A Simulation Framework for Knowledge Acquisition Evaluation

Tri M. Cao **Paul Compton**

School of Computer Science and Engineering
University of New South Wales,
Sydney 2052, Australia
Email: tmc@cse.unsw.edu.au

Abstract

Knowledge acquisition (KA) plays an important role in building knowledge based systems (KBS). However, evaluating different KA techniques has been difficult because of the costs of using human expertise in experimental studies. In this paper, we first address the problem of evaluating knowledge acquisition methods. Then, we develop an analysis of the types of errors a human expert makes in building a KBS. Our analysis suggests that a simulation of the key factors in building a KBS is possible. We demonstrate the approach by evaluating three variants of a practically successful KA methodology, namely Ripple Down Rules (RDR). The experimental results provide some fundamental insights into this family of KA techniques and suggest various hints for improvement.

Keywords: artificial intelligence, knowledge base construction, knowledge acquisition, simulation.

1 Introduction

Evaluation of KA tools and methodologies remains problematic (Menzies & Hamelen 1999, Shadbolt & O'hara 1999). The essential problem is that any evaluation requires people to actually build a KBS. If one wishes to compare two methods then ideally the same experts would use each of the methods, data would be collected and comparisons made. Of course if the same expert uses two methods for the same domain, his/her experience in building the first system will bias their performance on building the second system. Further, where does one find even one expert able to give their time to evaluation experiments, let alone the multiple experts a proper study would require?

The Sisyphus experiments have been the major attempt at evaluation by the KA community (Shadbolt & O'hara 1999). The Sisyphus experiments provided various types of material describing a problem area and references to other resources, which could be used by a non- expert in the problem domain to build a problem solver. The outcomes from these studies were both interesting and valuable, but both the outcomes and the studies themselves largely reflected the current focus of the knowledge acquisition (KA) community on knowledge level (KL) modelling. The various papers published showed how different problem solving methods (PSMs) could be applied to the same problem type and how domain knowledge might be represented for use with the PSM. However, it was impossible for the participants to develop a complete system or to properly test the system they had developed. There was simply not enough material provided, or even if there had been it probably would have been too costly for participants, who were not domain experts to build a real system. That is, these studies could only really evaluate whether a PSM and representation scheme appeared suitable for a domain and the costs of preparing these.

There is a fundamental problem with only evaluating the early stage of developing a KBS. KA research came about to address the "knowledge-engineering bottleneck". That is, in the early days of expert systems it was easy to demonstrate the technology on small problems, but there was considerable difficulty with the ad-hoc approach in dealing with large scale knowledge-acquisition. The Sisyphus experiments demonstrated a number of putative advantages for various PSMs but did not get to the stage where these could be really demonstrated. This problem also occurs in the one study with a comparison of a systematic modelling approach to an ad-hoc approach (Corbridge & Shadbolt 1995). This study showed better performance from ad-hoc methods, but the problem was so small that ad-hoc methods may have done better because there were not the overheads from using a modelling methodology.

As the Sisyphus experiments have developed (Shadbolt & O'hara 1999), there has been increasing emphasis on documenting the time and other costs involved in carrying out the initial development. There have also been attempts to make the domains richer by providing further data and resources. But outcomes have remained at the level of demonstrating the possible application of various PSMs and other KBS components rather than demonstrating actual performance in developing a KBS.

One can find a range of reports on the development of individual KBS. However, to be published as academic papers these tend to report on developments

of systems for new problems. They therefore tend to provide arguments on why the method or tool they used is suitable for the problem rather than quantitative data that can be used for comparisons. They also tend to be published at an early stage of development, rather than being a reflection on experience. Shadbolt et al. make the point that funding tends to be for new developments rather than evaluation (Shadbolt & O'hara 1999). There is a small amount of work on maintenance (Menzies 1999), but this does not provide a good basis for comparative evaluation as it tends to focus on experience with individual systems.

One approach to providing multiple experts for evaluative studies is to use simulated experts (Compton, Preston & Kang 1995). In these studies various KBS were developed by machine learning. When a case was processed by these systems, the rules that fired to correctly process the case and the conditions in these rules, were used as an expression of expertise. The level of expertise could then be varied by the number of conditions selected from the rule trace and the addition of random features from the case. These "experts" were then used to provide rules to build further KBS. The advantage of this approach was that it allowed complex domains to be considered and complete KBS to be developed. An obvious limitation of the approach is that a rule trace from a machine-learning-developed KBS does not necessarily correspond to human expertise. However, the use of simulated experts did allow for repeat experiments with a range of variations.

