The Explanatory Power of Symbolic Similarity in Case-Based Reasoning

Enric Plaza, Eva Armengol and Santiago Ontañón
IIIA - Artificial Intelligence Research Institute, CSIC - Spanish Council for Scientific Research, Campus UAB, 08193 Bellaterra, Catalonia (Spain).
({enric, eva, santi}@iiia.csic.es*)*

Abstract.
A desired capability of automatic problem solvers is that they can explain the results. Such explanations should justify that the solution proposed by the problem solver arises from the known domain knowledge. In this paper we discuss how explanations can be used in case-based reasoning (CBR) in order to justify the results in classification tasks and also for solving new problems. We particularly focus on explanations derived from building a symbolic description of the similar aspects among cases. Moreover, we show how symbolic descriptions of similarity can be exploited in the different processes of CBR, namely retrieve, reuse, revise, and retain.

Keywords: Case-Based Reasoning, Symbolic Similarity, Explanation, Lazy Learning

1. Introduction

A desired capability of automatic problem solvers is that they can explain the results they produce. Such explanations should justify that a solution proposed by the problem solver arises from the known domain knowledge. There are several ways to explain the results depending on the kind of problem solver and the representation it uses. For instance, problem solvers using rules can explain the result by showing the rules used to reach the solution whereas problems solvers based on cases can explain the result showing the set of cases supporting the solution. Showing the reasoning chain or the set of cases supports the user in the detection of failures such as the use of an incorrect rule, the lack of one or more rules, or the use of an inappropriate similarity measure assessment among the cases.

Leake (1994) distinguishes three key issues on explanations: what to explain, when to explain, and how to generate explanations. In this paper we will consider classification problems where the explanation has to justify the membership of a new problem in a solution class, and the generation of the explanation has to be made at the end of each new problem solving process. In particular, we want to analyze both how the explanations are generated and how they can be reused for

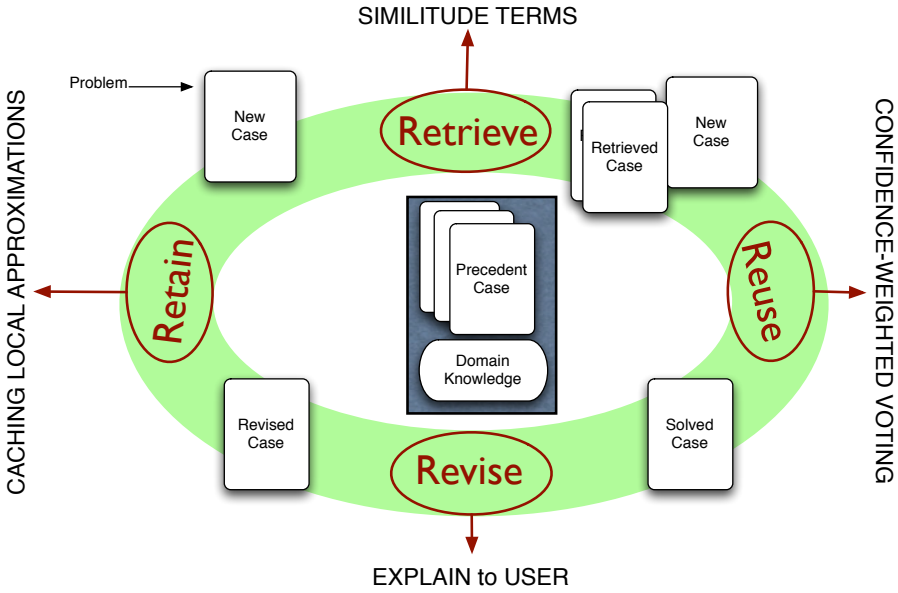

Figure 1. The Case-based Reasoning cycle showing how explanations can be used in the stages of retrieve, reuse, revise, and retain.

solving new problems. This paper discusses how the explanations can be used in case-based reasoning (CBR) in order to justify the solution and, moreover, how to use them to improve case-based problem solving. We particularly focus on explanations derived from building a symbolic description of the similar aspects among cases. Moreover, we show how symbolic descriptions of similarity can be exploited in the different processes of CBR, namely retrieve, reuse, revise, and retain.

2. Related Work

Case-based reasoning methods (Kolodner, 1993) are based on retrieving past experiences to solve a new problem. Figure 1 shows the four processes of the CBR problem solving cycle (Aamodt and Plaza, 1994). Given a new problem and a case base, the *Retrieve* process retrieves a subset of cases similar in some aspects to the new problem. From the retrieved cases the *Reuse* process determines how to transfer or adapt their solutions to the new problem. This suggested solution is analyzed in the *Revise* process to provide a final solution. Finally, the *Retain* process incorporates the new case (problem description and final solution) for further use.

