Evaluation of Incremental Knowledge Acquisition with Simulated Experts*

Paul Compton and Tri M. Cao

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

Abstract. Evaluation of knowledge acquisition (KA) is difficult in general. In recent times, incremental knowledge acquisition that emphasises direct communication between human experts and systems has been increasingly widely used. However, evaluating incremental KA techniques, like KA in general, has been difficult because of the costs of using human expertise in experimental studies. In this paper, we use a general simulation framework to evaluate Ripple Down Rules (RDR), a successful incremental KA method. We focus on two fundamental aspects of incremental KA: the importance of acquiring domain ontological structures and the usage of cornerstone cases.

1 Introduction

Knowledge acquisition is widely considered as a modelling activity [14,19]. Most of the KA approaches for building knowledge based systems support a domain analysis (including problem analysis and/or problem solving analysis) by the domain experts and the knowledge engineers. This process possibly involves steps like developing a conceptual model of the domain knowledge, distinguishing subtasks to be solved, differentiating types of knowledge to be used in the reasoning process, etc. Eventually, this knowledge engineering approach results in a model of the domain knowledge that can be turn into an operational system manually or automatically.

On the other hand, the incremental KA approach aims to skip the time consuming process of analysing expertise and domain problem by a knowledge engineer [4,6]. It rather allows the experts themselves to communicate more directly their expertise to the system. This communication is usually triggered by real data that the experts encounter in their normal workflow. Over time, a complex system will be gradually developed. As many knowledge based systems are classification systems, from this point on, we focus on classification based systems, even though most of our arguments are easy to adapt to other types of knowledge based systems. We also use the term production systems as a synonym for classification systems.

* A shorter version of this paper has been accepted to the conferrence EKAW 2006.

The following algorithm describes a general incremental knowledge acquisition process.

Algorithm 1. General Incremental KA

```
1. Start with an empty KB
2. Accept a new data case
3. Evaluate the case against the knowledge base (KB).
4. If the result is not correct, an expert is consulted to refine the KB
5. If the overall performance of the KB is satisfactory, then terminate, otherwise go
   to Step 2
```

It is important to note that this scheme largely corresponds to the maintenance phase of any knowledge based system (KBS). The KB processes cases; when its performance is judged unsatisfactory or inadequate, changes are made and the performance of the new KB is validated. RDR is an extreme example of this maintenance model as it starts the maintenance cycle immediately with an empty KB. The first industrial demonstration of this was the PEIRS system, which provided clinical interpretations for reports of pathology testing and had almost 2000 rules built by pathologists [5,13]. Since then, RDR has been applied successfully to a wide range of tasks: control [16], heuristic search [1] and document management [10]. The level of evaluation in these studies varies, but overall they clearly demonstrate very simple and highly efficient knowledge acquisition. There is now significant commercial experience of RDR confirming the efficiency of the approach.

Following the PEIRS example, one company, Pacific Knowledge Systems, supplies tools for pathologist to build systems to provide interpretative comments for medical Chemical Pathology reports. One of their customers now processes up to 14,000 patient reports per day through their 23 RDR knowledge bases with a total of about 16,000 rules, giving very highly patient-specific comments. They have a high level of satisfaction from their general practitioner clients and from the pathologists who keep on building more rules or rather who keep on identifying distinguishing features to provide subtle and clinically valuable comments. A pathologist generally requires less than one day's training and rule addition is a minor addition to their normal duties of checking reports; it takes an average of 77 seconds per rule [8].

Given the success of the knowledge representation scheme and the knowledge revision procedure, it is of interest to investigate the properties of RDR to account for its success and shape its future developments. In this paper, we use a general simulation framework proposed in [7] and developed in [2] to evaluate two interesting features of incremental knowledge acquisition: the usage of supporting data (aka cornerstone cases) in interactions with human experts and the importance of domain ontology acquisition.

The structure of the paper is as follows: in section 2, we sketch a brief description of the simulation framework that is used in the paper. Section 3 describes

Flat Ripple Down Rules (FRDR), the incremental knowledge acquisition method being investigated. In section 4, we experiment with the usage of cornerstone cases in FRDR. The importance of domain ontology acquisition is investigated in section 5, and we conclude in section 6.