Another problem occurs in these simulated expert studies which also occurs more generally in machine learning evaluation. Machine learning evaluation is well developed but is generally based on using databases of cases drawn from specific domains (Blake & Merz 1998). There are variations in performance of the different algorithms on different datasets. However the datasets are not characterised in a way that assists in explaining the variations. One simply has a result that method X works well on the soya bean dataset but not perhaps as well on the chess end-game dataset.

A final limitation of using machine learning to build simulated experts (Compton et al. 1995), was the provision of a domain representation with the data set, so the process of developing this could not be simulated. However, it is not always a requirement to develop a domain representation as there are many KBS applications which link to a larger information systems which then provide the domain representation. It was one of the strengths of the Sisyphus studies that they required a knowledge representation to be developed from descriptive material about a domain.

Finally, a basic tool in computer science is complexity analysis. Complexity analysis is conventionally concerned with the time and memory required for a program dealing with a certain class of problem. Recent work has attempted to carry out a kind of complexity analysis for the process of KA (Beydoun & Hoffmann 2001). However, the analysis here attempts to quantify how much knowledge, or how many interactions with an expert will be required to develop a KBS in the worst case.

2 Expert performance analysis

It seems that the aspects of expert performance that affect development of a KBS are over-generalisation or over-specialisation. For example if an expert provides a definition (e.g in the form of a classification rule), then this definition may cover objects to which the definition does not properly apply, or the definition may exclude objects which it should cover. For example, a poor definition of a horse may include cows; or more reasonably, donkeys. On the other hand a more precise definition may exclude miniature ponies. There are no absolutes in such a process; a definition of horses may well include donkeys or exclude miniature ponies quite appropriately; it depends what the definition is to be used for. Regardless of what the definition is used for, the two errors that will occur are that objects will be covered by the definition inappropriately or objects which should be covered will be excluded. In classification based systems, errors of over-generalisation or over-specialisation are often called false positives or false negatives respectively. Other terms such as sensitivity and specificity are also used to quantify of over-generalisation and over-specialisation in other disciplines.

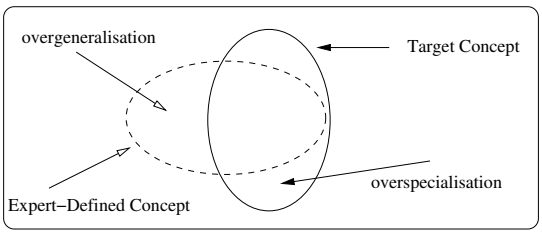

Figure 1: Overspecialisation and overgeneralisation

These errors not only apply to individual definitions, but to a KBS itself. It is clear they apply to classification systems. An error where an object which is not classified, but which should be, is a false negative. An error where an object is classified, but shouldn't be, is a false positive. The most common error is when an object is wrongly classified. Here there is a false positive error as the object is included in a class to which it does not belong. There is also a false negative error as the KBS failed to cover the object correctly.

This analysis seems to apply not only to classification system but all aspects of KBS. With a planning system, the KBS has error components that that cause an incorrect plan to be produced for the data provided. That is, the data was covered inappropriately; there was overgeneralisation. However, the system also failed to cover the data correctly, and that was overspecialisation. In a similar manner, these

errors also apply to ontologies acquisition. The definitions of concepts, or the relations between them result in objects failing to be covered or being covered inappropriately. If an expert provides too many repeated low level definitions rather than developing abstractions, there is an overspecialisation error. In summary it seems fair to say that any KA depends on declarative knowledge of one form or another. The errors in declarative knowledge, whether provided by an expert or obtained by machine learning are overgeneralisation and overspecialisation. This is not a very profound insight, it is simple common sense. However, it seems to provide a useful basis for developing a KA simulation.

In the study here we simulate obtaining rules from the expert and so apply these errors at the rule level. A given rule may cover cases for which the conclusion is not appropriate; that is it is too general. Or the rule is too specific and so excludes some cases that should be covered. The intuitive response to an overspecialised rule is to remove conditions and for an overspecialised rule to add conditions. However, whether one does this or corrects the system in some other way depends on the KA tool or method being used.

It should be noted that there are other perspectives beyond overgeneralisation and overspecialisation. One could have a hierarchy of errors and consider the cost of errors in the application domain (Menzies & Hamelen 1999). This may also include a notion of the degree of error. Some errors are close to the correct answer, others are far away. However, this judgement is made from some sort of perspective of how bad the error is in the domain. That is, it is from some sort of cost perspective. In the current simulation framework developed here we simply consider all errors as equal; an answer is either right or wrong. However, a cost scale would be easy to include in further development.

