Arguments and Counterexamples in Case-based Joint Deliberation

Santiago Ontañón[1] and Enric Plaza[2]

[1] CCL, Cognitive Computing Lab
College of Computing, Georgia Institute of Technology
266 Ferst Drive, Atlanta, Georgia 30332 (USA)
santi@cc.gatech.edu
[2] IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia (Spain).
enric@iiia.csic.es

Abstract. Multiagent learning can be seen as applying ML techniques to the core issues of multiagent systems, like communication, coordination, and competition. In this paper, we address the issue of learning from communication among agents circumscribed to a scenario with two agents that (1) work in the same domain using a shared ontology, (2) are capable of learning from examples, and (3) communicate using an argumentative framework. We will present a two fold approach consisting of (1) an argumentation framework for learning agents, and (2) an individual policy for agents to generate arguments and counterarguments (including counterexamples). We focus on argumentation between two agents, presenting (1) an interaction protocol (AMAL2) that allows agents to learn from counterexamples and (2) a preference relation to determine the joint outcome when individual predictions are in contradiction. We present several experiment to asses how joint predictions based on argumentation improve over individual agents prediction.

1 Introduction

Argumentation frameworks for multiagent systems can be used for different purposes like joint deliberation, persuasion, negotiation, and conflict resolution. In this paper, we focus on argumentation-based joint deliberation among learning agents. Argumentation-based joint deliberation involves discussion over the outcome of a particular situation or the appropriate course of action for a particular situation. Learning agents are capable of learning from experience, in the sense that past examples (situations and their outcomes) are used to predict the outcome for the situation at hand. However, since individual agents experience may be limited, individual knowledge and prediction accuracy is also limited. Thus, learning agents that are capable of arguing their individual predictions with other agents may reach better prediction accuracy after such an argumentation process.

In this paper we address the issue of joint deliberation among two learning agents using an argumentation framework. Our assumptions are that these two agents work in the same domain using a shared ontology, they are capable of learning from examples, and they interact following a specific interaction protocol. In this paper, we will propose an argumentation framework for learning agents, and an individual policy for agents to generate arguments and counterarguments.

Existing argumentation frameworks for multiagent systems are based on deductive logic. An argument is seen as a logical statement, while a counterargument is an argument offered in opposition to another argument [6, 18]. However, these argumentation frameworks are not designed for learning agents, since they assume a fixed knowledge base. Learning agents, however may generate several generalizations that are consistent with the examples seen at a particular moment in time; the *bias* of the generalization technique used determines which of the valid generalizations is effectively hold by a learning agent.

Having learning capabilities allows agents a new form of counterargument, namely the use of *counterexamples*. Counterexamples offer the possibility of agents learning during the argumentation process, and thus improving their performance (both individual and joint performance). Moreover, learning agents will allow us to design individual agent policies to generate adequate arguments and counterarguments. Existing argumentation frameworks mostly focus on how to deal with contradicting arguments, while few address the problem of how to generate adequate arguments (but see [18]). Thus, they focus on the issue defining a preference relation over two contradicting arguments; however for learning agents we will need to address two issues: (1) how to define a preference relation over two conflicting arguments, and (2) how to define a policy to generate arguments and counterarguments from examples.

In this paper we present a case-based approach to address both issues. The agents use case-based reasoning (CBR) to learn from past cases (where a case is a situation and its outcome) in order to predict the outcome of a new situation; moreover, the reasoning needed to support the argumentation process will also be based on cases. In particular, both the preference relation among arguments and the policy for generating arguments and counterarguments will be based on cases. Finally, we propose an interaction protocol called AMAL2 to support the argumentation process among two agents to reach a joint prediction over a specific situation or problem.

In the remainder of this paper we are going to introduce the multiagent CBR framework ($\mathcal{M}$AC) in which we perform our research (Section 2). In this framework, Section 2.1 introduces the idea of justified predictions. After that, Section 3 provides a specific definition of arguments and counterarguments that we will use in the rest of the paper. Then, Section 4 defines a preference relation between contradicting arguments. Section 5 presents specific policies to generate both arguments and counterarguments. Using the previous definitions, Section 6 presents a protocol called AMAL2 to allow two agents to solve a problem in a collaborative way using argumentation. Finally Section 7 presents an empiri-

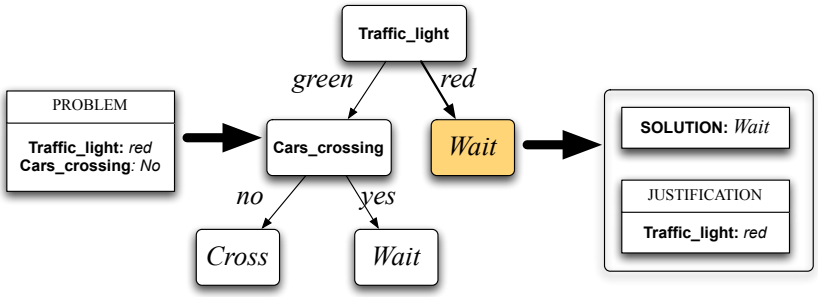

Fig. 1. Simple Justification generation example using a decision tree.

cal evaluation of the argumentation protocol presented. The paper closes with related work and conclusions sections.

2 Case-Based Multiagent Learning

In this section we are going to define the multiagent learning framework in which our research is performed [15].

