Facilitating CBR for Incompletely-Described Cases: Distance Metrics for Partial Problem Descriptions[⋆]

Steven Bogaerts and David Leake

Computer Science Department, Indiana University, Lindley Hall 215
150 S. Woodlawn Avenue, Bloomington, IN 47405, U.S.A.
{*sbogaert, leake*}*@cs.indiana.edu*

Abstract. A fundamental problem for case-based reasoning systems is how to select relevant prior cases. Numerous strategies have been developed for determining the similarity of prior cases, given full descriptions of the problem at hand, and situation assessment methods have been developed for formulating appropriate initial case descriptions. However, in real-world applications, attempting to determine all relevant features of a new problem before retrieval may be impractical or impossible. Consequently, how to guide retrieval based on partial problem descriptions is an important question for CBR. This paper examines the problem of assessing similarity in partially-described cases. It proposes a set of similarity assessment strategies for handling missing information, evaluates their performance and efficiency on sample data sets, and discusses their tradeoffs.

1 Introduction

Case-based reasoning (CBR) systems solve new problems by retrieving cases capturing the solutions of similar prior problems, and adapting their solutions to fit new needs. Determining the most relevant prior cases is a fundamental issue for CBR systems, and may require special methods when a full case description is not immediately available. For example, in a common applied CBR approach, *conversational CBR* (CCBR) (Aha & Breslow 1997), users build up a problem description by successively answering questions, as the system incrementally ranks candidate cases and questions based on the partial information available. The more accurate the distance measure used in this process, the more quickly the system will be able to point the user to the most applicable cases. Likewise, the ability to rank cases based on partial information may be essential when feature values are costly to determine or simply unavailable. Thus an important issue for CBR is how to assess the similarity of *partial problem descriptions*—problems with incomplete feature descriptions.

Considerable attention has been devoted to the process of refining initial problem descriptions through situation assessment (Kolodner 1993), and handling incomplete problem descriptions is a fundamental problem for CCBR. In diagnostic tasks, for example, only a partial set of features may initially be available. Consequently, every CCBR system includes methods for handling incrementally-built case descriptions,

[⋆] This material is based upon work supported by NASA under award No NCC 2-1216 and by the U.S. Department of the Navy, NSWC Crane Division, under contracts N00164-04-C-6514 and N00164-04-C-6515. **Copyright ©Springer Verlag 2004.**

and research has addressed the problem of deciding which features to request when elaborating a partially-described case during CCBR (e.g., (Carrick *et al.* 1999)). However, comparatively little attention has been given to examining alternative similarity assessment methods for cases with missing features. A better understanding of the performance of alternative strategies, their tradeoffs, and their applicability, could enable more effective retrieval of partially-described cases and could also provide useful information for guiding the CCBR process. Because cases in the case base may themselves have partial descriptions, understanding how to handle partial descriptions could also be valuable for case base maintenance (Leake *et al.* 2001), to determine when and how to augment partial descriptions of stored cases.

This paper first discusses general issues affecting similarity judgments for partially-described cases. It then examines a set of similarity assessment strategies, including two simple baseline strategies and two more complex strategies designed to take advantage of information offered by the case base to predict feature values. The strategies apply to feature-vector representations for any ordinal features, i.e., features whose values belong to an ordered set; these may be numeric, or may belong to other categories provided that notions of distance and average can be defined (e.g., for a finite set, the "average" might be determined by a vote). The first method, *Default Difference*, is a baseline method which simply assigns a fixed default distance whenever the values of one or both features are missing (e.g., if this distance is 0, missing features are assumed to match perfectly). The remaining methods use additional information extracted from the case base as a whole: *Full Mean*, another baseline, treats each missing value as if it were the mean feature value. *NN Mean* takes a similar approach, but relies on local information, using the mean values of "near-by" cases. A drawback of *NN Mean* is its increased expense to compute the predicted feature value, which can be extreme when many cases must be compared to the current situation. *Region Mean* addresses this problem by generating a case base of prototypical cases, providing a local approximation to use to predict missing features without additional computation. Finally, we consider the use of composite methods involving combinations of these strategies. An experimental evaluation compares (1) the ability of each method to select the most similar cases, for differing levels of partial information—which reflects the number of questions that must be answered for a CCBR system to achieve a desired level of accuracy, (2) their efficiency at providing their information, and (3) the potential benefit of combined strategies. After comparing these performance issues, we develop general hypotheses for the applicability of the methods and their tradeoffs.

1.1 Handling Unknown Features

A simple example illustrates the subtlety of handling unknown features. Consider a domain in which cases are described by a feature vector of four features, $[f_1, f_2, f_3, f_4]$, and for which the system must solve a problem p, for which only the values of the first three are known: $[5.0, 6.0, 7.0, -]$. Let $distance(p_1, p_2)$ denote the distance between two problems p_1 and p_2. Suppose that the case base contains two cases, c_1, described by $[5.1, 6.0, -, -]$ and c_2, described by $[5.0, -, -, -]$. Note that in the following, we will use the name of the case as a shorthand for referring to the problem it solves.

