When Experience is Wrong: Examining CBR for Changing Tasks and Environments*

David B. Leake and David C. Wilson

Computer Science Department
Indiana University, Lindley Hall
150 S. Woodlawn Ave
Bloomington, IN 47405, U.S.A.
{leake,davwils}@cs.indiana.edu

Abstract. Case-based problem-solving systems reason and learn from experiences, building up case libraries of problems and solutions to guide future reasoning. The expected benefits of this learning process depend on two types of regularity: (1) *problem-solution regularity,* the relationship between problem-to-problem and solution-to-solution similarity measures that assures that solutions to similar prior problems are a useful starting point for solving similar current problems, and (2) *problem-distribution regularity,* the relationship between old and new problems that assures that the case library will contain cases similar to the new problems it encounters. Unfortunately, these types of regularity are not assured. Even in contexts for which initial regularity is sufficient, problems may arise if a system's users, tasks, or external environment change over time. This paper defines criteria for assessing the two types of regularity, discusses how the definitions may be used to assess the need for case-base maintenance, and suggests maintenance approaches for responding to those needs. In particular, it discusses the role of analysis of performance over time in responding to environmental changes.

1 Introduction

Case-based reasoning (CBR) solves new problems by retrieving stored cases encapsulating records of similar problems, and adapting their lessons to fit the new circumstances. Case-based problem-solving is based on two central premises about the regularity of the problem-solver's world (e.g., Kolodner, p. 8). The

* The authors' research is supported in part by NASA under award No NCC 2-1035. The authors are currently on leave at the Computer Science Department of Northwestern University and gratefully acknowledge its support. The authors also thank the anonymous reviewers for their helpful comments.

first, which we call *problem-solution regularity,* describes the relationship between problem descriptions and solutions that assures that similar problems have similar solutions. This regularity is needed to guarantee that cases for similar prior problems are likely to be useful starting points for new reasoning. The second, which we call *problem-distribution regularity,* describes the relationship between new problems and those previously encountered. This regularity is needed to assure that the system will have the cases it needs for the problems it is called upon to solve.

The successes of numerous CBR systems bear out that for many tasks and domains, appropriate similarity metrics can be devised to provide sufficient problem-solution regularity, and that problem-distribution regularity is often sufficient to enable effective CBR. Unfortunately, no matter how good initial similarity metrics might be for a given task and domain, and no matter how complete a case library a system may build up, changes in task and domain characteristics may render obsolete prior similarity criteria or cases. Developers have cited the problem of dealing with changing task characteristics as the reason for rejecting CBR for some tasks (Talebzadeh *et al.*, 1995), and the long-term use of CBR systems makes such changes increasingly likely during a system's lifetime. In order to perform as well as possible despite changing circumstances, a CBR system must be able to evaluate how well the regularity assumptions apply and to signal the need for maintenance or to invoke its own maintenance strategies as needed.

This paper presents initial steps towards understanding and responding to deviations from desired regularities. First, it defines measures that can be used to calculate the amount of problem-solution regularity and problem-distribution regularity that exist for the problem sequences that a system encounters. Second, the paper discusses methods that may be used to respond to, and (ideally) to exploit changing characteristics of the problems the CBR system solves and of the environment in which its solutions must be applied.

In particular, the paper describes opportunities for maintenance strategies that perform their changes based on analysis of problem-solving and case-base characteristics over time—*diachronic case-base maintenance strategies* as described in (Leake and Wilson, 1998). In general, determining the right response to shifting context requires knowledge that is unlikely to be available from a single snapshot of the CBR system's state. However, by examining *trends* in retrieval performance, system errors, and presented problems, the system may be able to respond more effectively.

2 Defining Regularities for Case-Based Reasoning

It is well-known in the CBR community that case-based reasoning depends on two relationships: the relationship between *similarity of problems* and *similarity of solutions*, and the relationship between *prior problems* (solved by the system or provided as seed cases) and *new problems*. However, to our knowledge, there are not yet precise definitions of what these relationships mean. Such definitions would be useful to quantify and compare the relationships in order to understand the effects of different similarity metrics, case bases, and problem sequences on the performance of different CBR systems. Equally important, such definitions give criteria for monitoring the appropriateness of a system's similarity criteria and case library for dealing with current problems, in order to identify the need for system maintenance. This section proposes working definitions as a basis for future discussion and study.