Two key points of the CBR methods are how to assess the similarity between cases in order to retrieve appropriate precedents and how to adapt old solutions to the new case. Commonly, cases are indexed in the case base in such a way that indexes reflect the relevance of some case features. Nevertheless, as Leake (1995) points out, the relevance of a case feature depends on the context. This means that, given a new case, the specific subset of retrieved cases may vary, since the features that are relevant are different depending on those contexts. Moreover, the reuse of the solutions will also depend on the retrieved cases. In both situations, an explanation of the decisions taken in those processes can increase the user confidence in the final solution.

There are several ways in which explanation can be integrated into a CBR system. Sørmo et al. (2005) define four types of explanations: justification, transparency, relevance and learning. These types respectively explain the solution, the process to the solution, the relevancy of the questions (when there is user interaction) and what is important with respect to the domain (i.e. what to learn). Leake (1995) suggests that the type of explanation depends on the user. For instance, novice users may prefer explanations justifying why the solution has been proposed, whereas expert users may prefer explanations about how the solution has been found. In addition, the type of explanation can also change as the users increase their confidence in the system. Swartout and Moore (1993) try to capture the different stereotypes of users by defining explanation patterns for each type of user. However, if the system classifies the current user as belonging to an erroneous stereotype then the explanations are not satisfactory.

Commonly, the explanation of CBR systems is performed by showing, for the current problem, the most similar case (or a small set of most similar cases). The justification is that this is a common process done by the experts: they solve new cases based on the adaptation of past ones. Therefore, it is easy to understand why the system proposes a solution showing the case from which the solution has been reused. Nevertheless, this kind of explanation may be difficult to understand directly; for instance, when the most similar case solution has needed several adaptations to be reused; or when cases have complex structures (Doyle et al., 2004; McSherry, 2005) that make difficult to grasp at first sight those aspects that are considered relevant.

Some other authors propose that, for some domains (e.g. medicine), an expert understands better a description of differences among retrieved cases than just presenting the retrieved cases. In particular, Doyle et al. (2004) propose that the more explicative cases are those close to the frontiers of the classes. Nugent and Cunningham (2005) propose that the explanation has to include the contribution of each

feature value to the classification of a case. McSherry (2005) proposes that, in addition to the similarities between two cases, the features capable of discriminating between competing cases may also be useful.

In the rest of the paper we will focus on using explanations derived from symbolic descriptions of the similar aspects among cases. For brevity, we will define *similitude* to be the "similar aspects among cases" and *symbolic similitude* to be the "symbolic descriptions of the similar aspects among cases". Notice that we take the word *similitude* to mean *that which is like or similar*. In particular, we will show how symbolic similitudes can be used in the CBR processes of retrieve, reuse, revise, and retain shown in Figure 1.

3. Explanation and Structural Similarity

Explanation in AI systems is commonly understood as a form of symbolic description provided to the user after solving a problem; as such, an explanation is then used in the *Revise* process of the CBR cycle shown in Figure 1. In this section we will show how the process of building such an explanation and the process of retrieving the most similar (relevant) cases to solve a problem can be both unified into a single process. That is to say, the retrieval technique we will present selects the most relevant aspects shared by the problem and the cases in the case base and builds at the same time the explanation based on those shared aspects.

CBR approaches are based on finding the most similar (or relevant) cases for a particular problem. Commonly, the similarity among cases is estimated using metrics and assuming that cases are represented as attribute-value pairs. Nevertheless, case representation may be more complex (and more expressive) structures; in such situations the kind of similarity to be applied has to take into account the structural similarity between two cases (Börner, 1993; Bunke and Messmer, 1993; Armengol and Plaza, 2001b).

Another approach to assessing structural similarity is to consider the shared structure between two cases (Plaza, 1995). For this discussion, we will consider the problem space $\mathcal{P}$ as the collection of all possible case descriptions (i.e. without the case solutions) in a given language and their generalizations. Moreover, consider Figure 2 showing the "similarity space" $\mathcal{P}(A_i, A_j)$ between two cases: case A_i with solution S_{A_i} and case A_j with solution S_{A_j}. The *similarity space* $\mathcal{P}(A_i, A_j)$ is a subset of the problem space $\mathcal{P}$ formed by the collection of terms generalizing the problem descriptions of A_i and A_j. That is to say, a term $\sigma \in \mathcal{P}(A_i, A_j)$ satisfies both $\sigma \sqsubseteq A_i$ (σ subsumes or is more

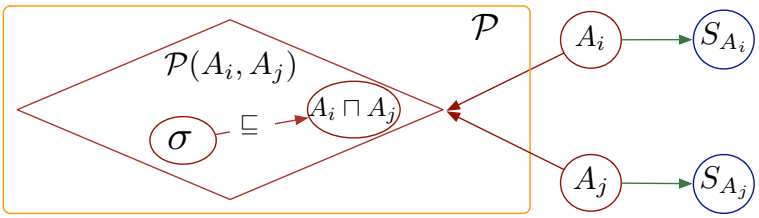

Figure 2. Schema of the problem space $\mathcal{P}$ and the similarity space $\mathcal{P}(A, B)$ of two cases A and B. $\mathcal{P}$ contains all possible similitude terms (σ) of cases A and B. Anti-unification $A \sqcap B$ is the similitude term that contains *all* that is common to A and B.