2 Simulation Framework

Evaluation of KA tools and methodologies is difficult [11,15]. The essential problem is the cost of human expertise to build a KBS. A solution to this is the use of simulated experts in evaluation studies. A simulated expert is not as creative or wise as a human expert, but it readily allows for repeated control experiments. The simulation framework we use in this paper is described in [2]. In this section, we outline the main features of this framework.

We characterise an expert by two parameters: overspecialisation and overgeneralisation. Overspecialisation is the probability that a definition excludes data which it should cover. Overgeneralisation, on the other hand, is the probability that a definition includes data which it should not cover. This is depicted in Figure 1. In this figure, the human expert tries to capture a target concept by providing the system a rule or rules; however as the expert is not perfect, the defined concept deviates from target concept. The deviation can be quantified through two parameters: overspecilisation and overgeneralisation.

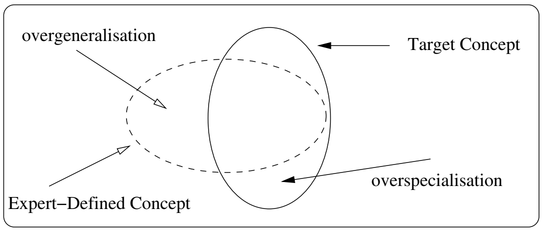


Fig. 1. Overspecialisation and overgeneralisation

In classification based systems, errors of overspecialisation and overgeneralisation are often called false negative and false positive, respectively. These errors not only apply to individual classification rules, but to complex classifiers too. Moreover, they also apply to other aspects of knowledge based system. With a planning system, the KBS has error components 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 ontology 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 this study, we simulate obtaining rules from the expert and so apply these errors at the rule level. A given rule may cover data for which the conclusion is not appropriate; that is, it is too general. Or the rule is too specific and so excludes some data that should be covered. The intuitive response to an overgeneralised rule is to add conditions and for an overspecialised rule to remove conditions. However, whether one does this or corrects the system in some other way depends on the KA tool or method being used.

These characterisations can be used to describe different levels of expertise (for example, experienced experts and trainees). These errors also increase with the difficulty of the domain. Trainees will be associated with higher overgeneralisation and/or overspecialisation errors than experienced experts in the domain. One major problem with previous work that used simulated experts is how to model levels of expertise. For example in [6], levels of expertise are represented by picking various subsets of the full condition. There is no such difficulty in our approach as we model the effects of different levels of expertise by using different combinations of overgeneralisation and overspecialisation.

As mentioned above, the simulation here is restricted to classification. Secondly the domain is assumed to be made up of non-overlapping regions. The minimum number of rules required is therefore the number of regions in the domain. This assumption is made for the sake of simplicity and can easily be relaxed to allow for more complex domains.

Expert effort is often measured by the number of rules created in the knowledge base. We suggest that the number of knowledge acquisition sessions is a better metric for expert effort. With Ripple Down Rules, it is shown in [8] that, in practice, a rule often takes around one minute to be actually encoded. So whether it takes one rule or 5 rules to fix a case is of little importance; the key issue is how to deal with a case that has been misclassified.

3 Flat Ripple Down Rules

The RDR variant we use in this experiment is the flat rule version. The reason behind this choice is that Flat Rule is a simplified version of multiple classification RDR which is used in practical systems, e.g. from Pacific Knowledge Systems. Flat RDR can be considered as a *n-ary* tree of depth two. Each node of the tree is labelled with a primary rule with the following properties:

- The root is a default rule which gives a default dummy conclusion (for example **unknown**).
- The rules in the nodes of depth 1 give a 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 2 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

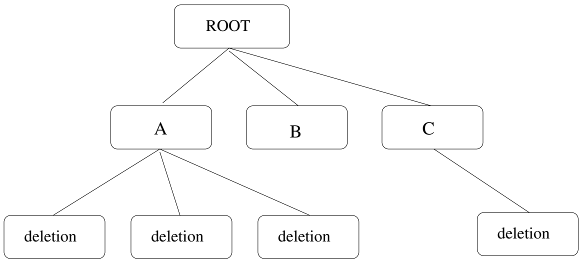


Fig. 2. Flat RDR

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.

FlatRDR is capable of handling the Multiple Classification Problem, i.e, a data case can be classified with more than one label. In this simulation framework, however, we just apply Flat RDR to the single classification problem.