The key insight in this paper is to attempt to describe expert performance in terms of overgeneralisation and overspecialisation errors. A related assumption is that the knowledge acquisition interaction with the domain expert will cover many separate items and that normally there will be some kind of iterative process. For example a number of different rules and/or a number of definitions related to an ontology may be provided. If the items of knowledge provided have errors of overgeneralisation and overspecialisation, there will also be some process of fixing or improving. This does not necessarily imply a distinct maintenance phase. In the initial development the expert may simply go back and improve earlier definitions as later ones are added. What a KA simulation does is to model the effect of combining errors from many different items and where appropriate to also model the process whereby these errors are then corrected.

It seems that the two errors can be used to describe the differences between different levels of expertise (for example, experienced experts and trainees). Trainees will have a greater number of false positives or false negatives, or both, than experienced experts in the domain. The type of error that is more acceptable will vary with the domain. One major problem with previous work that used simulated experts is how to model levels of expertise. For example in (Compton et al. 1995), levels of expertise are represented by picking various subsets of the full condition. There is no such difficulty in our approach as it allows us to demonstrate the effects of different levels of expertise by using different combinations of overgeneralisation and overspecialisation.

3 The Simulation Framework

For simplicity a number of assumptions are used in the simulation framework here. We do not claim that these assumptions perfectly capture the KA models. However, they allow various parameters to be adjusted so that the effect of using more accurate assumptions is encompassed anyway. This is not unusual in simulation as the benefits of simulation are not only from the results of the simulation, but from the process of developing the simulation. Many of the results of a simulation became obvious independent of the simulation. They become obvious through the process of trying to build a simulator.

As noted above, the simulation here is restricted to classification. Secondly the domain is assumed to be made up of disjunctive regions. The minimum number of rules required is therefore the number of disjuncts in the domain. In the simulation we assume that all disjuncts are equally probable. This is of course not realistic. In a real domain it is highly likely that some regions of the attribute space will be more probable than others. However, the range of results that would occur with a more realistic distribution can be covered by having a greater of less number of equally probable disjuncts.

Expert performance has been described in terms of overgeneralisation and overspecialisation, as depicted in Figure 1. For simplicity it is assumed that these will be the same for all pieces of knowledge added for a specific KBS. Again it would be more realistic if some pieces of knowledge provided by an expert were more reliable than others. Again, the range of results can be simulated by having all the errors at high or low levels. The simulation thus allows a kind of worst case analysis to be carried out. To simulate all the errors being large must be worse than some being large and some being small. However, one can also get some understanding of small errors, by simulating all the errors being small.

3.1 Notations and Definitions

Let U denote a non empty set as the universal *domain*. We will need to define some notations that will be used in the later sections.

Definition 1 *A* data case d *is a point in the domain* U. *We also write* $d \in U$.

We now consider finite partitions of U. A particular finite partition can be thought of as the target concept

structure (or the knowledge) that we would like to acquire from the expert and transfer to a system.

Definition 2 *A finite collection* $\{A_1, A_2, \ldots, A_n\}$ *of subsets of* U *for some nonzero natural number* $n \in \mathbb{N}^+$ *is called a* partition *of* U *if:*

- *For all* $1 \leq i \leq N$, $A_i \subseteq U$
- $\bigcup_{i=1}^{n} A_i = U$, *(the subsets cover the domain),*
- *For all* $1 \leq i, j \leq N$, $A_i \cap A_j = \emptyset$ *(the subsets are disjunctive).*

From now on, we fix a particular partition $\mathbb{A} = \{A_1, \ldots, A_n\}$ of U. Moreover, we refer to a *primary rule* (or a *rule* in short) as a production rule of the form $R : \alpha \rightarrow C$ where R is called the *label* of the rule, α is called the *condition* that can be considered as a constraint to define a region in the domain and C is called the *conclusion* which is a classification. As noted before, though the simulation here applies only for classification problems, it is easy to extend the ideas to more complex problems. A rule is intuitively an attempt of the expert to capture a particular region in the domain. Therefore, we say a condition α is *satisfied by* a data case d if d is in the region defined by α. In this case, we also say, the data case d is satisfied by the rule R or R *fires on* d.

Definition 3 *Let* $R : \alpha \rightarrow C$ *be a rule and* d *be a data case. We denote* $Fires(R, d)$ *if* d *is in the domain's region defined by* α.

