorithms.

Introduction

Ripple Down Rules (RDR)

RDR is a KA methodology and a way of structuring knowledge bases which grew out of long term experience of maintaining an expert system (Compton, Horn et al. 1989) . What became clear from this maintenance experience is that when an expert is asked how they reached a particular conclusion they do not and cannot explain how they reached their conclusion. Rather they justify that the conclusion is correct and this justification depends on the context in which it is provided (Compton and Jansen 1990) . The justification will vary depending on whether the expert is trying to justify their conclusion to a fellow expert, a trainee, layperson or knowledge engineer etc. This viewpoint on knowledge has much in common with situated cognition critiques of artificial intelligence and expert systems but here leads to a situated approach to KA.

The RDR approach was developed with the aim of using the knowledge an expert provided only in the context within which it was provided. For rule based systems it was assumed that the context was the sequence of rules which had been evaluated to give a certain conclusion. If the expert disagreed with this conclusion and wished to change the knowledge base so that a different conclusion was reached, knowledge was added in the form of a new rule of whatever generality the expert required, but this rule was only evaluated after the same rules were evaluated with the same outcomes as before. With this approach rules are never removed or corrected, only added. All rules provide a conclusion, but the final output of the system comes from the last rule that was satisfied by the data.

Initial experiments with this approach were based on rebuilding GARVAN-ES1, an early medical expert system (Horn, Compton et al. 1985; Compton, Horn et al. 1989) . This system was largely rebuilt as a RDR system and it was demonstrated that rule addition of the order of 20 per hour could be achieved and with very low error rates(Compton and Jansen 1990) . It was realised that the error rate could be eliminated by validating the rules as they were added (Compton and Preston 1990) . A valid rule is one that will correctly interpret the case for which it is added and not misinterpret any other cases which the system can already interpret correctly. One possibility is to store all the cases seen or their exemplars (an approach used by Gaines in his version of RDRs (Gaines 1991a)) and check that none of these are misinterpreted. The approach on which most RDR work has been based is to check that none of the cases that prompted the addition of other rules are misinterpreted. These cases are stored - they are the "cornerstone cases" that maintain the context of the knowledge base**.** In fact only one of these cases has to be checked at one time, the case associated with the rule that gave the wrong classification. To ensure a valid rule, the expert is allowed to choose any conjunction of conditions that are true for the new case as long at least one these conditions differentiates the new case from the cornerstone case that could be misclassified. To ensure this the expert is shown a difference list to choose from (Compton and Preston 1990) , a list of differences between the current case and the case attached to the last true rule.

RDR have also been used for the PEIRS system described below (Edwards, Compton et al. 1993) . They have also been used for a configuration task, but in this case a number of RDR knowledge bases were built inductively and a further algorithm was developed to reason across the various knowledge bases(Mulholland, Preston et al. 1993) . The version of RDR was a simple C implementation running under Unix.

Data Sets

The following data sets were used. Chess and Tic Tac Toe are from the University of California Irvine Data Repository. The Garvan data set comes from the Garvan Institute of Medical Research, Sydney

Chess	Chess End-Game -- King+Rook versus King+Pawn on a7. 36 attributes for 3196 cases and 2 classifications.
TicTacToe	Tic-Tac-Toe Endgame database. This database encodes the complete set of possible board configurations at the end of tic-tac-toe games. 9 attributes for 958 cases and 2 classifications.
Garvan	Thyroid function tests. A large set of data from patient tests relating to thyroid function tests. These case were run through the Garvan-ES1 expert system (Horn, Compton et al. 1985) to provide consistent classifications. The goal of any new system would be to reproduce the same classification for the cases. 32 attributes for 21822 cases and 60 different classifications. These are part of a larger data set of 45000 cases covering 10 years. The cases chosen here are from a period when the data profiles did not appear to be changing over time and could be reasonably reordered randomly (Gaines and Compton 1994) . The Garvan data in the Irvine data repository is a smaller subset of the same data. The Garvan data consists largely of real numbers representing laboratory results. Using the Garvan-ES1 pre-processor these were reduced to categories of high, low etc as used in the rules in the actual knowledge base. The preprocessed data was used in the studies below.

Machine Learning Methods

C4.5 (Quinlan 1992) is a well established machine learning program based on the ID3 algorithm. The extensions to the original ID3 are that it deals with missing data, real numbers, provides pruning and allows the KBS to be represented as a tree or rules, with some consequent simplification. The version of C4.5 used was provided by Ross Quinlan. It was used with the default settings, as the aim was not produce the best possible KBS but a reasonable simulated expert. There were no real numbers in the data but a lot of missing data in the Garvan data set.

Induct (Gaines 1989) is based on Cendrowska's Prism algorithm (Cendrowska 1987) . Induct can produce either flat rules or RDRs (Gaines 1991a) . Both versions of Induct were used. The versions used were provided by Brian Gaines as part of the KSSn system. The RDR representation is generally more compact (Gaines and Compton 1992) . Induct does not handle real numbers at this stage, but deals with missing data and provides pruning. No pruning was used in this study.

Although C4.5 and Induct both perform similarly there are important differences in their underlying algorithms. C4.5 attempts to find an attribute to go at the top of the decision tree whose values best separate the various classifications as assessed by the information calculation used. This separation is estimated as the best overall separation, so that there is no requirement that any leaf should contain only one class or a particular class. This process is repeated with the cases at each leaf. In contrast, Induct selects the most common classification and attempts to find an attribute value pair that provides the best selector for cases with this classification. Further attributes value pairs are added to the rule to improve the selection as long as this is statistically warranted. The process is repeated for the remaining cases. The difference with the RDR version of Induct is that it repeats the process separately for cases that are incorrectly selected by the rule and those that the rule does not select resulting in an RDR tree.