Definition 1. *A* Multiagent Case Based Reasoning System *(*$\mathcal{M}$AC*)* $\mathcal{M} = \{(A_1, C_1), ..., (A_n, C_n)\}$ *is a multiagent system composed of* $\mathcal{A} = \{A_i, ..., A_n\}$*, a set of CBR agents, where each agent* $A_i \in \mathcal{A}$ *possesses an individual case base* C_i*.*

Each individual agent A_i in a $\mathcal{M}$AC is completely autonomous and each agent A_i has access only to its individual and private case base C_i. A case base $C_i = \{c_1, ..., c_m\}$ is a collection of cases. Agents in a $\mathcal{M}$AC system are able to individually solve problems, but they can also collaborate with other agents to solve problem in a collaborative way.

In this framework, we will restrict ourselves to analytical tasks, i.e. tasks, like classification, where the solution of a problem is achieved by selecting a solution class from an enumerated set of solution classes. In the following we will note the set of all the solution classes by $\mathcal{S} = \{S_1, ..., S_K\}$. Therefore, a case is a tuple $c = \langle P, S \rangle$ containing a case description P and a solution class $S \in \mathcal{S}$. In the following, we will use the terms *problem* and *case description* indistinctly. Moreover, we will use the dot notation to refer to elements inside a tuple. e.g., to refer to the solution class of a case c, we will write $c.S$.

2.1 Justified Predictions

Many expert and CBR systems have an explanation component [19]. The explanation component is in charge of justifying why the system has provided a specific answer to the user. The line of reasoning of the system can then be examined by a human expert, thus increasing the reliability of the system.

Most of the existing work on explanation generation focuses on generating explanations to be provided to the user. However, in our approach we use explanations (or justifications) as a tool for improving communication and coordination among agents. We are interested in justifications to be used as arguments. For that purpose, we take benefit from the ability of some learning systems to provide justifications.

Definition 2. *A* justification *built by a CBR method to solve a problem P that has been classified into a solution class S_k is a description that contains the relevant information that the problem P and the retrieved cases $C_1, ..., C_n$ (all belonging to class S_k) have in common.*

For example, Figure 1 shows a justification build by a decision tree for a toy problem. In the figure, a problem has two attributes (traffic_light, and cars_crossing), after solving it using the decision tree shown, the predicted solution class is wait. Notice that to obtain the solution class, the decision tree has just used the value of one attribute, traffic_light. Therefore, the justification must contain only the attribute/value pair shown in the figure. The values of the rest of attributes are irrelevant, since whatever their value the solution class would have been the same.

In general, the meaning of a justification is that all (or most of) the cases in the case base of an agent that satisfy the justification (i.e. all the cases that are *subsumed* by the justification) belong to the predicted solution class. In the rest of the paper, we will use $\sqsubseteq$ to denote the subsumption relation. In our work, we use LID [3], a CBR method capable of building symbolic justifications. LID uses the feature term formalism (or ψ-terms) to represent cases [2].

We call *justified prediction* the justification for a prediction provided by a learning agent:

Definition 3. *A* justified prediction *is a tuple $\langle A, P, S, D\rangle$ containing the problem P, the solution class S found by the agent A for the problem P, and the justification D that endorses S as the correct solution for P.*

Justifications can have many uses for CBR systems [10, 14]. In this paper, we are going to use justifications as arguments, in order to allow agents to engage case based based argumentation processes.

3 Argumentation in Multiagent Learning

Let us start by presenting a definition of argument that we will use in the rest of the paper:

Definition 4. *An* argument *α generated by an agent A is composed of a statement S and some evidence D supporting that S is correct.*

In the remainder of this section we will see how this general definition of argument can be instantiated in specific kind of arguments that the agents can generate. In the context of $\mathcal{M}$AC systems, agents argue about the correct solution of new problems and can provide information in two forms:

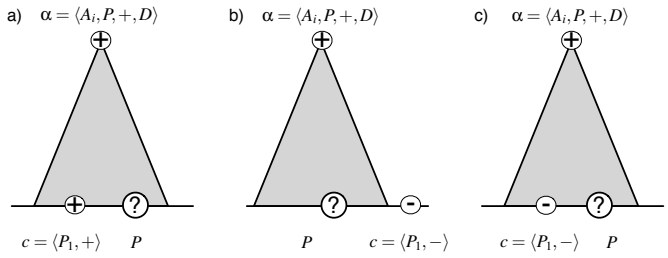

Fig. 2. Relation between cases and justified predictions. The case c is a counterexample of the justified prediction α in c), while it is not in a) and b)

.

- A specific case: $\langle P, S \rangle$,
- A justified prediction: $\langle A, P, S, D \rangle$.

In other words, agents can provide specific cases or generalizations learnt from cases. Using this information, and having in mind that agents will only argue about the correct solution of a given problem, we can define three types of arguments: justified predictions, counterarguments, and counterexamples.

