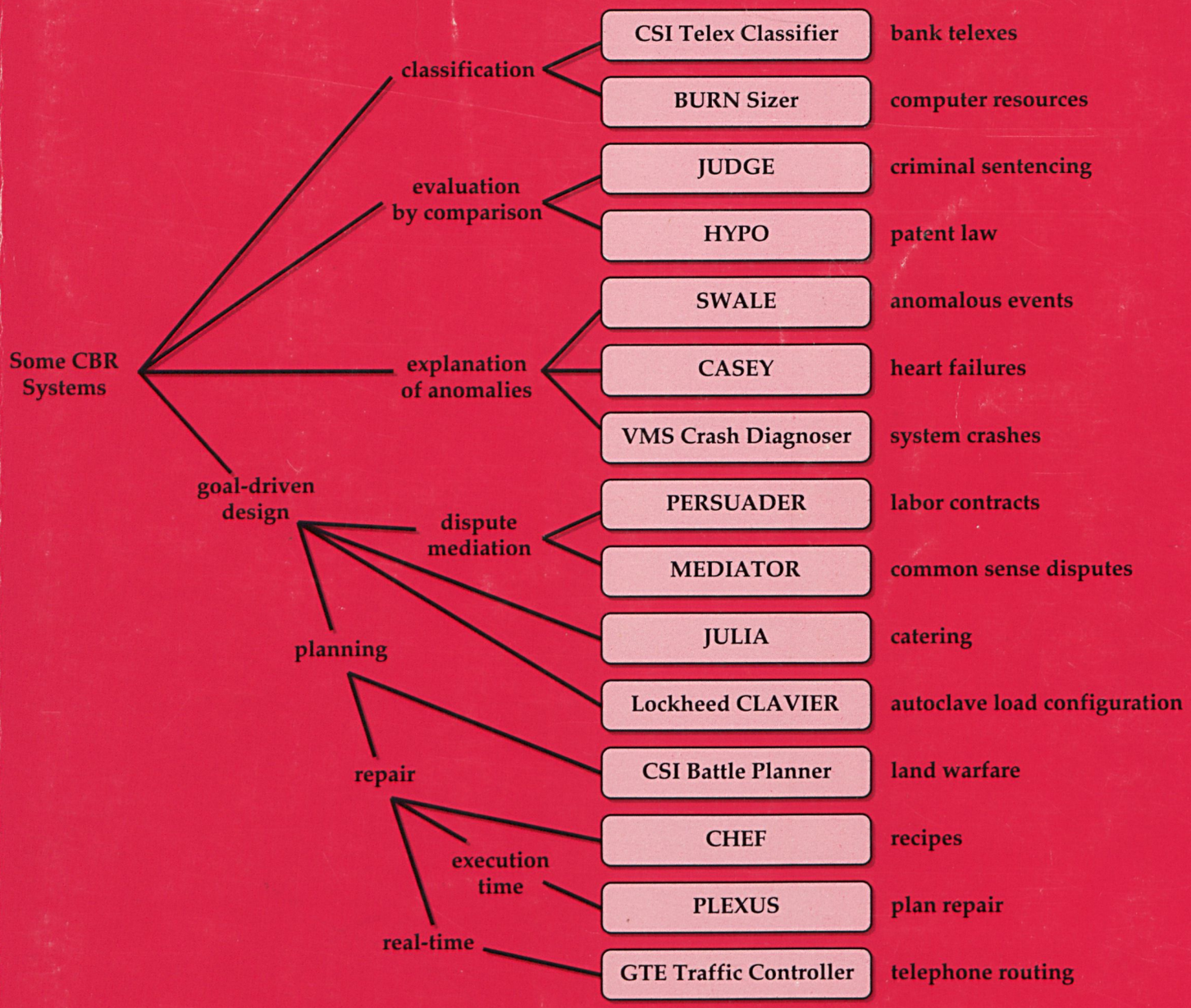
Proceedings:

Case-Based Reasoning Workshop



May 1989

Sponsored by:

DARPA

Defense Advanced Research Projects Agency
Information Science and Technology Office

CASE-BASED REASONING

from

DARPA: Machine Learning Program Plan[1]

INTRODUCTION

There is mounting evidence that human experts rely heavily on memory of past cases when solving problems in domains such as law, mathematics, design, and strategic planning. Thus, it seems natural to exploit this idea in constructing AI systems. This is the focus of systems using *case-based reasoning*; it constitutes a fifth major paradigm of machine learning research. A related approach is that of reasoning by analogy.

In case-based reasoning ("CBR"), one uses memory of relevant "past" cases to interpret or to solve a new problem case. Rather than creating a solution from scratch, a reasoner using case-based reasoning recalls cases similar to its current problem situation and solves or interprets a problem by reasoning with past solutions and interpretations. A reasoner using case-based reasoning can derive shortcuts and anticipate problems in new situations that might arise by having previously spotted and dealt with them. This can lead to improvement in the quality and efficiency of the reasoning.