2.1 Basic assumptions and definitions

Throughout our definitions, we will make some standard assumptions. First, we assume that there is a fixed CBR system that processes problems in a problem space P and that the solutions for these problems are elements of a solution space S. Cases are pairs $(p, s) \in C = P \times S$, the set of all possible cases. The system begins with a finite "seed" case base $B_1 \subseteq C$. As the system is used, it processes a sequence of problems Q = p_i, p_{i+1}, ... , p_j, where each $p_k \in P$ for $k = i, ..., j$. We define the sequence to start with an arbitrary index because, as we discuss in section 8.2, it is sometimes useful to consider the subsequence that starts after some initial set of problems has been processed.

Adding to the case base: We assume that after each problem is processed and the resulting solution has been evaluated, a new case with the problem and its correct solution are added to the case base. This means that each problem p_k is processed using an updated case base B_k that includes the results of previous processing. Note that this does not imply that the system can solve all problems presented to it: The correct stored solution may be based on external feedback if the system generates an incorrect solution or fails to generate a solution.

How problem distance guides retrieval: The CBR system uses a "problem distance" function $PDist : P \times C \rightarrow [0, \infty)$ to measure the distance between a new problem and the problem description of a stored case. $PDist(p, c)$ is zero if p is the same problem solved by c. Given a new problem, the CBR system retrieves the case closest to that problem according to $PDist$. However, there is no guarantee that the case considered closest by this function will actually be "close" to the problem in any useful way. This function simply reflects the similarity metric built into the system, whether or not it is useful.

How usefulness of retrievals is judged: The evaluator of the system uses a "real distance" function $RDist : P \times C \rightarrow [0, \infty)$ to measure how far the solution in a case is from the solution for a given problem. This function measures the usefulness of retrieved solutions according to the evaluator's goals for the retrieval process, which may not be classic "similarity." For example, if the evaluator's primary goal is to minimize the adaptation time required to generate a new solution, "real distance" could be measured in adaptation time: $RDist(p, c)$ could be the time to adapt the solution from case c to solve problem p, with some upper limit on the amount of time allowed. $RDist$ could also be defined to reflect other retrieval goals. For example, if reliability of adaptation is an issue, it could consider cases "closer" to a problem if they can be adapted to solve the problem using more reliable adaptations (regardless of adaptation time).

We stress that $RDist$ does not necessarily correspond to any function within the CBR system; it is an external criterion. For example, $RDist$ might be calculated off-line to determine the retrievals the CBR system *should have* made. Thus efficiency of calculating the $RDist$ function is comparatively unimportant. It might be possible, for example, to calculate $RDist$ for adaptability by simply adapting all stored cases to the new problem and seeing which adaptation was fastest.

In an ideal CBR system, the cases with the closest *problems* (according to $PDist$) would also have the closest *solutions* (according to $RDist$). In practice, of course, the actual similarity metric is likely to differ from the ideal (see Smyth and Keane, 1996, for an empirical demonstration). In some situations the deviations may be substantial enough to impair system performance.

2.2 Defining problem-solution regularity

The goal of our definition of problem-solution regularity is to capture how well $PDist$ approximates $RDist$ in practice. Because this depends on the specific context in which the CBR system is solving problems, our definition explicitly depends on:

- the goals for retrieval (as captured by $RDist$),
- the set of seed cases available to the system, and
- the problem sequence that the system is called upon to solve.

As background for our definition, for any input problem, we can calculate two sets of cases according to the formulas below. The first set of cases, which we designate by CCP for *Closest Cases to Problem*, contains all the cases within a case base B whose problem descriptions are closest to the input problem. The second, which we designate by RCC for *Real Closest Cases*, contains the cases whose solutions are within a user-specified neighborhood of the optimal solution. The size of the neighborhood is determined by a user-specified non-negative parameter ϵ.

$$CCP(PDist, p, B) = \{c \in B | PDist(p, c) = min_{c' \in B} PDist(p, c')\} \tag{1}$$

$$RCC(RDist, p, B, \epsilon) = \{c \in B | RDist(p, c) \leq min_{c' \in B} RDist(p, c') + \epsilon\} \tag{2}$$

If $\epsilon = 0$, RCC returns the optimal cases for solving the problem according to the "real" distance metric.