- A *justified prediction* $\alpha = \langle A, P, S, D \rangle$ is generated by an agent A_i to argue that A_i believes that the correct solution for a given problem P is $\alpha.S$, and the evidence provided is the justification $\alpha.D$. In the example depicted in Figure 1, an agent A_i may generate the argument $\alpha = \langle A_i, P, \mathsf{Wait}, (\mathsf{Traffic\_light} = \mathsf{red}) \rangle$, meaning that the agent A_i believes that the correct solution for P is Wait because the attribute $\mathsf{Traffic\_light}$ equals red.
- A *counterargument* β is an argument offered in opposition to another argument α. In our framework, a counterargument consists of a justified prediction $\langle A_j, P, S', D' \rangle$ generated by an agent A_j with the intention to rebut an argument α generated by another agent A_i, that endorses a different solution class than α for the problem at hand and justifies this with a justification D'. In the example depicted in Figure 1, if an agent generates the argument $\alpha = \langle A_i, P, \mathsf{Walk}, (\mathsf{Cars\_crossing} = \mathsf{no}) \rangle$, an agent that thinks that the correct solution is Stop might answer with the counterargument $\beta = \langle A_j, P, \mathsf{Stop}, (\mathsf{Cars\_crossing} = \mathsf{no} \wedge \mathsf{Traffic\_light} = \mathsf{red}) \rangle$, meaning that while it is true that there are no cars crossing, the traffic light is red, and thus the street cannot be crossed.
- A *counterexample* $c = \langle P, S \rangle$ is a case that contradicts an argument α. Specifically, for a case c to be a counterexample of an argument α, the following conditions have to be met: $\alpha.D \sqsubseteq c.P$ and $\alpha.S \neq c.S$. Figure 2 illustrates the concept of a counterexample: justified predictions are shown above the triangles while the specific cases subsumed by the justified predictions are at the bottom of the triangles. Figure 2 presents three situations: In a) c is not a counterexample of α since the solution of c is the solution predicted

by α; in b) c is not a counterexample of α since c is not subsumed by the justification $\alpha.D$; finally, in c) c is a counterexample of α).

By exchanging arguments and counterarguments (including counterexamples), agents can argue about the correct solution of a given problem. However, in order to do so, they need a specific interaction protocol, a preference relation between contradicting arguments, and a decision policy to generate counterarguments (including counterexamples). In the following sections we will present these three elements.

4 Case Based Preference Relation

The argument that an agent provides might not be consistent with the information known to other agents (or even to some of the information known by the agent that has generated the justification due to noise in training data). For that reason, we are going to define a preference relation over contradicting justified predictions based on cases. Basically, we will define a *confidence* measure for each justified prediction (that takes into account the cases known by each agent), and the justified prediction with the highest confidence will be the preferred one.

The confidence of justified predictions is assessed by the agents via an process of *examination of justifications*. During this examination, the agents will count how many of the cases in their case bases *endorse* the justified prediction, and how many of them are counterexamples of that justified prediction. The more endorsing cases, the higher the confidence; and the more the counterexamples, the lower the confidence.

Specifically, to examine a justified prediction α, an agent obtains the set of cases contained in its individual case base that are subsumed by $\alpha.D$. The more of these cases that belong to the solution class $\alpha.S$, the higher the confidence will be. After examining a justified prediction α, an agent A_i obtains the *aye* and *nay* values:

- The aye value $Y_\alpha^{A_i} = |\{c \in C_i |\ \alpha.D \sqsubseteq c.P \wedge \alpha.S = c.S\}|$ is the number of cases in the agent's case base *subsumed* by the justification $\alpha.D$ that belong to the solution class $\alpha.S$ proposed by $\mathbf{J}$,
- The nay value $N_\alpha^{A_i} = |\{c \in C_i |\ \alpha.D \sqsubseteq c.P \wedge \alpha.S \neq c.S\}|$ is the number of cases in the agent's case base *subsumed* by justification $\alpha.D$ that *do not* belong to that solution class.

When two agents A_1 and A_2 want to assess the confidence on a justified prediction α made by one of them, each of them examine the prediction and sends the *aye* and *nay* values obtained to the other agent. Then, both agents have the same information and can assess the confidence value for the justified prediction as:

$$\mathsf{C}(\alpha) = \frac{Y_\alpha^{A_1} + Y_\alpha^{A_2} + 1}{Y_\alpha^{A_1} + Y_\alpha^{A_2} + N_\alpha^{A_1} + N_\alpha^{A_2} + 2}$$

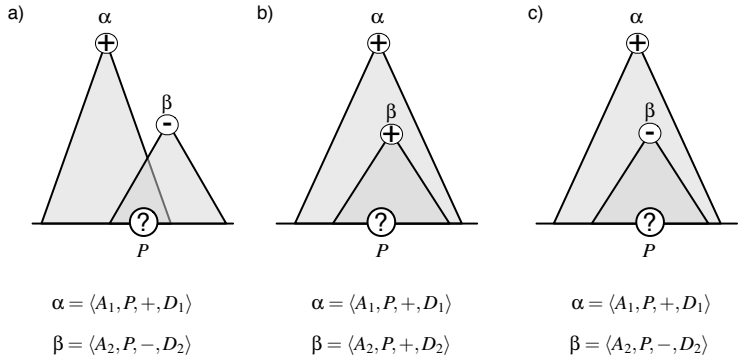

Fig. 3. Relation between arguments

i.e. the confidence on a justified prediction is the number of endorsing cases divided by the number of endorsing cases plus counterexamples found by each of the two agents. The reason for adding one to the numerator and 2 to the denominator is the Laplace correction to estimate probabilities. This prevents assigning excessively high confidences to justified predictions whose confidence has been computed using a small number of cases (in this way, a prediction endorsed by 2 cases and with no counterexamples has a lower confidence than a prediction endorsed by 10 cases with no counterexamples).

Using the previously defined confidence measure, the preference relation used in our framework is the following one: a justified prediction α is preferred over another one β is $C(\alpha) \geq C(\beta)$.

5 Generation of Arguments

In our framework, arguments are generated by the agents using CBR algorithms. However, any learning method able to provide a justified prediction can be used to generate arguments. In particular, we use the LID CBR method [3].

When an agent wants to generate an argument endorsing that a specific solution class is the correct solution for a given problem P, it generates a justified prediction as explained in Section 2.1.

