CBET: a Case Base Exploration Tool*

Paolo Avesani, Anna Perini and Francesco Ricci

Istituto per la Ricerca Scientifica e Tecnologica
Via Sommarive
38050 Povo (TN)
Italy
{avesani,perini,ricci}@irst.itc.it

Abstract. CBET is a software tool for the interactive exploration of a case base. CBET is an integrated environment that provides a range of browsing and display functions that make possible knowledge extraction from a set of cases. CBET is motivated by an application to training firemen. Here cases describe past forest fire fighting interventions and CBET is used to detect dependencies between data, acquire practical planning competences, visualize complex data, clustering similar cases. In CBET well rooted Machine Learning techniques for selecting relevant features, clustering cases and forecasting unknown values have been adapted and reused for case base exploration.

1 Introduction

This paper describes CBET (Case Base Exploration Tool), a system grown from the results of CHARADE, a decision support system for controlling environmental emergencies. A demonstrator based on the CHARADE platform was developed for managing forest fires emergencies [8]. A major component of the CHARADE demonstrator is a Case-Based Reasoning system for planning first interventions to forest fires. Case-Based Reasoning is a general problem solving methodology that searches for a solution of a problem by first retrieving a similar problem stored in a memory of cases and then adapting the solution found to the new problem [1]. When a forest fire is detected CHARADE retrieves similar emergencies from the memory and builds a plan through an interactive process where the user makes strategical choices and the system checks and propagates domain constraints.

An extensive evaluation of the system showed that, in this application domain, users are not keen to accept the system solutions even if those proposals can be even completely changed by user interaction. We were forced to reconsider the usefulness of the CHARADE case base. Rather that using it inside a program for building *solutions* we realized that the user needs a tool for *learning* how to plan first interventions and acquire knowledge from the data stored in the case base. For that reason we moved from a strict CBR paradigm to a

* This work has been partially supported by the EspritIV project CARICA #20401 (Cases Acquisition and Replay in Fire Campaign Ambiance).

KDD system [5]. Knowledge Discovery in Database (KDD) refers to the overall process of discovering useful knowledge from data, mapping low-level data into other forms, more compact, more abstract and more useful. A case base, a product of the CBR methodology, is a rich source of information that can be reused outside a strict CBR paradigm, it can be "mined" with KDD techniques. CBR tools can be used for that goal, but some changes are needed. For instance, the retrieval algorithm, the primary component of a CBR system, must discover similar case, but similarity must be redefined query by query according to the the target information searched by the user.

Case bases store knowledge combining numerical and symbolic data and often they adopt case representations based on complex structures (graph, semantic net) [3]. In our domain, for example, graphics is largely used to describe both the assessment of the emergency situation and the reaction plan. A set of icons displaced on a map illustrate the deployment of the resources and the activity they are engaged to. We argue that a rich description, plus a large collection of cases, become a real obstacle to the process of learning from the past experience contained in a case base.

In this paper we show how combining case-based reasoning and knowledge-discovering techniques can help significantly both teachers and students to acquire the knowledge contained in a case base. In our application, cases record the forest fire emergencies occurred in a department of the Southern France. Manipulation, browsing and enquiring tools enable users to directly extract knowledge in the form of: feature mutual dependencies; clusters and prototypes existing in the case base; feature statistical descriptors; results of CBR queries. In fact, the major focus of the system is retrieval for learning, that is addressed by using both CBR and KDD techniques. So for example retrieval based on a Nearest Neighbor algorithm, a standard techniques in CBR [2], can be supported by a selection of relevant features, performed with statistical and information theory algorithms [15]. CBET supports the user from the definition of the data structures, to the modification and maintenance of the case base.

The paper is organized as follow. Section 2 presents the main CBET functions by following an example in the forest fire domain. Section 3 reviews the techniques implemented in CBET for case retrieval, feature weighting, clustering and plotting. We end the paper with a discussion of the main contributions of CBET and future developments.

2 An Example: Fire Fighting

In this Section we illustrate the CBET functions by going through a typical interaction with the system.