Experimental Method

An RDR KBS is built by correcting errors, by adding new rules for cases which have not been given the correct classification. To do this the expert selects relevant conditions from the difference list for that case. The method used here is identical except that any expertise used in selecting important conditions from the difference list is provided from the rule trace from another expert processing the same case. It should not be expected that the simulation will perform better than a real expert or the machine learning techniques on which it rests. The best that could hope to be achieved is defined by the accuracy of the simulated expert (this essentially becomes a measure of the performance of base machine learning technique). Real experts do however perform better than machine learning techniques when there are small data sets (Mansuri, Compton et al. 1991) , and in general a little knowledge can replace a lot of cases for machine learning (Gaines 1991b) .

The following steps are required:

Preparation

- collect a set of cases, and produce a knowledge base using machine learning mechanism - this becomes the basis for the Simulated Expert (SE) (described more fully later).

- Randomise the data to avoid 'lumpiness'

- Start with fresh RDR Expert System (ES), essentially an empty knowledge base.

Processing

Step 1 • get the next case

Step 2 • ask the (simulated) expert for a conclusion

Step 3 • ask the ES under construction for a conclusion

Step 4 • if they agree, get another case and go back to Step 1

Step 5 • if the Expert and the ES disagree, make a new (valid) rule and go back to 1.

Step 5 is the crux. The new rule needs to be constructed and located in the KB. For a RDR simulation, this step consists of:

Step 5 • Run the case against the ML generated KBS and produce a rule trace. This is essentially the justification for the conclusion.

• Run the case on the developing RDR ES and identify the last rule satisfied and the last rule evaluated. The new rule will be attached to the true or false branch of the last rule evaluated according to the evaluation. That is, the new rule will be reached only if exactly the same conditions are satisfied.

• Obtain the difference list based on the current case and the case attached to the last true rule. This difference list provides a filter to select valid conditions that can be used in a new rule.

• Using a combination of the expert's justification (the ML KBS rule trace; i.e. the rules satisfied in reaching a conclusion) and the difference list create a new rule and attach it . The level of expertise of the simulated expert can be varied here by the mix of conditions selected

Analysis

• An examination of the simulation results, and consideration of various metrics that support or do not support the aim of the simulation.

The Simulated Expert

The human expert using an RDR system selects conditions from the difference list to go into a new rule. The simulated expert is simply the mechanism for similarly selecting conditions from the difference list to go in a new rule. The conclusion of the new rule is the conclusion specified in the database for that particular case. Note that if no rule is satisfied the difference list includes all the features of the present case. Three levels of expertise were used in this study.

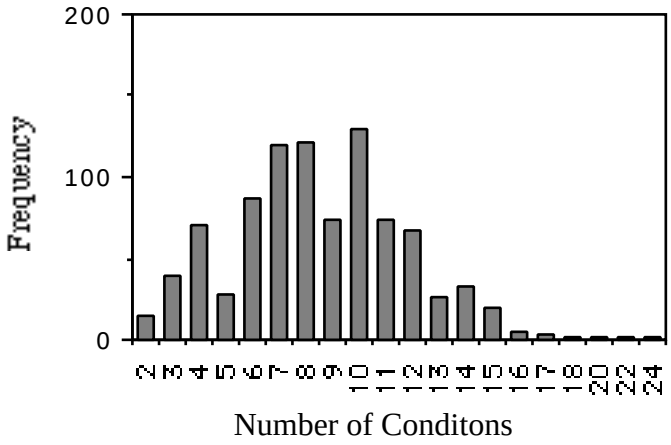
Smartest (S1)	choosing four conditions from the intersection of the ML KBS rule trace for the case and the difference list for the case. These conditions were selected from the top of the list rather than randomly. If the top conditions were less important and a correction had to be applied, the correction difference list would cover the more important conditions lower in the original list. If the intersection contains less than four conditions all the conditions it contains are selected. If it is empty four conditions are chosen form the difference list. In fact selecting four conditions gives the entire intersection in nearly all cases (Fig 1)
Smart (S2)	choosing a single condition from the intersection of the ML KBS rule trace for the case and the difference list for the case. If the intersection is empty one conditions is chosen from the difference list.
Dumb (D)	choosing all conditions from the difference list without reference to the ML KBS

The ML KBS referred to is a KBS built by using the entire data set and one of the machine learning algorithms. The entire data set is used to ensure complete expertise, but which is then weakened by the selection of conditions as above.

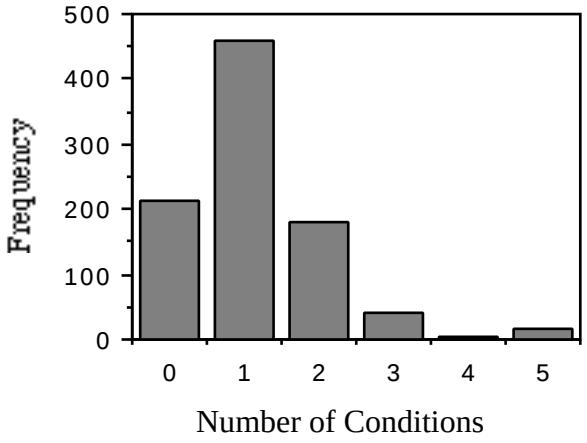
Obviously there are many other ways in which the expertise can be varied; however, these provide a crude separation of three levels of expertise. The third is trivial and ensures that there will be a rule for every different case profile in the database. Figs 1 and 3-5 indicate that the discrepancy between the smartest and the smart expert is not as great as one might expect. Firstly there are frequently zero conditions in the intersection of the difference list and the ML KBS rule trace. In this case conditions are selected from the difference list. Secondly there are hardly ever four conditions in the intersection. In this case the entire intersection is used. These problems are worst with the Garvan data set (Figs 1 and 4) whereas with the other data sets there is a greater difference between the smartest and the smart expert. A reason for zero conditions in the intersection of the difference list and the ML rule trace can be an empty rule trace. A case that is misclassified by the RDR KBS may actually have the default classification so that there is no rule trace from the ML KBS. Note that conditions selected from the difference list are guaranteed to produce a valid rule that will correctly classify the present case but not misclassify other cornerstone cases; this is a strength of RDR. However, the rule though valid may be fairly stupid. Where no rules are satisfied in the RDR KB and the difference list contains the entire case, the selection of conditions is more arbitrary but still determined by the ML KBS rule trace. Where the case has the default classification so that there is no ML KBS rule trace, the selection of conditions is very arbitrary, particularly since conditions are always selected from the top of the list.