In order to select the right case, the system must predict whether $distance(p, c_1)$ or $distance(p, c_2)$ is smaller. More features of c_1 are known than c_2; this guarantees that $c_1[1]$ has no difference from $p[1]$, and $c_1[0]$ has an apparently small difference from $p[0]$. However, c_2 might be more promising. Because $c_2[0]$ has no difference from $p[0]$, the potential minimum difference between c_2 and p is smaller, even though selecting c_2 entails more risk, due to possible differences in the unknown features. Likewise, a difference of 0.1 between $p[0]$ and $c_1[0]$ *could* be important, and perhaps even so significant that the exact match on feature 1 is inconsequential. Thus determining how to treat missing features depends on both (1) the importance of known differences and (2) the potential importance of unknown features, given their likely values and the user's tolerance for the level of uncertainty that they entail for the quality of results.

Even the selection of quality measure may involve subtle considerations. For example, possible quality measures could include *rank quality*, which measures how close the top-ranked cases are to the actual best match, or—if the specific values of the predicted distances are important—the *error* in the distance prediction. For example, error might be important in medical domains, if a differential diagnosis accepts a diagnosis when it appears sufficiently superior to its competitors. Error may also be important when the system provides the user with distance estimates to help guide the choice of cases to examine.

2 Strategies for Handling Unspecified Features

For any particular domain, domain knowledge may suggest specific assumptions or strategies for handling partially-described problems. Here we examine simple domain-independent strategies for assigning distances between pairs of corresponding features, within the framework of the standard distance function:

$$distance(r_1, r_2) = \sqrt{\sum_i w_i [d(r_1[i], r_2[i])]^2}$$

If F_i represents the set of possible feature values for the i^{th} feature, including a value used to designate unknown features, these are functions $d_i : F_i \times F_i \rightarrow [0, \infty)$.

2.1 Default Difference(x)

A simple baseline strategy is to treat the distances between unknown features as zero. This corresponds to a typical strategy of considering only differences in known features. This approach can be generalized to assign a fixed default difference, x, whenever either feature is unknown. *Default Difference*(x), the corresponding similarity assessment strategy, is defined as:

$$d_i(r_1[i], r_2[i]) = \begin{cases} x & r_1[i] \text{ and/or } r_2[i] \text{ unknown} \\ |r_1[i] - r_2[i]| & \text{otherwise} \end{cases}$$

Default Difference with x equal to 0 can be seen as an "optimistic" measure. When x equals 0, a completely unknown problem has distance 0 from all cases; this might be considered appropriate because every case is potentially a perfect match. Alternatively, if the maximum possible feature distances are bounded and equal across all features, setting x to the maximal possible difference corresponds to a "pessimistic" measure.

This simple metric illustrates an interesting asymmetry between handling partially-specified *input problems* and handling *stored cases* whose problem descriptions are partially-specified. When stored cases include complete problem descriptions, different values of x may change the magnitude of computed difference values and the spacing between cases ranked by similarity, but will not affect the cases' *ranking* by difference values. However, if features may be missing from problem descriptions in stored cases as well as input cases, increases in x may change the ranking, causing the metric to favor stored cases for which more features are known.

Default Difference assumes a fixed difference for a pair of features whenever they are missing from *either* the input case or a stored case. A problem with this simple approach is that it may neglect useful information: if a problem feature in either the input or stored case has an atypical value, it is reasonable to consider the missing feature's value less likely to be similar. This should affect the prediction of a good match between the cases, but *Default Difference* does not take this into account. The next method, *Full Mean*, addresses this deficiency.

2.2 Full Mean

Full Mean exploits global feature information to estimate missing values, by replacing missing feature values with the mean values for those features when calculating similarity. If the feature is not known in any of the stored cases, it assigns a default difference value x. More formally, let

$$CasesKnowingFeature(i, CB) = \{c \in CB | f_i \text{ known in c}\}$$

and let $\mu(i, CB, x)$, the average of known value of feature i in the case base, with default x for completely unknown features, be defined by:

$$\mu(i, CB, x) = \begin{cases} \frac{\sum_{CasesKnowingFeature(i,CB)} c[i]}{|CasesKnowingFeature(i,CB)|} & CasesKnowingFeature(i, CB) \neq \phi \\ x & \text{otherwise} \end{cases}$$

Then with *Full Mean*,

$$d(r_1[i], r_2[i]) = |EstimatedValue(r_1, i, x) - EstimatedValue(r_2, i, x)|$$

where

$$EstimatedValue(r, i, x) = \begin{cases} r[i] & r[i] \text{ is known} \\ \mu(i, CB, x) & \text{otherwise} \end{cases}$$

Because the means for each feature in the case base can be precomputed offline and updated cheaply online as cases are added or removed, this is a low-cost strategy.