Case-based reasoning as a learning paradigm has several advantages. First, there are several performance enhancements it provides for its associated performance element: shortcuts in reasoning, the capability of avoiding past errors; the capability of anticipating and therefore avoiding other previously made mistakes, the capability of focusing in on the most important parts of a problem first. Second, learning can be fairly uncomplicated. CBR doesn't require a causal model or a deep understanding of the domain, though either of these provides better performance. A third advantage is that individual or generalized cases can also serve as explanations; these explanations are trivial to generate, and probably more satisfactory, than the chains generated by expert systems. Connectionist systems, which can also learn to generalize from existing databases of previous decisions, cannot easily generate explanations at all. Fourth, case-based reasoning ought to be scalable. The biggest bottleneck is in choosing the best cases to reason with; this potentially massive search problem is handled currently through indexing. While the choice of indices still needs to be better understood, it is possible that the use of parallel implementations make this problem doable in real time. Fifth, the knowledge acquisition bottleneck is much easier in case-based reasoning than for other learning methods, which, in general need to have large amounts of knowledge available before learning processes can be helpful. The reason for this is that much of the knowledge needed for case-based reasoning is in the form of cases. Cases require a minimum of debugging of the interactions between them (unlike rules). Thus, initial knowledge acquisition can be "rote". In addition, many domains have existing case bases (e.g., medicine, law, mathematics, military planning, design) that can be used to "seed" a case-based system.

---

[1]DARPA gratefully acknowledges the efforts of Edwina L. Rissland (University of Massachusetts) who guided the drafting of this section, and of Janet Kolodner (Georgia Institute of Technology) and David Waltz (Thinking Machines, Inc.) who helped draft it.

With all of its advantages, there are still problems that need to be worked out for case-based reasoning to be feasible. Of primary concern are three problems: (1) choice of indexes to be used for organizing cases in the memory, (2) methods for choosing the most relevant cases from memory at reasoning time, and (3) general formulations of adaptation heuristics used to modify previous cases or their solutions to fit the new case. There are a variety of approaches being studied to solve these problems. For problem 1, these include the creation of a vocabulary for representing problem solving situations, investigation of memory structures, and the use of explanation-based generalization to choose salient features of a case. For problem 2, they include nearest neighbor analyses, statistical weighting methods, weight of evidence methods, and preference heuristics. For problem 3, they include generalize and refine heuristics and designation of particular kinds of dimensions that cases might have and general rules for each.

There are two major types of case-based reasoning: interpretive/classification CBR (often called precedent-based CBR) and problem solving CBR. In interpretive or classification CBR, such as found in strategic planning or legal reasoning, one argues that a new situation should or should not be treated like a past one based on similarities or differences with the past cases whose interpretation or classification has been settled. A typical interpretive CBR task is to generate a pro's and con's analysis of why a new fact situation should or should not be classified in particular way, for instance, that certain assets received by a taxpayer constitute "income" or that certain troop or resource movements are indicative of "military initiative".

Problem-solving CBR is used for problem solving tasks, such as design or planning. In problem-solving CBR, one formulates a solution suited to the new case by modification and adaptation of past solutions. A typical problem solving CBR task is to generate a plan or design to meet the demands of a new situation, for instance, treatment of a disorder in a sick patient who can't tolerate the "usual" treatment or creation of a battlefield plan. One solves the new problem by recalling previous cases with similar problem situations and then adapting the solutions from those problems. This may result in a full solution, or a previous case may address only part of the new problem, requiring additional reasoning to solve the remainder, or the previous case may suggest an abstract solution that must then be further refined. (Note, that in choosing a past solution for adaptation, one might need to argue in a precedent-based fashion why it, and not some other, is the best case to work on.) Case-based reasoning might be applied again to finish solving the problem, this time using the partially solved problem as a retrieval probe and recalling more specifically-matching cases from the memory.

Once a solution is proposed, it must be checked for appropriateness. This is particularly important when one is deriving solutions based on "unexplained" experience or when one is deriving solutions that are carried out in an unpredictable world (Kopeikina et al., 1988). Interpretative CBR can be used on this evaluation task. In essence, in interpretive CBR, the reasoner is attempting to better understand a situation. One tries to do this in understanding a problem, as a prerequisite to solve it, and in evaluating a solution. Also using interpretive CBR, the same military planner might try to evaluate a particular attack plan it has come up with based on what has and has not worked previously. While this CBR task is not learning in and of itself, it helps a reasoner decide on the appropriateness of using some solution or reusing a previous line of reasoning. Interpretive CBR, especially the use of hypo cases, provides a check on the use of knowledge derived from experience, no matter what learning method was used to derive that knowledge.