Definition 4 *Let* R *be a rule, we denote*

- *the set of data cases evaluated by* R *as* $\text{Input}(R)$,
- *the set of data cases that satisfy* R *as* $\text{Accept}(R)$,
- *the set of data cases that do not satisfy* R *as* $\text{Reject}(R)$.

Property 5 *For any rule* R *in the knowledge base,* $\text{Input}(R) = \text{Reject}(R) \cup \text{Accept}(R)$.

Definition 6 *Let* $d \in U$ *be a data case. We denote* $\text{Class}(d)$ *to be the actual classification of a data case* d, *i.e,* $\text{Class}(d)$ *is the unique* $A_i \in \mathbb{A}$ *such that* $d \in A_i$.

Definition 7 *Let* $D \subseteq U$ *be a set of data. We denote* $\text{Size}(D)$ *to be the cardinality of the set* D.

3.2 Analysis

In this section, we investigate the effects that overgeneralisation and overspecialisation factors have over a primary rule in the knowledge base. Note that due to the length of the paper, all the proofs are omitted.

Suppose we have the target concept A that we want to capture with a rule R as in the figure 2. Because the expert we consult is not perfect, the region R captures is A'. Furthermore, assume that the cardinalities of each region of the domain are denoted as in the figure.

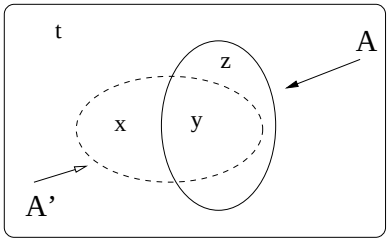

Figure 2: Captured concept

Then the overgeneralisation and overspecialisation factors in our framework are realised as

$$OS = \frac{z}{z+y}; \quad OG = \frac{x}{x+y} \tag{1}$$

Note that the more familiar *precision* and *recall* factors, (usually used in measuring how a classifier performs) can also be realised as

$$Precision = 1 - OS;\ Recall = 1 - OG.$$

Now, given a data case d that is drawn randomly from the domain U, we can compute $cover(R, d)$, the probability that d is accepted by rule R. If $\text{Class}(d)$ is A then:

$$cover(R, d) = \frac{y}{y+z} = 1 - OS$$

otherwise if $\text{Class}(d)$ is not A we have

$$\begin{aligned} cover(R, d) =& \frac{x}{x+t} \\ =& \frac{x}{y+z}\frac{P(A)}{1-P(A)} \\ =& \frac{(1-OS)*OG}{1-OG}\frac{P(A)}{1-P(A)} \end{aligned}$$

where $P(A)$ is the prior probability of a data case randomly drawn from U being in A. If we assume that all the classes are equally distributed, then $P(A) = \frac{1}{n}$.

The following proposition gives the size of the set of data accepted by a rule R with respect to the size of the input set to R.

Proposition 8 *Let*

- R *be a rule with target concept* A;
- OG *and* OS *be the overgeneralisation and overspecialisation factors respectively;*
- $P(A)$ *be the probability that a data case* d *in* $\text{Input}(R)$ *is actually of class* A

then we have

$$\text{Size}(\text{Accept}(R)) = P(A) * \frac{1-OS}{1-OG} * \text{Size}(\text{Input}(R)).$$

Proposition 9 *Let*

- R *be a rule with target concept* A;
- OG *and* OS *be the overgeneralisation and overspecialisation factors respectively;*

then we have

$$P(d \in A | d \in \text{Accept}(R)) = 1 - OG;$$

$$P(d \notin A | d \in \text{Accept}(R)) = OG.$$

This proposition states that the distribution of the A in the Accept(R) only depends on OG. It does not depend on the distribution of A in Input(R) or OS.

In the following sections, we will apply this framework to analyse knowledge acquisition methods.

4 Ripple Down Rules

In this section, we introduce three variants of a practically successful KA methodology, namely Ripple Down Rules (RDR). RDR was first introduced in (Edwards & Compton 1993) and applied in the medical expert system PEIRS. Formal specification of RDR can be found in (Colomb 1999), (Richards & Compton 1998) and (Cao *et. al* 2004). The approach has also been adapted to a range of tasks: control (Shiraz & Sammut 1997), heuristic search (Beydoun & Hoffmann 2000), document management (Kang *et. al* 1997), landscape management (Martinez *et. al* 2001) and configuration (Compton *et. al* 1998). The level of evaluation in these studies varies, but overall they clearly demonstrate very simple and highly efficient knowledge acquisition. We will look at three variants of RDR: Single Classification RDR (SCRDR), Flat RDR (Flat RDR) and Composite Rules (or Decision List (DL)).