We let B_k designate the case library used when processing problem p_k. This case library contains the initial seed cases and all the new cases added to the case base processing problems before p_k. Following the notion of *precision* in information retrieval, we then define:

$$SimPrecision(PDist, RDist, p_k, B_k, \epsilon) = \frac{CCP(PDist, p_k, B_k) \cap RCC(RDist, p_k, B_k, \epsilon)}{CCP(PDist, p_k, B_k)} \quad (3)$$

This function measures the probability that a case returned as optimal by the similarity function will actually be within ϵ of an optimal case.[1]

Given these definitions, we define the problem-solution regularity as the average *SimPrecision* over the problem sequence Q, starting with case base B_i, as follows:

$$ProbSolnReg(PDist, RDist, Q, B_i, \epsilon) = \frac{\Sigma_{k=i,...,j} SimPrecision(PDist, RDist, p_k, B_k, \epsilon)}{j - i + 1} \quad (4)$$

When ϵ is set to 0, this function calculates the average probability that a case for a maximally-similar problem will actually be optimal. With non-zero values for ϵ, this function provides information about the average probability that a maximally-similar problem (according to the system's similarity metric) will be acceptably close to a maximally useful case, which determines the quality of the similarity metric.

We note that when *ProbSolnReg* is used to compare the problem-solution regularity of different systems, *RDist* must be same for both systems. If different systems have different "real" costs (e.g., because of differences in adaptation capabilities), differences in the values of *ProbSolnReg* for the two systems may not predict their relative performances.

2.3 Defining problem-distribution regularity

The second regularity assumption of CBR is that new problems will tend to resemble the problems addressed in previous cases (either in the seed case base, or in cases learned during prior processing). We call this *problem-distribution regularity*. It determines the likelihood that, as new problems are processed (and new cases with their solutions are added to the seed case base), the case base will contain cases for similar problems. When the case base does contain similar problems, and when (in addition) there is sufficient problem-solution regularity, this will result in retrieval of cases whose solutions are close to the actual solutions according to *RDist*.

ProbDistReg calculates the percentage of cases in a problem sequence $Q = p_i, ..., p_j$ for which there are sufficiently close cases in the current case bases B_k

[1] Because we assume that the system will reason from a single most similar case, the IR notion of *recall* is not relevant here. It would be relevant if, e.g., the system attempted to increase reliability by generating and comparing solutions starting from multiple cases.

built up from the seed case base B_i, according to a user-specified distance limit $\epsilon \geq 0$.

$$ProbDistReg(Q, B_i, \epsilon) = \tag{5}$$
$$\frac{1}{j-i+1} * \Sigma_{k=i,\ldots,j} \begin{cases} 1, \text{If } min_{c \in B_k} PDist(p_k, c) < \epsilon \\ 0, \text{Otherwise} \end{cases}$$

Together, *ProbSolnReg* and *ProbDistReg* provide measures that describe the performance of a CBR system. Individually, each one identifies problems that can be addressed by either refining the similarity metric or the solutions stored in cases (for *ProbSolnReg*) or by adding to the case library (for *ProbDistReg*).

3 Perspective on Regularity-Related Research

In this section we consider the importance of the regularities and compare our perspective to related research; in the following sections we look at its practical application.

Work on Problem-Solution Regularity: The importance of problem-solution regularity underlies the considerable attention to similarity criteria in CBR research. Faltings (1997) uses probability theory to prove that for prediction tasks, the assumption that a problem with similar features to an earlier one is likely to have a similar solution is guaranteed to be true on average. The issue of how to define practical similarity metrics for particular tasks remains a central research focus of the field, making it useful to have criteria for comparing different similarity metrics.

Recent CBR work has developed methods for making retrieval criteria explicitly reflect the underlying "true" retrieval criterion that we have called *RDist*. A primary example is adaptation-guided retrieval (Smyth and Keane, 1996), which replaces the traditional similarity criterion with estimated cost of adaptation, in order to retrieve cases that satisfy the goal of easy adaptation.