The rule traces used come from three KBSs developed by three machine learning systems

- an Induct/RDR knowledge base (IR)
- Induct knowledge base (II) (a series of conventional independent rules without linkages as in RDR.)
- a C4.5 knowledge base (C4)



The number of conditions in an Induct/RDR rule trace for cases from a sample of the Garvan data set



The number of conditions in the intersection of the Induct/RDR rule trace and the difference list for cases from a sample of the Garvan data set.

Fig. 1 Availability of conditions to be used by the simulated expert in constructing a rule

Randomising the Data, Training and Test Sets

The order of the data sets were randomised to provide nine sets of experimental data for each domain. The first 75% of each of the randomised data sets were taken as training data and the remaining 25% as test data. RDR systems were then built for six different sized training sets representing 2.5%, 5%, 15%, 25%, 50%, 75% of the total number of cases available. Since RDR are built incrementally, it was crucial to obtain data on how the system developed.

Results

For Figs 2–5, the outer edges of shaded graphs following are the maximum and minimum of the nine results for each of the six points. The error data is calculated on the remaining 25% of the data for each of the randomisations. The data is incomplete in Fig 2. in that the Garvan data was used in randomised form only for the C4.5 ML studies.

Fig 2. shows results for the three machine learning methods alone when provided with various sized training sets. These results are not critical to the overall findings and merely provide control data on the performance of the machine learning methods. The smaller size of the Induct/RDR KB for each data set is not unexpected as it has been shown previously that Induct/RDR tends to produce more compact knowledge bases (Gaines and Compton 1992) . For this reason RDR has also been proposed independently as a mediating representation in machine learning (Catlett 1992) . For two of the data sets it appears that straight Induct does not produce as accurate a KB as Induct/RDR or C4.5, while overall, Induct/RDR seems to give the best performance. One should not draw significant conclusions from the absolute performance of the methods in this study as no attempt was made to optimise the methods. In previous studies Induct and C4.5 have performed similarly (Gaines 1989) . The strongest conclusion that should be made is that the methods perform differently in the studies here giving some variety in the simulated experts used. The gold standard for the studies here is to produce an RDR KBS as good as the ML KBS from which it is learning.

Figs 3-5 show the performance of the various RDR KBS built, one figure for each data set. The graphs show the increasing size and the improving performance as more and more of the training data is tested on the developing RDR systems. The three horizontal pairs of graphs on each figure show the results for the three different ML methods used to build the simulated experts. The shaded sections on the graphs indicate the range of results for the randomised data for the smartest and smart experts. Since the stupid expert KBS is independent of the ML method used, it is shown only once in the top panel for each data set and results are shown for only one randomisation. The other single line on each of the graphs represents the performance of the ML method when also given more and more of the training data. Only a single line is shown for clarity, but the full randomised data for the ML methods are given in Fig 2.

In terms of size, the Induct/RDR ML KBS are smaller than the smartest and smart RDR systems for all three data sets. However, for straight Induct and C4.5 the RDR systems are as small as the ML systems except for the Tictactoe domain where the Induct KB is clearly smaller that the RDR KBs. These are fairly surprising findings. It is frequently commented that RDR are likely to produce large badly organised KBS because there is no control of the order in which rules are added and since rules are used only in context the same rules may have to be reproduced in many places in the KBS. On the other hand it is generally assumed that ML methods will produce very compact if not optimally compact KBs, in fact the point of induction is to produce a compact representation. These results are therefore strongly supportive of the idea that RDR systems are suitable for practical applications and that repetition in the KB is a comparatively minor problem. On the other hand the results also show that lesser expertise results in larger knowledge bases, with presumably more corrections of less appropriate rules. However, there is no doubt that there is repetition because the ML Induct/RDR KBs are smaller than the manual RDR KBs. However the difference in size is the same as between Induct/RDR and the other ML methods.

The error results show that in general the RDR KBS have comparable errors to the ML KBS except for Induct where the RDR systems have less errors. However from Fig 2 it is not so much that the RDR system have less errors than Induct, but that Induct has greater errors than the other ML methods. However, as noted before, this may well be due to less than optimal use of Induct. There is also a tendency for the RDR systems to outperform induction when there are only small numbers of training cases as ML methods have difficulty with small training sets. This is most noticeable in the Tictactoe data, presumably because this dataset has the smallest number of cases. It is least noticeable with the Garvan dataset with the largest number of cases. Note that the ML KBS used for the simulated experts have been built using the entire data sets and so should be able to provide greater expertise to the RDR systems than attempting to apply ML to a small number of cases. These results are consistent with earlier findings that human experts outperformed ML with small amounts of data (Mansuri, Compton et al. 1991) . This earlier study showed a much greater outperformance of the human expert building RDR (12% errors) to the ML method (ID3) (72% errors) after training on 291 cases. This discrepancy can be explained because these cases covered all 60 classifications (Garvan data) resulting in very small numbers of training cases for each class. A human expert should perform better than the simulated experts because of the problems alluded to earlier of the intersection between the difference list and the ML rule trace often being empty and consequent inappropriate selection of conditions from the top of the difference list, often the complete case data. This is a far cry from true human expertise.