4.1 Single Classification Ripple Down Rules

A SCRDR knowledge base is a finite binary tree with two distinct types of edges, labelled with either *except* or *if-not*, whose nodes are labelled with primary rules. Figure 3 depicts an example of a binary RDR knowledge base. We call α the condition and C the conclusion of the rule.

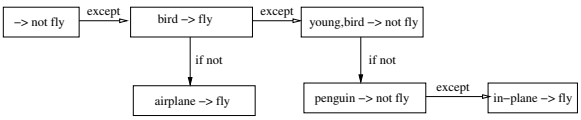

Figure 3: An RDR knowledge base

SCRDR works as follows: a data case d is passed to the tree starting from the root. If the data case entails the condition of the current node, the conclusion is temporarily recorded (and overrides the previous conclusion). The data case is then passed to the next node in the *except* branch. If the case does not entail the condition of the rule, it is passed to the next node in the *if-not* branch. The process continues until there is no next node to evaluate. This algorithm defines a path from the root of the tree to the node that gives classification for the case.

One of the strengths of the RDR framework is that rule bases are easy to revise. There is some supervision of the system so that incorrect classifications are detected. If an incorrect classification is given by the system which the expert wishes to correct, the expert identifies features in the case which suggest the new conclusion. This rule is automatically added to the knowledge base so that it goes at the end of the previous inference path for the case. That is, the case will be processed in exactly the same way, except that it will also be passed to a further rule the new rule added. The classification given by the system comes from the last rule satisfied. That is, the case is first given the erroneous conclusion as before. This is then replaced by the new conclusion if the correction rule fires. There is also a further component of the method that the expert has to select sufficient features in the case so that the new rule will not fire on a case for which the previous rule was correct.

4.2 Flat Ripple Down Rules

Flat RDR can be considered as a *n-ary* tree of depth two. Again, each node of the is labelled with a primary rule with the following properties:

- The root is a default rule which gives a dummy classification (for example UNKNOWN).
- The rules in the nodes of depth 1 give classification to a data case
- The rules in the nodes of depth 2 are called deletion rule and work as refinements to the the classification rules.

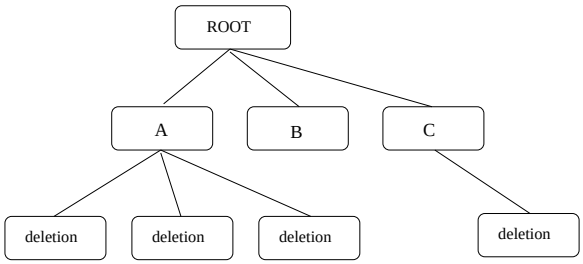

Figure 4: Flat RDR

Figure 4 shows an example of Flat RDR. Flat RDR works as follow: a data case is passed to root. As the root always fires, a dummy conclusion is recorded. After that, the case is passed to *all* the classification rules (the rules of depth 1). A conclusion of a rule is recorded (and overrides the dummy conclusion) if and only if the condition of this rule is satisfied and *none* of its children (its deletion rules) fires. The final classification given to the case is all the conclusions recorded from the classification rules. If there is an undesired conclusion, the human expert will be asked to provide a deletion rule to remove it. The new deletion rule is added as a child to the classification rule that misfires the case. On the other hand, if the expert decides that a classification should be given to this data case, a classification rule (rule of depth 1) will be added. The system will propose the expert

with conditions that will not affect the past performance.

FlatRDR is capable of handling the Multiple Classification Problem, i.e, a data case can be given more than one label. It can be argued that it is possible to decompose a multiple classification problem into a single classification problem. However, such decompositions will make the problem exponentially complex and lose the ontological structures of the domain. In this simulation framework, however, we just apply Flat RDR to the single classification problem.

