Knowledge Acquisition for Case-Based Reasoning Systems

Christopher K. Riesbeck
Dept. of Computer Science
Yale University
New Haven, CT 06520

Cognitive Systems, Inc.
New Haven, CT 06511

Abstract

Case-based reasoning (CBR) is a simple idea: solve new problems by adapting old solutions to similar problems. The CBR approach offers several potential advantages over rule-based reasoning: rules are not combined blindly in a search for solutions, solutions can be explained in terms of concrete examples, and performance can improve automatically as new problems are solved and added to the case library.

Moving CBR from the university research environment to the real world requires smooth interfaces for getting knowledge from experts. We describe the basic elements of an interface for acquiring three basic bodies of knowledge that any case-based reasoner requires: the case library of problems and their solutions, the analysis rules that flesh out input problem specifications so that relevant cases can be retrieved, and the adaptation rules that adjust old solutions to fit new problems.

Introduction

Case-based reasoning means reasoning from prior examples. A case-based reasoning system (CBRS) has a case library, containing 100s, 1000s or more cases. Each case describes a problem and a solution to that problem. The reasoner solves new problems by adapting relevant cases from the library.

Case-based reasoning is an alternative to rule-based reasoning. A rule-based reasoner has a large library of rules of the form, "IF A THEN B." These are chained together in various combinations to solve problems. A rule-based system will be flexible and produce nearly optimal answers, but it will be slow and prone to error. A case-based system will be restricted to variations on known situations and produce approximate answers, but it will be quick and its answers will be grounded in actual experience. In very limited domains, the trade-offs favor the rule-based reasoner, but the balance changes as domains become more realistically complex. Realistic domains involve a large number of number of rules, many of which are quite difficult to formalize properly, linked into long, tenuous chains of reasoning. With enough pre-stored solutions in a CBRS, there is almost always short path between the input case and the retrieved solution.

One important potential advantage of case-based reasoning is that human expertise seems more like a library of past experience than a set of rules. Hence cases should better support knowledge transfer (communication of expertise from domain experts to system) and explanation (justification of solution from system to domain experts). To increase the reasoner's knowledge, an expert would add more cases. To explain why it came to the conclusions it did, the reasoner would show the cases from which it built its solution.

To make the utility of the case-based approach clear, consider a case-based reasoner that only retrieved cases similar to a given situation, but did no reasoning at all. For example, a tax advisor that showed me examples of tax forms filled out by people in circumstances very similar to mine would be very helpful, even though it didn't try to do my taxes for me. A tax expert could extend the system simply by adding more examples. The system in this case serves as a fairly direct conduit, tranferring knowledge from expert to user.

This is the ideal situation, but there is one major problem that has to be solved when building a case library: indexing the cases so that the right cases are retrieved. The features that make one case similar to another are usually not explicitly in the input data, but are inferred using domain-specific knowledge. The domain expert has to add this knowledge to the system along with the cases.

Based on basic research in case-based reasoning [1], Cognitive Systems Inc. is developing a case-based reasoning shell whose primary function is to help a domain expert enter and organize a case library. In this paper we will describe the kinds of features that have been found useful for knowledge acquisition in a case-based reasoner.

A Case-Based Reasoning Shell

One way to look at a CBRS is to contrast it with a database management system (DBMS). The cases are like records in the database. (In fact, in application areas where databases exist, e.g., financial domains, cases really *are* records.) In a standard DBMS, records are retrieved with queries, such as "List all employee records where the salary field is over $50,000 and the position field is not 'manager'." The DBMS retrieves all records that satisfy the constraints of the query. To do this, a database administrator in advance has organized the database, splitting different kinds of information into different files, and telling the DBMS to create indices for certain fields of certain records. For example, an employee database might be split into a file of employee names and ID numbers, indexed by names, a file of personal information about each employee, e.g., employee ID, home address, age, and so on, indexed by employee ID, and a file of payroll information, also indexed by ID. With modern query languages, the user doesn't have to know how information is actually split up, although these choices will make some queries more efficient than others.