2.1 Case Base Loading

Cases bases are stored in two different formats. The first, (raw) is a flat text file used mainly to import data from other systems. The second (qbe) is a richer

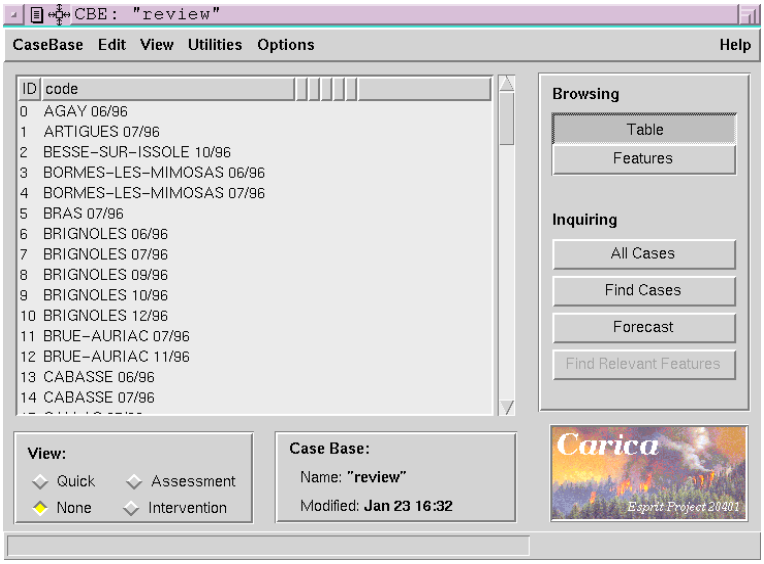

Fig. 1. The table browsing modality.

format that stores both data and info on data, e.g. feature relevance, feature distribution, etc. We shall refer to a case as a vector of feature values (e.g. `case1 = <id = 10232, hour = 15, risk-index = 5, .... >`), where a feature denotes a property of cases.

Many case bases can be loaded simultaneously, enabling parallel exploration of different knowledge sources. CBET supports editing actions like deleting a feature or creating a new feature whose values are obtained by the application of an arbitrary user defined function.

2.2 Case Base Browsing

A case base can be browsed in two modalities: table and features. The *table* modality shows a table where the columns display features (values) and the rows are cases (in Figure 1, "ID" and "code" features are shown). The user can select the features to show and the order in which they must appear on the table. That enables a simple customization of the interface.

In the second modality all the feature descriptors are listed. This provides another perspective to the information contained in the case base, which would help, for example, in selecting the most appropriate features to show in the *table* modality. A closer look to a feature is provided by another panel that shows summary information on the feature: maximum value, minimum value, type, the presence of missing values, mean, median, etc.

The system can also show summary information on a selected case (see Figure 2). In the Forest Fire application this panel shows a simple map of the zone

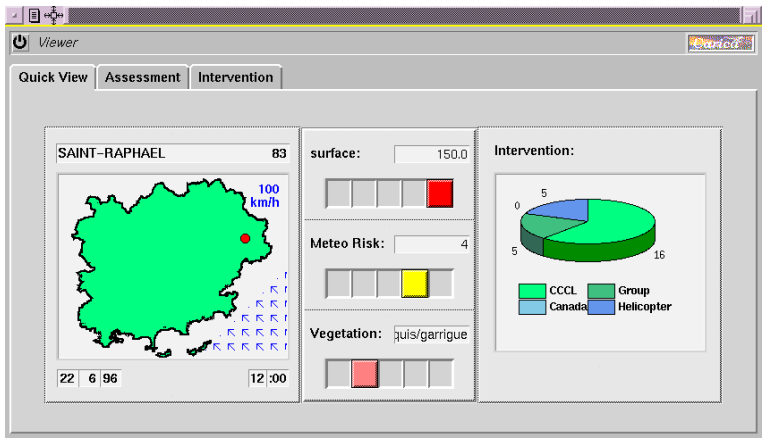

Fig. 2. The quick view panel.

surrounding the fire with the location of the fire and the description of the wind direction and speed. It also shows the values of the "surface", "meteo risk" and "vegetation" features and a pie chart of the types of resources used in the intervention.