Although we have described Flat RDR with the RDR tree and refinement structure, it also corresponds to the general case of simple classification, where new rules are added or rules are refined when incorrect.

4 Cornerstone Cases

Cornerstone cases are data cases that trigger the creation of new rules. One of the hallmark features of RDR is the employment of cornerstone cases. They serve two purposes:

– as a means of maintaining past performance by imposing consistency
– as a guide to help the experts make up the new rules.

The cornerstone cases are used in the following manner: when a data case is misclassified by the system, an expert is consulted and asked to provide a new rule (or rules) to deal with this case. The new rule then is evaluated against all the cornerstone cases stored in the system. If any of the cornerstone cases is affected by the new rule, the expert is asked to refine it. Only when the system confirms that the new rule does not affect any of the cornerstone cases then it is added to the knowledge base, and the current data case becomes the new rule's cornerstone. In practice, the expert might decide to allow the rule to apply an existing cornerstone case, but this evaluation excludes this.

The first question for the evaluation is the importance of cornerstone cases. Or more generally, what is the importance of validating performance against test data after modifying a KBS.

4.1 Experimental Settings

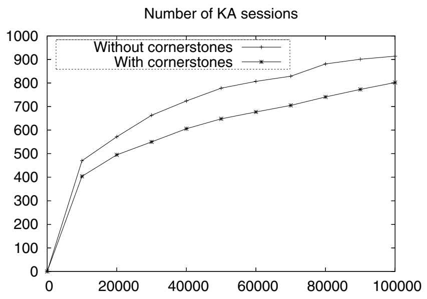
The simulations here are restricted to two artibrary levels of expertise:

- 'Good' Expert: the human expert is characterised by $(0.2, 0.2)$, i.e a rule made by this expert will include cases that it should not with probability 0.2 and exclude cases that it should cover also with probability 0.2.
- 'Average' Expert: the human expert is characterised by $(0.3, 0.3)$.

Our naming of these levels of expertise is arbitrary; our intention is simply to distinguish higher and lower levels of expertise. With each level of expertise, we run the simulation with two options: with or without cornerstone cases. The simulation is run with 100000 data cases from a domain of 20 regions, and the number of required KA sessions is recorded.

4.2 Result and Discussion

The following figures show the number of KA acquisition sessions as a function of data cases presented to the system. As a KA is required each time a data case is misclassified, the slope of this graph can also be considered as the error rate for the acquired system.



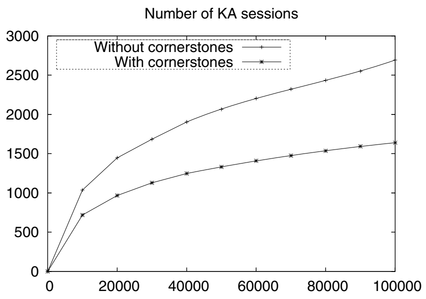


Fig. 3. Good and Average Expertise Levels

It can be seen from the graphs that when a good level of expertise is available, there is not much difference in the performance of the acquired knowledge base whether or not cornerstone cases are employed. However, when the available expertise is average, the system with cornerstone cases clearly outperforms the one without, in terms of the number of KA sessions (or error rate). In a KA session with the system that uses cornerstone cases, the expert is usually asked to create more primary rules. However, this is perfectly acceptable since the number of KA sessions is a better measure of human experts' time than the number of primary rules.

5 Domain Ontology Acquisition

In recent years, the use of explicit ontologies in knowledge based systems has been intensively investigated [18,9,12,17]. Heuristic classification was first introduced by [3] and remains a popular problem solving method (PSM). It can be understood as a PSM using a very simple ontological structure of intermediate conclusions. It is comprised of three main phases:

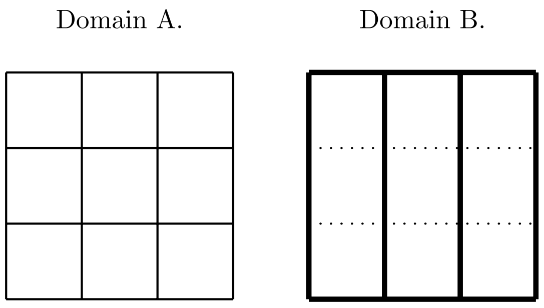
- abstraction from a concrete, particular problem description to a problem class definition that applies to
- heuristic match of a principal solution to the problem class
- refinement of the principal solution to a concrete solution for the concrete problem