A case-based reasoning system differs from a DBMS in two fundamental ways, First, a CBRS query is simply a

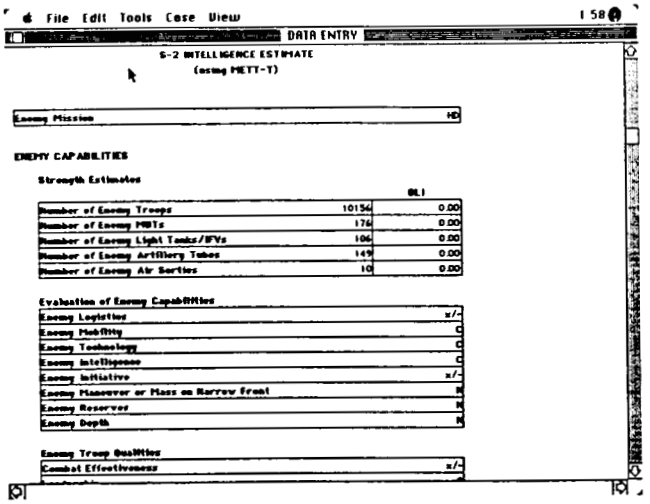

Figure 1: On-Screen Input Form ©Cognitive Systems, Inc. 1988

partially filled-in record, e.g., a tax form with personal and income information filled in, a loan application with the loan request and income information filled in, or a battle situation assessment form. The query is not a pattern of constraints, such as "greater than $50,000", but a concrete description of some current situation. There is no query language to learn.

Second, the CBRS retrieves cases by *best partial match*. Retrieved cases usually do not match the input exactly. Instead, they are the cases that are closest to the input query, based on domain-specific information about what matters and what doesn't. In a Normal DBMS, records either match the query pattern or they don't. If a user wants to find records *similar* to a particular case, the user has to form a query pattern, replacing particular values with ranges of possible values. The user has to know what's in the database, what extra fields to calculate, how to replace particular input values with ranges, and so on. Example-based interfaces, such as Query-by-Example [2], make it easier to generate query patterns, but they still do not allow a user to simply enter a description of the current situation and let the system take care of everything else.

To use a CBRS, a user enters the current case, by filling out an on-screen form that, ideally, mimics exactly forms the user is already familiar with. Figure 1 shows an example of a battlefield intelligence estimate form that can be filled in on-screen in Cognitive System's shell-based battlefield advisor demo.

The CBRS application then retrieves forms describing similar cases. These retrieved forms contain additional fields, called *output fields*, indicating actions taken and outcomes observed in those previous situations. Armed with these exemplars, the user can make an intelligent choice about what to do in the current situation, and, optionally, add the new case to the library for future reference.

Knowledge Acquisition

To determine best matches, a CBRS has to know how to

- infer values for internal fields that a user would not be expected to input, e.g., the debt to income ratio in a loan application, or the attacker to defender ratio in a battle situation
- relate different field values to each other, e.g., incomes fall into brackets for tax purposes, and a prepared defense is closer to a fortified defense than it is to a hasty defense in a battle situation
- rank different fields as more or less important in determining best matches, e.g., income is more important than name for tax advice purposes, and strength ratios are more important than absolute sizes in assessing battles

The job of the Domain Expert Interface component of the Case-Based Reasoning Shell is to enable a domain expert, with little or no programming knowledge, to add these three kinds of knowledge, plus cases, of course, to form a *case library* that, in conjunction with the Shell, forms a CBRS application usable by end users.

The Domain Expert Interface, by necessity, is somewhat more complicated than the end user interface, but, by using ideas from DBMS interfaces and taking advantage of the concrete, real-world-based nature of case-based reasoning, it is still possible to have an interface that is much simpler and easier to use than those commonly found in rule-based expert systems.