CBR directed at the construction of solutions has been the focus of "problem solving" CBR; primary examples are the work of Kolodner and students (Kolodner, 1983 ;Kolodner, 198;7 Kolodner, Simpson, & Sycara; 1985; Sycara, 1988a,b) and Hammond and others on certain planning tasks (Hammond, 1986a; Hammond, 1986b; Hammond, 1987; Alterman,, 1985; Collins, 1987; Barletta & Mark, 1988a). CBR directed at interpretation/classification tasks has largely been the province of "precedent-based" CBR; the primary examples are the work of Rissland and Ashley in the legal domain (Ashley, 1988a,b; Ashley & Rissland, 1988; Rissland & Ashley, 1988) and the work of Bareiss and Porter in the medical domain (Bareiss, 1988; Bareiss & Porter, 1988). While this division has been made in the past, it should be pointed out that most problems have components of both types of CBR. Labor mediation (Sycara, 1987), for example, requires both interpreting the problem and then deriving a solution based on precedents, then evaluating that solution using interpretive CBR, based on still other precedents. And diagnosis, normally thought of as an interpretive problem, can be addressed using adaptation methods that come from problem solving CBR (as in Koton, 1988a). The most effective case-based learners are going to have to use a combination of both methods.

Although most CBR approaches maintain a memory of individual cases whose individual impact and contribution to the reasoning is apparent, certain related "memory intensive" approaches use more statistical or numerical methods, for instance, the work of Stanfill and Waltz on word recognition and the work of Lehnert on certain classic state space problem solving domains (Stanfill & Waltz, 1986; Bradtke & Lehnert, 1988; Lehnert, 1987). Regardless of which approach is being used, in CBR cases and past experiences ("case memory"), ways to retrieve them ("indexing") and assess their "relevance", "similarity", "difference", etc. as well as techniques to adaptively modify and argue from them are paramount. CBR has its own set of methods for handling these tasks, and its own set of issues arising from implementing these methods. In this section we examine some of these methods and issues.

FUNDAMENTALS OF CASE-BASED REASONING

CBR involves several basic operations. Upon accepting a new case, CBR proceeds as follows:

1. **Recall relevant cases from case memory**. The goal of this step is to retrieve "good" cases that can support the reasoning that comes in the next steps. Good cases are those that have the potential to make relevant predictions about the new case. Retrieval is done by using features of the new case that were relevant in solving past cases as indexes into the case-base. Cases indexed by subsets of those or derived features are recalled. Cases that are recalled from memory might be real ones or might be hypothetical ones (that is, cases thought about during previous reasoning but that never really occurred). They can be composites of several cases, stereotypical, or specific. Techniques in this initial and critical step depend on the structure of case memory, what information is actually stored in a case, indices into the case base, notions of similarity and relevance, and of course, what the case is to be used for and what general knowledge about the domain is available. Different methods for structuring and manipulating case memory (e.g., EMOP-style hierarchical memory (Kolodner, 1983) or a flat memory) and for computing indices (e.g.,

using past failures/successes, statistical clusters, groupings based on derived features) lead to different styles of processing.

Note, in assessing relevance one must view cases from the point of view of the case at hand. So, for instance, just because a known case was a landmark case, it does not necessarily follow that it is important in the present case, since the two might not share any relevant similarities. Assessment of relevancy is highly case- and context-dependent. The importance of processing information with respect to the case at hand applies for the rest of the steps described here also.

2. **From the collection of cases retrieved in Step 1, select the most promising case or cases to reason with.** The purpose of Step 2 is to winnow down the set of relevant cases retrieved in Step 1 to a few most-on-point candidates worthy of intensive consideration as the foundations of the interpretation or solution to be generated in the next step. In assessing "on-pointness", CBR systems use various metrics and ranking schemes, for instance, the overlap of salient features, or the importance of those shared features in addressing the current reasoning goal. In the case of problem-solving CBR, the emphasis has been on choosing one best case to reason with. In interpretive/precedent-based CBR, the emphasis is on working with a select handful of best or most-on-point cases (of course, there might be a clear winner but probably not). Typically for each of the various lines of attack on an analysis, there will be a few key, most on point cases.

3. **Construct a solution or interpretation for the new case.** This step produces a solution, an interpretation, or an evaluation of the new case, along with the necessary justification or supporting argumentation. During this step in problem-solving CBR, a solution is constructed for the new case by adapting solutions from old ones. Previous cases are also used in this step to warn of the potential for failure, allowing the reasoner to anticipate and therefore avoid problems that have been encountered previously. For interpretive/precedent-based CBR, the cases selected in Step 2 are used to construct arguments pro/con an interpretation; this involves drawing analogies by focusing on shared, relevant similarities with "positive" supporting cases, that is cases speaking for a proposed interpretation, as well as distinguishing cases and breaking analogies with "negative", oppositively pointing, cases by focusing on critical differences.

4. **Test and criticize the output from Step 3.** In domains such as law or foreign policy where there is no unique "right" answer, and in domains rooted in the real world where it is impossible to predict all consequences of a plan, proposed solutions must be tested and criticized. There are several ways to do this. One way is by proposing hypotheticals and counterexamples to test the robustness of an interpretation; for instance, one might construct a slippery slope argument to show that the dividing line between the interpretations is more chimera than real and that while interpretations in the extremes may be clear, those in the middle ground are not. Another way is to use the solution as a probe to memory to see if instances are already known of the proposed solution or some similar solution failing. Another way is to simulate the solution and check the results of simulation against expected results. Note that the first two methods are case-based themselves and may require additional probes to memory (Steps 1 and 2) in order to be

done. This phase is very helpful in giving the consumer of the result, a tactical decision maker, for instance, a feeling for its utility, robustness, and weak points.

