Ripple-Down Rules

Impact Summary

Ripple-Down Rules (RDR) allow end-users to build their own knowledge bases themselves, without needing a knowledge engineer, or needing to become a knowledge engineer. Rules are added to a knowledge base while it is already in use, whenever a case is noted for which the system has not made an appropriate decision. Because of its ease of use, RDR has had significant industry uptake with at least nine companies, including IBM, providing RDR technology or RDR-based services for different applications across a range of industries [1].

One of these, Pacific Knowledge Systems (PKS) (https://pks.com.au) provides RDR technology for chemical pathology laboratories to add diagnostic and management advice to laboratory reports. There are over 800 PKS RDR knowledge bases in use worldwide, some with over 10,000 rules. These have been developed largely by chemical pathologists themselves, after minimal training, and the data shows that it only takes a minute or two on average to add a rule [2,3]. These 800 plus knowledge bases are all separately developed, rather than the same knowledge base deployed in many places, and so represent a very significant uptake of AI technology in medicine. PKS was floated on the Australian Stock Exchange in early 2019.

Underpinning research

GARVAN-ES1, an expert system interpreting thyroid laboratory results, was reported as one of the first four medical expert systems in routine clinical use. As such it was carefully monitored and maintained. It became obvious that the "knowledge engineering bottleneck", is not because experts have difficulty in reporting on their mental processes, rather what they do is identify features in the data which justify their conclusion as opposed to other possible conclusions in the context. This led to the key foundational paper for RDR, arguing that there is a strong philosophical basis for the idea that experts always provide knowledge in a specific context [4].

There are various versions of RDR, but in summary the essential features of any RDR as described in [1] are:

- Rules are added to deal with specific cases to either replace an incorrect conclusion given for a case, or to add a conclusion. Rules are only added, never removed or edited
- "Cornerstone cases" are cases for which a rule has been added. If a new rule gives a different conclusion for a stored cornerstone case, the cornerstone is shown to the user. The principle of rationality requires that the user must be able to identify some feature that distinguishes the case from the cornerstone case or accept that they should have the same conclusion. Nearly always a user will add extra conditions to the rule to exclude the cornerstone.
- RDR are linked production rules. Every rule not only specifies a rule action, such as asserting a fact, but also specifies which rule will be evaluated next, depending on whether it evaluates true or false. This completely determines the rule evaluation sequence so there is no need for an inference engine conflict resolution strategy as required with conventional production rules.
- A rule reached by an if-true link can either replace the conclusion given or add an extra conclusion, as specified by the user (or by the type of RDR used).

There are further extensions to RDR that allow for repeat inference for tasks such as configuration. A key idea is that during inference RDR rules should only assert facts, not retract them. If incorrect conclusion is asserted it should not be retracted in some later inference cycle, rather a rule should be added to prevent the incorrect conclusion being asserted in the first place [1].

The first RDR system in routine use was the PEIRS system in Chemical Pathology at St Vincent's hospital. It was a large system, with 2000 rules but was based on single-classification RDR [5]. The most widely used form of RDR is now multiple-classification RDR (MCRDR), able to provide multiple independent conclusions for a set of data which was developed by Byeong Kang as part of his PhD. Data on PKS experience with its version of MCRDR, shows that end-users, gradually over time, can build systems with many thousands of rule, taking only a minute or two per rule [2, 3].

There have been a range of RDR algorithms developed for different types of applications, but the notion of linked production rules outlined above, provides a generalisation of these various algorithms. Other RDR research includes:

- An RDR representation has been used in a number of machine learning algorithms. RIDOR from the WEKA is very widely used, and is based on Brian Gaines Induct-RDR.

- There has been a range of research on prudence and credentials with RDR to produce RDR systems which are able to recognise when a case is outside their range of experience, like a medical trainee.

- There has also been research, where an RDR system can generalise from what it has learned.

- Of particular significance, there has considerable research on using an RDR system as a "wrapper" around another system, .e.g a system built by machine learning. That is, rules are added to correct errors from an underlying system, or to it customise the underlying system to a particular domain. This can result in a huge improvement [6].

The RDR book provides brief outlines of the known industry applications and significant applications validated in a research setting [1]. It also argues that RDR can be more suitable than machine learning for many problems. Machine learning requires sufficient cases to be reliably labelled with all the specific classifications that it is important to learn. Such data is rarer than might be expected and can be costly to prepare. Given the simplicity and low cost of adding RDR rules an RDR system can be built as part of the labelling process. This not only produces a working system so that learning is no longer required, but avoids issues with rare classes and unbalanced data and ensures consistent labelling to whatever degree of refinement is required.

References

[1] Compton, P. and Kang B.H (2021) *Ripple-Down Rules: the Alternative to Machine Learning*, CRC Press (Taylor and Francis)

[2] Compton, P. (2013). "Situated cognition and knowledge acquisition research." *International Journal of Human-Computer Studies* **71**: 184-190.

(This was one of 11 invited papers for a special issue of the journal to commemorate 25 years of knowledge acquisition research)

[3] Compton, P., L. Peters, T. Lavers and Y. Kim (2011). *"*Experience with long-term knowledge acquisition*." Proceedings of the Sixth International Conference on Knowledge Capture, KCAP 2011*, Banff, ACM. 49-56
(a version of this paper with minor corrections is at https://pks.com.au/wp-content/uploads/2015/03/WhitePaperExperiencewithKnowledgeSystemsPKS.pdf).

[4] Compton, P. and R. Jansen (1990). "A Philosophical Basis for Knowledge Acquisition." *Knowledge Acquisition* **2**: 241-257.

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

[6] Ho, V. H., P. Compton, B. Benatallah, J. Vayssière, L. Menzel and H. Vogler (2009). "An Incremental Knowledge Acquisition Method for Improving Duplicate Invoice Detection." *Proceedings of the 25th IEEE International Conference on Data Engineering, ICDE 2009*, Shanghai, IEEE. 1415-1418