A fire emergency is normally *assessed* by people in a control center by running mathematical models that forecast the fire evolution and by checking the state of the resources (human, aerial and ground means). That information and the description of the intervention made are described in other application oriented panels.

2.3 Plotting

Many different types of plots can be showed with CBET. For example, simple feature histograms can be shown for assessing the distribution of both numerical and symbolic features. Two dimensional plots of numerical features can also be showed. When one of the two features to plot has symbolic values CBET shows the distributions (histograms) of the numeric feature values conditioned to the different values of the symbolic feature. It results a set of histograms (one for each different value of the symbolic feature) that show the influence of the symbolic feature on the distribution of the numeric feature. Finally if two numeric features are plotted a third symbolic feature can be shown by using different icons.

2.4 Finding Relevant Features

CBET can show dependencies between data contained in a case base, i.e. how the variation of a feature impact on another. The user must select the target feature (the feature he/she is interested in) and the algorithm to be used for evaluating the relevance of the other features in describing the behavior of the

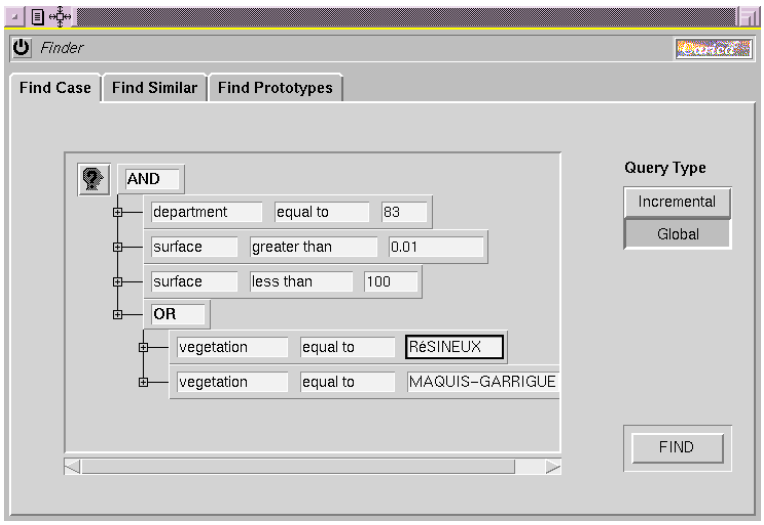

Fig. 3. A query by example.

target. The result can be viewed in two ways: as an ordered list of features, or with a plot showing for each feature the precise value of the relevance. The evaluation of the relevance of features can be also stored in the case base, so next time these data can be only retrieved from the information stored in the case base. [2]

In a similar way the relevance of pairs of features can be estimated. This provides an heuristic for deciding what pairs of features produce the most clear and effective plots.

2.5 Searching the Case Base

CBET searches a case base in two ways: the classical query by example and the case base search. Queries by example like that depicted in Figure 3 can be easily composed.

A second type of search is the "fuzzy" search performed by a nearest neighbor algorithm with feature relevance computed as described before. Every case can be selected as probe to search for similar cases. Some feature values of the selected case and feature relevance can also be modified, therefore building a completely new case (probe).

[2] This a very general technique: one can always store the result of the the most computational expensive algorithms and access them afterwards without recomputing it.

2.6 Finding Prototypes and Forecasting Values

Searching for prototypes is another important function supported by CBET. A prototype is a case that have a cluster of similar cases in the case base. These similar cases can be described by referring to this prototype. The prototypes created can be loaded and the partition of cases in the clusters represented by prototypes can also be shown to the user.

The last collection of functions in CBET are those related with forecasting of unknown feature values. In order to solve this problem a classifier is built and then the classifier is used for evaluating the most likely value of the unknown feature. CBET supports the many types of classifiers, including: k-NN and IB family, ID3, C4.5, NGE.

3 Techniques

In this Section we describe the major groups of techniques used in CBET. As we have shown in Section 2 they can be grouped in five classes: retrieval; feature selection; clustering; graphics; value forecasting. In this paper we concentrate on the first three.