Work on Problem-Distribution Regularity: The key question of problem-distribution regularity is whether the case library will contain the cases a system needs to solve the problems it encounters. The importance of problem-distribution regularity is recognized by developers of CBR applications, who attempt to gather representative and well-distributed sets of cases for their systems (e.g., (Kriegsman and Barletta, 1993; Watson, 1997)).

Recent work on case-base competence (Smyth and McKenna, 1998; Zhu and Yang, 1998) has developed methods for estimating the range of problems that can be solved by a system with a given case-base. The purpose of this work is to assure that problem-solution regularity is sufficient, to give an indication of the likely system success rate, and to help identify regions of the case base in which additional cases may be needed.

Problem-distribution regularity is closely related to case-base competence, but our work differs from that work in two ways. The first difference concerns the role of problem distribution. Analysis of case-base competence assumes a uniform distribution of problems in order to make analysis more tractable. Likewise, it is customary for empirical evaluations of CBR systems to use a randomly-generated set of problems uniformly distributed in the problem space (e.g., (Veloso, 1994)). However, our definition explicitly references the particular problem sequence on which the behavior is measured. While we agree with Smyth and McKenna (1998) that assuming a uniform distribution can provide a very useful overall view, considering specific details of problem presentation order and distribution can be useful as well. For example, the quality of a CBR system's performance can depend strongly on the order of case presentation (Fox, 1995; Redmond, 1992), making it desirable for the formulas to be usable for exploring the effects of different orderings. Likewise, as we discuss later in this paper, if the system can identify "hot spots" in case-base accesses, examining problem distribution regularity may make it possible to reorganize the case base to speed likely retrievals, or to delete (or deactivate, e.g., by placing in secondary storage) cases that are not being used.

Second, our definition of problem-distribution regularity depends on a user-defined threshold for what constitute sufficiently similar stored cases, rather than considering only whether the problem can or cannot be solved. Using a user-defined criterion for whether a stored case is "close enough," rather than simply whether *some* solution can be generated, is important when the quality of solutions depends on the amount of adaptation performed, or when there are changeable limits on the amount of effort that can be expended on adaptations. For example, in some domains, available domain theories are strong enough for local adaptations but are not sufficiently reliable for more substantial changes (e.g., Cheetham and Graf, 1997).

Work on Case-Base Maintenance: Many researchers are examining issues in *case-base maintenance* (CBM) for improving the performance of CBR systems (for an overview, see Leake and Wilson, 1998). CBM research addresses issues such as assuring that the cases in the case base cover the space of possible problems (Smyth and McKenna, 1998; Zhu and Yang, 1998) and deleting superfluous cases to improve space efficiency or utility of retrieval (Smyth and Keane, 1995). These do not address, however, how to maintain the case-base in response to specific task needs—for example, to build coverage in precisely those areas that tend to arise in current problems—or how to predict the need for future maintenance from current problems, in order to proactively revise the case base before problems occur. Salganicoff (1997) has studied the problem of learning time-varying functions in instance-based learning, and proposes a method based on de-activating old instances when similar new ones are available, and selectively re-activating those that are consistent with new data. Ideally, augmenting CBR systems with the ability to detect regularity problems and respond to problem trends will improve their ability to avoid future failures and organize their case bases for efficient access.

4 Calculating the Regularity Values

In order to apply the formulas to trigger maintenance, practical means are needed to calculate their values. Because $ProbDistReg$ depends only on the levels of similarity between new problems and the cases retrieved to deal with them (which are available as a byproduct of normal processing), $ProbDistReg$ can be calculated easily.

On the other hand, calculating $ProbSolnReg$ is problematic, because calculating $RDist$ requires complete information about the "right" retrievals. (If this information could be calculated inexpensively at retrieval time, the system could always make perfect retrievals.) Nevertheless, it is sometimes possible to take advantage of information available after a problem is solved to estimate whether the right case was retrieved. The ROBBIE system (Fox and Leake, 1995), for example, detects problems in its similarity criteria by first solving the current problem, and then using the solution as the index for another retrieval, to determine if the solution from another case is more similar to the final result. If so, perfect similarity criteria would have favored that case, so the failure to retrieve it shows a flaw in problem-solution regularity.[2]

Alternatively, ProbSolnReg calculations could be done off-line at times when high processing cost is acceptable, to trigger off-line maintenance to improve future on-line performance.