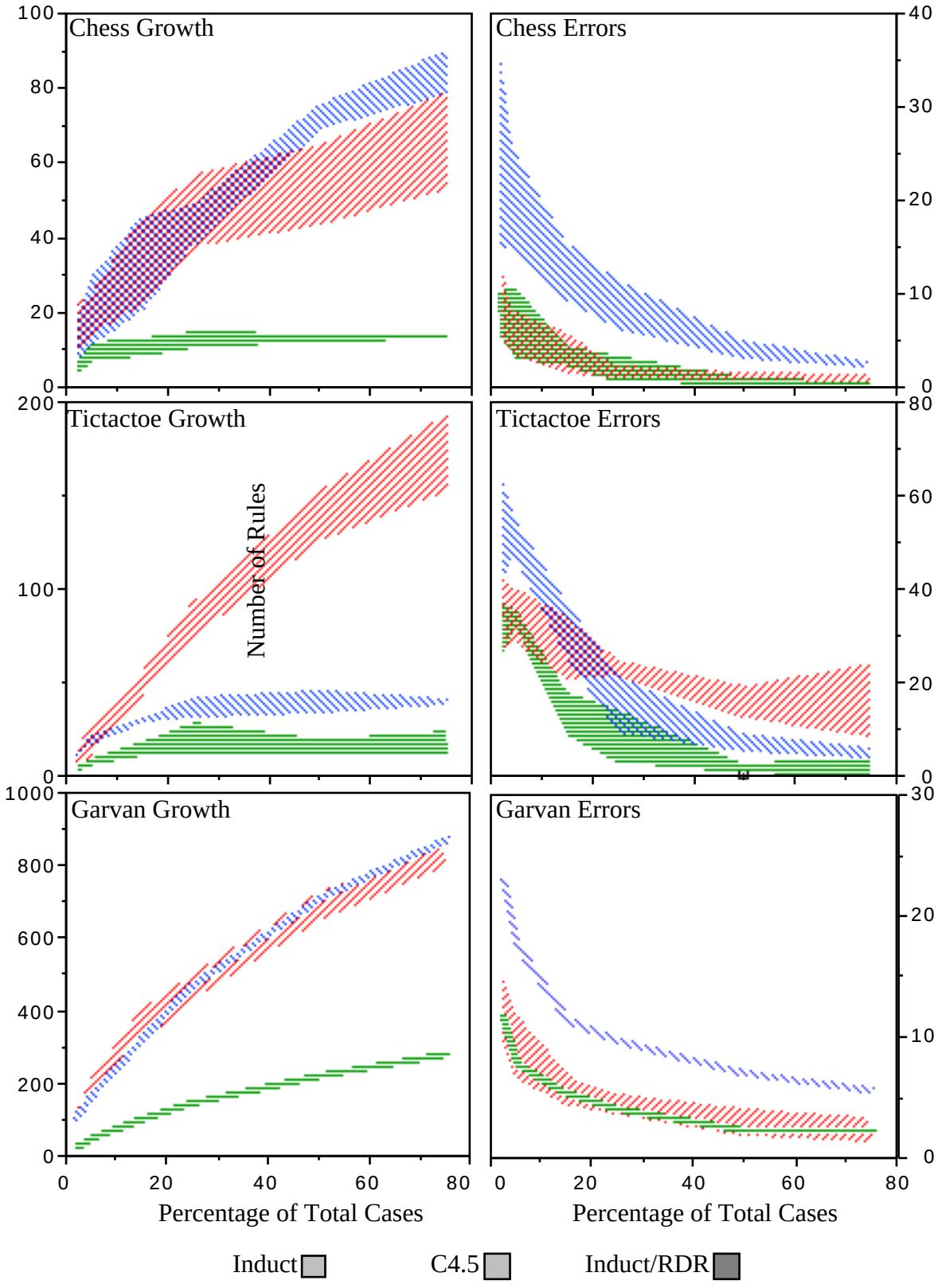


Fig 2. Knowledge Base size and accuracy for ML KBS

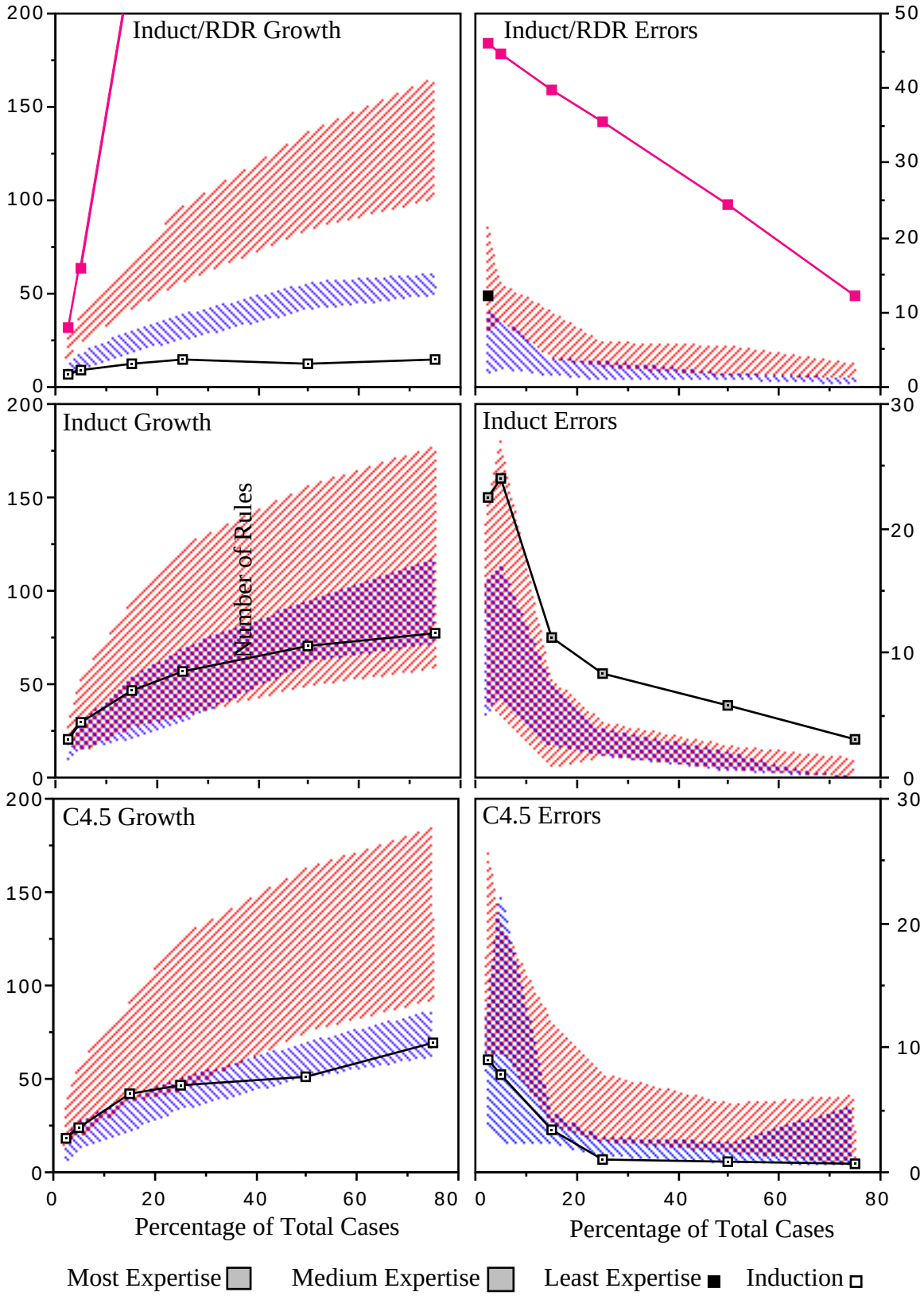


Fig 3. Knowledge base size and accuracy for KBS built for the Chess domain

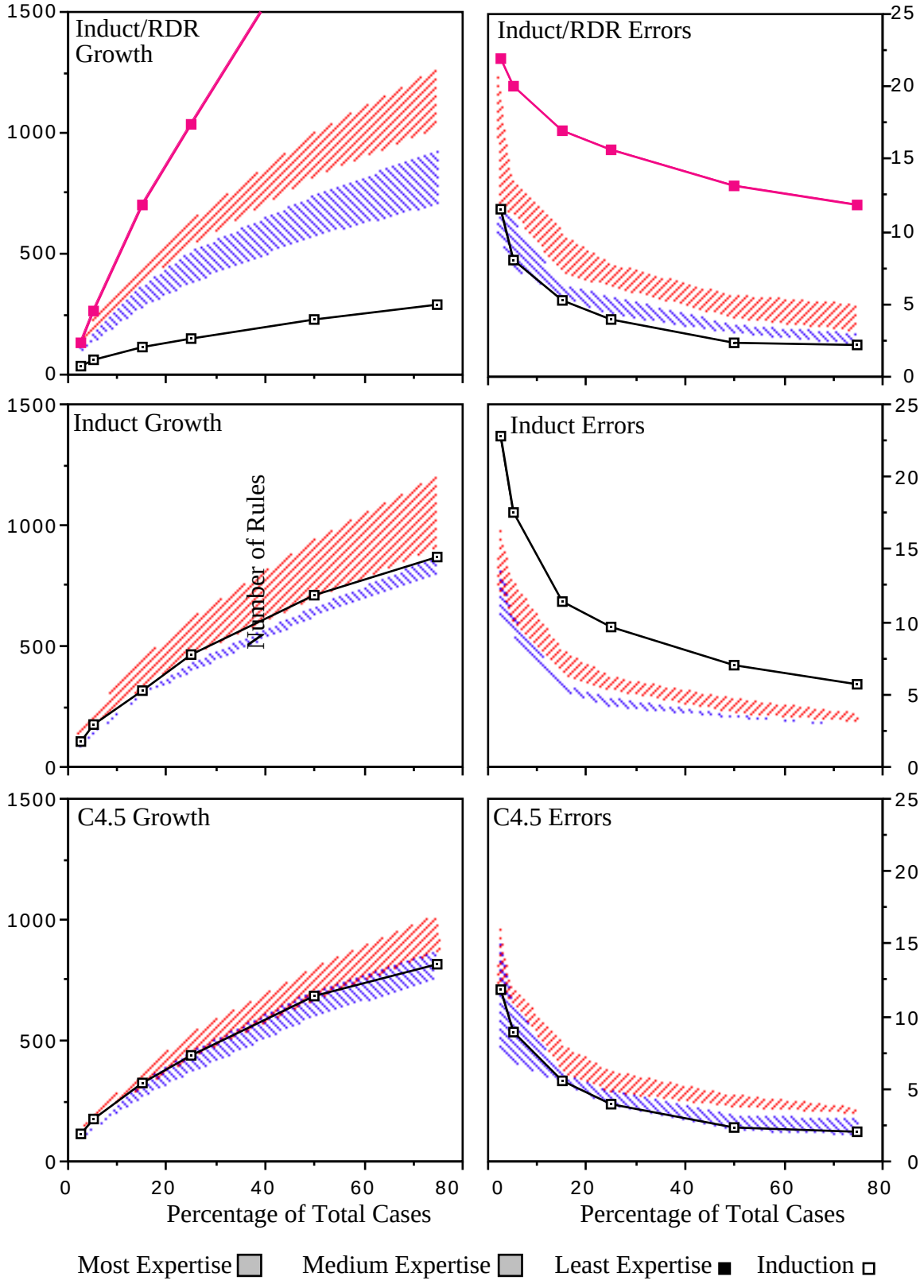


Fig 4. Knowledge base size and accuracy for KBS built for the Garvan domain

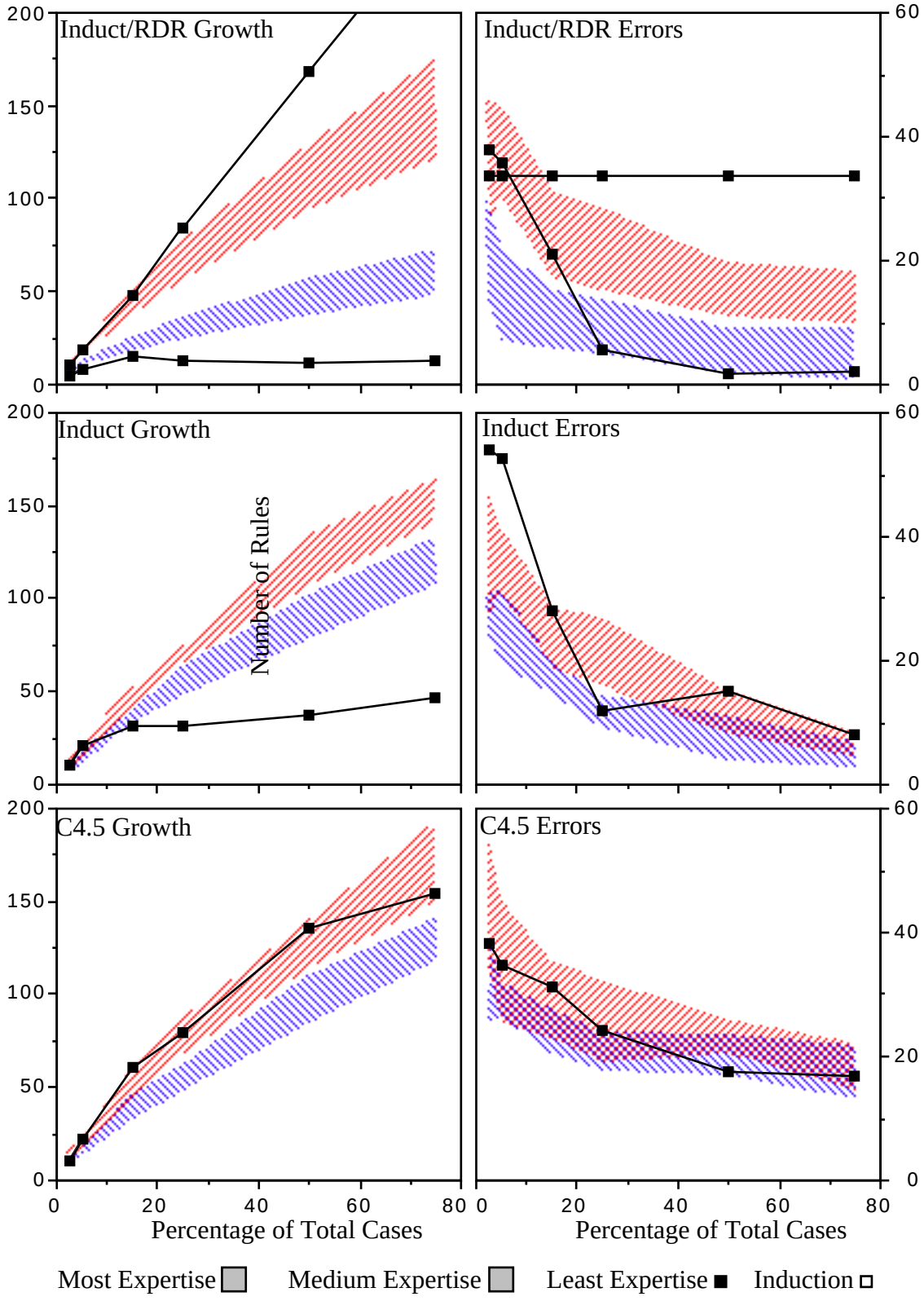


Fig 5. Knowledge base size and accuracy for KBS built for the Tictactoe domain

Discussion

RDR Evaluation

These results demonstrate that manual RDR expert systems produce a KB of similar size and performance to inductively built expert systems. Since inductively built KBs are expected to be reasonably compact there is no evidence to suggest that RDR have a fundamental flaw in the repetition and disorganisation of the KB they might produce. The exception to this is that manual RDR systems are from 2.5 to 4 times as large as the inductively built