Although *Full Mean* makes better use of global feature information than *Default Difference*, it has potential drawbacks. First, like all average-based approaches in this paper, it considers only the average feature value, independent of the feature's distribution (which might be better captured, e.g., by the mean or mode). Second, it ignores possible dependencies between features, although the expected value of a feature may change dramatically based on the value of other features. For example, even if the average age of passenger cars is 8 years, predicting 8 years of age for a car would be misleading if it were also known that the car had only been driven 100 kilometers. The next strategy, *NN Mean*, attempts to better reflect local dependencies by taking a more case-based approach, using similar cases to predict feature values.

2.3 NN Mean

Nearest Neighbor Mean, or *NN Mean*, responds to *Full Mean*'s problems with a more case-based approach, predicting feature values based on the values of similar cases. Its premise is that nearby cases will be good predictors of feature values. Intuitively, if $r[i]$ is unknown and $Near_{r,i}$ is the set of all cases near r that know feature i, then a good predictor is the average feature value over $Near_{r,i}$. Unfortunately, there is one catch to this approach: Defining "nearby" cases requires predicting inter-case distances, which is the very problem that *NN Mean* is intended to address.

In *NN Mean*, we address this problem by recursively drawing on the distance metrics from this paper for the "internal" similarity computation. For example, *Default Difference*(x) can be used as an internal strategy to estimate distances for finding nearby cases, and then the k closest cases, or all cases within a distance threshold, can be used to obtain a mean according to *NN Mean*. We will denote the "internal" strategy as an argument to *NN Mean*, as in *NN Mean*(*Default Difference*(x)). If feature dependence information is available (though this often is not the case), an additional variant on the *NN Mean* strategy is to use only the dependent features in the search for nearby cases. We call this approach *NN Mean* Dep.

NN Mean is an expensive strategy. Unlike *Full Mean*, the average values for a feature cannot be precomputed, because they depend on r. No matter what internal strategy is used, at a minimum a new retrieval is required for each unknown feature.

2.4 Region Mean

Region Mean attempts to avoid the expense of *NN Mean* yet maintain its advantages over *Full Mean* by precomputing near means at various points in the case space, and predicting means based on the nearest precomputed cases to the input problem. In the offline process, the precomputation algorithm is:

- Cluster the case base and find a prototype for each class. We apply k-medoid clustering.[1]
- For each prototype p_j
    - For each feature i

[1] k-medoid clustering is robust to outliers and independent of the order in which objects are considered. For a comparison with other clustering methods, see (Kaufman & Rousseeuw 1990).

∗ Let $Class_{p_j,i}$ be the set of cases in the equivalence class with prototype p_j that know feature i. Determine $\mu(i, Class_{p_j,i}, x)$, the mean value of feature i over $Class_{p_j,i}$, as follows:

$$\mu(i, Class_{p_j,i}, x) = \begin{cases} \frac{\sum_{c \in Class_{p_j,i}} c[i]}{|Class_{p_j,i}|} & Class_{p_j,i} \neq \phi \\ x & \text{otherwise} \end{cases}$$

The online computation for *Region Mean* is analogous to *Full Mean*. The key difference is that $EstimatedValue(r, i, x)$ retrieves the p_j closest to r, and uses $\mu(i, Class_{p_j,i}, x)$, instead of *Full Mean*'s $\mu(i, CB, x)$.

As for *NN Mean*, an internal difference metric is required, this time to determine the nearest prototype p_j to r, as well as to measure the difference between problems in clustering. Again this can be found by recursively using any method described in this paper, provided the final method is defined. For example, *Default Difference*(x) could be used for finding the nearest prototype and for clustering. We would denote this strategy as *Region Mean*(*Default Difference*(x)).

2.5 Composite Strategies Exploiting Dependency Information

If information can be obtained about feature dependencies—which may itself be a significant challenge—it may be beneficial to apply a composite strategy, using one strategy for independent features and another for dependent features.

If a feature is independent, then, by definition, the values of other features are not helpful in predicting its value. Thus the best that can be hoped for is to simply use global information such as the average value across the entire case base; that is, to use *Full Mean*. Because *Full Mean* is inexpensive to compute, it is an obvious choice given a priori knowledge that a feature is independent. Only when handling dependent features are other strategies much more likely to be successful.

Because composite strategies use one strategy for independent features and another for dependent features, we write them in the form *independent-strategy/dependent-strategy*. For example, a composite strategy using *Full Mean* for independent features and *Region Mean*(*Default Difference*(0)) for dependent features is written *Full Mean/-Region Mean*(*Default Difference*(0)).

3 Experiments

The previous discussion raises a number of general questions for comparing similarity assessment strategies for partial problem descriptions:

1. Their efficiency
2. Their accuracy for ranking candidate cases
3. Their accuracy for estimating difference levels between candidate cases
4. Their accuracy when different levels of information are available

It also raises some strategy-specific questions, on how performance is affected by:

1. Choice of internal strategy for *NN Mean*
2. Cluster count during initial clustering for *Region Mean*
3. Internal strategy for *Region Mean*
4. Composite strategies with different methods for independent/dependent features

To answer these questions, we tested the previous strategies for a number of domains. Our experiments focused on the ability of the methods to identify similar cases when some input features were missing, primarily for case bases in which all cases had complete problem descriptions.