5. **Results evaluation**. In this step, the result of problem solving or the decisions made as a result of some interpretation are tried out in the real world. Feedback about the real things that happened as a result of executing the solution are obtained and analyzed. If results were as expected, further analysis is not necessary in this step, but if they were different than expected, explanation of the anomalous results is necessary. This requires figuring out what caused the anomaly and what could have been done to prevent it. This step requires all the machinery of credit and blame assignment required by every other learning method that learns from failures. With one caveat: case-based reasoning provides a simple method of explaining failures if similar ones have been encountered in the past. That is, by recalling a similar failure and adapting its explanation, it is sometimes possible to explain a new failure. This step is one of the most important for an automated case-based problem solving system, since it gives the problem solver a way of evaluating its decisions in the real world. Without feedback and analysis of feedback, a case-based problem solver is doomed to repeat its mistakes. This step provides the case-based reasoner with a means of anticipating and later avoiding mistakes it has been able to explain sufficiently. When explanation is not possible, it still provides the case-based reasoner with warnings.

6. **Update memory by storing the new case** (that is, its solution or interpretation plus underlying facts and supporting reasoning) into case memory. In this step, the new case is stored in the memory so that it can be used during later reasoning. The most important process that happens at this time is choosing the ways to "index" the new case in memory. Indices must be chosen in such a way that the case can be recalled during later reasoning at times when it can be helpful. On the other hand, it should not be indexed in so many ways that it will be recalled irrelevantly. This means that the reasoner must be able to anticipate the importance of the case to later reasoning. Additional processing that happens during this step is adjustment of memory's indexes and organizational structure. If many cases are indexed in the same way, for example, an adjustment of the indexing scheme is probably necessary.

   This step, the knowledge acquisition step, is the one that results in learning. If a case was adapted in a novel way, if it was solved using some method other than case-based reasoning, or if its result came from a combination of results from several previous cases, when it is recalled during later reasoning, the steps required to solve it won't have to be repeated. If an error in the way the case was resolved was found during evaluation or discovered when executing it in the real world, this step indexes it in such a way that if a later situation is encountered that might fail similarly, the case will provide a warning. In this step too, necessary changes in indexing mechanisms are discovered and made. In the best of all indexing schemes, not only are cases indexed to provide answers and warn of problems during later reasoning, but they also serve to change reasoning mechanisms themselves. For instance, the reasoner could learn not to rely on cases which are "two-edged swords" or not to attempt modification of cases with certain brittle characteristics

or to ask for certain information before attempting to solve particular types of problems.

CBR can also be used in the service of machine learning to aid in the intelligent selection of training instances. Too often, learning research has overlooked the potential power possessed by the mechanism – teacher or machine – choosing the training instances. This oversight occurs both in inductive and explanation based learning research; in both, but especially the latter, the learner is particularly reliant on having good training instances. This major source of bias can be used in a positive way, by a teacher, to facilitate learning. Since so much of CBR is focused on techniques to analyze and select cases, it seems natural that some of these techniques be applied to the problem of selecting (or generating) training instances. This would be especially helpful in domains where a large corpus of instances already exists and part of the knowledge acquisition bottleneck involves knowing how to pick and choose cases for consideration.

In summary, case-based reasoning incorporates a six-step process: (1) accepting a new experience and analyzing it (e.g., by computing features, relations and indices) to retrieve relevant cases from case memory; (2) selecting a set of best cases from which to craft a solution or interpretation for the problem case; (3) derivation of a solution or interpretation complete with supporting arguments in the case of precedent-based CBR and with implementation details in the case of problem solving CBR; (4) testing of the solution or interpretation with an eye to assessing its strengths, weaknesses, generality, etc.; (5) executing it in the world and analyzing the feedback; and (6) storing the newly solved or interpreted case into case memory and appropriately adjusting indices and other mechanisms.

A FUNDAMENTAL ISSUE FOR CBR: CASE RETRIEVAL AND SELECTION

The most important issues in case-based reasoning are retrieval and selection of cases from the case memory. Adaptation, interpretation, and evaluation processes can be powerful only if they have the right previous experiences to do their work from. Solving these two problems requires several things. First, cases need to be indexed in the memory so that they can be retrieved with appropriate probes. This happens at memory update time. Second, search algorithms must efficiently use retrieval probes to find matching cases in the memory. Third, best-matching or most appropriate cases must be chosen from among those retrieved from memory. These two processes comprise retrieval. Fourth, memory must be organized (structured) in such a way that search algorithms can do their jobs well.

INDEXING: Retrieving cases from memory is essentially a massive search problem. In many complex real-world domains, thousands or tens of thousands of cases might be stored, each with considerable structure. In addition, since a new case (used as the retrieval probe) and any stored case are unlikely to match exactly, one must perform some form of partial matching, and this problem is inherently nonpolynomial. In such situations, one cannot afford to exhaustively match against all cases stored in memory against a retrieval probe. Rather retrieval of **relevant cases** becomes a central issue.