Induct/RDR KBs. Even if this were the experience with human experts we suggest that this would be acceptable to the experts because of the ease of building RDR systems and certainly the results show that the performance of these systems are acceptable. Now the results also show that the size of the RDR knowledge base depends on the level of expertise, so that we suggest that in fact when genuine human experts are used with better expertise that smaller knowledge bases will result. Certainly there have been no complaints from the expert building PEIRS that after 2000 rules he feels that he is adding the same knowledge repeatedly. Note that the PEIRS expert spends about 15 minutes per day adding rules so that even if there is some repetition it is unlikely to be seen as a significant problem.

The critical issue is why this simple approach of constantly patching errors produces reasonably compact knowledge bases and that greater expertise produces smaller knowledge bases. The answer seems to lie in the difference between trying to make a rule and trying to build a decision tree. With RDR the expert, human or simulated, builds rules. Each addition to the knowledge base is a rule to correctly classify the case in hand and any other cases like it. Ideally, a classification rule will cover all the members of the class but none of the members of other classes. Repetition in the KB will occur where this goal is not achieved and there are false positives and negatives to be handled by further rules. False positives (in this context) are where a rule is too general and some of the cases that fall under it have to be corrected by further rules (which also occur elsewhere in the KB). False negatives (in this context), will occur if the expert makes rules that are too specific so that more rules have to be added (which themselves subsume earlier rules).