general than A_i) and $\sigma \sqsubseteq A_j$. In particular, the anti-unification $A_i \sqcap A_j$ (or most specific generalization) is the term that contains *all* that is shared between A_i and A_j; other terms $\sigma \in \mathcal{P}(A_i, A_j)$ contain only *some* commonalities between A_i and A_j. Plaza (1995) proposed to retrieve cases (from a case base $\mathbf{B}$ for a problem P) by computing the *similitude terms* $\Sigma(P, \mathbf{B}) = \{\sigma_i | \sigma_i = P \sqcap A_i \wedge A_i \in \mathbf{B}\}$; that is to say, computing all the anti-unifications σ_i of the problem P with the cases in the case base. Then the similitude terms σ_i are ranked using an information entropy measure and the cases with top-ranking σ_i are retrieved as those more relevant for solving P.

However, taking into account *all* the similar aspects (i.e. using anti-unification) is not necessary. In fact other similitude terms σ_i may be better for selecting the most relevant cases. In general, we may conceive implementing the retrieval phase of CBR as a search process in the problem space $\mathcal{P}$; this process aims at finding the *best* similitude term σ^P for a problem P. The similitude term is *best* in the sense that σ^P *retrieves* the cases *most* similar to P, i.e. it is a generalization of P and the cases $A_i \ldots A_k$ that would be the best precedents for solving P. Therefore, the issue to be solved now is how to estimate the *best* similitude term. Specifically, the issue is to define a search process over the problem space such that the similitude term and the corresponding *best* cases are found. Notice that for analysis tasks (e.g. classification) we need to estimate the likelihood that the retrieved cases are in the same solution class of the problem P, while in synthesis tasks we need to estimate the likelihood that the *solution(s)* of the retrieved case(s) can be *adapted easily* to the problem P. In the following we will present a technique for classification tasks that follows a top-down heuristic search strategy to build similitude terms.

3.1. Lazy Induction of Descriptions

Lazy Induction of Descriptions (LID) is a CBR technique that uses this problem space search approach. Given a problem P, LID carries out a heuristic top-down search in $\mathcal{P}$. Top-down search in $\mathcal{P}$ means that LID follows a general to specific search strategy in the space of the generalizations of P. LID starts with the most general generalization, i.e. the empty similitude term $\sigma = \bot$. At each step, LID uses a heuristic measure to add a new feature (a new predicate) to the current generalization (the similitude term σ). Thus, the search process of LID specializes the generalized description until a termination criterion is met; at that point the cases subsumed by σ are those retrieved as those more relevant to P.

At each step of the search process LID has a similitude term σ and the associated *discriminatory set* $\mathcal{D}(\sigma) = \{A_i \in \mathbf{B} | \sigma \sqsubseteq A_i\}$; i.e. the cases subsumed by the similitude term. Therefore LID has to determine whether this is a good enough set of cases to be retrieved or if it is better to specialize σ and reduce $\mathcal{D}$ to a subset of *more relevant* cases. Clearly, finding the "best" similitude term depends crucially on the heuristic used to select the feature (predicate) that specializes σ.

Since LID is a CBR technique for classification tasks, the heuristic used is an information theoretic one, the López de Mántaras (LM) distance (López de Mántaras, 1991). The LM distance assesses how similar two partitions are, in the sense that the lesser the distance the more similar they are. A feature f_i with value range $[v_1^i \dots v_{n^i}]$ induces a partition over the case base $\mathbf{B}$, where each partition set contains the cases A_j with the same value v_j^i for the feature f_i. LID computes the distance between this partition and the *correct partition* P_c, i.e. the partition over the case base $\mathbf{B}$ given by the solution classes. The most discriminatory feature f_d is that producing a partition P_d having the minimum distance to the correct partition P_c. A further explanation of LID and its evaluation on two application domains can be found in (Armengol and Plaza, 2001a).