5 Using the Formulas as Maintenance Triggers

The previous definitions provide a basis for judging the levels of regularity for particular systems, case bases, and problem sequences. By monitoring the levels of regularity and their changes, it is possible to identify needs for maintenance. For example,

- When problem-solution similarity falls below acceptable levels, it may signal:
  - Failure of the similarity metric to capture features that have become important in predicting $RDist$ for current problems (e.g, if a route planner does not consider the direction of old paths when doing retrieval, and is called upon to plan paths in a new area with many one-way streets).

[2] This approach does not apply to all domains, however. For example, if solutions are a single numeric value, the fact that a case in memory happens to have the correct value may be coincidental. If a CBR system estimates the price of a bunch of carrots based on the price of a bunch bought the week before, even if its estimate is wrong it is probably not appropriate to adjust its similarity to consider the carrots more similar to a light bulb that happens to cost precisely the correct amount.

  - Changes in the problem-solving environment that require adjusting the solutions that would have applied to the same problems in the past, so that *RDist* itself has changed and *PDist* must be adjusted to be consistent (e.g, if roads have been closed, blocking paths that would previously have been successful).
- When problem-distribution regularity falls below acceptable levels, it may signal:
  - Insufficient case coverage of the current problems (additional cases would increase the chance of having one available within the acceptable neighborhood).
  - Flawed or insufficient adaptation knowledge (improving adaptation knowledge would increase the size of the neighborhood of cases that is usable).
- When problem-distribution regularity is high for a subset of the case base, it may signal:
  - A "hot spot" in the case base (which enables reorganizing the case base to facilitate access to active regions, or deactivating cases from less frequently used regions.)

6 Determining How to Respond: The Role of Diachronic Analysis

Once a regularity problem has been found, it is necessary to select strategies for responding. Normally, CBR systems consider only the current problem and state of the case base when responding to processing failures (e.g., by revising the indices for a case or storing a new case with the correct solution). However, considering trends in problems may enable better response strategies. For example, knowing that problem-solution regularity has dropped from acceptable levels to a current unacceptable level is more informative than simply knowing that the level is unacceptable, because a change in performance must be caused by changes in either the problem distribution or the environment. For example, if a system for estimating building costs consistently generates estimates that are too low, that trend suggests that a general change is needed to prevent that class of failures in the future.

One response strategy is to simply update the cases in the case base (e.g., increasing the recorded prices), but this may lose useful historical information. It may also require monitoring the update history and ages of cases, in order to make sure that all cases are updated properly. Another alternative is to keep the values of cases unchanged, but to add a "lazy" maintenance rule to adjust case solutions after they have been retrieved (Leake and Wilson, 1998)).

Leake and Wilson (1998) describe a class of maintenance strategies that collect data over time, over a sequence of snapshots of system processing, in order to identify trends in how case-base contents and usage are changing. They call policies based on analyzing the performance of the case-base over time *diachronic maintenance policies*. Diachronic analysis is useful, for example, for determining whether coverage problems—shown by low problem-distribution regularity—should prompt the search for additional cases. If problem-distribution regularity shows an *increasing trend*, showing that the cases being processed are filling the important regions of the case base, it may suffice to simply let the normal case learning process fill the case base. However, if the level of problem-distribution regularity is low and stable, or even decreasing, steps must be taken to increase the coverage of the case library.

Diachronic analysis is also useful to find and exploit trends in problems presented to the case base. If the problems that the system must solve consistently fall within a small neighborhood, it may suggest that the system should exploit the locality of the "hot spot" by reorganizing the case base to make cases in that region easier to access. In a distributed case base, cases in the hot spot are candidates for pre-fetching. If space limitations require that some cases be deleted, for efficiency reasons the system should also focus competence-preserving deletion (Smyth and Keane, 1995) on regions other than the hot spot, in order to minimize adaptation cost on likely problems by keeping the active regions more densely populated with nearby cases.

