On the Benefits of Argumentation-Derived Evidence in Learning Policies

Chukwuemeka David Emele[1], Timothy J. Norman[1], Frank Guerin[1], and Simon Parsons[2]

[1] University of Aberdeen, Aberdeen, AB24 3UE, UK
[2] Brooklyn College, City University of New York, 11210 NY, USA
{c.emele,t.j.norman,f.guerin}@abdn.ac.uk
parsons@sci.brooklyn.cuny.edu

Abstract. An important and non-trivial factor for effectively developing and resourcing plans in a collaborative context is an understanding of the policy and resource availability constraints under which others operate. We present an efficient approach for identifying, learning and modeling the policies of others during collaborative problem solving activities. The mechanisms presented in this paper will enable agents to build more effective argumentation strategies by keeping track of who might have, and be willing to provide the resources required for the enactment of a plan. We argue that agents can improve their argumentation strategies by building accurate models of others' policies regarding resource use, information provision, etc. In a set of experiments, we demonstrate the utility of this novel combination of techniques through empirical evaluation, in which we demonstrate that more accurate models of others' policies (or norms) can be developed more rapidly using various forms of evidence from argumentation-based dialogue.

Keywords: Argumentation, Machine learning, Policies, Norms, Evidence

1 Introduction

Distributed problem solving activities often require the formation of a team of collaborating agents. In such scenarios agents often operate under constraints placed on them by the organisations or interests that they represent. Such constraints, typically, determine the behaviour of representatives of organisations. When these constraints are part of the standard operating procedures of the agents or the organisations in question, we refer to them as *policies* (also known as norms). Members of the team agree to collaborate and perform joint activities in a mutually acceptable fashion. Often, agents in the team represent different organisations, and so there are different organisational constraints imposed on them. Even within a single organisation, team members often represent sub-organisations with different procedures and constraints. For example, the sales department of an organisation may possess certain operating procedures, which differ from those of the purchasing department. Furthermore, team members

may possess individual interests and goals that they seek to satisfy, which are not necessarily shared with other members of the team. These individual motivations largely determine the way in which members carry-out tasks assigned to them during joint activities.

In this paper, we focus on policy and resource availability constraints, and define policies as explicit prohibitions that members of the team are required to adhere to. Policy constraints may be team-wide or individual. We focus on individual policies. These policies are often private to that individual member or subset of the team, and are not necessarily shared with other members of the team. In order to develop effective plans, an understanding of the policy and resource availability constraints of other members in the team is beneficial. However, tracking and reasoning about such information is non-trivial.

Our conjecture is that machine learning techniques may be employed to aid decision making in this regard. Although this is not a new claim [7], it is novel to combine it with evidence derived from argumentation-based dialogue, which we call argumentation-derived evidence (ADE). We present a system where agents learn from dialogue by automatically extracting useful information (*evidence*) from the dialogue and using these to model the policies of others in order to adapt their behaviour in the future. We describe an experimental framework and present results of our evaluation in a resource provisioning scenario [4], which show empirically: (1) that evidence derived from argumentation-based dialogue can indeed be effectively exploited to learn better (more complete and correct) models of the policy constraints that other agents operate within; and (2) that through the use of appropriate machine learning techniques more accurate and stable models of others' policies can be derived more rapidly than with simple memorisation of past experiences.

To illustrate the sorts of interaction between agents in this framework, consider the following examples. Let i and j be two agents collaborating to hang a picture [11].

Example 1:	**Example 2:**
i: Can I have a *screw-driver*?	i: Can I have a *screw-driver*?
j: What do you want to use it for?	j: What do you want to use it for?
i: To hang a picture.	i: To hang a picture.
j: No.	j: I can provide you with a *hammer* instead.
	i: I accept a *hammer*.

Fig. 1. Dialogue between Collaborating Agents (Examples 1 and 2)

Following from the interaction in example 1 (see Figure 1), there is very little that we can learn from that encounter. It is unclear why agent j said no to agent i's request. It could be that there exists some policy that forbids agent j from providing the *screw-driver* to agent i, or it could be that the *screw-driver* is not available at the moment. On the other hand, suppose we have an argumentation framework that allows agents to suggest alternatives as in example 2 (see Figure

1) or ask for and receive explanations as in examples 3 and 4 (see Figure 2), then agent i can, potentially, gather more evidence regarding the provision of the resources involved.