The top-down heuristic search process ends when one of two conditions are met. The first condition is that all cases subsumed by a similitude term belong to the same class S_i; the search needs not refine σ further since LID has found S_i to be the solution class for the problem. The second possibility is that cases subsumed by a similitude term σ belong to two or more classes but there is no discriminatory feature left to further discriminate among the retrieved cases; in this situation LID also terminates and yields as solution the majority class among the retrieved cases. As we will see later in Section 4, the degree to which

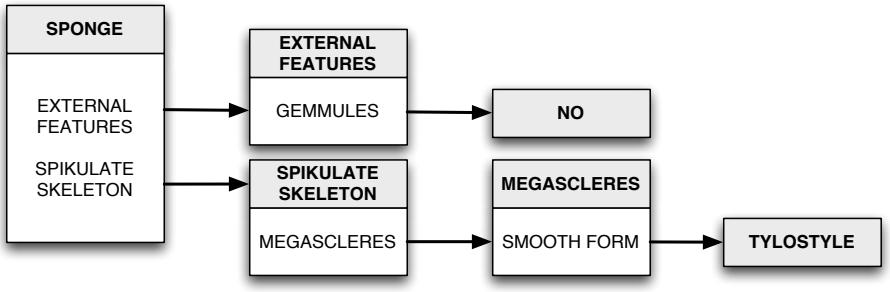

Figure 3. An explanation built by LID in the domain of marine sponge identification.

a solution is endorsed by a large majority can be used to estimate the confidence that the system has in its own solution.

When a problem P is solved by LID, the outcome provided is three-fold: a solution class S_i, a similitude term σ and a set of similar cases $\mathcal{D}$; the meaning of the outcome is as follows: σ is the explanation of why P belongs to class S_i and this solution is endorsed by the cases in $\mathcal{D}$ that share the class S_i as solution. In other words, class S_i is the solution for P because it shares the description σ with the cases in $\mathcal{D}$ that are also of class S_i. Thus, σ explicitly shows the important aspects shared by the problem P and the retrieved cases $\mathcal{D}$.

Figure 3 shows an example of an explanation built by LID in the domain of marine sponge identification while solving a problem P. More precisely, this figure shows the similitude term built by the retrieval process of CBR. The complete explanation includes the set of cases $\mathcal{D}$ that endorse the solution predicted by the system, and that share with the problem those aspects depicted by the similitude term of Figure 3.

Since the similitude term is composed of a set of relevant features shared by a subset of cases belonging to the same solution class it represents a symbolic description of the structural similarity among the new case and a subset of cases. Also, the similitude term can be seen as a partial description of that class. Notice that this description is only partial because, in contrast with inductive learning methods, a similitude term does not characterize all the examples of a solution class but only a subset of them. LID only assures that the relevant features are shared by a subset of cases all belonging to the same solution class. The similitude term depends on the new problem, for this reason there are several partial descriptions for the same solution class. Notice also that this partial description can be seen as an explanation of why the new case is classified as belonging to a certain class. Thus, from the explanation viewpoint, LID is able to justify the classification of new

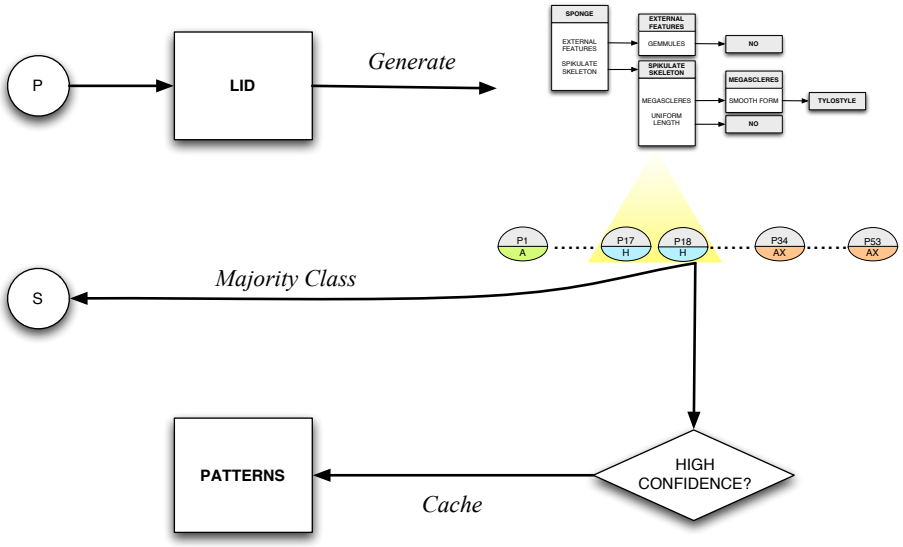

Figure 4. Schema of caching policies for C-LID.

problems by giving a partial description of a solution class. This partial description can be used to solve further problems, as is also done in explanation-based learning.

4. Caching LID

This section investigates the idea of *storing* and *reusing* the similarity terms that LID has generated in solving past cases to improve the solving of future problems.

C-LID is a lazy learning technique for CBR that caches the explanations built in solving past problems — with the purpose of reusing them to solve future problems. In fact, the main issue of lazy learning is that it builds a *local approximation* of the target concept (*local* since it is around the problem P), while *eager learning* (e.g. rule induction) aims at building *global approximations* of the target concept, that should therefore be useful to solve any future problem (e.g. a collection of rules to describe all instances of a concept). Specifically, LID builds a symbolic similarity description (an explanation) that is the local approximation used to solve a problem P.