Finally, diachronic analysis is useful for monitoring and guiding the maintenance process itself: The history of maintenance operations applied will affect choices of which operations *should* be applied. For example, if maintenance has just added a large set of cases to the case base to improve problem-distribution regularity, the choice of whether to search for still more cases should be determined by observing the effects of the new cases over some period of time, rather than simply based on the value of $ProbDistReg$ as soon as the next input problem is processed.

7 Tools for Trend Detection

Performing diachronic maintenance requires methods for detecting underlying trends in sequences of values over time. Trend detection for numeric values can be done by a number of statistical techniques. These include simple methods such as linear regression models that attempt to find the equation of the line that best fits the data as well as time series analysis techniques such as autoregressive moving averages (ARMA) and autoregressive integrated moving averages (ARIMA). Research in machine learning has studied "concept drift," in which hidden changes in context over time cause learned experiences to become inaccurate (e.g., Salganicoff, 1997). A number of techniques have been applied to concept drift problems in time ordered domains for learning hidden context (Harries *et al.*, 1998; Lane and Brodley, 1998), and could be applied to adjusting similarity criteria when problem-solution regularity becomes insufficient due to concept drift.

8 Two Examples: Error Trends and Hot Spots

In this section we illustrate the usefulness of trend-based reasoning for responding to drops in problem-solution regularity and to patterns in problem distribution.

8.1 Addressing Solution Error Trends:

As a simple example of the use of trend detection, we show how regression techniques can augment a case-based price estimating system, in order to make its predictions more robust despite inflation. Trend-based corrections are triggered by drops in problem-solution regularity: When the solutions predicted based on similar prior problems are no longer close to the real solutions determined by feedback to the program, maintenance is performed. The method we describe is still primarily case-based, rather than regression-based: Detected trends influence case adaptation, but the primary information source is still cases.

As our case data changing over time, we selected a college summary from the magazine *U.S. News and World Report.*[3] The data we used included information on 1302 colleges, with 28 features for each one (e.g., enrollment, student test scores, etc.). The task was to predict tuition costs. Because multi-year data was not available, we simulated the increase as a normal distribution of increases around an annual inflation rate.

We used the following simple strategy for detecting and responding to error trends. The system records and monitors the percent errors between retrieved cases and evaluations. A cumulative error level is maintained by summing successive error percentages, with the expectation that accumulated percent errors due to random fluctuations (both positive and negative) will remain below a reasonable threshold magnitude. If the activation level persists above the threshold value for a specified amount of time, the system triggers a statistical analysis for possible underlying error trends. In the current system, the percentage error trend is approximated by performing a simple linear regression analysis on the sequence of error data. A maintenance rule is then installed that uses the computed regression line to forecast the percentage error for the current year and modifies cases according to the predicted error value as they are retrieved.

Experiments used query samples of 5 to 20 probes from the case set for each year over a 10 to 20 year span, selecting queries by two methods. The first method constructed a random problem distribution by selecting query cases at random. The second method constructed a highly regular problem distribution by restricting the query population to a set of similar instances, according to the system's similarity metric. The samples were used as probes in their respective years, over the varying year spans. The underlying annual inflation rate was varied in separate experiments between 2 and 5 percent for each year, which fluctuated according to a random normal distribution to represent yearly variations. Average error rates were measured for the baseline (no learning), case learning alone, maintenance alone, and combined case learning/maintenance. Each experiment was repeated 10 times, each time re-selecting the query sample, to obtain results on average.

While the results did not give a clear picture of how adjustments in individual parameters affected the outcomes, a general picture did emerge. With a random problem distribution, case learning performed better than the baseline, trend-based maintenance performed better than case learning, and the combination

[3] Available from http://lib.stat.cmu.edu/datasets/.

gave equivalent or better results. With the regularized problem distribution, the combination performed best, followed by case learning, then maintenance, and finally the baseline. A representative trial with an inflation rate of 2 percent over 15 years and sample size of 5 queries/year gave the following results. The randomized distribution showed average errors of 18 percent in the baseline, 17 percent in case learning, and 14 percent in both maintenance and combined trials. The regularized distribution showed average errors of 18 percent in the baseline, 14 percent with maintenance, 13 percent with case learning, and 12 percent in the combined trials.