Example 3:	**Example 4:**
i: Can I have a *screw-driver*?	i: Can I have a *screw-driver*?
j: What do you want to use it for?	j: What do you want to use it for?
i: To hang a picture.	i: To hang a picture.
j: No.	j: No.
i: Why?	i: Why?
j: I'm not permitted to release *screw-driver*.	j: *Screw-driver* is not available.

Fig. 2. Dialogue between Collaborating Agents (Examples 3 and 4)

Considering examples 3 and 4 (see Figure 2), it is worth noting that without the additional evidence, obtained by the information-seeking dialogue, the two cases are indistinguishable. This means that the agent will effectively be guessing which class these cases fall into. The additional evidence allows the agent to learn the right classification for each of the cases. It should be noted here that although in example 3, we now have an explanation that the resource is not to be provided for policy reasons, the question remains: what are the important characteristics of the prevailing circumstances that characterise this policy? Additional evidence is beneficial, and could be used to identify the important features that characterise an agent's prevailing circumstance. Each piece of evidence can be used to improve the model, hence, the quality of decisions made in future episodes. Section 3 discusses how we do this.

In a domain where there are underlying constraints that could yield similar results, standard machine learning techniques will have limited efficacy. Using argumentation to gather additional evidence could improve the accuracy of the information learned about the policies of others. We claim that significant improvements can be achieved because argumentation can help clarify reasons behind decisions made by others (e.g. the *provider*).

In the research presented in this paper, we intend to validate the following hypotheses: (1) Allowing agents to exchange arguments during practical dialogue (like negotiation) will mean that the proportion of correct policies learned during interaction will increase faster than when there is no exchange of arguments; and (2) Through the use of appropriate machine learning techniques more accurate and stable models of others' policies can be derived more rapidly than with simple memorisation of past experiences.

The remainder of this paper is organised as follows: In section 2 we briefly describe argumentation-based dialogue and introduce the protocol employed. Learning policies is discussed in section 3 and section 4 describes our simulation environment. Experimental results are reported in section 5. Section 6 discusses related work and future direction, and the paper concludes in section 7.

2 Argumentation-based Dialogue

In this section we present the argumentation-based negotiation protocol which will be used in guiding the negotiation process, and for obtaining additional evidence from the interaction. This protocol uses information-seeking dialogue [13, 18] to probe for additional evidence.

2.1 The Negotiation Protocol

The negotiation for resources takes place in a turn-taking fashion, where the *seeker* agent sends a request for resource to a *provider* agent. Figure 3 captures the negotiation protocol in a AUML-like interaction diagram (*www.fipa.org*). If the *provider* agent has the requested resource in its resource pool and it is in a usable state then it checks whether there is any policy constraint that forbids it from providing the resource to the *seeker* or not. If the *provider* agent needs more information from the *seeker* in order to make a decision, the *provider* agent would ask for more information to be provided. This is the information gathering stage. The information gathering cycle will continue until the *provider* has acquired enough information (necessary to make the decision), or the *seeker* refuses to provide more information and the negotiation ends.

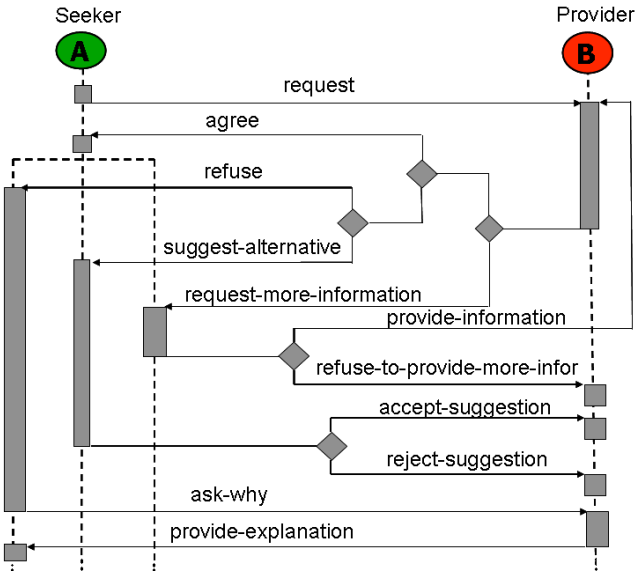

Fig. 3. The negotiation protocol.