The natural response to this problem is to *index* cases by appropriate features, thus making the retrieval process more selective and reducing the effect of memory size. The first step in indexing cases involves selecting an appropriate set of indices. The programmer can fix the

indices or types of indices at the outset, but this produces an inflexible system that cannot adapt to new domains. Some researchers (e.g., Lebowitz, 1987) have invoked inductive learning methods to identify predictive features, which are then used as indices. Others have used explanation-based techniques to determine relevant features for each case and index on these instead (e.g., Mark & Barletta, 1988b). Problem-solving domains are especially well-suited to the latter approach, since the trace of problem-solving behavior (e.g., goal trees) provides ready-made information for explaining success or failure (Carbonell, 1986; Hammond, 1986b), which can then be used to index quite complex cases. The notion of "derivational replay" based on such problem-solving traces has received attention in many circles, including software engineering and automated VLSI design. Most recently, researchers have been attempting to define a vocabulary for describing planning problems, adversarial disputes, and other types of problems in an attempt to discover the content of indices that allow reminding across particular domains (e.g., Hammond, 1986b).

ORGANIZING MEMORY

However, indexing by itself is not sufficient to allow efficient retrieval of relevant cases. For large knowledge bases, one must also *organize* memory into some manageable structure. Discrimination networks (Feigenbaum, 1963) are one approach to memory organization, but retrieval of a case depends on a conjunction of features being present. This leads to fragility in domains where features can be missing, but the basic approach can be extended to support redundant indexing (Kolodner, 1983; Lebowitz, 1983). In addition, one can store abstract summary descriptions at internal nodes in the network, giving generalization beyond individual cases. This provides the memory with a way of knowing that several cases with different details can be treated the same for some retrieval probes. Only the most prototypical of those cases need be retrieved under these circumstances. An interesting byproduct of this process is a conceptual clustering of cases. Thus, the memory component of case-based reasoning can provide one way of doing incremental conceptual clustering, and in fact, some researchers in conceptual clustering have drawn on the case-based approach (e.g., Fisher, 1988).

Another organization problem that must be considered is the structure of cases themselves. Some case-based systems store cases in their entirety in one place in memory, e.g., Cognitive System's Case Based Reasoning Shell (Riesbeck, 1988), Waltz and Stanfill's MBR (Stanfill & Waltz, 1986), Koton's CASEY (Koton, 1988b), Hammond's CHEF (Hammond, 1986b), and Rissland's HYPO (Rissland & Ashley, 1988). The advantage of this is that one case might provide an almost complete solution to a new one. The disadvantage is that it makes it hard to use pieces of old cases to solve pieces of new cases. An alternative is to break cases into pieces, and to store the pieces individually along with a set of pointers that can be used to reconstruct the whole, as in, e.g., Carbonell's derivational analogy (Carbonell, 1983), and Kolodner's JULIA (Kolodner, 1988; Hinrichs, 1988). This makes it easier to access parts of old cases to solve parts of new ones, allowing complex problems to be solved by combining partial solutions of several other problems. It also makes it easier to assess the applicability of a part of a previous problem to a new situation. However, while it doesn't preclude deriving a new answer from a complete previous problem, the piecemeal representation of a case requires additional processes to reconstruct it before using the full case.

RETRIEVAL ALGORITHMS

As stated previously, retrieving cases from a case memory is a massive search problem. It is made harder by the fact that search is for partial matches rather than complete ones. When doing searches for partial matches, one must be careful to make sure that the whole database doesn't get retrieved with each probe. There have been two approaches to this problem: concept refinement search methods and parallel retrieval methods.

Concept refinement search methods depend on a memory being organized in generalization/specialization hierarchies. Search starts at the top (most general point) in a hierarchy and progresses downward only when a match is possible at the more general level. Search is cut off at any branch in the hierarchy where a match is not possible. Thus, specific cases in memory can be retrieved only when their abstractions in the hierarchy match the retrieval probe. Concept refinement search is well suited to indexed memories in which the indices and intermediate nodes are chosen well. They derive the power to bypass large parts of memory from the content of the abstraction nodes in the hierarchy being searched. The earliest implementations of concept refinement search were implemented on memories organized by redundant discrimination networks, as described above (Kolodner, 1983; Lebowitz, 1983). Later implementations of concept refinement search methods have been implemented on memories with more distributed representations (i.e., cases are stored in pieces) (Martin & Riesbeck, 1986; Kolodner, 1988). Output of concept refinement algorithms is a set of cases that are in the right contextual ballpark with the case used as a retrieval probe.

Parallel implementations bypass the problem of having to deal with the whole memory by making that a feature of the solution rather than a sore point. MBR methods (Stanfill & Waltz, 1986) for example, assume that there are enough processors available so that the match between the retrieval probe and every case in the memory can be done at the same time. Partial matching is done by applying an evaluation function to the match done on each item in the case base, and the items that are above threshold are retrieved. Stanfill (1987) has applied this technique to the task of mapping letters to phonemes, achieving 88% predictive accuracy on a test set of 1024 instances. He has also shown that the method degrades gracefully as one adds noise and as one decreases the size of the case base. The big issue with such methods, of course, is choosing the evaluation function well (see section on case selection).