3.1 Retrieval

Case-Base retrieval is a central activity for knowledge acquisition and learning from a case-base. Case-base retrieval consists of looking for a set of cases that are in the case-base and are similar to a given partially described case (probe). Case-base retrieval relates to Information Retrieval and DBMS querying. In fact, new approaches are trying to integrate CBR and Information retrieval [11]. CBET provides both ways to case retrieval: query by example (QBE) and nearest neighbor (NN).

QBE is one traditional approach to query a data base. CBET allows this type of query: the user provides ranges for a set of features and the system returns those cases that satisfy the given constraints. The result of a QBE is a new case base and can be further queried by QBE or NN, or can be used for other studies supported by CBET.

Nearest Neighbor is a common algorithm in Pattern Recognition [4] and Case-Based Reasoning[1]. Given a set of cases $X = \{x_1, \ldots, x_n\}$ and a probe case y, NN searches in X for the k most similar cases to y. Similarity is defined by means of a distance function, $(\sum_{i=1}^{m} w_i d_i(x_{ij}, y_i)^2)^{1/2}$, where $d_i(x_{ij}, y_i)$ is the distance between the $i-$th feature values, x_{ij} is the value of the i-th feature of the j-th case, m is the number of features, w_i is a weight used for balancing feature relevance.

- **The feature distance.** Feature values are compared by using the feature distances d_i. If the $i-$th feature is numeric then $d_i(x_{ij}, y_i) = |x_{ij} - y_i|/r_i$, where r_i is the range of the $i-$th feature values. Otherwise $d_i(x_{ij}, y_i) = 0$ if $x_{ij} = y_i$ and $d_i(x_{ij}, y_i) = 1$ if $x_{ij} \neq y_i$. If x_{ij} or y_i are missing then

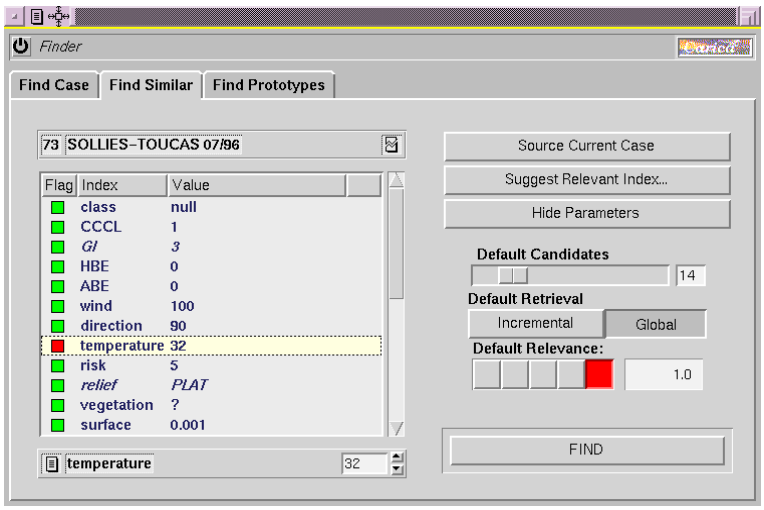

Fig. 4. Changing the relevance the "temperature" feature in searching for a similar case.

$d_i(x_i, y_i) = 0.5$. Missing values originates from missing information in the case base or from the fact that the probe y is normally only partially defined, i.e., some features' values are not specified.

- **The feature relevance.** The contribution to the total distance given by the i-th feature value distance is weighted by the positive constant w_i. There is a vast literature on feature weighting [15] and a number of different distance functions can be defined by varying the weights. Case bases are often used for classification tasks, i.e. a symbolic feature f_T (class) in the probe is unknown and the f_T feature value of the nearest neighbor provides a prediction of f_T in the probe. In nearest neighbor classification weights are chosen in such a way that the classification accuracy is maximized (learning weights).

CBET retrieval has not been primarily designed for classification, even if that can be supported. Retrieval, generally speaking, must satisfy user's information needs. For that reason we have developed a number of different approaches to feature weighting that are targeted to the satisfaction of the user information needs when a query is executed by the system. We describe these techniques in the next section.