Although the rule provided by the expert can be viewed as being added to a binary tree, RDR should not be confused with trying to build a decision tree. Normally with a decision tree, the top nodes do not immediately reach a classification and it is assumed that the classification will be produced after further splitting at lower nodes. It is generally (but not necessarily) assumed that each node will refer to a single attribute and the branches correspond to each value of the attribute. In this framework the choice of which decisions go at the top is critical and algorithms like ID3 attempt to find the best attribute. Another way of looking at this is that when a single attribute decision tree node is selected there is no way of controlling false positives till further nodes are added. The values for the attribute split the population and that is that. In contrast, with RDR, each rule attempts to deal with a particular class, so each rule can be selected to limit both false positives and false negatives. In particular, further conditions can be added to the rule to limit false positives. If we assume that each rule is perfect and there are no false positives or negatives, it does not matter whether the initial rules deal with classes with many members or few. If a rule (using attribute value pair conditions) is specific to the class, then cases for other classes will fail to satisfy the rule and will result in new rules being created. The order doesn't matter as long as the rules have no false positives. As false positives are introduced the order in which cases occur and rules are added become more critical. A rule with irrelevant conditions will have a lot of false positives. This suggests that the higher the level of expertise the less important to the final size is the order in which rules are added .

For an expert to build a decision tree he or she must think about the whole domain in selecting an attribute to go at the top of the tree. With RDR the expert is asked simply to justify their conclusion in the context in the same way as normal human discourse. An expert is someone who knows the right conclusion and is likely to provide a justification that is both as general and as

error free as possible in the context; i.e. it implicitly minimises the false positives and false negatives in the context; i.e. it is a good practical explanation. The simulated expert using the ML rule trace also does this to some degree. The RDR approach thus provides a framework where the natural performance of an expert will tend to produce a compact KB. The experts response will also probably encourage the development of the KBS in a way that is appropriate for the domain. In a domain where false positives are problematic the expert will tend to give more specific rules resulting in gradual but very accurate coverage of the domain. Where false negatives are not desirable the expert will tend to make more general rules resulting in the need for greater correction.

The results for PIERS suggest human experts tend to produce rules that are very general, and so would produce few false negatives while at the same time producing few false positives needing further correction (Compton, Edwards et al. 1991) . The rules tend to have only one or two conditions in them and on average only two to three rules have to be satisfied to reach the correct conclusion. The resulting structure is closer to a long decision list with each rule having a further decision list refinement than to a tree. The depth of decision list corrections is only two or three but the length of the decision lists is about 50.

Of course some repetition does occur as shown in the results This repetition can be dealt with by Gaines strategy (Gaines 1991a) as discussed. However, reorganisation of the KB is likely to be a secondary or rare requirement, with the results here not indicating any pressing need to reorganise the KB.

Since RDR have been compared here to induction, the question arises of why not simply use induction. For induction all the training cases need to be well classified by an expert prior to use and in many domains such cases are not available. For RDR the cases only need to be well classified by the expert when they are used to add a rule to a system. Hence, a good way of preparing cases for induction would be to build an RDR system which produces a version of the required KB in the process. As the PIERS experience shows this can occur while the developing KB is in routine use.

A conclusion from this work is that the effort that is put into trying to organise knowledge into an optimal model is often unnecessary. It is perfectly adequate merely to keep patching errors with local corrections. Not only is this practical, but the task does not require a knowledge engineer or knowledge engineering skills, thus transforming the practical possibilities of KBSs.

Finally, this is seen as a counter intuitive result, probably because of the underlying Platonic motivation in much AI of finding the right model, the right representation or the right knowledge . Something that works and is simple but is not elegant and demanding is perhaps not seen as attractive. The situated cognition perspective on the other hand suggests that since the knowledge experts construct is always going to be justification constructed on the fly in context, a local patch approach is an appropriate solution.

The critical question for the RDR approach is whether a local patch approach can be found for tasks other than single classification. A solution has been found for multiple classification problems (Kang, Compton et al. 1994; Kang, Compton et al. 1995) and the size of the knowledge acquisition task is similar to ordinary RDR. It remains a (hopeful) conjecture as to whether this can be used as a basis for other tasks, removing the need for task analysis (Compton, Kang et al. 1993) . More radically it has been suggested that testing and repairing KBs is a more important and more fundamental issue in developing KBs than analysing the tasks to be carried out (Menzies and Compton 1995) .

Simulated Experts

Most KA research at present is concerned with modelling the problem solving activity required for a particular task and the task domain. The KA method may provide a framework in which this is approached or it might provide a way of developing KA tools specific to the problem and this in turn may be carried out by building a tool de novo or by assembling a tool from components or modifying existing tools. Evaluation of these methods, such as in the Sisyphus studies, have primarily focussed on trying to make clear the different ways in which these tools and methods work and their relative ease of use. Very little attention has been paid to the task of actually populating or maintaining the knowledge base. It seems to be assumed that if the initial analysis and tool building is done well enough this will become a trivial problem. To us a two fold evaluation seems necessary, an evaluation of the ease of building the shell for the task and an evaluation of the ease and efficiency of populating the KB and its performance.

In theory this approach can be used to populate any knowledge base once a shell has been set up. To populate a knowledge base there must be some way of systematically getting the expert to provide expertise covering the domain. In the studies here cases were used, and no order in the availability of the cases was required. We believe that eventually all knowledge acquisition reduces to asking an expert, at least implicitly, to deal with cases. These can be real cases that the KBS fails to deal with or synthetic cases constructed by the knowledge engineer or automatically, asking the expert "what if . . . ?". There is no reason why in the evaluation method here one cannot use synthetic cases or order the cases in some way. Similarly there is no reason why algorithms cannot be set to answer questions from the simulated expert KBS such as: "Which attribute is most important in distinguishing between conclusion A and conclusion B?", "What is the most common conclusion made?", "What conclusion do you have the most knowledge about?", etc. Setting up such a framework would seem possible for all KBS albeit with varying difficulty depending on the interaction with the expert required.

Clearly the answers to these questions by a simulated expert will not be as good as answers from real experts, just as there were problems with the quality of the simulated experts here. However the issue is whether one can provide useful data for comparisons. In the study here the use of a simulated expert approach made it possible to build 180 different RDR systems (3 domains x 3 ML experts x 2 levels of expertise x 9 randomisations) and provide data on the repetition in and performance of RDR KBs. This approach allows one to compare different KA methods and to modify the type and level of expertise provided. The differences in the performance of methods should be readily identifiable and the simulated expert algorithm readily modified to explore these differences even if the expertise is not ideal. The expertise will be less than and different from human expertise but the volume and variety of data that can be produced is quite impossible using real experts. We suggest that the simulated expert approach is not only suitable but is probably the only way in which to evaluate how different methods and tools facilitate populating knowledge bases.