Concept refinement has also been implemented on a parallel machine (Kolodner, 1988). While the algorithm returns the same cases that would be recalled using the serial algorithm, the parallel implementation runs in time linear in the size of the retrieval probe. This leads us to believe in the scalability of the methods.

Even with the best retrieval algorithms and indexing methods, uncontrolled growth of memory can be a problem. Uncontrolled growth of the case base is a natural concern in this paradigm, though it becomes less of a problem with parallel architectures. Still, one may want to store cases selectively and delete others on occasion. Kibler and Aha (1987) have taken this approach, storing new cases only when the existing knowledge base leads to a classification error. Their approach compares favorably to methods for inducing decision trees in diagnostic accuracy for thyroid diseases. Bradshaw (1987) has successfully applied a similar approach to the challenging domain of speech recognition.

CHOOSING THE BEST CASES

The indexing methods, memory organizations, and retrieval algorithms available can recall a set of partially-matching cases for a case-based reasoner to use, but often provide too many cases. Thus, an additional problem to be considered is choice of best-matching or most-on-point cases. Even MBR, which chooses best-matching cases during retrieval needs a principled way of deciding which is best.

While it might seem that a count or weighted count of matching features could do the work here, this is not possible because the importance of some features can only be derived in context. Often, it is the cases already in the memory (i.e., those retrieved) that determine which of the features of the new case are most important ones for matching. There are several methods proposed for doing this: preference heuristics (Kolodner, 1988), dimensional analysis (Rissland & Ashley, 1988), and dynamically-changing weighted evaluation functions (Stanfill, 1987). Though the methods vary, what they all have in common is that the set of items retrieved by retrieval algorithms each contribute their know-how about what was important in solving them so that the case selector can decide what to take into account in determining which cases match best. The best or most on point cases are the ones that address the reasoner's current problem in the best way.

MATCHING

Choosing the best case requires being able to match two cases together to generate a correspondence between their parts. Given feature-based or attribute-based representations, the process of matching two cases is relatively simple and inexpensive although even in relatively simple, flat data bases matches may require cleverness. However, in some domains like planning the need to consider structural or relational representations is inescapable, and these introduce serious complexities into the match process. A set of symptoms presented at some visit to a doctor, for example, may represent an improvement or worsening from a previous visit. Thus, a "trajectory" of symptom values may be the right candidate for matching two "cases", rather than a single record of the experience of one visit. In this case, matching is more complex, requiring search and judgment. To make these types of judgments, more is necessary than the features of the case itself. In particular, the relationships between the parts of the case, the case's relationship to other related cases, and the reasons underlying the derivation of the solution to a case may all be necessary.

Researchers concerned with the process of *analogy* have devoted considerable attention to this issue, with most approaches involving some form of heuristic search through the space of partial matches. In this framework, the main issue becomes finding ways to constrain and direct the search for a useful match. For instance, Falkenhainer, Forbus, and Gentner's (1986) Structure-Mapping Engine finds mappings that preserve higher-order relations between two cases in preference to ones that preserve simple features shared by the cases. Holyoak and Thagard propose five types of constraints that must be filled in matching and a connectionist-type relaxation algorithm that finds the best correspondences between two cases. Carbonell (1983) addresses the ways derivations are used in this process. Other researchers (e.g., Winston, 1984) have proposed different but closely related methods. Of course, we still need to discover how to integrate matching and retrieval methods with each other.

POTENTIAL TOOLS AND APPLICATIONS

Although interest in case-based learning has emerged relatively recently, a few tools are ready for distribution. For instance, Falkenhainer, Forbus, and Gentner's SME (Structure-Mapping Engine), is a flexible system for analogical matching that is available from the University of Illinois. Thinking Machines has built memory-based reasoning software that runs on the Connection Machine with a Symbolics front end. The system accepts a database in relational form, and produces the nearest match or several nearest matches using a variety of statistical and user programmable similarity metrics. The system is not a supported product, but the software is being actively developed, using the Framingham Heart Study database as a test domain. In addition, at least one DARPA contractor (Cognitive Systems) is actively working on prototype tools for manipulating and using extensive case bases. Additional systems are also being constructed in university settings, for instance, the CAsed-BAsed REasoning Tool (CABARET) being developed by Rissland's group at the University of Massachusetts. Nevertheless, more such tools are needed, and a DARPA initiative could accelerate their development and evaluation.

Case-based methods are relevant to a variety of application domains involving classification and problem solving. Since we have discussed many such domains in earlier sections, we will not recount them here. However, we should mention that, like inductive methods, case-based techniques seem best suited to domains in which many training cases are available, perhaps with many exceptional cases, and where it is difficult to specify appropriate behavior using abstract rules.