3.1 Performance Measures

Three performance measures were used in the experiments:

- **Time:** The efficiency of the approaches is compared by measuring the CPU time required for the strategies to calculate distance values between the target and all the cases in the case base.
- **Normalized Absolute Error:** Given a strategy s, a partially-described target problem $\hat{t}$ generated by removing feature values from a completely-described problem t, and a case c in the case base, we define the absolute error as the difference between the actual distances between t and c, and the distance predicted when only $\hat{t}$ is known: $error_s(\hat{t}, p) = |distance(t, c) - distance_s(\hat{t}, c)|$.
  This metric is useful within a domain, to indicate of how misleading a predicted distance value may be. However, it may be less useful for comparing performance across domains, because it is sensitive to factors such as scaling of distance values. In order to facilitate comparison of errors across domains, we normalize absolute errors onto [0, 1], by dividing the absolute distance by the maximum observed distance in that case base (the distance between the two maximally distant cases in the case base). Our results report the percent of maximum observed distance between cases.
- **Rank Quality:** The rank quality measure reflects the ability of a distance metric to generate a ranking in which the quality of the top suggested cases is similar to the quality of the top cases which would be suggested if all features were known. Given a strategy s and a partial problem $\hat{t}$, rq_s measures the percent increase in distance between the input problem and the top suggested cases, compared to the true top cases. Thus it measures how much worse the top suggested cases are when only $\hat{t}$, rather than t, is known. To reflect that users in a CCBR system may be most likely to focus on the top-ranked cases, our metric weights suggestions by their order in the ranking: having a top-ranked case closest to the real top-ranked case is considered most important, with lower-ranked suggestions less important. More precisely, let $ClosestProb$ be the problem of the case that is nearest to t (when all features are known). The rank quality ratio is defined as:

  $rq_s = \frac{\sum_p w(p)*ratio(p)}{\sum_p w(p)}$, where $ratio(p) = \frac{distance(t,p)}{distance(t,ClosestProb)}$

  and $w(p)$ is a function that assigns a weight to $ratio(p)$ that favors higher-ranked cases. Let $rank_s(p)$ be the 0-based rank of problem p according to strategy s. In our experiments, we set $w(p) = max(5 - rank_s(p), 0)$. Thus, only the top 5 cases had a non-zero weight.

Note that normalized absolute error and rank quality both measure the ability of the strategies to predict real inter-case distances, according to a given distance measure. They do not directly compare solution accuracies, which would depend on the quality of the given distance measure.

3.2 Experimental Domains

The experiments were conducted in four domains, three from the University of California, Irvine, Machine Learning Repository (Blake & Merz 1998), and one artificial domain to observe performance for strongly-correlated problem description features:

1. Ecoli: 336 cases, 7 numerical features, predicting one of 8 protein localization sites.
2. Pima: 768 cases, 8 numerical features, predicting positive or negative diabetes test results in members of the Pima Native American population.
3. Liver: 345 cases, 6 numerical features, predicting the presence or absence of a liver disorder.
4. Dep7: Artificial domain, 300 cases, 7 numerical features, predicting a single numerical value. There are strong dependencies between the features: $f_0 \sim N(0, 10)$, $f_1 \sim N(f_0, 2)$, $f_2 \sim N(f_1^2 - f_0, 10)$, $f_3 \sim N(0, \pi)$, $f_4 \sim N(sin(f_3), 10)$, $f_5 \sim N(10, 20)$, $f_6 \sim N(-20, 10)$.

For all experiments, the underlying similarity metric was the Euclidean distance function of section 2, with all features given a weight of 1.

3.3 General Procedure

All implementations and experiments were done using the Indiana University Case-Based Reasoning Framework (IUCBRF) (Bogaerts & Leake 2004). IUCBRF is a Java framework, freely-available for research, designed to facilitate rapid and modular CBR system development. The general experimental procedure was as follows. Let t be the fully known target problem, $\hat{t}$ a partial target problem generated by removing features of t, p a problem (fully known) from a case in the case base, and s a similarity assessment strategy. All experiments were done for each feature prediction strategy, and were repeated 300 times per strategy (except as stated otherwise), with results averaged. For each feature prediction strategy s, steps were:

- Perform any required initialization (e.g., any strategy involving *Region Mean* builds a partition of the case base)
  - Perform leave-one-out testing. For each case c in the case base,
    - $*$ Hide c, and use c as the basis for generating a partial target $\hat{t}$. Initially, no features of $\hat{t}$ are revealed.
    - $*$ For each case c in CB - $\{t\}$, measure $distance_s(c, \hat{t})$ according to strategy s, and sort the case base by these distances.
    - $*$ Obtain performance measurements as described above

- If $\hat{t} = t$, exit loop.
- Else randomly choose a feature to "reveal" in $\hat{t}$ (obtained from t), and loop. The random choice simulates a user presenting a feature to the system, outside the system's control.