For instance, Figure 4 shows an argument generated by LID in the sponge data set, used in our experiments. Specifically, the argument shown in Figure 4 endorses the solution *hadromerida* for a particular problem P. The justification D_1 in that argument can be interpreted saying that "the prediction for P is hadromerida because the smooth form of the megascleres of the spikulate skeleton of the sponge is of type tylostyle, the spikulate skeleton has no uniform length, and there is no gemmules in the external features of the sponge".

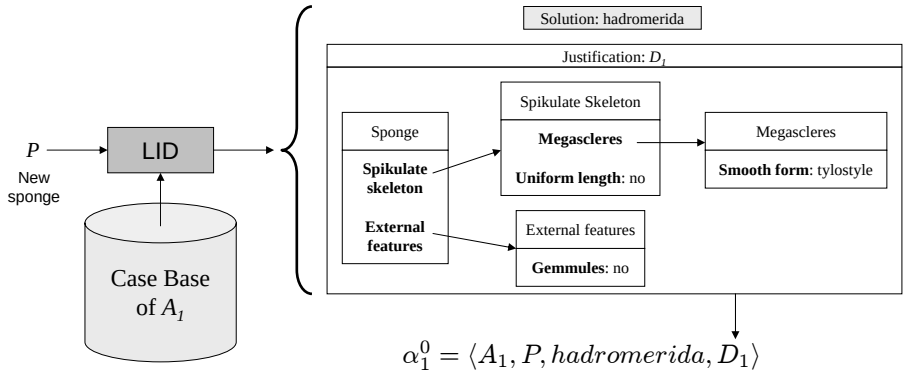

Fig. 4. Example of an argument generated using LID in the marine sponges domain (used in our experiments)

5.1 Generation of Counterarguments

When an agent A_i generates a counterargument β to rebut an argument α, A_i expects that β is preferred over α. With that purpose, in this section we are going to present a specific policy to generate counterarguments based on the *specificity* criterion [16].

The specificity criterion is widely used in deductive frameworks for argumentation, and states that between two conflicting arguments, the most specific should be preferred since it is, in principle, more informed. Thus, counterarguments generated based on the specificity criterion are expected to be preferable (since they are more informed) to the arguments they try to rebut. However, there is no guarantee that such counterarguments will always win, since we use a preference relation based on joint confidence. Moreover, one may think that it would be better that the agents generate counterarguments based on the joint confidence preference relation; however that is not feasible, since collaboration is required in order to evaluate joint confidence. Thus, the agent generating the counterargument should constantly communicate with the other agents at each step of the CBR algorithm used to generate counterarguments.

Therefore, when an agent wants to generate a counterargument β to an argument α, it will generate a counterargument that it is more specific than α. Figure 3 illustrates this idea. In Figure 3.c) β is a counterargument of α, and is more specific than α. However in Figure 3.a) β is not more specific than α and in Figure 3.c) both arguments endorse the same solution, and thus β is not a counterargument of α.

The generation of counterarguments using the specificity criterion imposes some restrictions over the learning method, although LID or ID3 can be easily adapted to generate counterarguments. For instance, LID is an algorithm that generates a description starting by the empty term and heuristically adding features to that term. Thus, at every step, the description is made more specific

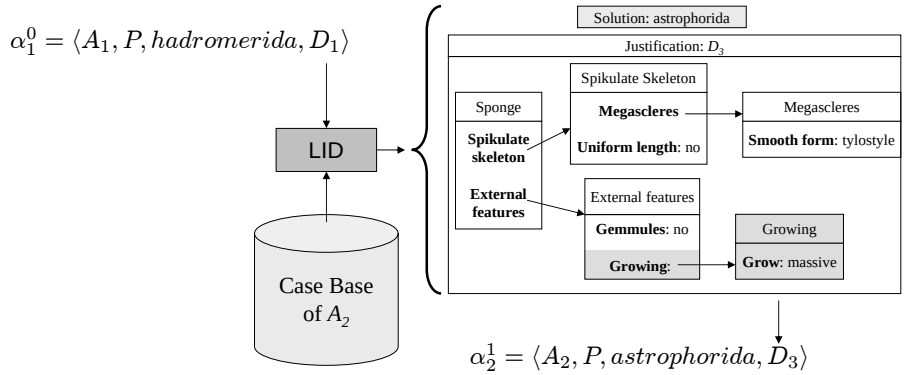

Fig. 5. Example of a counterargument generated using LID in the marine sponges domain (used in our experiments)

than in the previous step, and the number of cases that are subsumed by that description is reduced. When the description only covers cases of a single solution class LID terminates and predicts that solution class. To generate a counterargument to an argument α LID just has to use as starting point the description $\alpha.D$ instead of the empty term. In this way, the justification provided by LID will always be subsumed by $\alpha.D$, and thus the resulting counterargument will be more specific than α. However, notice that LID may sometimes not be able to generate counterarguments, since LID may not be able to specialize the description $\alpha.D$ any further, or because the agent does not own any cases subsumed by $\alpha.D$ to run LID.

For instance, in response to the argument in Figure 4, an agent may generate the counterargument shown in Figure 5. The interpretation of the justification is similar as the previous one, but now "the grow rate of the external features of the sponge is massive". Finally, notice that D_3 is more specific than D_1.