The experiments point to some interesting observations. First, they suggest that maintaining existing cases can be as effective as learning new cases, and that augmenting case learning with diachronic maintenance can be beneficial. Second, it is worth noting that the individual trials of maintenance alone produced highly consistent results, while the individual trials involving case learning fluctuated a great deal in producing the average. This may indicate that detecting general trends is a more stable method of dealing with change over time than case learning. Third, we note that typical problem distributions will likely fall somewhere between the extremes of uniform sampling (where maintenance strategies alone were better than case learning) and highly focused sampling (where case learning worked better). Consequently, more experiments will be required to determine the interplay of the two along varying levels of problem-distribution regularity.

8.2 Addressing Hot Spots

A second potential use of trend detection is to respond to "hot spots" in the case base. In practice, case accesses are often non-uniform. For example, a primary motivation for the development of the GizmoTapper CBR support system for Broderbund computer games was to aid the Broderbund help desk in handling the increased queries it received soon after Christmas (**?**). The problem patterns for any domain are likely to be strongly domain-specific, but if those patterns can be detected automatically the system may be able to optimize access to information that is likely to be in demand.

To observe query distribution patterns in a real-world information source, we gathered data on accesses to Indiana University web pages for various on-line information repositories. These pages provide academic information (e.g., requirements for the BA degree) as well as homework assignments, etc. A sampling of access results for a year of logs are shown in Figure 1, with each band reflecting the total accesses to files within the directory. (Numbers of accesses are normalized to show the percent of maximum accesses per month from January 1998 to February, 1999. Patterns that might not have been expected (but that are easily explainable) emerge. For example, department academics pages are heavily accessed in the Fall (presumably by new students), but less frequently accessed in the Spring, as students become familiar with policies, and seldom in the summer. Pages for classes offered in Spring and Fall reflect that in their accesses. Temporal patterns are not always present—no pattern is apparent in the "Types Forum" accesses at the front of the graph—but there appears to be considerable regularity.

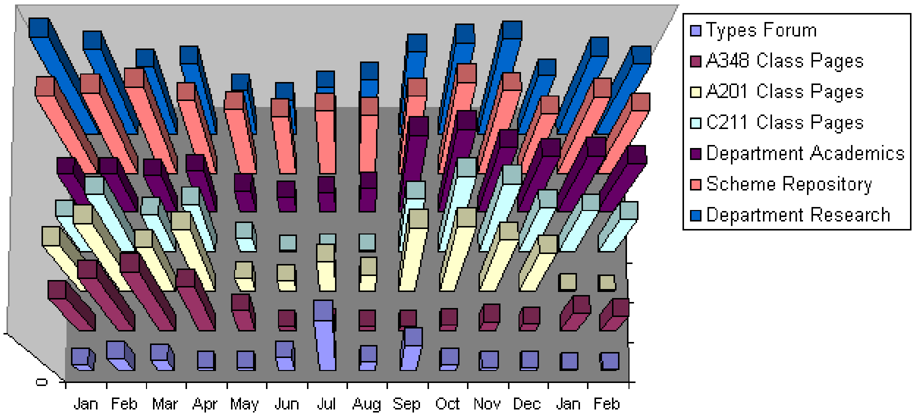

Fig. 1. Web page accesses by month.

Various methods could be used to detect or predict hot spots, such as clustering on the problems processed, predicting problem distributions from a model of the task the CBR system serves (if available), or collecting user profiles that associate users with particular access patterns. Once a hot spot has been hypothesized, the problem-distribution regularity formula can be applied to measure the adequacy of its coverage. Insufficient coverage is a sign to examine the current problem sequence for new hot spots. We are preparing an experiment to compare different hot spot detection strategies for different input problem sequences.

9 Considerations for costs and benefits

The processes described here depend on processing steps that increase the overhead of the CBR system, such as processes for trend detection and for reorganizing the case base in response to hot spots. More study must be done on the costs involved, but there may be important mitigating factors. First, trend analysis can be done off-line, when the system is otherwise idle. Second, in interactive CBR systems, cost and benefit analysis must weigh not only the costs incurred by the system, but also those avoided by the user. If trend analysis can, for example, warn the user of environmental changes that render prior cases obsolete, the real-world benefits may be substantial (e.g., for a realtor setting the price of a house). This may counterbalance increased computation costs.