Thus for each t, data is collected for each strategy's calculated distances between the target problem and the remainder of the case base, from 0 features of the target problem revealed, through all its features revealed.

3.4 Classes of Tests

There were three classes of experiments, with different independent variables:

1. Cross-Domain Comparison: We compared performance for 12 strategies and variants, for four case bases of 300 cases, with *Region Mean* based on partitions created from 20 clusters.
2. Cluster Count Comparison: Performance of the 12 strategies was compared to three versions of *Region Mean*, respectively using 50, 20, and 6 clusters, each for the full ecoli case base of 336 cases.
3. Comparison for Unknown Features in the Case Base: Instead of using a fully-known case base, as is done in the other experiment classes, this experiment assessed performance with partially-described cases in the case base, for 100%, 75%, 50%, and 25% chance that a feature in the case base is known. This used the Pima case base of 768 cases, with *Region Mean* using 10 clusters.

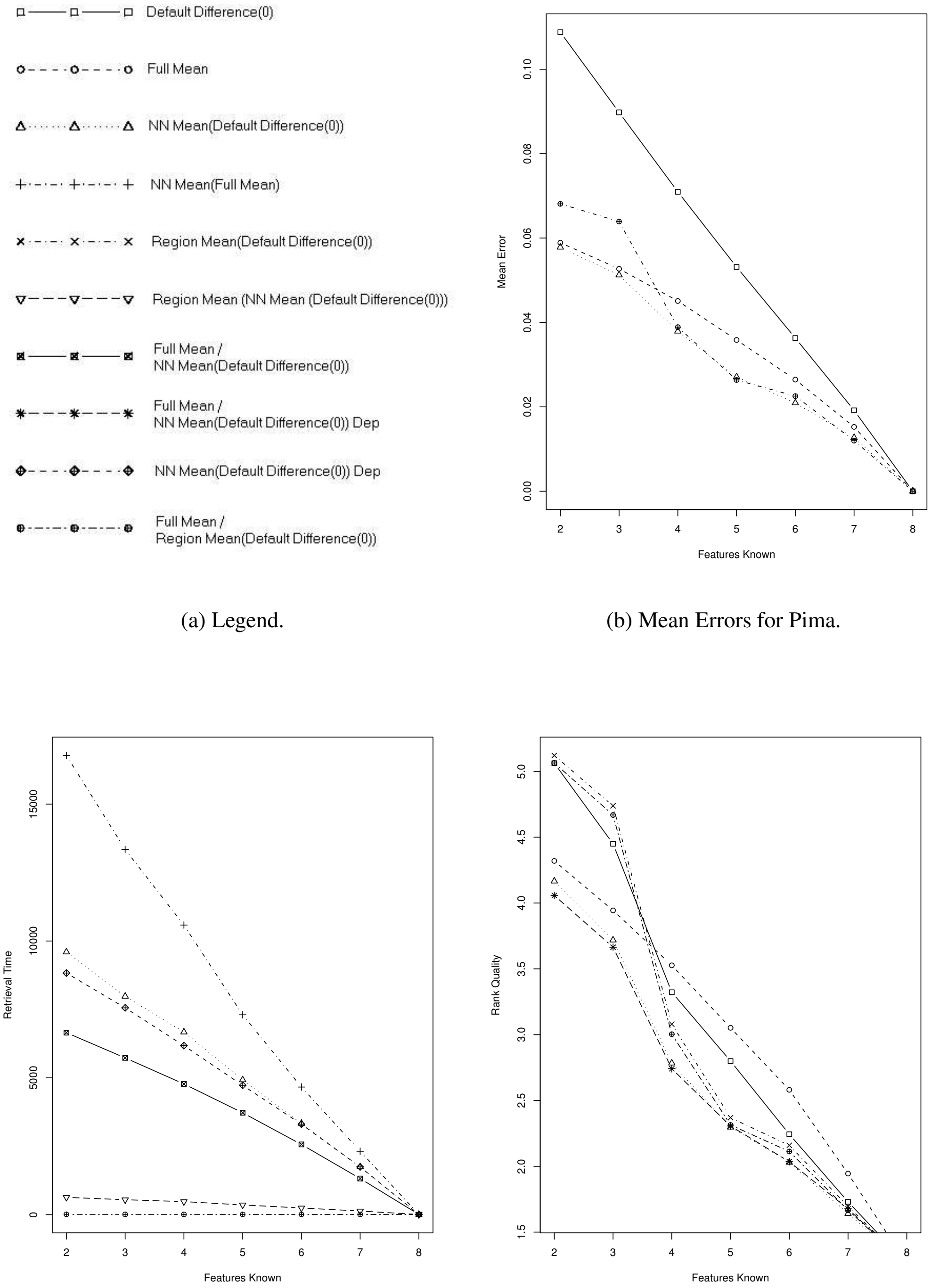
4 Results

The comparative results in each of the UCI domains were remarkably similar. Figure 1 illustrates them with examples from the Pima domain. Figure 1 (a) lists the range of strategies considered in the experiments. However, because some strategies had almost identical performance, only a subset of lines is included in each figure, with similarities described in the text.

