Explaining similarity in CBR

Eva Armengol, Santiago Ontañón and Enric Plaza

Artificial Intelligence Research Institute (IIIA-CSIC)
Campus UAB, 08193 Bellaterra, Catalonia (Spain)
Email: {eva, enric}@iiia.csic.es

Abstract. A desired capability of automatic problem solvers is to explain their 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 the explanations can be used in CBR methods in order to justify the results in classification tasks and also for solving new problems.

1 Introduction

A desired capability of automatic problem solvers is to explain their results they produce. Such explanations should justify that the 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 the explanation of problem solvers based on cases could be 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 [10] distinguishes three key issues for 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 such explanations 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 solving new problems.

There is a family of machine learning methods, called *explanation-based learning* that give an explanation of the solution. Then this explanation is generalized and it can be further used for solving new problems. This kind of explanation does not take benefit of the previous experience since they are built from scratch. The best explanation is chosen using probability or plausibility and without taking into account changes in the information that has motivated the explanation. Leake [11] contrasts this kind of explanation, that he calls *based on deductive proofs* with the *explanations based on plausibility*. This kind of explanation, commonly used in *case-based reasoning*, works under the assumption that similar situations have similar explanations. Thus, there is a set of explanation patterns

able to explain anomalous situations and when a similar anomaly is detected in the current situation, the associated explanation pattern can be used to explain it. Therefore, the similarity assessment among situations is a key issue for the generation of appropriate explanations.

In the next sections we briefly describe both explanation-based learning (EBL) and case-based reasoning (CBR) and the role that explanations can play in both kind of methods. Then, we discuss some aspects of the justification provided by the LID method [2] comparing it with both EBL and CBR. LID is a lazy learning method that justifies the classification of a new problem by means of a structure containing the most relevant features allowing the classification of the new problem.

2 Related Work

In this section we review existing approaches in Machine Learning and Case-based Reasoning that use the concept of explanation.

2.1 Explanation-based Learning

The main idea of explanation-based learning methods (also called *analytical methods* or *deductive learning methods*) is to use domain knowledge and to apply logical deduction to solve a problem. The basis of the EBL methods is the *explanation-based generalization* (EBG) method proposed by Mitchell et al. [14]. Given a domain theory, a description of the goal concept, an operationality criterion and a training example, the EBG method tries to improve the domain theory in order to obtain a more efficient (operational) definition of the goal concept.

EBL has two main steps (Fig. 1): 1) to build an explanation justifying why the input example is a positive instance of the goal; and 2) generalizing the explanation as much as possible while the explanation holds. The explanation is the proof tree build by the system during the problem solving process. Therefore, the explanation is generated in a deductive way and its generalization will be correct since deductive methods are truth-preserving. Finally, from the generalized explanation new rules can be generated and stored as part of the domain theory and they can be used for solving further problems.

The main advantage of the EBL methods is that the explanations they produce are correct since they are deductively constructed. Therefore, in principle, when an explanation can be applied to a new situation, we can assure that such explanation is appropriate. Nevertheless, this is only true when the domain theory is complete and sound which is not always the case. Frequently, complex domains do not have a complete theory and, even if they did, it could be very difficult to formalize appropriately. As a consequence, EBL produces two kinds of problems [7]: theory revision and theory reformulation. The theory revision problem is a consequence of the available domain theory. Theory reformulation is to assure the learned knowledge is really useful. Both problems are out of the scope of this paper.

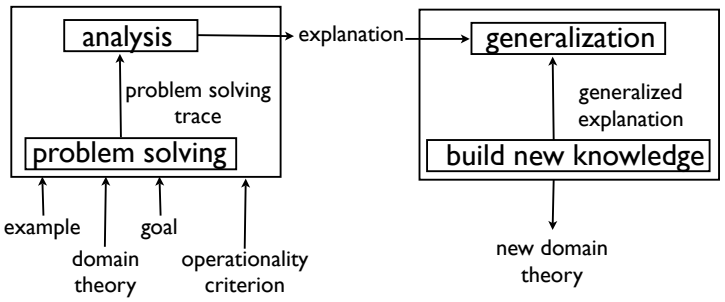

Fig. 1. Main concepts in explanation-based learning.

2.2 Case-based Reasoning

Case-based reasoning methods [9] are based on the retrieval of past experiences to solve a new problem. Figure 2 shows the steps of CBR [1]. Given a new case and a case base the first step is the retrieval of a subset of cases similar in some aspect to the new case. From this subset of cases the next step is to decide how to adapt their solution to the new case. This suggested solution can be revised to provide the final solution and also to retain the new case and the solution for further use.