The *provider* agent releases the resource to the *seeker* agent if there is no policy that prohibits the *provider* agent from doing so. Otherwise, the *provider*

agent offers an alternative resource (if there are no policies that forbid that line of action and the alternative resource is available). When an alternative resource is suggested by the *provider* agent, the *seeker* agent evaluates it. If it is acceptable, the *seeker* agent accepts it and the negotiation ends. Otherwise, the *seeker* agent refuses the alternative (in principle, this cycle may be repeated until an alternative is accepted or the negotiation ends). However, for simplicity and brevity, only one suggest-refuse cycle is permitted per request.

From a learning point of view, the suggestion of alternative resources could be a positive evidence that the *provider* agent does not have any policy that forbids the provision of the alternative resource to the *seeker*. In addition, it provides an evidence that the alternative resource is also available. This extra evidence, we anticipate, may help to improve the performance of the learner in predicting the policy constraints of *provider* agents in future encounters.

If there is a policy constraint that forbids the provision of the resource, or the resource is not available then the *provider* agent will refuse to provide the resource to the *seeker* agent. From the *seeker*'s perspective, the refusal could be as a result of policy constraint or because the resource is not available. In order to disambiguate which of these constraints are responsible for the refusal, the *seeker* agent switches to argumentation-based dialogue. The *seeker* agent asks for explanations for the refusal so as to gather further evidence and thereby identify the underlying constraints. The *provider* agent, therefore, responds with some explanations and the negotiation ends. Three categories of explanations are investigated in this framework: (1) Policy constraints (whereby the agent attributes the refusal to policy constraints); (2) Resource not available (here, the agent attributes the refusal to resource availability constraints); (3) Won't tell you (in this case, the agent refuses to give any explanation for the refusal). These pieces of evidence will be explored further in Section 2.3.

2.2 Policies

In our model, policies are conditional entities (or rules) and so are relevant to an agent under specific circumstances only. These circumstances are characterised by a set of features. Some examples of features may include: (1) the height of a tower; (2) the temperature of a room; (3) the manufacturer of a car, etc. In other words, policies map feature vectors, $\bar{F}$, of agents to appropriate policy decisions.

$\mathbb{P}_1$**:** You are **permitted** to release a *ladder* to an agent
if the *ladder* is required for the purpose of *hanging a picture*.
$\mathbb{P}_2$**:** You are **prohibited** from releasing a *screw-driver* to an agent
if the *screw-driver* is to be used for *hanging a picture*.
$\mathbb{P}_3$**:** You are **permitted** to release a *hammer* to an agent.
$\mathbb{P}_4$**:** You are **permitted** to release a *nail* to an agent.
$\mathbb{P}_5$**:** You are **permitted** to release a *table* to an agent.

Fig. 4. Sample Agent Policies

In order to illustrate the way policies may be captured in this model, we present the following examples (see Figure 4). Let F be the set of all features that characterise an agent's prevailing circumstance such that $f_1, f_2, \ldots \in F$. Assuming, f_1 is resource, f_2 is purpose, f_3 is location, f_4 is the affiliation of the agent, and f_5 is the day the resource is required then an agent's policies may be captured as shown in Figure 4.

2.3 Argumentation-Derived Evidence

Following the argumentation-based negotiation protocol described earlier, the agents could ask for more information (with respect to a request or the response to a request), which indicates what constraints others may be operating within. For instance, let us assume that a *provider* agent has a policy that forbids it from providing a *screw-driver* to any *seeker* agent that intends to use it for *hanging a picture*. Then, whenever a *screw-driver* is requested the *provider* agent will probe for more information to ascertain that the purpose the *seeker* intends to use the *screw-driver* for is not *hanging a picture*. This extra evidence could be useful. Similarly, whenever a *seeker* agent's request is refused then the *seeker* agent will ask for explanations/justifications for the refusal. These additional evidence are beneficial, and we expect them to improve the quality of the models of other agents that can be inferred in future encounters.