The underlying notion of C-LID is that of reusing past local approximations (explanations) to improve the classification of new problems in CBR. C-LID is defined on top of LID by defining two policies: the *caching policy* and the *reuse policy*. The *caching policy* determines

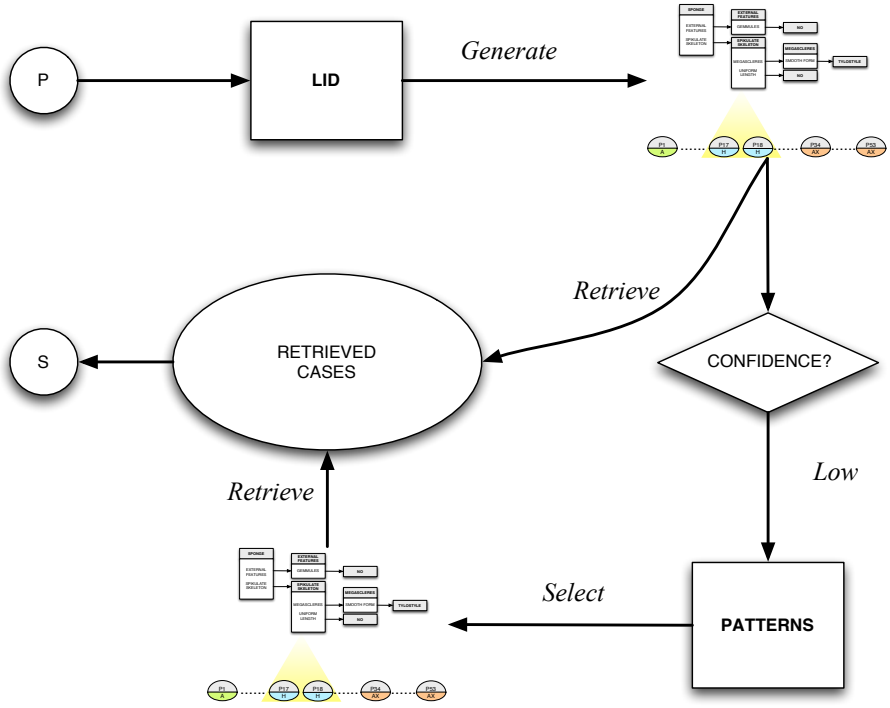

Figure 5. Schema of reuse policies for C-LID.

which similitude terms (explanations) are to be retained. The *reuse policy* determines when and how the cached explanations are used to solve new problems.

Thus, when a problem P is solved with solution class S and explanation σ, the caching policy decides whether σ is retained or discarded. Figure 4 show the schema of a *caching policy*. First, a similitude term σ is generated after solving a problem P using LID; then some estimation of the confidence in the usefulness of σ is performed and, if this confidence is positive enough, then σ is cached into the set of *patterns* (local approximations) for class S. Different confidence estimations implement different caching policies. Typically, a cache policy of C-LID is to keep only those similitude terms that perfectly classify the subsumed cases (i.e. those similitude terms such that all the cases in their discriminatory sets $\mathcal{D}$ belong to just one class). The rationale of this policy is that it is worth caching those similitude terms that are good approximations. Also other, less restrictive policies, are possible —such as caching all similitude terms that have a clear majority class among the cases in $\mathcal{D}$. This policy retains more patterns (and thus

increases their scope) but their uncertainty is also increased when they are reused.

The *reuse policy* decides when and how the patterns (the cached explanations) are reused to solve new problems. Figure 5 shows the schema of a reuse policy. Notice that when a problem P is in the process of being solved there is a set of cached patterns that subsume P, i.e. if Σ is the collection of cached patterns, a problem has a set of applicable patterns $\Sigma(P) = \{\sigma \in \Sigma | \sigma \sqsubseteq P\}$.

C-LID starts by solving a problem P using LID; this generates a similitude term σ just as we saw before. Then, some estimation about the confidence in the usefulness of $\mathcal{D}(\sigma)$ (the retrieved cases) is performed. The set $\mathcal{D}(\sigma)$ is then the initial set of *retrieved cases* in Figure 5. If this confidence is *not* positive enough then the cached patterns that subsume P (those $\sigma_k \in \Sigma(P)$) are reused; specifically, those cases $\mathcal{D}_k$ subsumed by each of these patterns σ_k are also added to the set of *retrieved cases*. Finally, the problem P is solved using the cases in the *retrieved cases* set, the content of which varies depending on the way the confidence is estimated. As before, different confidence estimations implement different caching policies.