3.2 Feature Selection and Weighting

Cases are described by a set of numeric and symbolic features. A central problem in Machine Learning relates to the identification of those features that provide predictive or descriptive value for a given target feature [7]. In CBET feature selection and more in general feature weighting [15] has two types of applications. First, in retrieval, when features are to be balanced in computing the similarity

function. Second, when data are to be visualized we must identify the most relevant dimensions along which to plot data.

Feature selection and weighting algorithms can be divided into two general classes: filter models and wrapper models [7]. Filter models selects relevant features before applying the chosen induction algorithm. Conversely wrapper models search for a good subset using the induction algorithm itself as part of the evaluation function. Even if wrapper models seem more accurate as the bias of the feature selection algorithm matches the induction algorithm they are not easily applicable to CBR. In CBET, feature weighting is mainly used inside the retrieval process where there is no reasonable definition of classification accuracy with respect to adapt the weights. Therefore, filter models seem more appropriate to CBET, moreover they are computationally cheaper and are based on some general theory of data correlation and relevance.

We have implemented in CBET weighting methods based on Information Theory. Assume that the user is *focussed* on a feature f_T, and he/she is interested in discovering what features may account for the behavior of f_T. Let us assume first that f_T is discrete and finite (symbolic features), in this situation we have borrowed some ideas from the theory of top down induction of decision trees (TDIDT) for classification [14]. A decision tree recursively selects a feature and partitions the data set into many subsets as the number of possible values of the chosen (discrete) feature. A central problem in TDIDT is therefore selecting features in such a way that the associated partition maximize a given "purity" objective function.

We hypothesize that the features that maximize the "purity" objective functions used for TDIDT can be also valuable for evaluating the relevance of feature when the user is generally interested in judging what are the most important features for understanding the behavior of f_T. We present in the following the functions used in CBET for evaluating feature relevance.

The *info* of the distribution of the f_T values is defined as:

$$Info(f_T) = -\sum_{j=1}^{k} p(j)\log_2(p(j)) \tag{1}$$

where $p(j)$ is the probability of value j among the possible values of f_T [9]. The information brought by a feature f_d with values $\{x_{d1}, \ldots, x_{dm}\}$ is defined:

$$Entropy(f_d) = -\sum_{j=1}^{k}\sum_{i=1}^{m} p(x_{di})p(j|x_{di})\log_2(p(j|x_{di})) \tag{2}$$

where $p(x_{di})$ is the probability of observing x_{di} given that f_d is not unknown, i.e. given that $x_{di} \neq ?$. In TDIDT test on continuous features are usually performed by selecting a threshold and dividing the cases into those whose value of feature f_d is greater of the threshold and those whose value is less than the threshold

[9]. We have followed a different approach, we discretize the feature f_d into a finite number of values and then we apply the above definitions. [3]

A first index of the relevance of feature f_d in describing feature f_T is given by the Information Gain [9]:

$$Gain(f_T, f_d) = (1 - p(f_d =?))(Info(f_T) - Entropy(f_d)) \tag{3}$$

where $p(f_d =?)$ is the probability that feature f_d has unknown value. So features f_d that have high information gains provide more knowledge to the understanding of the behavior of f_T. It is well known that the Gain criterion tend to favor features with many outcomes. The Information Gain Ratio has been introduced for overcoming that problem.

$$GainRatio(f_T, f_d) = \frac{Gain(f_T, f_d)}{SplitInfo(f_d)} \tag{4}$$

where SplitInfo is

$$SplitInfo(f_d) = -\sum_{i=1}^{m+1} p'(x_{di}) \log_2(p'(x_{di}))$$

and $p'(x_{di})$ is the a priori probability of x_{di} (unknown is treated as another possible value of f_d). SplitInfo is the information generated by dividing the case base into as many subsets as the number of possible values of f_d.