This process can be seen in the following figure

$$\begin{tikzcd}
\text{Problem Class} \arrow[r, "\text{match}"'] & \text{Principle Solution} \arrow[d] \\
\text{Concrete Problem} \arrow[u] & \text{Refinement Solution}
\end{tikzcd}$$

In practice, it is not always the case that all three phases of heuristic classification are employed. The example we look at in the next subsection will show how a simple taxonomy is used with classification systems.

5.1 Example



We look at two domain structures as in the picture above. The task here is to acquire a classifier for this domain from human experts. There are nine elementary classifications as shown in case A. In case B, however, we assume that there is a known taxonomy of classifications: the domain is divided into three general classes and each general class contains three elementary classifications. This taxonomy can be considered as a very simple ontology. We now describe how this explicit taxonomy of classification is used in a classification system and how we evaluate its usage.

In case A, the classifier produces one of the nine classifications. Revision of the knowledge base when a data case is misclassified is done similarly as in Section 3. On the other hand, in case B, classification is done in a two-step process. First, the classifier assigns a general class (from three classes in this particular example) to the input data. After that, the data is passed to a second sub-classifier which (based on the general class assigned) gives the sub-classification associated with this case. When there is a misclassification, the classifier (or classifiers) will be revised. As a consequence, one can argue that, revision in this case is likely to be more complex than that in case A. However, in our experiments, we still count each revision to deal with a case as a KA session.

5.2 Experiment Settings

The simulations here are restricted to two arbitrary levels of expertise:

- 'Average' Expert: the human expert is characterised by $(0.3, 0.3)$,
- 'Bad' Expert: the human expert is characterised by $(0.4, 0.4)$.

and two domain structures

- (A) the domain is composed of 25 non-overlapping regions
- (B) the domain is composed of 5 non-overlapping regions, and each region, in turn, is composed of 5 sub-regions.

Again, the naming of the levels of expertise is arbitratry. The simulation is run with 100000 data cases and the number of required KA sessions is recorded.

5.3 Result and Discussion

The following figure shows the number of KA sessions as a function of number of data cases presented to the system. The result is surprising because even with a fixed taxonomy in the experiments, a difference in expertise level can lead to such a difference in the performance of the acquired knowledge bases. While there is a reasonable expertise available, the classifier with a domain taxonomy clearly

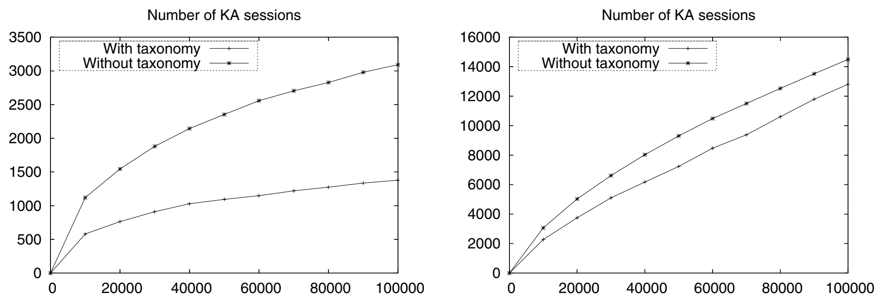


Fig. 4. Average and Bad Expertise Levels

outperform the one without. However, when the level of available expertise is poor, performance is similar so it might be better not to use the domain ontology because knowledge acquisition is simpler.

6 Conclusion

In this paper, we use the simulation framework developed in [2] to investigate two interesting aspects of incremental knowledge acquisition, namely, the usage of supporting data cases and explicit domain ontology. We do not claim that our model accurately reflects the real life situation, or our results quantitatively apply to the a real knowledge based system, the simulation still shows interesting observations.

We observe that the use of cornerstone cases in Ripple Down Rules system shows a real improvement of the knowledge base performance. While the expert has work a bit more at each knowledge acquisition session, the number of KA sessions will be less over time. In particular, when a high level of expertise is not available, the use of cornerstone cases significantly improves the experts' performance.