Moreover, notice that agents can also try to rebut the other agents arguments using counterexamples. Specifically, in our experiments, when an agent A_i wants to rebut an argument α, uses the following policy:

1. The agent A_i tries to generate a counterargument β more specific than α (in our experiments agents se LID). If A_i succeeds, β is sent to the other agent as a counterargument of α.
2. If not, then A_i searches for a counterexample $c \in C_i$ of α in its individual case base C_i. If such a case c is found, then c is sent to the other agent as a counterexample of α.
3. If no counterexamples are found, then A_i cannot rebut the argument α.

Notice that agents only send specific cases to each other if a counterargument cannot be found. To understand why have we done that, we must have in mind a known result in ensemble learning stating that when aggregating the predictions of several classifiers (i.e. agents) correlation between their predictions must be

low in order to have a good classification accuracy [13]. Therefore, since when a counterexample is sent to the other agent the degree of correlation between the two agents case bases increases, agents prefer to send a counterargument whenever possible, and only send a counterexample only when it is not.

The next section presents the interaction protocol we propose to perform argumentation in our learning framework.

6 Argumentation-based Multiagent Learning

In this section we will present the Argumentation-based Multiagent Learning Protocol for 2 agents (AMAL2). The idea behind AMAL2 is to allow a pair of agents to argue about the correct solution of a problem, arriving at a join solution that is based on their past learning and the information they exchange during argumentation.

At the beginning of the protocol, both agents will make their individual predictions for the problem at hand. Then, the protocol establishes rules allowing one of the agents in disagreement with the prediction of the other to provide a counterargument. Then, the other agent can respond with another counterargument, and so on.

In the remaining of this section we will present all the elements of the AMAL2 protocol. First, we will formally present the specific performatives that the individual agents will use in the AMAL2 protocol, that will allow them to state a prediction, to rebut an argument, and to withdraw an argument that the other agents arguments have rendered invalid. Then we will present the AMAL2 protocol.

6.1 Protocol Performatives

During the AMAL2 protocol, each agent will propose arguments and counterarguments to argue about which is the correct solution for a specific problem P. The AMAL2 protocol consists on a series of rounds. In the initial round, both agents state with are their individual predictions for P, then, at each iteration an agent can try to rebut the prediction made by the other agent, or change his own prediction. Therefore, at each iteration, each of the two agents holds a prediction that it believes is the correct one.

We will use $H_t = \langle \alpha_1^t, \alpha_2^t \rangle$ to note the pair of predictions that each agent holds at a round t. When at a certain iteration an agent changes its mind and changes the prediction it is holding (because it has been convinced by the counterarguments of the other agent), it has to inform the other agent using the withdraw performative.

At each iteration, agents can send one of the following performatives to the other agent:

- $assert(\alpha)$: meaning that the prediction that the agent is holding for the next round will be α.

– $rebut(\alpha, \beta)$: meaning that the agent has found a counterargument or a counterargument α to the prediction β.
– $withdraw(\alpha)$: meaning that the agent is removing a justified prediction α, since the counterarguments presented by the other agent have rendered it invalid.

In the next section the AMAL2 protocol is presented that uses the performatives presented in this section.

6.2 Case Based Argumentation Protocol

The AMAL2 protocol among two agents A_1 and A_2 to solve a problem P works in a series of rounds. We will use t to denote the current round (initially $t = 0$). The idea behind protocol is the following one. Initially, each agent makes its individual prediction. Then, the confidence of each prediction is assessed, and the prediction with the highest confidence is considered the winner. However, if the agent that has provided the prediction with lower confidence doesn't agree, it has the opportunity to provide a counterargument. Agents keep exchanging arguments and counterarguments until they reach an agreement or until no agent is able to generate more counterexamples. At the end of the argumentation, if the agents have not reached an agreement, then the prediction with the highest confidence is considered the final prediction.

Notice that the protocol starts because one of the two agents receives a problem to be solved, and that agent sends the problem to the other agent requesting him to engage in an argumentation process. Thus, after both agents know the problem P to solve, round $t = 0$ of the protocol starts:

1. Initially, each one of the agents individually solves P, and builds a justified prediction (A_1 builds α_1^0, and A_2 builds α_2^0). Then, each agent A_i sends the performative $assert(\alpha_i^0)$ to the other agent. Thus, both agents know $H_0 = \langle \alpha_1^0, \alpha_2^0 \rangle$.
2. At each round t, the agents check whether their arguments in H_t agree. If they do, the protocol moves to step 4, otherwise the agents compute the confidence for each argument and use the preference relation (presented in Section 4) to determine which argument in H_t is preferred. After that, the agent that has provided the non preferred argument may try to rebut the other agent's argument. Each individual agent uses its own policy to rebut arguments:
   – If an agent A_i generates a counterargument α_i^{t+1}, then it sends the following performatives to the other agent, A_j, in a single message: $rebut(\alpha_i^{t+1}, \alpha_j^t), withdraw(\alpha_i^t), assert(\alpha_i^{t+1})$. This message starts a new round $t + 1$, and the protocol moves back to step 2.
   – If an agent A_i selects c as a counterexample of the other agent's justified prediction, then A_i sends the following performative to the other agent, A_j: $rebut(c, \alpha_j^t)$. The protocol moves to step 3.
   – If no agent provides any argument the protocol moves to step 4.

3. The agent A_j that has received the counterexample c retains it and generates a new argument α_j^{t+1} that takes into account c. To inform A_i of the new argument, A_j sends A_i the following performatives $withdraw(\alpha_j^t), assert(\alpha_j^{t+1})$. This message starts a new round $t+1$, and the protocol moves back to step 2.
4. The protocol ends yielding a joint prediction, as follows: if both arguments in H_t agree, then their prediction is the joint prediction; otherwise the prediction in H_t with the higher confidence is considered the joint prediction.