4.1 Cross-Domain Comparison

Figure 1 (b) shows the normalized absolute error as a function of the number of features known, for selected strategies in the pima domain. Note that *Default Difference*(0) is the worst strategy, with *Full Mean* in next to last place when 4 or more features are known. *Full Mean*/*Region Mean*(*Default Difference*(0)) initially performs worse, but becomes comparable to *NN Mean*(*Default Difference*(0)) when 4 or more features are known. The competetive performance of *Full Mean* is interesting in light of its much lower cost than *NN Mean*(*Default Difference*(0)), as shown in Figure 1 (c).

Figure 1 (c) compares the strategies' efficiency. *NN Mean*(*Full Mean*) is slowest, followed by *NN Mean*(*Default Difference*(0)), *NN Mean*(*Default Difference*(0)) Dep, and *Full Mean*/*NN Mean*(*Default Difference*(0)). The remaining strategies, essentially



(a) Legend.

(b) Mean Errors for Pima.

(c) Retrieval Times for Pima (ms).

(d) Rank Quality for Pima.

Fig. 1. a. Legend for (b), (c), and (d). b. Error versus number of features known, for selected strategies. c. Time in ms to select top-ranked case versus number of features known, for selected strategies. d. Rank quality ratios versus number of features known, for selected strategies.

Strategy	**Error,**	**Error,**	**Time,**	**Time,**
	4 Known	**6 Known**	**4 Known**	**6 Known**
Default Difference(0)	0.0710	0.0363	5.70	6.03
Full Mean	0.0451	0.0265	8.77	8.10
NN Mean(*Default Difference*(0))	0.0379	0.0209	6671.06	3332.70
NN Mean(*Full Mean*)	0.0447	0.0260	10583.25	4663.49
Region Mean(*Default Difference*(0))	0.0392	0.0226	9.88	8.75
Region Mean(*Full Mean*)	0.0433	0.0247	10.23	8.96
Region Mean(*NN Mean*(*Default Difference*(0)))	0.0373	0.0209	476.92	243.49
Full Mean/*NN Mean*(*Default Difference*(0))	0.0380	0.0209	4776.45	2566.61
Full Mean/*NN Mean*(*Default Difference*(0)) Dep	0.0374	0.0209	4648.30	2552.25
NN Mean(*Default Difference*(0)) Dep	0.0374	0.0209	6175.21	3305.66
Full Mean/*Region Mean*(*Default Difference*(0))	0.0389	0.0225	10.48	9.07
Full Mean/*Region Mean*(*NN Mean*(*Default Difference*(0)))	0.0373	0.0209	10.51	9.07

Table 1. Mean Errors for Pima

all those not involving *NN Mean* except as a prototype finder of *Region Mean*, were fast, requiring 5-12 ms on a Sun Blade 1000 (750Mz) to rank all cases in the pima domain.

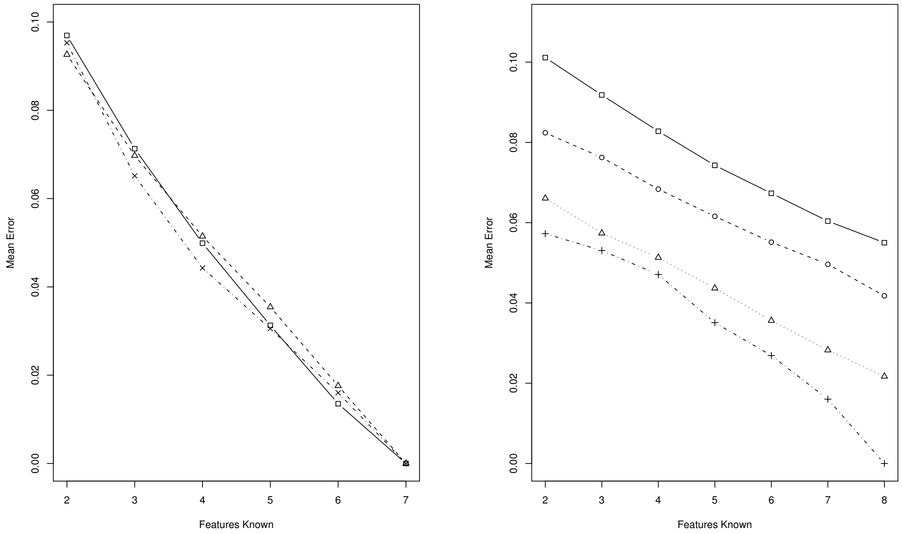
Figure 1 (d) shows the rank quality for selected strategies in the pima domain. Note that this graph can also be used to determine the number of questions, on average, that a CCBR system would require to achieve a particular rank quality. Here *Full Mean* generally has the worst performance, followed by *Default Difference*(0). *Region Mean*(*Default Difference*(0)) and *Full Mean*/*Region Mean*(*Default Difference*(0)) start comparatively poorly, but catch up quickly to *NN Mean*(*Default Difference*(0)) and *Full Mean*(*NN Mean*(*Default Difference*(0))) *Dep*. Table 1 summarizes performance for all the strategies tested, for 4 and 6 features known.