We have experimented with several reuse policies. The *univocal policy* examines the set $\mathcal{D}$ of cases retrieved by LID to check whether all of them belong to a single solution class; if this is the case σ is *univocal* and C-LID is confident in the solution predicted; otherwise the patterns $\Sigma(P)$ are reused. Another policy we have used is a threshold γ to assess the confidence. Let n be the number of cases retrieved in the set $\mathcal{D}$, and let m be the number of cases belonging to the majority class in $\mathcal{D}$; the policy states that if $\frac{m}{n} \geq \gamma$ (the ratio is larger than the threshold) then C-LID is confident in the solution predicted and otherwise the patterns are reused. The evaluation of different policies (Armengol and Plaza, 2003) shows that C-LID's accuracy improves over LID, although the degree of improvement depends on the dataset.

5. Justification-Based Multi-Agent Learning

An explanation can have other uses in addition to explaining the solution found by a CBR system. In this section we describe how explanations can be used during the Reuse process of the CBR cycle (see Figure 1) to improve the problem solving performance of CBR agents; specifically by improving classification accuracy in multi-agent CBR systems.

When a CBR system builds an explanation J of why it has found S_k to be the correct solution for a problem P, the explanation J is the

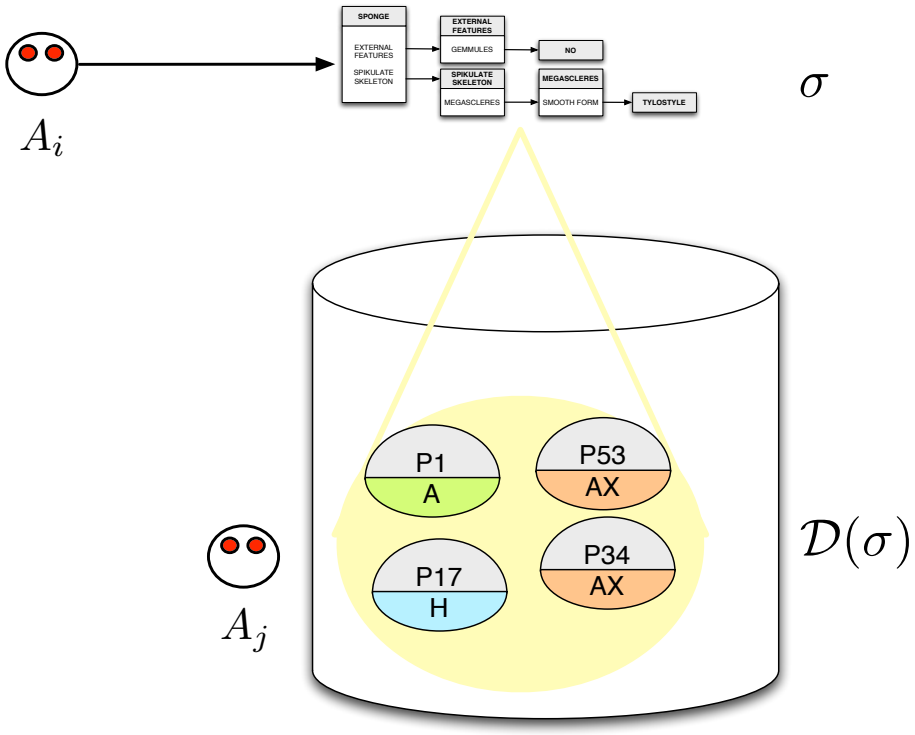

Figure 6. Examination of justifications in multi-agent CBR.

justification of the answer of the system, i.e. the system "believes" that S_k is the correct solution for P because of J. This interpretation of an explanation as a justification can be used in multi-agent systems to improve collaborative problem solving (Ontañón and Plaza, 2003).

Imagine a committee of CBR agents in which each CBR agent A_i owns a private case base C_i. This committee of agents works in the following way: when a new problem P has to be solved, each individual agent receives a copy of P and solves it; this solution of P is called an *individual prediction*. Each agent casts a vote for its predicted solution and the most voted solution will be considered the solution to the problem. As the individual case bases of the agents are private, the agents do not know the areas of expertise of the rest of the agents. Notice that in this approach the *retrieve* process is individual (since an agent has access only to the cases in its individual case base) and the *reuse* process is distributed (since class solution voting is the way to determine the solution of the new problem from the solutions of the retrieved cases).

However, when the committee is solving a problem, if there are some agent members that are not very knowledgeable in the area of the problem space needed to solve the current problem, the other agents in the system will not detect their shortcomings. This may cause some of the agents to cast unreliable votes that can affect the outcome of the voting system. Justifications can help to solve that problem. The problem in this scenario is that all the agents have the same weight in the voting process, regardless of whether they provide a reliable vote or not. However, if every agent can generate a *justification* to endorse their individual predictions the other agents could check the degree of *confidence* of an individual prediction made by every agent. The agents that have provided predictions with high confidence should have a larger weight in the voting process, and agents that have provided predictions with a low confidence should have a smaller weight.