Moreover, in order to avoid infinite iterations, if an agent sends twice the same argument or counterargument, the protocol also terminates.

Finally notice that when an agent A_i submits a counterargument α that defeats the other agents argument, then α becomes A_is argument, and thus the other agent may try to rebut it using another counterexample.

6.3 Exemplification

Let us consider two agents A_1 and A_2. One of the agents, A_1, receives a problem P to solve, and decides to use AMAL2 to solve it. In particular, the problem consists on identifying the proper order of a given marine sponge. For that reason, invites A_2 to take part in the argumentation process. A_2 accepts the invitation, and the argumentation protocol starts.

Initially, each agent generates its individual prediction for P, and assert it using the *assert* performative. Thus, both of them can compute $H_0 = \langle \alpha_1^0, \alpha_2^0, \rangle$. In particular, in this example:

- $\alpha_1^0 = \langle A_1, P, hadromerida, D_1 \rangle$ (specifically, the argument generated by A_1 in this example is the one shown in Figure 4).
- $\alpha_2^0 = \langle A_2, P, astrophorida, D_2 \rangle$

Then, the agents check whether their arguments agree. Since they don't agree (one predicts that the order of the sponge is *hadromerida* and the other one says that it is *astrophorida*), they evaluate the confidence of each of the arguments to see which is the preferred one. Specifically, they obtain the following confidence values:

- $C(\alpha_1^0) = 0.69$
- $C(\alpha_2^0) = 0.50$

Therefore, the preferred argument is the one of A_1, since it has the highest confidence. For that reason, A_2 will try to generate a counterargument to it. Specifically, A_2 generates the counterargument $\alpha_2^1 = \langle A_2, P, astrophorida, D_3 \rangle$ (shown in Figure 5). Then, A_2 uses the *withdraw* performative to withdraw his previous argument α_2^0, the *assert* performative to assert its new argument α_2^1, and the *rebut* performative to announce that α_2^1 is a counterargument of α_1^0.

This starts a new round, where $H_1 = \langle \alpha_1^0, \alpha_2^1, \rangle$. The agents check again if their arguments agree, but they still don't agree. Thus, they evaluate again the confidence of the arguments, and they obtain the following:

- $C(\alpha_1^0) = 0.69$
- $C(\alpha_2^1) = 0.71$

This time, it is A_1 who has to generate a counterargument, since the preferred argument is α_1^0, the one of A_2. In particular, in this example, A_1 fails to find a counterargument, but finds a counterexample c of α_2^1. Thus, A_1 sends c to A_2 using the *rebut* performative.

After receiving the counterexample c, A_2 incorporates it into its case base and tries to generate an updated prediction for the problem P that takes into account the recently learnt counterexample. The prediction generated is $\alpha_2^2 = \langle A_2, P, hadromerida, D_4 \rangle$. Thus, A_2 withdraws his previous prediction with the *withdraw* performative and asserts the new one using the *assert* performative.

This starts a new round, where $H_2 = \langle \alpha_1^0, \alpha_2^2, \rangle$. The agents check again if their arguments agree this time, which they do since they both predict *hadromerida*. Thus, the protocol ends yielding *hadromerida* as the final prediction for problem P. Moreover, as a side effect of the argumentation process A_2 has learnt a new case (the counterexample c sent by A_1) that not only has been useful to correct this prediction but will help to improve the future performance of A_2.

7 Experimental Evaluation

In this section we empirically evaluate the AMAL2 argumentation protocol. We have made experiments in two different data sets: sponge, and soybean. The sponge data set is a marine sponge classification problem, contains 280 marine sponges represented in a relational way and pertaining to three different orders of the Demospongiae class. The soybean data set is a standard data sets from the UCI machine learning repository, with 307 examples pertaining to 19 different solution classes.

In an experimental run, training cases are distributed among the agents without replication, i.e. there is no case shared by two agents. In the testing stage problems arrive randomly to one of the agents. The goal of the agent receiving a problem is to identify the correct solution class of the problem received.

Each experiment consists of a 10-fold cross validation run. An experiment consists of training and test phases as usual; during the training phase the training cases are distributed among the two agents in different ways, as we will see later. During the test phase learning is disabled, i.e. the agents cannot learn from one test case to the next (in order to evaluate all test cases uniformly). This is relevant here because the agents solving a test case can also learn from other cases (the counterexamples in the argumentation process). To keep test case uniformity the agents discard the cases learnt during the argumentation of a test case before moving to argue about the next test case.

Moreover, we have made experiments in four different scenarios: in the first scenario, a 100% of the cases of the training set are distributed among the agents; in the second scenario, the agents only receive a 75% of the training cases; in the third scenario, they only receive a 50%; finally in the fourth scenario agents only

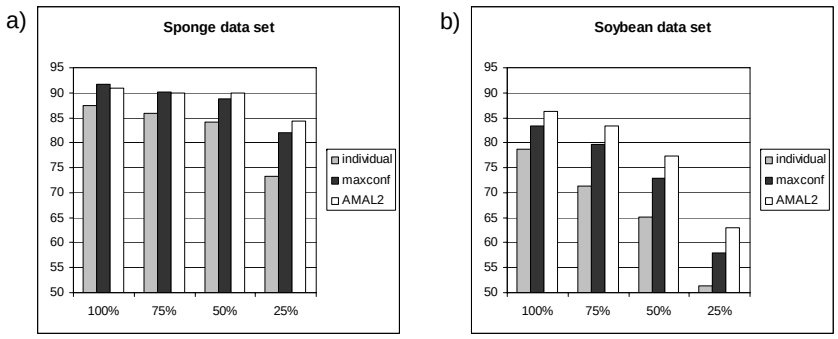

Fig. 6. Classification accuracy results in the Sponge and Soybean domains.

receive a 25% of the training cases. So, for instance, in the sponge data set (that has 280 cases), since we use 10-fold cross validation, 254 cases (a 90%) form the training set and 28 cases (a 10%) from the test set in each experimental run. In the scenario where a 50% of the training cases are distributed, then, only 127 of the cases in the training set will be given to the agents, thus each agent will receive 63.5 cases in average (since the training cases are split among the two agents).