Acknowledgments

This work has been funded by the Australian Research Council. The authors are grateful to Brian Gaines and Ross Quinlan for making available their software available and to Brian Gaines for his assistance in preparing the Garvan data set.

Bibliography

Bachant, J. and McDermott, J. (1984). R1 revisited: four years in the trenches. **The AI Magazine** Fall: 21-32.

Catlett, J. (1992). Ripple-Down-Rules as a Mediating Representation in Interactive Induction. **Proceedings of the Second Japanese Knowlege Acquisition for Knowledge-Based Systems Workshop**, Kobe, Japan, 155-170

Cendrowska, J. (1987). An algorithm for inducing modular rules. **International Journal of Man-Machine Studies** 27(4): 349-370.

Compton, P., Edwards, G., Kang, B., Lazarus, L., Malor, R., Menzies, T., Preston, P., Srinivasan, A. and Sammut, C. (1991). Ripple down rules: possibilities and limitations. **6th Bannf AAAI Knowledge Acquisition for Knowledge Based Systems Workshop**, Banff, 6.1-6.18

Compton, P., Horn, R., Quinlan, R. and Lazarus, L. (1989). Maintaining an expert system, in J. R. Quinlan (Eds.), **Applications of Expert Systems.** London, Addison Wesley. 366-385.

Compton, P., Kang, B., Preston, P. and Mulholland, M. (1993). Knowledge Acquisition without Analysis, in N. Aussenac, G. Boy, B. Gaineset al (Eds.), **Knowledge Acquisition for Knowledge Based Systems. Lecture Notes in AI (723).** Berlin, Springer Verlag. 278-299.

Compton, P. and Preston, P. (1990). A minimal context based knowledge acquisition system. **"Knowledge Acquisition: Practical Tools and Techniques" AAAI-90 Workshop**, Boston,

Compton, P. J. and Jansen, R. (1990). A philosophical basis for knowledge acquisition. **Knowledge Acquisition** 2: 241-257. (Proceedings of the 3rd European Knowledge Acquisition for Knowledge-Based Systems Workshop, Paris 1989, pp 75-89)

Edwards, G., Compton, P., Malor, R., Srinivasan, A. and Lazarus, L. (1993). PEIRS: a pathologist maintained expert system for the interpretation of chemical pathology reports. **Pathology** 25: 27-34.

Fensel, D. and Poeck, K. (1994). A Comparison of Two Approaches to Model-Based Knowledge Acqusition, in L. Steels, G. Schreiber and W. Van de Velde (Eds.), **A Future for Knowledge Acqusition: Proceedings of EKAW'94.** Berlin, Springer-Verlag. 46-62.

Gaines, B. (1989). An ounce of knowledge is worth a ton of data: quantitative studies of the trade-off between expertise and data based on statistically well-founded empirical induction. **Proceedings of the Sixth International Workshop on Machine Learning.**, San Mateo, California, Morgan Kaufmann. 156-159

Gaines, B. (1991a). Induction and visualisation of rules with exceptions. **6th AAAI Knowledge Acquisition for Knowledge Based Systems Workshop**, Bannf, 7.1-7.17

Gaines, B. (1991b). The tradeoff between knowledge and data in data acquisition, in G. Piatetsky-Shapiro and W. Frawley (Eds.), **Knowledge Discovery in Databases.** Cambridge, Massachusetts, MIT Press. 491-505.

Gaines, B. and Compton, P. (1994). Induction of Meta-Knowledge about Knowledge Discovery. **IEEE Transactions on Knowledge and Data Engineering** 5(6): 990-992.

Gaines, B. and Musen, M., Ed. (1994). Proceedings of the 8th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. Banff, Canada,

Gaines, B. R. and Compton, P. J. (1992). Induction of Ripple Down Rules. **AI '92. Proceedings of the 5th Australian Joint Conference on Artificial Intelligence**, Hobart, Tasmania, World Scientific, Singapore. 349-354

Horn, K., Compton, P. J., Lazarus, L. and Quinlan, J. R. (1985). An expert system for the interpretation of thyroid assays in a clinical laboratory. **Aust Comput J** 17(1): 7-11. (Aust Comput J)

Kang, B., Compton, P. and Preston, P. (1994). Multiple Classification Ripple Down Rules. **Proceedings of the Third Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop (JKAW'94)**, Tokyo, Japanese Society for Artificial Intelligence. 197-212

Kang, B., Compton, P. and Preston, P. (1995). Multiple Classification Ripple Down Rules : Evaluation and Possibilities. **Proceedings of the 9th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop**, Banff, Canada, in press

Linster, M., Ed. (1992). Sisyphus'91: Models of Problem Solving. GMD (Gesellschaft fur Mathematik und Datenverarbeitung mbH).

Mansuri, Y., Compton, P. and Sammut, C. (1991). A comparison of a manual knowledge acquisition method and an inductive learning method. **Australian workshop on knowledge acquisition for knowledge based systems**, Pokolbin, 114-132

Menzies, T. and Compton, P. (1995). The (Extensive) Implications of Evaluation on the Development of Knowledge Based Systems. **Proceedings of the 9th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop**, Banff, in press

Mulholland, M., Preston, P., Hibbert, B. and Compton, P. (1993). An expert system for ion chromatography developed using machine learning and knowledge in context. **Proceedings of the Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems**, Edinburgh, 12 pages

Quinlan, J. (1992). C4.5: Programs for Machine Learning. Morgan Kaufmann.

Shaw, M. (1988). Validation in a knowledge acquisition system with multiple experts. **Proceedings of the International Conference on Fifth Generation Computer Systems**, Tokyo, 1259-1266