The second observation is that explicit domain ontology brings significant improvement in the resulting system's performance if high levels of expertise are available. However, explicit ontologies do not have as much positive effect when the domain is dynamic (due to its changing nature, or unestablished tacit knowledge).

Aspects of these conclusions are entirely obvious and be accepted by all: that validation and ontologies are both useful. However, the methodology also raises the question that as we move into less well defined area relating to personal and business preferences, validation becomes more critical while perhaps ontologies are less valuable.

In the future, we would like to investigate other aspects of evaluating KA: more complex domain structure or in multiple experts settings.

Acknowledgements

This work is supported by CRC for Smart Internet Technology, Australia.

References

1. G. Beydoun and A. Hoffmann. Incremental acquisition of search knowledge. *Journal of Human-Computer Studies*, 52:493–530, 2000.
2. T. Cao and P. Compton. A simulation framework for knowledge acquisition evaluation. In *Proceedings of 28th Australasian Computer Science Conference*, pages 353–361, 2005.
3. W. J. Clancey. Heuristic classification. *Artificial Intelligence*, 27:289–350, 1985.
4. P. Compton, G. Edwards, A. Srinivasan, P. Malor, P. Preston, B. Kang, and L. Lazarus. Ripple-down-rules: Turning knowledge acquisition into knowledge maintenance. *Artificial Intelligence in Medicine*, 4:47–59, 1992.

5. P. Compton and R. Jansen. A philosophical basis for knowledge acquisition. *Knowledge Acquisition*, 2:241–257, 1990.
6. P. Compton, P. Preston, and B. Kang. The use of simulated experts in evaluating knowledge acquisition. In B. Gaines and M. Musen, editors, *9th Banff KAW Proceeding*, pages 1–12, 1995.
7. Paul Compton. Simulating expertise. In *PKAW*, pages 51–70, 2000.
8. Paul Compton, Lindsay Peters, Glenn Edwards, and Tim Lavers. Experience with ripple-down rules. In *Applications and Innovations in Intelligent Systems*, pages 109–121, 2005.
9. Asuncion Gomez-Perez. Evaluation of ontologies. *Int. J. Intelligent Systems*, 16:391–409, 2001.
10. B. Kang, K. Yoshida, H. Motoda, and P. Compton. A help desk system with intelligence interface. *Applied Artificial Intelligence*, 11:611–631, 1997.
11. T. Menzies and F. Van Hamelen. Editorial: Evaluating knowledge engineering techniques. *Journal of Human-Computer Studies*, 51(4):715–727, 1999.
12. Natalya Fridman Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W. Fergerson, and Mark A. Musen. Creating semantic web contents with protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.
13. P. Preston, G. Edwards, and P. Compton. A 2000 rule expert system without a knowledge engineer. In B. Gaines and M. Musen, editors, *8th Banff KAW Proceeding*, 1994.
14. Guus Schreiber, Bob J. Wielinga, Robert de Hoog, Hans Akkermans, and Walter Van de Velde. Commonkads: A comprehensive methodology for kbs development. *IEEE Expert*, 9(6):28–37, 1994.
15. N. Shadbolt and K. O'Hara. The experimental evaluation of knowledge acquisition techniques and methods: history, problem and new directions. *Journal of Human-Computer Studies*, 51(4):729–775, 1999.
16. G. Shiraz and C. Sammut. Combining knowledge acquisition and machine learning to control dynamic systems. In *Proceedings of the 15th International Joint Conference in Artificial Intelligence (IJCAI'97)*, pages 908–913. Morgan Kaufmann, 1997.
17. York Sure, Asunción Gómez-Pérez, Walter Daelemans, Marie-Laure Reinberger, Nicola Guarino, and Natalya Fridman Noy. Why evaluate ontology technologies? because it works!. *IEEE Intelligent Systems*, 19(4):74–81, 2004.
18. G. van Heijst, A. Th. Schreiber, and B. J. Wielinga. Using explicit ontologies in kbs development. *Journal of Human-Computer Studies*, 45:183–292, 1997.
19. Gertjan van Heijst, Peter Terpstra, Bob J. Wielinga, and Nigel Shadbolt. Using generalised directive models in knowledge acquisition. In *EKAW*, pages 112–132, 1992.