4.2 Cluster Count Comparison

In this experimental setup, the *Region Mean* strategies were run for 6, 20, and 50 clusters, in the ecoli domain with a case base of 336 cases. Note that these cluster counts correspond to approximately 2%, 6%, and 15% of the number of cases.

Figure 2 (a) shows a sample of results, for *Region Mean*(*NN Mean*(*Default Difference*(0))). In the lines in this figure, points marked with □ correspond to performance for 50 clusters, × to 20 clusters, and △ to 6 clusters. The results show small improvements when more clusters are used, but also show that the behavior is generally robust to the cluster count. These results were typical for each of the *Region Mean* strategies, and also for the rank quality measure. This suggests that low cluster counts may be sufficient. Because speed increases with lower cluster counts, this result is encouraging for the efficiency of *Region Mean*.

The relative processing times versus cluster counts were also consistent across each of the *Region Mean* strategies. Experiments showed that strategies with 6 clusters were fastest, followed by 20, with 50 the slowest, although for each cluster count the difference in computation time was fairly small. This is as expected: Fewer clusters decreases time to find the applicable cluster, but even for a large number of clusters, there are at most a few dozen more prototypes that must be examined to find the nearest cluster, and



(a) Effect of Cluster Count on Errors for *Region Mean* Strategies.

(b) Effect of Chance Known on Errors for Composite Strategy.

Fig. 2. (a). The error for *Region Mean*(*NN Mean*(*Default Difference*(0))) for 50, 20, and 6 clusters. □ is for 50 clusters, × is for 20, and △ is for 6. (b) Error in Full Mean / Region Mean(Default Difference(0)) for a chance known of 100% (+), 75% (△), 50% (◯), and 25% (□).

the difference has limited effect on execution time due to the relatively larger constant overhead cost of the strategy.

4.3 Chance Known Comparison

In this experimental setup, each strategy's performance was examined for varying levels of missing information in the case base, in the pima domain, with 768 cases. As discussed above, previous experiments used a fully-known case base (a 100% chance that a case in the case base knows any given feature). This setup, however, examines not only a 100% chance known, but also 75%, 50%, and 25%. Runs were repeated a minimum of 186–270 times, with results averaged.

Figure 2 (b) shows the error using Full Mean / Region Mean(Default Difference(0)) for a chance known of 100% (+), 75% (△), 50% (◯), and 25% (□). Note that, as expected, error increases as the chance that a feature in the case base is unknown increases. Results were very similar for the other strategies.

5 Discussion

The experiments illustrate a number of interesting properties. First, we note that the commonly-used trivial strategies, *Default Difference*(0) and *Full Mean*, consistently produce comparatively poor results, as shown in the error and rank quality measures. Thus we would expect a considerable boost to the prediction accuracy of a CBR system faced with partial problems (or a conversational CBR system in which not all questions have been answered for a given session) when any of the more advanced strategies are used.

The results also provide information to help in selecting a more advanced strategy. As hypothesized and experimentally verified, strategies involving *NN Mean* are prohibitively slow, requiring on the order of several seconds to sort a fully-known case base of a few hundred cases against a single partial problem. However, the *Region Mean* strategies were developed to address this are fairly fast and provide comparable accurate to *NN Mean* strategies.

It is interesting to note that in limited circumstances, *NN Mean* may still be appropriate. Specifically, experiments show that *Region Mean*(*NN Mean*(*Default Difference*(0))) (and the related composite method) is not slowed down dramatically by the use of a *NN Mean* strategy as its prototype finder. This can be explained because, when the case base is fully-known, there is only one partial problem, the target, for which the nearest cluster must be found. Thus, *NN Mean* must be used only once to determine the nearest cluster. Once the nearest cluster is determined, the mean values associated with that cluster can be used for the partial problem in comparing it with the entire case base. In fact, for a partial case base, once the partition is created off-line, the nearest cluster of any case is already known and need not be recomputed. So even for a partial case base, *NN Mean* must only be computed once for a target partial problem.

If information regarding the dependence between features is available (either as domain knowledge, or calculated via statistical analysis) then a composite method can be used. As argued above, *Full Mean* is the suitable when a feature is independent, and strategy for dependent features should use *Region Mean*. Assuming that the dependence information is accurate, a composite method should be just as accurate, or even more accurate, than the dependent part alone applied to all features. In addition, because *Full Mean* is fast, a composite method of the form *Full Mean*/*Region Mean* would be faster than *Region Mean* alone, applied to all features.

We note that none of the approaches exploit statistical information about feature distributions. When that information is available, it may provide even more useful information. An area for future study is the application of the *representativeness assumption* (Smyth & McKenna 1999) to use existing cases in the case base to estimate feature value distributions.