OPEN ISSUES IN CASE-BASED LEARNING

Research on case-based approaches has led to a number of promising methods, some of which have been tested on challenging domains. However, a number of open issues remain to be addressed:

- Matching metrics. Many existing systems employ ad hoc schemes for matching against cases in memory. We need to develop more principled methods. We also need to experiment with metrics for determining best matches.
- Selective matching. In structural domains, the match process itself can be very expensive. We need to explore methods for determining relevant features, thus reducing the cost of finding high-quality matches.
- Selecting indices. A number of methods exist for selecting indices, but we need more studies of this process. We also need methods for generating new indices dynamically.
- Memory organization. Some initial work has addressed the organization of memory, but we need more work in this area. We also need techniques that dynamically reorganize memory as new cases are encountered.

- Connections between cases. In some domains, each case may have a complex, internal structure, effectively consisting of many component cases. We need principled schemes for representing connections between component cases, and mechanisms for storage and retrieval that support such structure.

- Dealing with sparse memory. In some domains, one needs to be able to deal with a lack of known relevant cases, especially in the presence of a novel problem situation. One approach is to generate hypotheticals to help fill in and "interpolate" case memory. Hypotheticals are also effective in testing the robustness of an analysis or solution.

- Forgetting. Although most cases are useful, storage of all cases can lead to an overly "cluttered" case memory and to overfitting effects in noisy domains. We need methods that tell when to forget redundant, superfluous, or even harmful cases.

- Mixing CBR with other paradigms. There is a need in some domains to combine CBR with other modes of reasoning, such as model-based or rule-based reasoning. This is especially so when a large component of the domain knowledge is other than case knowledge, for instance, in statutory legal domains where there are statutes (rules), but cases are necessary to interpreting and applying this other knowledge. We need to better understand how to build mixed paradigm systems in order to complement the strengths of the individual paradigms.

- Selecting cases as training instances. Techniques from CBR can probably help reduce the knowledge acquisition bottleneck, especially with regards to the problem of selecting/generating training instances to give to a learner. We need to discover how to intelligently select training instances.

Research on case-based reasoning has produced some promising techniques, but we need to more fully explore the space of such methods, and we need to carefully evaluate alternative approaches in terms of their performance on real-world domains. DARPA could play an important role in helping this happen.

References

Alterman, R. (1985). Adaptive Planning: Refitting Old Plans to New Situations. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, Irvine, CA.

Ashley, K.D. (1988a). Arguing by Analogy in Law: A Case-Based Model. *Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science, and Philosophy*, Helman, D. (Ed.), D. Reidel.

Ashley, K.D. (1988b). *Modelling Legal Argument: Reasoning with Cases and Hypotheticals.* Ph.D. Thesis. COINS TR-88-01, Department of Computer and Information Science, University of Massachusetts. To be published by The MIT Press/Bradford Books.

Ashley, K.D. and Rissland, E.L. (1988). *A Case-Based Approach to Modeling Legal Expertise.* Counselor Project Technical Memo No. 23, Department of Computer and Information Science, University of Massachusetts, Amherst. to appear in *IEEE Expert,*, Summer, 1988.

Ashley, K.D. and Rissland, E.L. (1987). Compare and Contrast, A Test of Expertise. In *Proceedings AAAI-87*, Seattle, July 1987.

Bareiss, R. (1988). *Protos: A Unified Approach to Concept Representation, Classification, and Learning.* Ph.D. Thesis. Technical Report CS-88-10, Department of Computer Science, Vanderbilt University, Nashville, TN.

Bareiss, R., Banting, K. and Porter, B. (1988). The Role of Explanation in Exemplar-Based Classification and Learning. *Proceedings of the Case-Based Reasoning Workshop*, AAAI-88, Minneapolis, MN.

Barletta, R. and Mark, W. (1988a). Explanation-Based Indexing of Cases. *Proceedings of the Seventh National Conference on Artificial Intelligence.* Minneapolis, MN.

Barletta, R. and Mark, W. (1988b). Explanation-Based Indexing of Cases. *Proceedings of the DARPA Workshop on Case-Based Reasoning*, Clearwater, FL.

Bradshaw, G. (1987). Learning about Speech Sounds: The NEXUS Project. *Proceedings of the Fourth International Workshop on Machine Learning.* Irvine, CA.

Bradtke, S. and Lehnert, W.G. (1988). Some Experiments with Case-Based Search. *Proceedings of the Seventh National Conference on Artificial Intelligence.* Minneapolis, MN.

Carbonell, J.G. (1986). Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. *Machine Learning, An Artificial Intelligence Approach, Vol. 2,* R. Michalski, J. Carbonell and T. Mitchell (Eds)., Morgan Kaufmann, Los Altos, CA.

Carbonell, J.G. (1983). Derivational Analogy and its Role in Problem Solving. *Proceedings of the Third National Conference on Artificial Intelligence*, Washington, DC.

Collins, G. (1987). *Plan Creation: Using Strategies as Blueprints.* Ph.D. Thesis. Department of Computer Science, Yale University, New Haven, CT.

Falkenheimer, B. (1988). The Utility of Difference-Based Reasoning. *Proceedings of the Seventh International Conference on Artificial Intelligence*, Minneapolis, MN.