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 by means of indexes that reflect the relevance of some case features. Nevertheless, as Leake points out in [11], the relevance of a case feature depends on the context. This means that, given a new case, the subset of retrieved cases should depend on the context since the relevant features could be different. Moreover, the adaptation of the solutions will also depend on the retrieved cases. In both situations, the explanation of the decisions taken supports the confidence in the final solution.

There are several CBR methods that explain their reasoning. CHEF [8] is a case-based planner that uses its own experience to develop new cooking recipes that accomplish some user-decided goals. CHEF contains all the necessary background knowledge and also a simulator able to perform the proposed recipe. CHEF detects the points where the new plan does not perform as is expected and tries to explain the reasons of the failures.

SWALE [17] is a program for story understanding that detects anomalous facts and uses CBR to explain anomalies. The explanation process is based on the retrieval and application of cases that store previous explanations called *explanation patterns* (XP). Anomalous situations are characterized in terms of a set of indices and asks which XP explains similar situations. SWALE needs a high amount of domain knowledge in the form of relations between concepts and

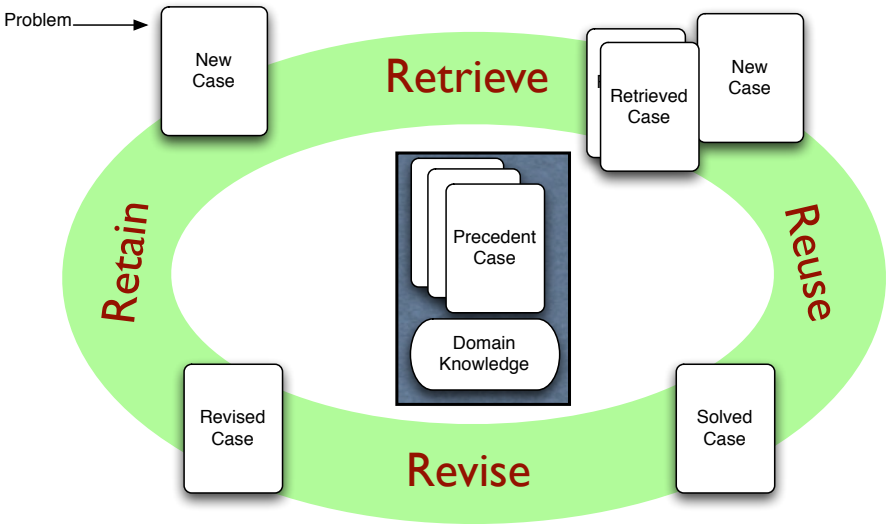

Fig. 2. Case-based reasoning

explanation patterns. New explanations are achieved by directly entering them or by learning from wrong applications of other explanations.

The main difference between CHEF and SWALE is that CHEF selects only one past recipe and tries to adapt it. Instead, SWALE retrieves several stories and builds an explanation for each one of them trying to justify the similarity to the new case. Then, it has to select the best explanation. In fact, SWALE uses a CBR method to find appropriate explanations.

3 Explanation and structural similarity

As we saw in the previous section, 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 considering that cases are represented as attribute-value pairs. Nevertheless, cases can also be represented as structures and in such situation the kind of similarity to be applied has to take into account the structural similarity between two cases [5, 6, 3].

Another approach to assess structural similarity is to consider the shared structure between two cases [16]. For this discussion, we will consider the problem space $\mathcal{P}$ as the collection of all possible case descriptions (ie. without the case solutions) in a given language and their generalizations. Morover, consider Figure 3 showing the "similarity space" $\mathcal{P}(A, B)$ between two cases: case A with solution S_A and case B with solution S_B. The *similarity space* $\mathcal{P}(A, B)$ is a subset of the problem space $\mathcal{P}$ formed by the collection of terms generalizing the problem

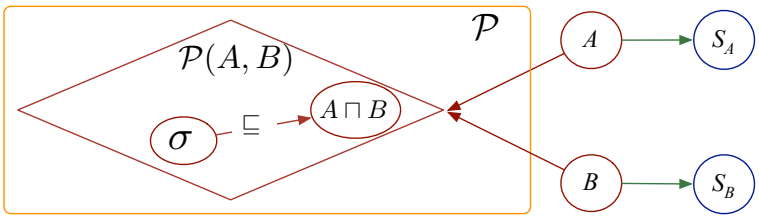

Fig. 3. Schema of the problem space $\mathcal{P}$ and the similarity space $\mathcal{P}(A, B)$ of two cases A and B. $\mathcal{P}$ contains all possible similitude terms (σ) of cases A and B. Antiunification $A \sqcap B$ is the similitude term that contains *all* that is common to A and B.

descriptions of A and B. That is to say, a term $\sigma \in \mathcal{P}(A, B)$ satisfies both $\sigma \sqsubseteq A$ (σ subsumes or is more general than A) and $\sigma \sqsubseteq B$. In particular, the antiunification $A \sqcap B$ (or most specific generalization) is the term that contains *all* that is common or shared between A and B; other terms $\sigma \in \mathcal{P}(A, B)$ contain only *some* commonalities between A and B. Plaza [16] proposed to retrieve cases for a case base $\mathbf{B}$ for a problem P by computing the *similitude terms* $\Sigma(P, \mathbf{B}) = \{\sigma_i | P \sqcap A \wedge B \in \mathbf{B}\}$. Then the similitude terms σ_i are ranked using an information entropy measure and the cases with top-ranking σ_i are retrieved.