6 Related Work

Distance metrics have been the subject of extensive study in CBR (e.g., (Wettschereck, Aha, & Mohri 1997; Bergmann 2002)). CCBR systems must always include methods for handling partially described problems, and a number of methods have been applied.

One common approach in CBR and instance-based learning is to assume a maximal difference between missing features (Witten & Frank 2000, p. 115), which is similar to *Default Difference*(x) for a large value of x.

To our knowledge, how to handle missing features in distance metrics and the tradeoffs between alternative strategies have received only limited study in the CBR community. However, missing features have been considered in a number of studies in machine learning. For example, in decision tree induction, Mingers (1989) uses a strategy similar to *Full Mean* as well as a strategy that assigns the most common feature value among training instances with the same classification, and Quinlan (1993) uses probability information on feature values while descending multiple paths of the tree. Other work has examined the theoretical learnability of a target function when features are missing (Decatur & Gennaro 1995; Goldman, Kwek, & Scott 1997).

CBR research has examined how to select the next question to ask in a dialogue (Aha, Breslow, & Munoz-Avila 2001; Kohlmaier, Schmitt, & Bergmann 2001), and how to select useful sets of cases to present in light of similarity and diversity concerns (Smyth & McGinty 2003). McSherry (2003) studies a related problem, the determination of when recommendation dialogues can be terminated without loss of solution quality, and compares the efficiency of alternative attribution-selection strategies, given a similarity metric in the spirit of *Default Difference*(0). However, these approaches assume a pre-existing method for assessing similarity based on partial descriptions; they do not examine which similarity metrics to use. Increased understanding of how to assess similarity for partial descriptions could have substantial benefits both for CCBR and for case-based recommender systems.

7 Conclusion

Being able to retrieve appropriate cases, based on partial information, is a fundamental problem for CCBR systems. This paper examines alternative strategies for addressing this problem. It compares a set of difference measures, evaluates their performance and efficiency on sample data sets, and discusses their tradeoffs as suggested by the experiments. It identifies difficulties in handling partial problem descriptions that may not be initially apparent, illustrates high-cost, high-accuracy strategies based on CBR, and shows that they may be effectively approximated by more efficient methods. This work provides a set of tools for building distance metrics for incompletely-described cases, and provides an initial foundation for further study of this area.

References

Aha, D., and Breslow, L. 1997. Refining conversational case libraries. In *Proceedings of the Second International Conference on Case-Based Reasoning*, 267–278. Berlin: Springer Verlag.

Aha, D.; Breslow, L.; and Munoz-Avila, H. 2001. Conversational case-based reasoning. *Applied Intelligence* 14:9–32.

Bergmann, R. 2002. *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*. Berlin: Springer.

Blake, C., and Merz, C. 1998. UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Bogaerts, S., and Leake, D. 2004. IUCBRF: A framework for rapid and modular CBR system development. In preparation.

Carrick, C.; Yang, Q.; Abi-Zeid, I.; and Lamontagne, L. 1999. Activating CBR systems through autonomous information gathering. In *Proceedings of the Third International Conference on Case-Based Reasoning*, 74–88. Berlin: Springer Verlag.

Decatur, S., and Gennaro, R. 1995. On learning from noisy and incomplete examples. In *Proceedings of the Eighth Annual ACM Conference On Computational Learning Theory*. ACM Press.

Goldman, S.; Kwek, S.; and Scott, S. 1997. Learning from examples with unspecified attribute values. In *Computational Learing Theory*, 231–242.

Kaufman, L., and Rousseeuw, P. 1990. *Finding Groups in Data: an Introduction to Cluster Analysis*. Wiley.

Kohlmaier, A.; Schmitt, S.; and Bergmann, R. 2001. A similarity-based approach to attribute selection in user-adaptive sales dialogues. In *Case-Based Reasoning Research and Development: Proceedings of the Fourth International Conference on Case-Based Reasoning*, 306–320. Berlin: Springer Verlag.

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

Leake, D.; Smyth, B.; Wilson, D.; and Yang, Q., eds. 2001. *Maintaining Case-Based Reasoning Systems*. Blackwell. Special issue of *Computational Intelligence*, 17(2), 2001.

McSherry, D. 2003. Increasing dialogue efficiency in case-based reasoning without loss of solution quality. In *Proceedings of the eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 121–126. San Mateo: Morgan Kaufmann.

Mingers, J. 1989. An empirical comparison of selection measures for decision-tree induction. *Machine Learning* 3(4):319–342.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Smyth, B., and McGinty, L. 2003. The power of suggestion. In *Proceedings of the eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 127–132. San Mateo: Morgan Kaufmann.

Smyth, B., and McKenna, E. 1999. Building compact competent case-bases. In *Proceedings of the Third International Conference on Case-Based Reasoning*, 329–342. Berlin: Springer Verlag.

Wettschereck, D.; Aha, D.; and Mohri, T. 1997. A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review* 11(1-5):273–314.

Witten, I., and Frank, E. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco: Morgan Kaufmann.