CBET uses also another index for evaluating features, namely the Calinski-Harabasz. This index has been used in clustering for evaluating how well the classes (different values of f_T), regarded as clusters, are separated in a dataset[12]. In our application, clusters are defined by grouping cases having equal values of the feature f_d (continuous features are dealt with discretization). The Index is defined as:

$$CH(f_T, f_d) = (TraceB/(g-1))/(TraceW/(n-g)) \tag{5}$$

Where $Trace$ is the trace of the matrix (the sum of the diagonal elements); W and B are the within-class covariance matrix and the between-classes covariance matrix respectively [10]; n is the number of cases and g is the number of different values of f_d

In case of f_T is numeric more classical statistical indices of correlation are used. We describe here only the use made in CBET of the Mean Squared Error. The MSE is given by the following:

$$MSE(f_T, f_d) = \sum_{i=1}^{g} p_i \sigma^2(X_i) \tag{6}$$

where $\bigcup_{i=1}^{g} X_i = X$ is a partition of the case base X, $p_i = |X_i|/|X|$ and σ^2 is the variance of the feature f_T in X_i. The sets X_i are produced by splitting

[3] Many alternative discretization algorithms can be selected through the CBET user interface.

a continuous feature f_d in g equally spaced bins, or by grouping together the values of f_T that are paired with equal values of f_d (i.e. $X_i = \{x_T \ : \ x_d = i\}$). The littler the MSE the better correlation between features f_d and f_T.

So, resuming the process, when the user want to retrieve cases, the focus of the user is acquired, i.e. the target feature f_T. For example if the planning techniques for forest fire fighting are the focus, the number of firemen involved or the total area burned can be the target feature. The other features are then weighted with one of the methods described above. The weights are computed and fed into the distance definition used for Nearest Neighbor retrieval to finally run the search.

3.3 Clustering

Usually case base systems do not provide any indication about the distribution of cases in the features' space. Exploring a new case base this information can help focusing on a smaller subset of "significantly" different cases. The same problem occurs when we examine the result of some query to a case base: if the answer is represented by a wide collection of cases we need to identify quickly the variability of cases. Moreover, it could be useful to summarize the answer replacing similar cases by a barycentric prototype.

Clustering means grouping together cases that possess strong internal similarity. In order to cluster data we must be able to decide if a given case is more similar to the examples in one cluster rather than those in another. That yields two issues: first, defining a similarity measure between cases; second, a partitioning criteria to divide a set of examples into two clusters.

In CBET we have borrowed two methods: COBWEB [6] and RPCL (Rival Penalized Competitive Learning) [17], they are based on unsupervised and supervised learning techniques respectively [14].

COBWEB organizes the examples into a hierarchy using four kind of operators: *incorporate* when an examples fits well into an existing cluster; *create* when an examples has very different characteristics from any existing example in the current cluster; *merge* when the hierarchy is overly branched and combining two clusters provides a better one; *split* when the hierarchy contains a cluster that is too general and therefore less accurate than breaking it into two clusters.
COBWEB scores alternative operators to apply to a case with an evaluation function called *category utility*. This function is defined as follows:

$$CU = 1/n(\sum_{k=1}^{n} P(C_k) \sum_i \sum_j P(f_j = x_{ij}|C_k)^2 - \sum_i \sum_j P(f_i = x_{ij})^2)$$

where $P(C_k)$ is the probability of occurrence of a particular cluster C_k, $P(f_j = x_{ij}|C_k)$ is probability of observing a particular feature value x_{ij} in the cluster C_k, and $P(f_i = x_{ij})$ is the a priori probability of observing the feature value x_{ij}.

The first sum represents the probability to correctly guess the feature values given the cluster membership, the second one the probability to correctly guess the feature values without any cluster knowledge. Category utility gives

a high score to partitions which maximize similarity among cluster examples (intra-cluster similarities) and dissimilarity between examples of different clusters (inter-cluster dissimilarities). This function trades off the *predictiveness* of each feature value (the probability of an example to be a member of a cluster given its feature value) and the *predictability* of the value (the probability of the value, given that an example is a member of a cluster).