However, taking into account *all* similarities (i.e. using antiunification) 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" similarity term σ_{i_P} for a problem P. The similarity term is the best in the sense that σ_{i_P} *retrieves* the cases most similar to P, i.e. is a generalization of P and the cases $A_i \ldots A_k$ that would be the best precedents for solving P

Lazy Induction of Descriptions (LID) is a CBR technique that uses this problem space search approach. Given a problem P, LID follows a heuristic top-down approach search in $\mathcal{P}$. Top-down search in $\mathcal{P}$ means that LID follows a general to specific strategy in the generalizations of P. LID starts with the most general generalization, i.e. the empty 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 most relevant to P.

At each step of the search process LID has a similitude term σ and the associated *discriminatory set* $\mathcal{D}(\sigma) = \{A_i \in \mathbf{B} | \sigma \sqsubseteq A_i\}$ ie. the cases subsumed by the similarity term. Therefore LID has to determine whether this is 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" similarity 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 [12]. 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*, i.e. the partition over the case base $\mathbf{B}$ given by the solutions classes.

When LID solves a problem P provides as outcome the 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 is of class S_i and this solution is endorsed by the cases $\mathcal{D}$. 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 $\mathcal{D}$.

Since σ is a generalization, but subsumes only a subset of the cases in class S_i, it only partially characterizes S_i, while induction methods typically try to build generalizations that completely characterize each class. In fact LID will build a new explanation, a new partial generalization, for each new problem to be solved. The next section explores how to exploit these explanations further.

4 Caching LID

C-LID (Caching LID) [4] 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, learning can be seen as a method that builds a *local approximation* of the target concept (*local* since it is around the problem P); induction (or eager learning) aims at building *global approximations* of the target concept that has to be useful to solve any future problem. 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 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 as classified in class S_i with explanation σ, the caching policy decides whether σ is retained or discarded. If σ is retained it is cached into the set of *patterns* (local approximations) for class S_i. 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 whose cases in the discriminatory set $\mathcal{D}$ all belong to a unique 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 increase their scope) but they increase the uncertainty when they are reused.

The *reuse policy* decides when and how the patterns (the cached explanations) are reused to solve new problems. We performed experiments [4] using several policies in order to compare the accuracy of them. The first policy is the following: 1) A new problem P is solved using LID; 2) if LID cannot uniquelly classify P then the patterns are used. A different policy is to prefer the patterns, i.e. if P satisfies patterns that classify it in one solution class S_i, then P is classified as belonging to S_i; otherwise P is solved using LID. Finally, there is a third reuse policy consisting of solving P using both LID and the patterns. In such situation P is classified as belonging to the solution class proposed by the majority of LID and patterns. We ran experiments on several databases of the UCI repository and the results show, as expected, that the best strategy depends on domain characteristics but definitively CLID accuracy improves LID.

5 Justification-based Multiagent Learning

An explanation can have other uses in addition to explaining the solution found. In fact, when a CBR system builds an explanation J of why S_k is the correct solution for a problem P, the explanation J is the *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 has been used in multiagent systems to improve collaborative problem solving [15].

Consider a committee of CBR agents in which each CBR agent owns a private case base. This committee of agents works in the following way: when a new problem P has to be solved, each individual agent solves it and makes its individual prediction. Then each agent casts a vote for the predicted class and the most voted class 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. Therefore, when the committee is solving a problem, if there are some agent members that are not very knowledgeable of the area of the problem space needed to solve the current problem the other agents in the system will not notice it. This can mean that some of the agents to cast unreliable votes and justifications can help to solve that problem.

Therefore the problem is that an agent does not know if the other agents are casting votes that are reliable or not. However, if each agent can generate a *justification* of the solution found the other agents could see if the justification is strong or if the agent has given a very weak justification and its vote can be ignored. The voting process among the committee using justifications can be performed in the following way [15]:

1. Given a new problem P to solve, each agent AG_i in the committee makes individual predictions and generates a justification record $\langle J_i, S_i \rangle$ that is sent to the rest of agents in the committee (meaning that the agent AG_i believes that the correct solution class is S_i and the justification is J_i).