Falkenheimer, B., Forbus, K.D., and Gentner, D. (1986). The Structure Mapping Engine *Proceedings of the Sixth National Conference on Artificial Intelligence*, Philadelphia PA.

Feigenbaum, E.A. (1963). The Simulation of Verbal Learning Behavior. In Feigenbaum and Feldman (eds) *Computers and Thought*, New York: McGraw Hill, 297–309.

Fisher D.H. (1988). A Computational account of Basic Level and Typicality Effects' *Proceedings of the Seventh International Conference on Artificial Intelligence*, Minneapolis, MN.

Hammond, K. (1987). Explaining and Repairing Plans that Fail. *Proceedings of the IJCAI-87.*

Hammond, K. (1986a). *Case-based Planning: An Integrated Theory of Planning, Learning and Memory.* Ph.D. Thesis, Yale University, New Haven, CT.

Hammond, K. (1986b). CHEF: A Model of Case-Based Planning. *Proceedings of AAAI-86*, Philadelphia, PA.

Hinrichs, T.R. (1988). Towards an Architecture for Open World Problem Solving. *Proceedings of the DARPA Workshop on Case-Based Reasoning*, Clearwater, FL.

Kibler, D. and Aha, D. (1988). Case-Based Classification. *Proceedings of the Case-Based Reasoning Workshop, AAAI-88*, Minneapolis, MN.

Kopeikina, L, Brandau, R. and Lemmon, A. (1988). Extending Cases Through Time. *Proceedings of the Case-Based Reasoning Workshop, AAAI-88*, Minneapolis, MN.

Kolodner, J.L. (1988). Retrieving Events from a Case Memory: A Parallel Implementation. *Proceedings of the DARPA Workshop on Case-Based Reasoning*, Clearwater, FL.

Kolodner, J.L. (1987). Capitalizing on Failure Through Case-Based Inference. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society.*

Kolodner, J.L. (1983). Towards and Understanding of the Role of Experience in the Evolution from Novice to Expert. *International Journal of Man-Machine Studies, Vol. 19.*

Kolodner, J.L., Simpson, R.L. and Sycara, K. (1985). A Process Model of Case-Based Reasoning in Problem Solving. *Proceedings of IJCAI-85*, Los Angeles, CA.

Koton, P. (1988a). *Using Experience in Learning and Problem Solving*, Ph.D. Thesis, MIT.

Koton, P. (1988b). Reasoning about Evidence in Causal Explanations. *Proceedings of the DARPA Workshop on Case-Based Reasoning*, Clearwater, FL.

Lebowitz, M. (1987). Experiments with Incremental Concept Formation: UNIMEM. *Machine Learning* Vol. 2 No. 2. pp 103–138

Lebowitz, M. (1983). Generalization from Natural Language Text. *Cognitive Science*, Vol. 7 No. 1.

Lehnert, W.G. (1987). *Case-Based Reasoning as a Paradigm for Heuristic Search.* Technical Report No. 87-107, Department of Computer and Information Science, University of Massachusetts.

Mark, W. and Barletta, R. (1987). *Case-Based Reasoning in Manufacturing.* Manuscript, Lockheed AI Center, Palo Alto, CA.

Riesbeck, C.K. (1988). An Interface for Case-Based Knowledge Acquisition. *Proceedings of the DARPA Workshop on Case-Based Reasoning*, Clearwater, FL.

Rissland, E.L. and Ashley, K.D. (1988). Credit Assignment and the Problem of Competing Factors in Case-Based Reasoning. *Proceedings of the DARPA Workshop on Case-Based Reasoning*, Clearwater, FL.

Rissland, E.L. and Ashley, K.D. (1986). Hypotheticals as Heuristic Device. *Proceedings AAAI-86*, Philadelphia, August 1986.

Rissland, E. L. and Skalak, David B. (1989) Case-Based Reasoning in a Rule-Governed Domain. *Proceedings of the Fifth IEEE Conference on Artificial Intelligence Applications*, Miami, FL.

Stanfill, C. (1987). Memory-Based Reasoning Applied to English Pronunciation. *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA.

Stanfill, C. and Waltz, D. (1986). Toward Memory-Based Reasoning. *Communications of the ACM.* Vol. 29, No. 12. pp. 1213-28.

Sycara, K. (1988a). Using Case-Based Reasoning for Plan Adaptation and Repair. *Proceedings of the DARPA Workshop on Case-Based Reasoning*, Clearwater, FL.

Sycara, K. (1988b). Resolving Goal Conflicts Via Negotiation. *Proceedings of the Seventh National Conference on Artificial Intelligence*, Minneapolis, MN.

Sycara, K. (1987). *Resolving Adversarial Conflicts: An Approach to Integrating Case-Based and Analytic Methods.* Ph.D. Thesis. Technical Report No. GIT-ICS-87/26, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA.

Winston, P.H. (1984). Learning New Principles from Precedents and Exercises. *Artificial Intelligence.* 19:321-350.

Distributed by:
Morgan Kaufmann Publishers, Inc.
P.O. Box 50490
Palo Alto, CA 94303

Printed in the United States of America

ISBN 1-55860-068-X