For example, adding new cases to the case library is simply a matter of filling out the same forms that the end user sees. The only difference is that the domain expert also fills in the output fields. For example, in the battle assessment domain, the domain expert would fill in not only the initial battle situation, but also the battle outcomes, post-battle analyses of why things happened as they did, graphical annotations, and so on. If an on-line database already exists for a domain, the shell can convert the database records to cases, so that the domain expert just has to add any output fields not included in the original database, and organize the cases, as described below. For example, the Quantified Judgment Model (QJM) battle database, developed at Data Memory Systems Incorporated, was used to initialize the case library for Cognitive Systems' battlefield advisor.

Cognitive Systems' CBR Shell provides the domain expert with two interfaces to assign relative importances to fields in a case. One interface allows the expert to "color" the fields of the input form, where each color represents a different level of importance. The other interface, present when different cases are being compared on-screen, lists fields in order or importance, and allows the expert to drag fields up or down, to fine-tune the relative rankings. Furthermore, the Shell can assign an initial set of importance levels to fields, by looking at which cases have similar values in their output fields. Our initial experience suggests that field importances are hard for experts to determine, and not as robust for determining case similarities as the other two kinds of information described next. The initial values assigned by the Shell are usually best left as is, except in special circumstances.

To specify how different field values relate to each other, e.g., to allow the expert to say that a prepared de-