4.3 Composite Rules

Composite Rules (Crawford, Kay & McCreath 2002*a*, Crawford, Kay & McCreath 2002*b*) is another knowledge representation schema applied in the IEMS (Intelligent Email Sorter) system. Composite Rules use clausal form to represent the knowledge base: the clauses are interpreted as first order decision list where the first clause that explains a query is used. Each clause in the decision list is of the form

$$(R_i) \quad Action \longleftarrow C \text{ unless } C_1, C_2, \ldots$$

where (R_i) is the label of the clause (rule), *Action* is the conclusion of the rule which specifies what needs to be done to the input date (e.g, which folder the email should be placed in the IEMS application), and $C, C_1, C_2, \ldots$ are relational conditions. The knowledge base operates as follow: if C is satisfied with the input data and *none* of the C_i is satisfied, then *Action* is applied to the data and the evaluation stops. Otherwise, i.e if C is not satisfied or any of the exception $C_1, C_2, \ldots$ is satisfied, the input data is passed to the next clause in the decision list. When an action is wrongly applied for an input datum, an exception is added to the current rule and the case is passed to the *next rule*. If none of the rules applies to the input datum, a new clause is added at *the end* of the list. Because of this property, Composite Rules are also termed Decision List (DL). We can see that Composite Rules are very similar to Flat RDR except for the way the top rules are evaluated. It can be argued that Composite Rules are more suitable to single classification.

5 Implementation

The three methods we simulate share the following generic process:

1. Create a default rule (which always gives a dummy conclusion).
2. Accept a new data case.
3. Evaluate the case against the knowledge base.
4. If the case if not correctly classified then the expert is consulted, a primary rule (or rules) will be added to the knowledge base to take care of the new case.
5. Go to Step 2.

This process can be seen in the following pseudo-code.

proc $GenericSim()$
  $KB = \{DEFAUL\_RULE\}$;
  while $StoppingCondition$! = **true** **do**
    $d = GenerateNewDataCase()$;
    $result = Evaluate(KB, d)$;
    if $Incorrect(result, d)$
      then $Update(KB, d, result)$;
    fi
  od

Algorithm 5.1 Generic Simulation Process

To evaluate of data against a knowledge base, we need to know how to evaluate it against a primary rule. Given a data case d and a primary rule R, we can compute the probability that R is satisfied by d based on our analysis in section 3.2. The following pseudo-code shows how this probability (cover) is computed.

proc $cover(Rule, d)$
  if $Rule.Conclusion = d.TrueClass$
    then $cover = 1 - OS$;
    else $cover = \frac{(1-OS)*OG}{(1-OG)} * \frac{P(d.TrueClass)}{1-P(d.TrueClass)}$
  fi

Algorithm 5.2 Computation of cover

Note that $P(d.TrueClass)$ depends on the distribution of the set Input(R). We decide that a case is fired by a rule if the cover above is greater than a random number drawn from the uniform distribution. Furthermore, each KA technique has its own implementation of the $Evaluate()$ and $Update()$ procedures depending how the rules are structured in their knowledge base and how knowledge base revision are done.

6 Result and Discussion

We run the simulation on an artificial domain of 20 disjunctive regions. The number of data cases processed is $20,000$. The OS and OG factors are set to combinations of $\{0.1, 0.2, 0.3, 0.4\}$. The simulation is run 10 times for each technique. Figure 5 shows the average number of knowledge acquisition sections as function of number of cases processed for each combination of overspecialisation and overgeneralisation factors.

The graphs show that Composite Rules have the better convergence rate when the expert has a high level of expertise. In all the cases where $OG < 0.3$, Composite Rules outperform the other methods and shows good convergence. However, when the expert is not doing well, particularly when he tends to overgeneralise, SCRDR outperforms other methods. FlatRDR seems not to do as well as the other methods in all of the tests. However, as we argued before, Flat RDR was designed for Multiple Classification Problem, and therefore these tests do not favour this method. This is a possible explanation for the good results achieved in (Crawford et al. 2002*a*) as in the email domain, the users are able to provide highly