Example A
i: **request**(i, j, *screw-driver*)
j: **ask-infor**(j, i, need(*screw-driver*, f_2, f_3, f_5))
i: **provide-infor**(i, j, need(*screw-driver*, $f_2 = x$, $f_3 = y$, $f_5 = z$))
j: **refuse**(j, i, *screw-driver*)
i: **why**(i, j, refuse(*screw-driver*))
j: **inform**(j, i, *screw-driver*, reason(policy-constraints))
Example B
i: **request**(i, j, *nail*)
j: **refuse**(j, i, *nail*)
i: **why**(i, j, refuse(*nail*))
j: **inform**(j, i, *nail*, reason(wont-tell-you))
i: **request**(i, j, *table*)
j: **agree**(j, i, *table*)

Fig. 5. Dialogue between agents i and j

Figure 5 shows two simple examples of the kind of dialogue that may occur between two collaborating agents, i and j. For the purpose of the example, we use **need**(R, f_2, f_3, f_5) to denote that the *seeker* agent intends to use the resource R for a purpose (captured by feature f_2) at a location (represented by f_3) on a given day (captured by f_5). Note that although this is presented as

a dialogue between two agents, in reality the initiator (agent i, the agent that wishes to resource its plan) may engage in multiple instances of this dialogue with other agents.

3 Learning Policies

In this section we discuss the machine learning techniques that we have explored for learning policies through argumentation-derived evidence. These techniques include decision tree learning (C4.5), instance-based learning (k-Nearest Neighbours, abbreviated as k-NN) and rule-based learning (Sequential Covering, abbreviated as SC).

Our approach seeks ways to combine argumentation analysis with already existing machine learning techniques with a view to improving the performance of agents at predicting the policy constraints of others. We anticipate that this could enable them to build more effective argumentation strategies. In other words, we argue that evidence derived from argumentation-based dialogue can indeed be effectively exploited to learn better (more complete and correct) models of the policy constraints that other agents operate within. Also, we claim that through the use of appropriate machine learning techniques more accurate and stable models of others' policies can be derived more rapidly than with simple memorisation of past experiences. In future encounters, the *seeker* agent attempts to predict the policies of *provider* agents based on the model it has built.

When an agent has a collection of experiences with other agents described by feature vectors, we can make use of existing machine learning techniques for learning associations between sets of discrete attributes (e.g. f_1, f_2, f_3, f_4, f_5) and policy decisions. The following sections discuss specifically, the three classes of machine learning algorithms[3] [8], namely: decision tree learning (using C4.5), instance-based learning (using k-nearest neighbours), and rule-based learning (using sequential covering).

We define a *learning interval*, ϕ, which determines the number of interactions[4] an agent must engage in before building (or re-building) its policy model[5]. Once an agent has had ϕ interactions, the policy learning process proceeds as follows. For each interaction, which involves collaborating with provider j for the provision of a resource, we add the example $(\bar{F}_j, grant)$ or $(\bar{F}_j, deny)$ to the training set, depending on the evidence obtained from the interaction. The model is then constructed. In this way, an agent may build a model of the relationship between observable features of agents and the policies they are operating under. Subsequently, when faced with resourcing a new task, the policy model can be used

[3] We use the Weka [19] implementation of these algorithms. Weka is a popular open-source machine learning toolkit written in Java

[4] By interaction, we mean, an entire plan resourcing episode.

[5] This requirement relates, specifically, to C4.5 and sequential covering because they are non-incremental learning algorithms. A detailed discussion of non-incremental learning is covered in [19].

to obtain a prediction of whether a particular provider has a policy that permits him to provide the resource or not. This satisfies our requirement for a policy learning mechanism.

3.1 Decision Tree Learning (C4.5)

C4.5 [14] builds decision trees from a set of training data, using the concept of information entropy [8] (beyond the scope of this paper). Generally, the training data is a set $S = s_1, s_2, ..., s_n$ of already classified samples. Each sample $s_i = x_1, x_2, ..., x_m$ is a vector where $x_1, x_2, ..., x_m$ represent attributes of the sample. The training data is augmented with a vector $C = c_1, c_2, ..., c_n$ where $c_1, c_2, ..., c_n$ represent the class to which each sample belongs.

Integrating this algorithm into our system with the intention of learning policies is appropriate since the algorithm supports concept learning and policies can be conceived as concepts/features of an agent. Agent policies are represented as a vector of attributes (e.g. resource, purpose, location, etc.) and these attributes are communicated back and forth during negotiation. The C4.5 algorithm is then used to classify each set of attributes (policy instance) into a class. There are two classes: grant and deny. Grant means that the *provider* agent will possibly provide the resource that is requested while deny implies that the *provider* agent will potentially refuse. The leaf nodes of a decision tree hold the class labels of the instances while the non-leaf nodes hold the test attributes. In order to classify a test instance, the C4.5 algorithm searches from the root node by examining the value of test attributes until a leaf node is reached and the label of that node becomes the class of the test instance. Figure 6 outlines the C4.5 algorithm in pseudo-code.