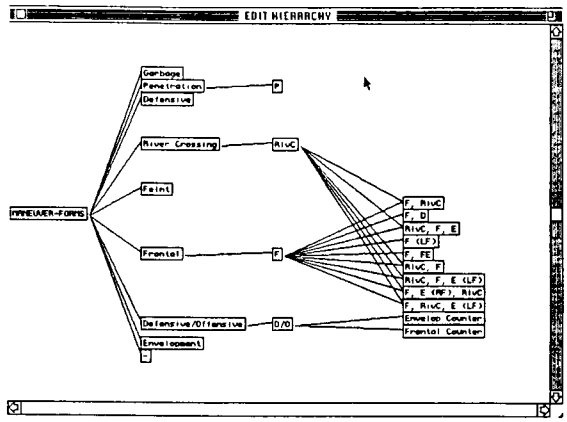

Figure 2: Abstraction Editor Screen ©Cognitive Systems, Inc. 1988

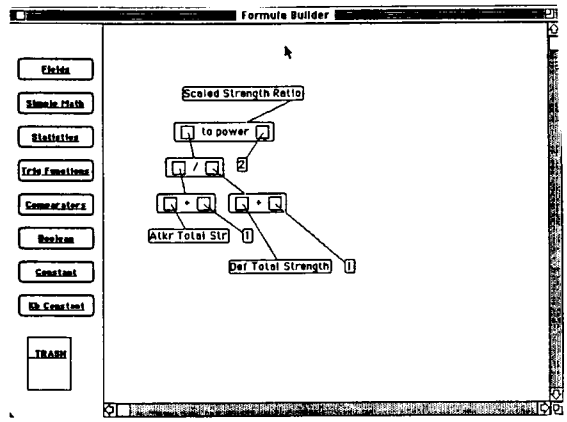

Figure 3: Formula Editor Screen ©Cognitive Systems, Inc. 1988

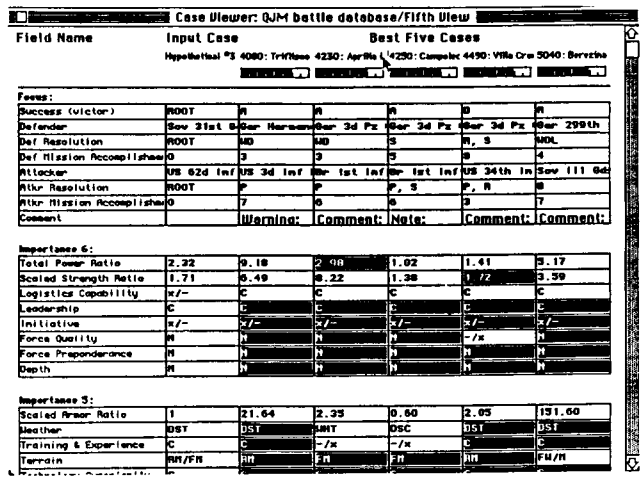

Figure 4: Best Match Comparison Screen ©Cognitive Systems, Inc. 1988

fense is closer to a fortified defense than to a hasty one, the expert uses a graphical *abstraction hierarchy editor*, similar to those found in other AI shells such as KEE and ART [3]. Each field in a form has its own hierarchy of values, since the values, especially if they are numbers, might mean different things in different fields. The hierarchy is initialized to a simple list of all the values seen in that field in the entire case library. The domain expert groups these values together to form the hierarchy. Thus, in the field for defensive posture, the expert might group prepared and fortified defenses together as strong defenses, and group strong and hasty defenses together as defenses. The closer two values are grouped, the better they are considered to match when the CBRS is looking for similar cases. Figure 2 shows an abstraction hierarchy screen for Cognitive's battlefield advisor.

Finally, to specify derived fields, e.g., to create a field that is the attacker to defender strength ratio, the CBR shell uses a graphic formula building interface similar to that found in database systems such as Double Helix [4]. The expert selects various fields from the case form and links them to arithmetic, comparison, and other kinds of operation icons. Such an interface avoids problems with syntactic errors, and lets the domain expert know what options are available at any time for constructing a formula. Figure 3 shows a formula screen for Cognitive's battlefield advisor.

The usefulness of a CBRS of course depends on how well it actually does in find similar cases. Therefore, the center of interaction in the Domain Expert Interface is a screen that displays, for a given test input case, the five best matches that currently exist, given the current importances, hierarchies, and derived fields. The display uses a "compare and contrast" columnar format listing the output fields, followed by the fields ranked as most important. If the expert sees cases being retrieved that he or she considers very dissimilar to the input case, the expert can tell from this display if truly similar cases are not matching because certain field values are poorly grouped in their hierarchies, or if an important combination of values is not being calculated, or if some field of information is simply missing from the database. Figure 4 shows a compare and contrast screen for Cognitive's battlefield advisor.

Conclusions

This paper has described the basic features of a knowledge acquisition interface for a case-based reasoning shell. The shell uses a form-filling metaphor for case entry, and a graded match mechanism for displaying case similarity. A domain expert organizes the library by specifying what fields in a form are important, how well different values of a field match each other, and what additional derived fields have to be calculated to capture important non-input features. The result of the domain expert's efforts is a case-based reasoning application that can take an input situation and retrieve relevant cases from the case library to assist a human decision maker or problem solver.

Acknowledgments

Basic research on case-based reasoning was funded in part by the Air Force Office of Scientific Research under con-

tract F49620-82-K-0010. Application of Cognitive System's Case-Based Reasoning Shell to the battlefield assessment domain is being funded in part by the Defense Advanced Research Projects Agency. The battlefield advisor was built with Cognitive Systems' case-based reasoning shell, using Data Memory Systems' Quantified Judgment Model battle database. The screens shown in the figures are copyrighted by Cognitive Systems, Inc.

References

1. Kolodner, J., editor, *Proceedings of the First Case-Based Reasoning Workshop*, Morgan Kaufmann Publishers, Inc., Los Altos, CA., 1988.
2. Ullman, J. D., *Principles of Database Systems, Computer Software Engineering*, Computer Science Press, Rockville, MD, second edition, 1982.
3. Gevarter, W., The nature and evaluation of commercial expert system building tools, *IEEE Computer*, 20(5), May 1987, pages 24–41.
4. Hirschberg, G., Double helix or nothing, part 1, *MacUser*, 4(4), April 1988, pages 108–116.