The confidence in a solution S_k predicted by an agent A_i must be understood as a measure of the likelihood of S_k to be a correct solution. The process of assessing the confidence value of a prediction made by an agent by analyzing the justification provided is called the *justification examination* process. Let an agent A_j provide a justification J endorsing a prediction S_k. In order to examine this justification an agent A_i contrasts J against the cases in its case base C_i looking for *counterexamples* (cases that satisfy the justification, but that belong to a different class than S_k, the one predicted) and *endorsing cases* (cases that satisfy the justification and that belong to the predicted solution class S_k). The more endorsing cases and fewer counterexamples the higher the confidence value computed for a justification. Figure 6 shows an agent A_i examining a justification built by an agent A_j for the marine sponge classification domain. We will note by Y^{A_i} the number of endorsing cases and by N^{A_i} the number of counterexamples found by A_i. Now, A_j has predicted *Axinellida* (AX) as the solution for the problem; from the set of cases subsumed by the justification in the case base of A_i, two of them are endorsing cases (since they have AX as the solution class), and therefore $Y^{A_i} = 2$, and two of them are counterexamples (since they have a solution different from AX), and thus $N^{A_i} = 2$.

Using this justification examination process, an adaptive voting process among the committee members can be defined using justifications in order to determine which agents should have a higher weight in the voting system (Ontañón and Plaza, 2003):

1. Given a new problem P to solve, each agent A_i in the committee builds its individual prediction and generates a justification record $\langle J_i, S_i \rangle$ containing the individual prediction S_i and the justification

J_i that endorses it. Then, the justification record is sent to the rest of the agents in the system.

2. Each agent A_j *examines* the justification records $\langle J_i, S_i \rangle$ built by each other agent by contrasting them to its local case base C_j by looking for counterexamples and endorsing cases.

3. Then, a confidence value for each justification record $\langle J_i, S_i \rangle$ is computed as a function of the number of endorsing cases and counterexamples that each agent has found for each justification. Specifically, the confidence value of a justification record is computed as:

$$\mathsf{C}(\langle J_i, S_i \rangle) = \frac{\sum_{i=1\ldots n} Y^{A_i}}{\sum_{i=1\ldots n} Y^{A_i} + N^{A_i}}$$

where n is the number of agents in the committee, Y^{A_i} is the number of endorsing cases found by A_i for that justification record, and N^{A_i} is the number of counterexamples found by A_i for that justification record.

4. Finally, the confidence values can be used as weights in a weighted voting scheme were each agent votes for its individually predicted solution class, but its vote is weighted by the confidence computed by the rest of the agents on its prediction.

This weighted voting system is much more robust than a voting system where every individual agent has the same weight. If an agent A_i has not enough cases in the area of the problem space needed to solve the problem P and the solution found is not properly endorsed by a strong justification, the other agents will assign a low confidence value to the prediction made by A_i and its vote will not have much influence in the final outcome of the committee for problem P. Notice that this is an adaptive scheme since the weights of the agent prediction are computed for each specific problem. Notice also that the interpretation of an explanation as a justification is the key to allow the rest of the agents to assess the confidence of the individual predictions made by the agents.

The experimental evaluation (Ontañón and Plaza, 2003) shows that the committee's accuracy increases using justifications to estimate the confidence about individual predictions. This result proves that weakly justified individual predictions are effectively detected and their influence on the voting outcome is diminished accordingly. The accuracy increment is specially noticeable when some of the agents in the committee are less competent (higher individual error rate) than others; in

this situation, the accuracy increment using justifications is capable of achieving the accuracy of a competent committee without justifications.

We can conclude that explanations are not only useful to a human expert, but can also be a tool to improve problem solving. We have presented a multi-agent setting that can take advantage of the ability of CBR agents to generate explanations (used as justifications of individual predictions) to improve the *Reuse* process of the CBR cycle. Moreover, we believe that explanations have other potential uses in multi-agent systems, where the performance of an agent is usually dependent on information provided by other agents and thus it would be highly desirable that any information coming from others is properly justified.

6. Conclusions

In this paper we discussed several ways in which explanations can be useful for a CBR system. Thus, we briefly introduced LID, a CBR method capable of giving a justification of the result. This justification (a similitude term) contains the relevant aspects shared by the new problem and a subset of precedents belonging to one solution class. Moreover, the similitude term is built not only for user feedback purposes (in the *Revise* process of the CBR cycle) but is part and parcel of the *Retrieve* process in LID. In fact, we saw that LID performs a heuristic top-down search in the space of problem descriptions the result of which is the selection of a "best" symbolic similitude description and the retrieval of the subset of cases that satisfy that description.

Moreover, since lazy learning builds local approximations of the target concepts, LID can be conceived of as a system that builds a *symbolic local approximation* in the form of a similitude term (an explanation). We presented then the idea of caching and reusing those explanations in C-LID to improve classification accuracy using CBR. C-LID needs two policies: a caching policy and a reuse policy. The idea of both policies is to select similitude terms (patterns) that could be useful to solve new problems and then to decide when to use them. Thus, C-LID can be seen as an example of how explanations can be used in the *Retain* process of the CBR cycle.