The C4.5 algorithm has three base cases.

- All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
- None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
- Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

Algorithm 1. The C4.5 algorithm

1: **Check** for base cases
2: **For** each attribute *D*,
   Find the normalised information gain from splitting on *D*
3: **Let** *D_best* be the attribute with the highest normalised information gain
4: **Create** a decision node that splits on *D_best*
5: **Recurse** on the sublists obtained by splitting on
   D_best, and add those nodes as children of the node

Fig. 6. The C4.5 algorithm.

The problem with this algorithm is that it is not incremental, which means all the training examples should exist before learning. To overcome this problem, the system keeps track of the *provider* agent's responses. After a number of interactions, predefined by ϕ, the decision tree is rebuilt. Without doubt, there is a computational drawback involved in periodically reconstructing the decision tree. However, in practice, we have evaluated C4.5 to be fast and the reconstruction cost to be small. Our approach is similar to the incremental induction of decision trees proposed in [17].

3.2 Instance-based Learning (*k*-NN)

The k-nearest neighbours algorithm (k-NN) [3] is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The universal set of all the policies an agent may be operating within could be conceived as a feature space (or a grid) and the various policy instances represent points on the grid. Using k-NN, a policy instance is classified by a majority vote of its neighbours, with the policy instance being assigned to the class most common amongst its k nearest neighbours, where k is a positive integer, typically small. The k-NN algorithm is incremental, which means all the training examples need not exist at the beginning of the learning process. This is a good feature because the policy model could be updated as new knowledge is learned.

The k-nearest neighbour algorithm is sensitive to the local structure of the data and this, interestingly, makes k-NN a good candidate for learning policies because slight changes in the variables/attributes of a policy could trigger different action. For example:

Policy1: *You are* ***permitted*** *to release resource R to team member X if his affiliation is O and R is to be deployed at location L for purpose P on day* 1.

Policy2: *You are* ***prohibited*** *from releasing resource R to team member X if his affiliation is O and R is to be deployed at location L for purpose P on day* 2.

In order to identify neighbours, the policy instances are represented by position vectors in a multidimensional feature space. In this approach, new policy instances are classified based on the closest training examples in the feature space. A policy instance is assigned to the class c if it is the most frequent class label among the k nearest training samples. It is usual to use the Euclidean distance, though other distance measures, such as the Manhattan distance, Hamming distance could in principle be used instead. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the actual classification phase, the test sample is represented as a vector in the feature space. Distances from the new vector to all stored vectors are computed and k closest samples are selected.

A major drawback to using this technique to classify a new vector to a class is that the classes with the more frequent examples tend to dominate the prediction of the new vector, as they tend to come up in the k nearest neighbours when

the neighbours are computed due to their large number. The distance-weighted k-NN algorithm, which weights the contribution of each of the k neighbours according to their distance to the new vector, uses distance weights to minimise the bias caused by the imbalance in the training examples by giving greater weight to closer neighbours. In our work, the weight of a neighbour is computed as the inverse of its distance from the new vector.

3.3 Rule-based Learning (Sequential Covering)

Since policies guide the way entities within a community (or domain) act by providing rules for their behaviour it makes sense to learn policies as rules. Sequential covering algorithm [2, 8] is a rule-based learning technique, which constructs rules by sequentially covering the examples. The sequential covering algorithm, SC for short, is a method that induces one rule at a time (by selecting attribute-value pairs that satisfy the rule), removes the data covered by the rule and then iterates the process. SC generates rules for each class by looking at the training data and adding rules that completely describe all tuples in that class. For each class value, rule antecedents are initially empty sets, augmented gradually for covering as many examples as possible. Figure 7 outlines the sequential covering algorithm in pseudo-code.