10 Conclusions

The definitions presented here are useful for three reasons. First, they delineate the factors that affect regularity assumptions for CBR and their relationships—that regularity is not a property of the system or world individually but of the relationship between task, system, and the external world. Second, they provide a quantitative criterion for comparing the performance of particular CBR systems. Third, and most important for this paper, is that by giving standards for measuring regularity, they also give standards for detecting changes that require maintenance.

As CBR systems are more widely fielded for long-term use, it will become necessary to monitor both problem-solution regularity and problem-distribution regularity assumptions and to respond intelligently when they fail. This paper provides a practical starting point for how to detect and respond to situations in which the reuse of experiences goes wrong.

References

[Cheetham and Graf, 1997] W. Cheetham and J. Graf. Case-based reasoning in color matching. In *Proceedings of the Second International Conference on Case-Based Reasoning*, pages 1–12, Berlin, 1997. Springer Verlag.

[Faltings, 1997] Boi Faltings. Probabilistic indexing for case-based prediction. In *Proceedings of the Second International Conference on Case-Based Reasoning*, pages 611–622, Berlin, 1997. Springer Verlag.

[Fox and Leake, 1995] S. Fox and D. Leake. Using introspective reasoning to refine indexing. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 391–397, San Francisco, CA, August 1995. Morgan Kaufmann.

[Fox, 1995] S. Fox. *Introspective Reasoning for Case-based Planning*. PhD thesis, Indiana University, 1995. Computer Science Department.

[Harries *et al.*, 1998] M. Harries, K. Horn, and C. Sammut. Learning in time ordered domains with hidden changes in context. In *Papers from the AAAI 1998 Workshop on Predicting the Future: AI Approaches to Time-Series Problems*, pages 29–33. AAAI, 1998.

[Kolodner, 1993] J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Kriegsman and Barletta, 1993] M. Kriegsman and R. Barletta. Building a case-based help desk application. *IEEE Expert*, 8(6):18–26, December 1993.

[Lane and Brodley, 1998] T. Lane and C. Brodley. Approaches to online learning and concept drift for user identification in computer security. In *Papers from the AAAI 1998 Workshop on Predicting the Future: AI Approaches to Time-Series Problems*, pages 64–70. AAAI, 1998.

[Leake and Wilson, 1998] D. Leake and D. Wilson. Case-base maintenance: Dimensions and directions. In P. Cunningham, B. Smyth, and M. Keane, editors, *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, pages 196–207, Berlin, 1998. Springer Verlag.

[Redmond, 1992] M. Redmond. *Learning by Observing and Understanding Expert Problem Solving*. PhD thesis, College of Computing, Georgia Institute of Technology, 1992. Technical report GIT-CC-92/43.

[Salganicoff, 1997] M. Salganicoff. Tolerating concept and sampling shift in lazy learning using prediction error context switching. *Artificial Intelligence Review*, 11(1-5):133–155, 1997.

[Smyth and Keane, 1995] B. Smyth and M. Keane. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 377–382, Montreal, August 1995. IJCAI.

[Smyth and Keane, 1996] B. Smyth and M. Keane. Design à la Déjà Vu: Reducing the adaptation overhead. In D. Leake, editor, *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press, Menlo Park, CA, 1996.

[Smyth and McKenna, 1998] B. Smyth and E. McKenna. Modelling the competence of case-bases. In P. Cunningham, B. Smyth, and M. Keane, editors, *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, pages 208–220, Berlin, 1998. Springer Verlag.

[Talebzadeh *et al.*, 1995] Houman Talebzadeh, Sanda Mandutianu, and Christian Winner. Countrywide loan-underwriting expert system. *AI Magazine*, 16(1):51–64, 1995.

[Veloso, 1994] M. Veloso. *Planning and Learning by Analogical Reasoning*. Springer Verlag, Berlin, 1994.

[Watson, 1997] I. Watson. *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann, San Mateo, CA, 1997.

[Zhu and Yang, 1998] J. Zhu and Q. Yang. Remembering to add: Competence-preserving case-addition policies for case based reasoning. 1998.