We have made those experiments to see how the argumentation protocol (and how the argument generation policies) work when the agents have different amount of data.

Figures 6.a and 6.b show the classification accuracy achieved by agents using the AMAL2 argumentation protocol in the sponge and soybean data sets. For each of the 4 scenarios (100%, 75%, 50% and 25%) three bars are shown: individual, *maxconf* and AMAL2. The individual bar represents the classification accuracy achieved by agents solving problems individually, the *maxconf* bar represents classification accuracy of the two agents using the following simple strategy: both agents solve the problem individually, then they evaluate the confidence of both predictions, and the prediction with the highest confidence is selected (notice that this is equivalent to using the AMAL2 protocol without any agent providing any counterargument). Finally, the AMAL2 bar represents the classification accuracy of the two agents using the AMAL2 protocol.

Figures 6.a and 6.b show several things. First, that using collaboration is always beneficial, since both *maxconf* and AMAL2 systematically outperform the individual agents in terms of accuracy. Moreover, both figures also show that the accuracy achieved by AMAL2 is higher than that of *maxconf* (in fact, AMAL2 is better or equal than *maxconf* in all the experiments except in the 100% scenario of the sponge data set). Moreover, the less data the individual agents have the greater the benefits of AMAL2 are. When each individual agent has enough data, then predictions and confidence estimations are reliable, and thus little or nothing is gained form the argumentation. However, when agents have access to limited data, the argumentation process helps them finding predictions

that take into account more information, thus making the joint prediction more accurate.

To show that our approach is proficient we can compare our results with that of a single agent owning all the cases. In this centralized scenario the accuracy is 89.64% for the sponge data set, and 89.12% for the soybean data set. These results should be compared with the 100% scenarios, where individual agents achieve a much lower accuracy but using AMAL2 they achieve a comparable performance to that of the centralized approach. Specifically, in the sponges data set the accuracy of 89.64% goes down to 87.43% for individual agents, and using AMAL2 the accuracy is 90.86%, that recovers and even surpasses the centralized accuracy. In the soybean data set the accuracy of 89.12% goes down drastically to 78.63% for individual agents, and using AMAL2 the accuracy is 86.25%, that significantly recovers but not surpasses the centralized accuracy. The difference between these two data sets is that the soybean data set has a large number of classes and thus performance drastically diminishes when dividing the data set among two agents (since the likelihood of an agent having cases of each specific class diminishes). In practical terms this accuracy can be recovered by adding redundancy to the case bases of the agents, i.e. allowing some duplicated cases (cases that are present in both case bases) [11].

Summarizing, collaborating agents (either using argumentation or the simple *maxconf* method) always increase their performance with respect to their individual performance. Similarly, using argumentation generally improves with respect to just using the simple *maxconf* aggregation function. However, when each individual agent has enough data, little is gained form the argumentation with respect to using *maxconf* aggregation function. Finally, when agents have access to limited data, there is ample opportunity for them to learn from communicating with another agent; the experiments reflect this hypothesis by the fact that argumentation in this situations increases performance to a larger degree.

8 Related Work

Research on MAS argumentation focus on several issues like a) logics, protocols and languages that support argumentation, b) argument selection and c) argument interpretation. Approaches for logic and languages that support argumentation include defeasible logic [6] and BDI models [18]. An overview of logical models of reasoning can be found at [5]. Moreover, the most related area of research is case-based argumentation. Combining cases and generalizations for argumentation has been already used in the HYPO system [4], where an argument can contain both specific cases or generalizations. Moreover, generalization in HYPO was limited to selecting a set of predefined dimensions in the system while our framework presents a more flexible way of providing generalizations. Furthermore, HYPO was designed to provide arguments to human users, while we focus on agent to agent argumentation. Case-based argumentation has also been implemented in the CATO system[1], that models ways in which experts compare and contrast cases to generate multi-case arguments to be presented

to law students. Moreover, the goal of CATO differs from the goal of our work, since it is designed to allow law students to learn basic case-based argumentation law skills.

Concerning CBR in a multiagent setting, the first research was on negotiated case retrieval [17] among groups of agents. Our work on multiagent case-based learning started in 1999 [8]; while Mc Ginty and Smyth [9] presented a multiagent collaborative CBR approach (CCBR) for planning. Finally, another interesting approach is multi-case-base reasoning (MCBR) [7], that deals with distributed systems where there are several case bases available for the same task and addresses the problems of cross-case base adaptation. The main difference is that our MAC approach is a way to distribute the Reuse process of CBR while Retrieve is performed individually by each agent; the other multiagent CBR approaches, however, focus on distributing the Retrieve process.

9 Conclusions and Future Work

In this paper we have presented a learning framework for argumentation. Specifically, we have presented AMAL2, a protocol that allows two agents to argue about the solution of a given problem. Finally, we have empirically evaluated it showing that the increased amount of information that the agents use to solve problems thanks to the argumentation process increases their problem solving performance, and specially when the individual agents have access to a limited amount of information. Clearly, an agent that knows all it needs does not need external help (nor, by the way, needs to continue learning if there is no room for improvement).