	Algorithm 2. The Sequential Covering Algorithm
1:	**Input** the training data (D) and the classes (C)
2:	**For** each class $c \in C$
3:	**Initialise** E to the instance set
4:	**Repeat**
5:	**Create** a rule R with an empty left-hand side (LHS) that predicts class c:
6:	**Repeat**
7:	**For** each $(Attribute, Value)$ pair found in E
8:	**Consider** adding the condition $Attribute = Value$ to the LHS of R
9:	**Find** $Attribute = Value$ that maximises p/t
10:	(break ties by choosing the condition with the largest p)
11:	**Add** $Attribute = Value$ to R
12:	**Until** R is perfect (or no more attributes to use)
13:	**Remove** the instances covered by R from E
14:	**Until** E contains no more instances that belong to c

Fig. 7. The Sequential Covering Algorithm.

In this study we used three different machine learning mechanisms: *Decision tree learning*, *Instance-based learning* and *Rule-based learning*. These three mechanisms represent very different classes of machine learning algorithms. The

rationale for exploring a range of learning techniques is to demonstrate the utility of argumentation-derived evidence regardless of the machine learning technique employed. Thus, we hypothesize that the use of evidence acquired through argumentation significantly improves the performance of machine learning in the development and refinement of models of other agents. Also, we claim that through the use of appropriate machine learning techniques more accurate and stable models of others' policies can be derived more rapidly than with simple memorisation of past experiences.

4 Simulation Environment

To test our hypotheses, we developed a simulation environment that combines mechanisms for agents to engage in argumentative dialogue and to learn from dialogical encounters with other agents. For the purpose of resourcing plans, agents may act as resource seekers, which collaborate and communicate with potential providers to perform joint actions. The enactment of both *seeker* and *provider* roles are governed by individual policies that regulate their actions. A *seeker* agent requires resources in order to carry out some assigned tasks. The *seeker* agent generates requests in accordance with its policies and negotiates with the *provider* agents based on these constraints. On the other hand, *provider* agents have access to certain resources and may have policies that govern the provision of such resources to other members of the team.

Although agents may have prior assumptions about the policies that constrain the activities of others, these models are often incomplete and may be inaccurate. *Provider* agents do not have an unlimited pool of resources and so some resources may be temporarily unavailable. By a resource being available we mean that it is not committed to another task (or agent) at the time requested and the resource is in a usable state. Both *seeker* and *provider* agents have access to the team-wide policies but not the individual policies of others. Agents in this domain play the role of a *seeker* or a *provider* in different interactions.

4.1 Architecture

Figure 8 depicts our architecture. Each agent has two main layers, the communication layer and the reasoning layer. The communication layer embodies the dialogue controller, which handles all communication with other agents in the domain. The dialogue controller sends/receives messages to/from other agents, and the reasoning layer reasons over the dialogue. If an agent is playing the role of a *seeker* agent then the dialogue controller sends out the request for resources. On the other hand, if the agent is a *provider* agent then the dialogue controller receives a request and passes it on to the reasoning layer.

The reasoning layer consists of two modules: the reasoner and the learner. Upon receiving a message (e.g. a request), the reasoner evaluates the message and determines the response of the agent. In most cases, the reasoner looks up policy constraints from the knowledge-base and generates the appropriate response

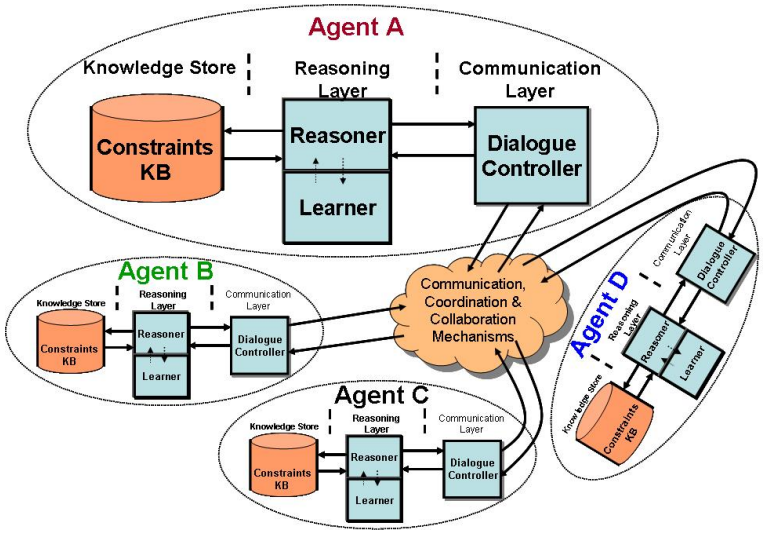

Fig. 8. Architecture of the framework for learning policies in team-based activities using dialogue.

for the agent. Policy and non-policy constraints are stored in the constraints knowledge-base. Whenever the agent observes a new pattern of behaviour the agent uses this experience as evidence for learning, and updates the model of the other agent accordingly. The learner uses standard machine learning techniques to learn policies based on the perceived actions of other agents. The learning techniques are discussed in Section 3.