RPCL algorithm is an adaptive version of the classical *k-means* clustering algorithm. RPCL addresses and solves a limitation of *k-means*, i.e. the need to hypothesize, before starting clustering, the right number of clusters. RPCL starts by randomly choosing a set of k, p_i for $i = 1, \ldots, k$, candidate prototypes. Then an iterative procedure executes the following three steps:
Step 1: randomly take a case x in the case base and select the closest prototype p_c to x (Euclidean metric).
Step2: if $f_T(x) = f_T(p_c)$ then update the selected candidate prototype $p_c := p_c + \alpha_c(x - p_c)$
Step3: if $f_T(x) = f_T(p_c)$ and also the second closest prototype p_s has $f_T(x) = f_T(p_s)$ then update also $p_s := p_s - \alpha_r(x - p_s)$.

α_c and α_r are two positive (< 1) constants called learning rates for the winner and rival respectively. While p_c moves towards x the second closest is moved far from x. The rival penalized mechanism tries to prevent that two prototype move both towards the center of the same cluster. At the same time when the number of the initially selected prototypes is larger than the number of clusters in the given case base, the RPCL algorithm automatically selects the correct number of prototypes.

The prototype candidates are abstract representations of cases. They don't really belong to the case base but could be considered as a generalization of all the examples in the cluster represented by a prototype. The final set prototypes used by CBET are then obtained replacing each prototype with its nearest neighbor case.

4 Conclusions

In this paper we have presented CBET a tool aimed at the exploration of rich and complex sources of knowledge (case bases). CBET is based on the application of a range of Machine Learning techniques including: feature weighting, clustering, classification. We have illustrated the application of CBET to a real problem: acquiring competence in fire fighting techniques.

CBET has been integrated with another tool for knowledge acquisition (not described here) that makes simple the acquisition of cases in the fire fighting domain. In this way CBET completes the learning cycle, from case acquisition to knowledge discovery. Our primary future objective is a thoroughly evaluation with the user. The goal is to score alternative algorithms for feature weighting and clustering according to the user evaluation and to compare that results with more abstract (Machine Learning) indicators.

We also aim to introduce a better metric for dealing with symbolic features. That will mimic the VDM metric [13] and could be integrated with other metrics on numerical features following the approaches proposed in [16].

References

1. Agnar Aamodt and Enric Plaza. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.
2. David W. Aha, Dennis Kibler, and Mark K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
3. L. K. Branting. Techniques for the retrieval of structured cases. In *Working Notes of the AAAI Spring Symposium on Case-Based Reasoning*, Palo Alto, CA, 1990.
4. B. V. Dasarathy, editor. *Nearest neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer Society Press, Los Alamitos, CA, 1991.
5. Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, pages 37–54, fall 1996.
6. Douglas H.Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.
7. G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Machine Learning Conference*, pages 121–129, New Brunswick, NJ, 1994. Morgan Kaufmann.
8. Anna Perini and Francesco Ricci. An interactive planning architecture. In M. Ghallab and A. Milani, editors, *New directions in AI Planning*, pages 273–283. IOS Press, 1996.
9. J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.
10. B. D. Ripley. *Pattern recognition and neural networks*. Cambridge U.P., 1996.
11. E.L. Rissland and J.J. Daniels. Using cbr to drive ir. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 400–407, Montreal, Canada, 1995.
12. David B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the Eleventh International Machine Learning Conference*, pages 293–301, New Brunswick, NJ, 1994. Morgan Kaufmann.
13. Craig Stanfill and David Waltz. Toward memory-based reasoning. *Communication of ACM*, 29:1213–1229, 1986.
14. Sholomon M. Weiss and Casimir A. Kulikowski. *Computer Systems that Learn*. Morgan Kaufmann, 1991.
15. Dietrich Wettschereck, Takao Mohri, and David W. Aha. A review and comparative evaluation of feature weighting methods for lazy learning algorithms. *AI Review Journal*, 11:273–314, 1997.
16. D. Randall Wilson and Tony R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 11:1–34, 1997.
17. Lei Xu, Adam Krzyzak, and Erkki Oja. Rival penalized competitive learning for cluster analysis, RBF net, and curve detection. *IEEE Transaction on Neural Networks*, 4(4):636–649, 1993.

This article was processed using the LaTeX macro package with LLNCS style