2. Each agent AG_j *examines* the justification records $\langle J_i, S_i \rangle$ built by each other agent against its local case base C_j. This examination is performed by searching for cases that are counterexamples of the justification and also cases that endorse the justification (a case c is a counterexample of a justification if c satisfies J_i but belongs to a different class than S_i, and a case c endorses the justification if c satisfies J_i and belongs to S_i).
3. 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 is a function of the sum of all the counterexamples found by all the agents and of the sum of all the endorsing cases found by all the agents (the more endorsing cases, the higher the confidence will be and the more counterexamples, the smaller the confidence).
4. Finally, the confidence values can be used as weights in a weighted voting scheme where each agent votes for its individually predicted solution class, but its vote is weighted by the confidence computed by the rest of the agents.

This voting scheme is much more robust, and if an agent AG_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 AG_i and its vote will not influence very much in the final outcome. Notice that this is a dynamic scheme where the agent prediction weights are computed for each specific problem.

6 Conclusions

In this paper we show how the explanation concept could be introduced in a CBR method. Thus, we briefly introduced LID, a CBR method capable of giving a justification of the result. This justification, called similitude term, contains the relevant aspects shared by the new problem and a subset of precedents belonging to one solution class. Thus, the similitude term can be seen as a local approximation of the target concept (i.e. the solution class). In C-LID a lazy learning technique built on top of LID, we used the similitude terms for solving further problems. 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.

We have also seen that explanations are not only useful to provide to a human expert, but can also be a tool to improve problem solving. We have presented a multiagent setting that can take advantage of the ability of CBR agents to generate explanations (used as justifications of the solutions found for problems). Moreover, we strongly believe that explanations can have many other uses in multiagent systems, where the performance of an agent is usually dependant on information provided by other agents and thus it would be highly desirable that any information coming from others is properly justified.

Acknowledgements This work has been supported by the SAMAP project (TIC2002-04146-C05-01).

References

1. Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–59, 1994.
2. E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In *Machine Learning: ECML-2001*, number 2167 in Lecture Notes in Artificial Intelligence, pages 13–24. Springer-Verlag, 2001.
3. E. Armengol and E. Plaza. Similarity assessment for relational cbr. In David W. Aha and Ian Watson, editors, *CBR Research and Development. Proceedings of the ICCBR 2001. Vancouver, BC, Canada.*, number 2080 in Lecture Notes in Artificial Intelligence, pages 44–58. Springer-Verlag, 2001.
4. E. Armengol and E. Plaza. Remembering similitude terms in case-based reasoning. In *3rd Int. Conf. on Machine Learning and Data Mining MLDM-03*, number 2734 in Lecture Notes in Artificial Intelligence, pages 121–130. Springer-Verlag, 2003.
5. K. Börner. Structural similarity as guidance in case-based design. In *Proc. of the First European Workshop on Case-based Reasoning, Posters and Presentations, 1-5 November 1993. Vol. I*, pages 14–19. University of Kaiserslautern, 1993.
6. H. Bunke and B.T. Messmer. Similarity measures for structured representations. In *Topics in Case-based Reasoning. EWCBR-94*, pages 106–118, 1994.
7. T. Ellman. Explanation-based learning: A survey of programs and perspectives. *Computing Surveys*, 21:163–222, 1989.
8. K.J. Hammond. *Case-Base Planning. Viewing Planning as a Memory Task*, volume 1. Academic Press, Inc, 1989.
9. Janet Kolodner. *Case-based reasoning.* Morgan Kaufmann, 1993.
10. D.B. Leake. Issues in goal-driven explanation. *Procs of the AAAI Spring symposium on goal-driven learning*, pages 72–79, 1994.
11. D.B. Leake. Abduction, experience and goals: a model of everyday abductive explanation. *Journal of experimental and theoretical Artificial Intelligence*, 7:407–428, 1995.
12. Ramon López de Mántaras. A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6:81–92, 1991.
13. S. Minton. *Learning effective search control knowledge: An explanation-based approach.* PhD thesis, Carnegie Mellon, Computer Science Department, 3 1988.
14. T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based learning: A unifying view. *Machine Learning*, 1(1):47–80, 1986.
15. Santiago Ontañón and Enric Plaza. Justification-based multiagent learning. In *Proc. 20th ICML*, pages 576–583. Morgan Kaufmann, 2003.
16. Enric Plaza. Cases as terms: A feature term approach to the structured representation of cases. In M. Veloso and A. Aamodt, editors, *Case-Based Reasoning, ICCBR-95*, number 1010 in Lecture Notes in Artificial Intelligence, pages 265–276. Springer-Verlag, 1995.
17. R.C. Schank and D.B. Leake. Creativity and learning in a case-base explainer. *Artificial Intelligence*, 40:353–385, 1989.