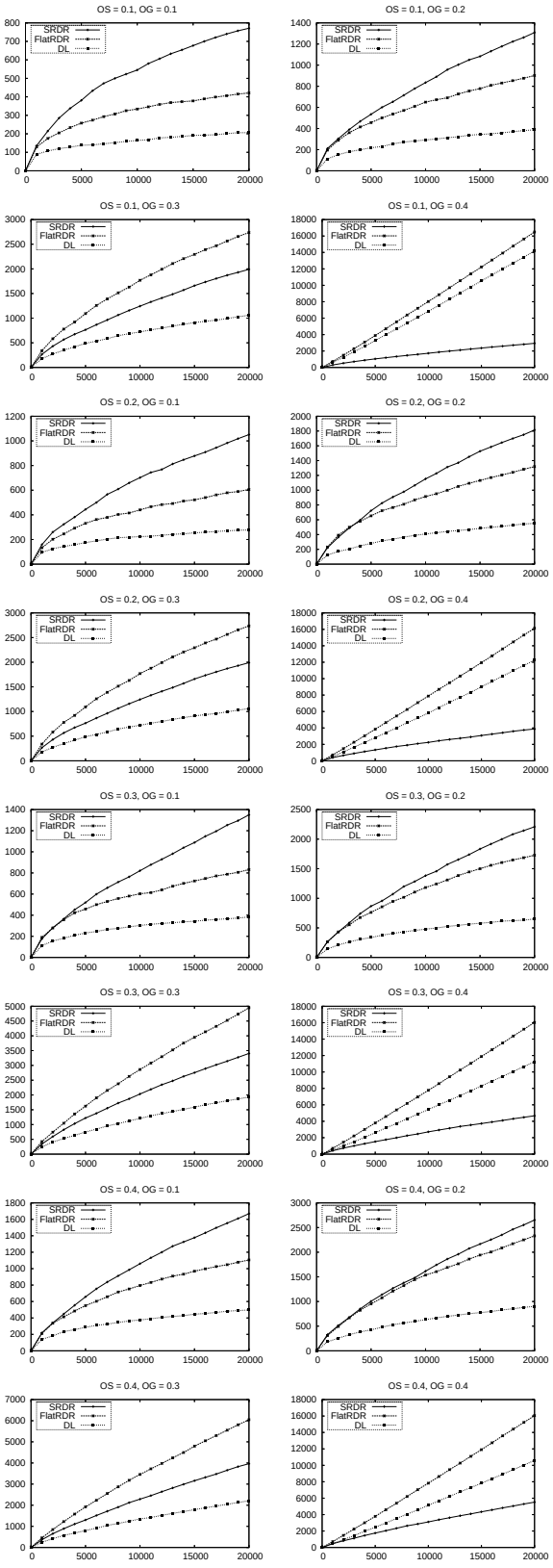

Figure 5: No. of KA sessions vs No. of cases with various expertise levels

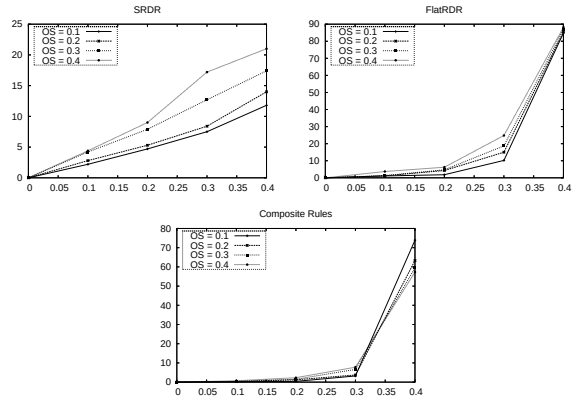

Figure 6: Error vs OG factor

accurate rules. Nevertheless, in an application of a domain where high levels of expertise are not available, SCRDR might be the technique to start with. When sufficient knowledge has been acquired, it is possibly more efficient to switch to another method.

Another interesting feature we can see is that of the two types of the experts' error, overgeneralisation seems to play a more important role in the resulting knowledge base performance. Figure 6 shows the error as a function of *OG* factor for the methods. In each graph, the lines represent different *OS* factors. It is clear that while the lines are similar in each graph, they all tend to increase exponentially. The only method which performs satisfactorily is SRDR as adding a rule in SRDR only affects the knowledge base locally. This finding is particularly helpful in building real life KBS as we now have some measure of a favourite KB. That is, a KB tends to overspecialise is arguably easier to maintain and extend than a KB that tends to overgeneralise.

7 Conclusion and Future Work

The purpose of this paper is twofold. Firstly, we address the problem of evaluating knowledge acquisition methods and its difficulties. We then propose a parameterised framework to quantify the levels of expertise based on data-model theory. Secondly, we apply this framework to simulate three variants of a practically successful incremental knowledge acquisition technique, namely Ripple Down Rules. The simulation shows interesting findings of the relations between levels of expertise and performance of resulting knowledge bases.

At the early presentations of RDR, there were always comments that the size of the KB would explode and reservations about the counter claim that this doesn't seem to occur in practice. A simulation does not answer this question by the results it produces, but rather the results together with the explicit assumptions needed to think more carefully about how the system works.