We have also seen that explanations can also be used in the *Reuse* process of the CBR cycle for multi-agent CBR systems. We have presented a multi-agent setting that can take advantage of the ability of CBR agents to generate explanations (used as justifications of the solutions found for problems). Moreover, the performance of an agent usually depends on information provided by other agents and thus it

would be highly desirable that any information coming from others is properly justified.

More generally, explanations have been used in several processes of the CBR cycle as a tool to estimate the *confidence* in the outcome of a case-based process. Explanations used in the *Revise* process are used to obtain feedback from the user but also to improve the confidence that the user has in the solution proposed by a CBR system. Moreover, C-LID uses explanations to assess the confidence in the predicted solution in order to decide whether to use cached explanations to enlarge the set of retrieval cases with the purpose of improving confidence in the final prediction. Finally, multi-agent CBR systems using justifications can assess the confidence of an individual agent's prediction.

Acknowledgments This work has been partially supported by project SAMAP (TIC2002-04146-C05-01) and grant CIRIT FI/FAP 2001.

References

Aamodt, A. & Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Communications*, **7**(1):39–59.

Armengol, E. & Plaza, E. (2001a). Lazy Induction of Descriptions for Relational Case-Based Learning. In de Raedt, L. & Flach, P. (eds.), *Machine Learning: ECML-2001*, volume 2167 of *Lecture Notes in Artificial Intelligence*, 13–24. Springer Verlag.

Armengol, E. & Plaza, E. (2001b). Similarity Assessment for Relational CBR. In Aha, D. W. & Watson, I. (eds.), *Case-Based Reasoning Research and Development. ICCBR-2001*, volume 2080 of *Lecture Notes in Artificial Intelligence*, 44–58. Springer Verlag.

Armengol, E. & Plaza, E. (2003). Remembering Similitude Terms in Case-Based Reasoning. In Perner, P. & Rosenfeld, A. (eds.), *Machine Learning and Data Mining: MLDM-03*, volume 2734 of *Lecture Notes in Artificial Intelligence*, 121–130. Springer Verlag.

Börner, K. (1993). Structural Similarity as Guidance in Case-Based Design. In Wess, S., Althoff, K. D. & Richter, M. M. (eds.), *Proc. of the First European Workshop on Case-Based Reasoning*, volume 1, 14–19. University of Kaiserslautern.

Bunke, H. & Messmer, B. (1993). Similarity Measures for Structured Representations. In Wess, S., Althoff, K. D. & Richter, M. M.

(eds.), *Topics in Case-Based Reasoning*, volume 837 of *Lecture Notes in Computer Science*, 106–118.

Doyle, D., Cunningham, P., Bridge, P. & Rahman, Y. (2004). Explanation-Oriented Retrieval. In González-Calero, P. A. & Funk, P. (eds.), *Advances in Case-Based Reasoning*, volume 3155 of *LNAI*, 157–168. Springer Verlag.

Kolodner, J. (1993). *Case-Based Reasoning*. Morgan Kaufmann.

Leake, D. (1994). Issues in Goal-Driven Explanation. In *Proc. of the AAAI Spring Symposium on Goal-Driven Learning*, 72–79. AAAI Press.

Leake, D. (1995). Abduction, Experience and Goals: A Model of Everyday Abductive Explanation. *Journal of Experimental and Theoretical Artificial Intelligence*, **7**:407–428.

López de Mántaras, R. (1991). A Distance-Based Attribute Selection Measure for Decision Tree Induction. *Machine Learning*, **6**:81–92.

McSherry, D. (2005). Explanation in Recommender Systems. *Artificial Intelligence Review*. This Issue.

Nugent, C. & Cunningham, P. (2005). A Case-Based Explanation System for Black-Box Systems. *Artificial Intelligence Review*. This Issue.

Ontañón, S. & Plaza, E. (2003). Justification-Based Multiagent Learning. In Fawcett, T. & Mishra, N. (eds.), *Proc. 20th Int. Conf. on Machine Learning*, 576–583. Morgan Kaufmann.

Plaza, E. (1995). Cases as Terms: A Feature Term Approach to the Structured Representation of Cases. In Veloso, M. & Aamodt, A. (eds.), *Case-Based Reasoning Research and Development. ICCBR-95*, volume 1010 of *Lecture Notes in Artificial Intelligence*, 265–276. Springer Verlag.

Sørmo, F., Cassens, J. & Aamodt, A. (2005). Explanation in Case-Based Reasoning: Perspectives and Goals. *Artificial Intelligence Review*. This Issue.

Swartout, W. & Moore, J. (1993). Explanation in Second Generation Expert Systems. In David, J. M., Krivine, J. P. & Simmons, R. (eds.), *Second Generation Expert Systems*, 543–585. Springer Verlag.