The main contributions of this work are: a) an argumentation framework for learning agents; b) a case based preference relation over arguments, based on computing a joint confidence estimation of arguments (this preference relation has sense in this learning framework since arguments are learnt from examples); c) a specific and efficient policy to generate arguments and counterarguments based on the specificity relation (commonly used in argumentation frameworks); d) a principled usage of counterexamples in the argumentation process, and e) a specific argumentation protocol for pairs of agents that collaborate to decide the joint solution of a given problem.

Moreover, in this work, we have focused on argumentation as a process to improve overall performance. However, notice that the proposed argumentation framework can also be used as a learning framework. Specifically, in [12] we show that the argumentation framework presented in this paper can be used by a group of agents to learn from each other. By engaging in argumentation processes, an agent might find weak points in the arguments generated by another agent and send him counterexamples of those wrong arguments. Notice that the agent that generated the wrong argument is certainly interested in learn from those specific counterexamples, since retaining them as cases in their case base will prevent him to generate the same wrong argument in the future. Therefore, by engaging in multiple argumentation processes with each other, each individual agent in a

group of agents can easily improve its individual accuracy by learning from the communication content of an argumentation process.

Finally, the work presented in this paper concerns only pairs of agents. However, as future work we plan to generalize the AMAL2 protocol to work with a larger number of agents. A possibility to do that is a token based protocol where the agent owner of the token engages in a 1-to-1 argumentation dialog with every other agent that disagrees with its prediction. When all these 1-to-1 argumentation dialogs have finished, the token passes to the next agent. This process continues until no agent engages in any new 1-to-1 argumentation. Then, from the outcome of all the 1-to-1 argumentation processes, a joint prediction will be achieved just as now on step 4 of the AMAL2 protocol: either the agreed prediction or the one with higher confidence.

Acknowledgments. This research was partially supported by the MID-CBR project TIC2006-15140-C03-01.

References

[1] Vinvent Aleven. *Teaching Case-Based Argumentation Through a Model and Examples*. PhD thesis, University of Pittsburgh, 1997.

[2] E. Armengol and E. Plaza. Bottom-up induction of feature terms. *Machine Learning Journal*, 41(1):259–294, 2000.

[3] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In *Proceedings of the 10th European Conference on Machine Learning, ECML'2001*, pages 13–24, 2001.

[4] Kevin Ashley. Reasoning with cases and hypotheticals in hypo. *International Journal of Man-Machine Studies*, 34:753–796, 1991.

[5] Carlos I. Chesñevar, A. Mguitman, and R. Loui. Logical models or argument. *Computing Surveys*, 32(4):336–383, 2000.

[6] Carlos I. Chesñevar and Guillermo R. Simari. Formalizing Defeasible Argumentation using Labelled Deductive Systems. *Journal of Computer Science & Technology*, 1(4):18–33, 2000.

[7] D. Leake and R. Sooriamurthi. Automatically selecting strategies for multi-case-base reasoning. In S. Craw and A. Preece, editors, *Advances in Case-Based Reasoning: Proceedings of the Fifth European Conference on Case-Based Reasoning*, pages 204–219, Berlin, 2002. Springer Verlag.

[8] Francisco Martín, Enric Plaza, and Josep Lluís Arcos. Knowledge and experience reuse through communications among competent (peer) agents. *International Journal of Software Engineering and Knowledge Engineering*, 9(3):319–341, 1999.

[9] Lorraine McGinty and Barry smyth. Collaborative case-based reasoning: Applications in personalized route planning. In I. Watson and Q. Yang, editors, *ICCBR*, number 2080 in LNAI, pages 362–376. Springer-Verlag, 2001.

[10] Santi Ontañón and Enric Plaza. Justification-based multiagent learning. In *Int. Conf. Machine Learning (ICML 2003)*, pages 576–583. Morgan Kaufmann, 2003.

[11] Santi Ontañón and Enric Plaza. Justification-based case retention. In *European Conference on Case Based Reasoning (ECCBR 2004)*, number 3155 in LNAI, pages 346–360. Springer-Verlag, 2004.

[12] Santi Ontañón and Enric Plaza. Case-based learning from proactive communication. In *IJCAI-2007*, pages 999–1004, 2007.

[13] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hydrid neural networks. In *Artificial Neural Networks for Speech and Vision*. Chapman-Hall, 1993.

[14] Enric Plaza, Eva Armengol, and Santiago Ontañón. The explanatory power of symbolic similarity in case-based reasoning. *Artificial Intelligence Review*, 24(2):145–161, 2005.

[15] Enric Plaza and Santiago Ontañón. Ensemble case-based reasoning: Collaboration policies for multiagent cooperative cbr. In I. Watson and Q. Yang, editors, *In Case-Based Reasoning Research and Development: ICCBR-2001*, number 2080 in LNAI, pages 437–451. Springer-Verlag, 2001.

[16] David Poole. On the comparison of theories: Preferring the most specific explanation. In *IJCAI-85*, pages 144–147, 1985.

[17] M V Nagendra Prassad, Victor R Lesser, and Susan Lander. Retrieval and reasoning in distributed case bases. Technical report, UMass Computer Science Department, 1995.

[18] N. R. Jennings S. Parsons, C. Sierra. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8:261–292, 1998.

[19] Bruce A. Wooley. Explanation component of software systems. *ACM CrossRoads*, 1998.