Simulation can only be used to provide precise quantification in domains where the assumptions behind the simulation are very well founded. In domains

where the assumptions are more speculative, the benefits of building a simulator are usually in developing a better understanding of the problem. For example there could be no claim from the results here that in real applications an expert could be characterised as making a 10% overgeneralisation error which would then result in an error of 0.2% after 20,000 cases. However, for example, in the case of SCRDR one gets a much clearer insight into how a refinement structure must provide fairly reasonable convergence partly independent of the error level. The refinement structure provides for a geometric decrease in overall error which reduces the impact of the actual size of the error. In that sense, this work complements the logic-based analysis in (Cao *et. al* 2004) for convergence of incremental knowledge base construction.

In the future, we would like to look into more realistic representations of the domain. For the time being, the domain structure is just a disjunctive partition. Hierarchical structures (such as taxonomy and/or ontology) will be investigated. It is expected that in a more complex domain, Flat RDR will show a better performance.

We also intend to apply the knowledge acquisition frameworks other than Ripple Down Rules. However, this will be much more problematic as it will often involve modelling the way in which the expert communicates with a knowledge engineer. The starting point that the only errors are overgeneralisation and overspecialisation will remain the same, but developing a reasonable model of the process may prove to be difficult.

References

Beydoun, G. & Hoffmann, A. (2000), 'Incremental acquisition of search knowledge', *Journal of Human-Computer Studies* **52**, 493–530.

Beydoun, G. & Hoffmann, A. (2001), 'Theoretical basis for hierarchical incremental knowledge acquisition', *Journal of Human-Computer Studies* **54**, 407–452.

Blake, C. & Merz, C. (1998), 'UCI repository of machine learning databases'.
*http://www.ics.uci.edu/∼mlearn/

Cao, T., Martin, E. & Compton, P. (2004), On the convergence of incremental knowledge base construction, *in* 'Proceedings of 7th International Conference of Discovery Science', pp. 207–218.

Colomb, R. (1999), 'Representation of propositional expert systems as partial functions', *Artificial Intelligence* **109**(1-2), 187–209.

Compton, P., Preston, P. & Kang, B. (1995), The use of simulated experts in evaluating knowledge acquisition, *in* B. Gaines & M. Musen, eds, '9th Banff KAW Proceeding', pp. 1–12.

Compton, P., Ramadan, Z., Preston, P., Le-Gia, T., Chellen, V. & Mullholland, M. (1998), A trade-off between domain knowledge and problem solving method power, *in* B. Gaines & M. Musen, eds, '11th Banff KAW Proceeding', pp. 1–19.

Corbridge, C. & Shadbolt, N. (1995), Models exposed: an empircial study, *in* '9th Banff KAW Proceeding'.

Crawford, E., Kay, J. & McCreath, E. (2002*a*), Iems the intelligent email sorter, *in* 'Proceedings of the Nineteenth International Conference on Machine Learning', Morgan Kaufmann Publishers Inc., pp. 83–90.

Crawford, E., Kay, J. & McCreath, E. (2002*b*), An intelligent interface for sorting electronic mail, *in* 'Proceedings of the 2002 Conference on Intelligent User Interfaces', ACM, pp. 182–183.

Edwards, G. & Compton, P. (1993), 'Peirs: A pathologist maintained expert system for the interpretation of chemical pathology reports', *Pathology* **25**, 27–34.

Kang, B., Yoshida, K., Motoda, H. & Compton, P. (1997), 'A help desk system with intelligence interface', *Applied Artificial Intelligence* **11**, 611–631.

Martinez-Bejar, R., Ibanez-Cruz, F., Compton, P. & Cao, T. M. (2001), 'An easy-maintenance, reusable approach for building knowledge-based systems: application to landscape assessment', *Expert Systems with Applications* **20**(2), 153–162.

Menzies, T. (1999), 'Knowledge maintennance: The state of art', *Knowledge Engineering Review* **1**, 1–46.

Menzies, T. & Hamelen, F. V. (1999), 'Editorial: Evaluating knowledge engineering techniques', *Journal of Human-Computer Studies* **51(4)**, 715–727.

Richards, D. & Compton, P. (1998), 'Taking up the situated cognition challenge with ripple down rules', *Journal of Human-Computer Studies* **49**, 895–926.

Shadbolt, N. & O'hara, K. (1999), 'The experimental evaluation of knowledge acquisition techniques and methods: history, problem and new directions', *Journal of Human-Computer Studies* **51(4)**, 729–775.

Shiraz, G. & Sammut, C. (1997), Combining knowledge acquisition and machine learning to control dynamic systems, *in* 'Proceedings of the 15th International Joint Conference in Artificial Intelligence (IJCAI'97)', Morgan Kaufmann, pp. 908–913.