The knowledge store in Figure 8 acts as a repository where an agent stores the constraints it has learned by interacting with other agents in the domain. The information includes the features that an agent requires in order to make a decision about providing a resource or not. For example, following from [11], a *provider* agent j may need to know what the purpose for requesting a *screw-driver* is before deciding whether to release the *screw-driver* or not. The *seeker* agent stores such information about agent j in the knowledge store. Also, the decision of j after the purpose has been revealed will also be learned for future interactions.

To achieve this, we have developed a simple dialogue game[6] involving *seeker* agents and *provider* agents operating under different constraints. The players take turns and the game starts with an agent, i, sending a request to another agent, j, for the use of some resources needed to fulfill a plan. The other agent (j) responds with an agree or refuse based on the prevailing context, e.g. policy constraints. The requesting agent could ask for explanations and reasons for an action, and so on until the game ends.

[6] Dialogue games have proven extremely useful for modeling various forms of reasoning in many domains [1].

4.2 Implementation

We implemented a simulation environment for agent support in team-based problem solving and integrated our learning and argumentation mechanisms into the framework. The policies are encoded as rules in a rule engine [6]. The application programming interface in Weka [19] was used to integrate standard machine learning algorithms into the framework. We note that, although these three learning algorithms were used, the framework is configured such that other machine learning algorithms can be plugged in. As discussed in the previous section, we evaluated the performance of a decision tree learner (C4.5), an Instance based learner (k-Nearest Neighbour algorithm) and a rule based learner (Sequential Covering) in learning policies through argumentation-derived evidence.

The simulation environment allows us to generate multiple *providers* with randomised policies, *seeker* agents with randomised initial models of the policies of *providers* in the simulation and randomised problems for the *seeker* to solve (that is, random resource requirements). The *seeker* predicts (based on the model of the *provider*) whether the *provider* has a policy that forbids/permits the provision of such resource in that context. The *seeker* requests the required resource from the *provider* agent and the *provider* uses a simple decision function (See Figure 9) to decide whether to grant or deny the request.

If the decision of the *provider* agent deviates from the predictions of the *seeker* agent then the *seeker* agent seeks additional evidence (through dialogue) to disambiguate whether the deviation was as a result of policy or resource availability constraints. The dialogue follows the protocol specified in Figure 3, and at the end of the interaction the outcome is learned by the *seeker* and the model of the *provider* is updated accordingly. This adaptive learning process serves to improve the quality of the models of the other agents that can be inferred from their observable actions in future interactions.

Assume *seeker* i **requests** resource R **from** *provider* j
IF (is_available(R) $\wedge$ NOT ($forbid$(release(R, i)))
THEN
agree(release(R, i))
ELSE
refuse(release(R, i))

Fig. 9. *Provider* agents' pseudo decision function

5 Experiments and Results

In a series of experiments, we show how learning techniques and argumentation can support agents engaging in collaborative activities, increase their predictive accuracy, avoid unnecessary policy conflicts, hence improve their performance.

The experiments show that agents can effectively and rapidly increase their predictive accuracy of the learned model through the use of dialogue.

The scenario adopted in this research involves a team of five software agents (one *seeker* and four *provider* agents) collaborating to complete a joint activity in a region over a period of three days. The region is divided into five locations. There are five resource types, and five purposes that a resource could be used to fulfill. A task involves the *seeker* agent identifying resource needs for a plan and collaborating with the *provider* agents to see how that plan can be resourced.

Argumentation-derived evidence (ADE) was incorporated into the learning process of the three machine learning techniques (C4.5, k-NN, and SC) described earlier, and their performances in learning the policy constraints of others were evaluated. A simple lookup table (hereafter called, LT) was used as a control condition and it serves as a structure for simple memorisation of outcomes from past encounters.

5.1 Results

Table 1. Average percentage of policies classified correctly and standard deviation

Approach \ Tasks	1000	2000	3000	4000	5000	6000
LT-ADE	65.1±6.5	70.3±10.3	75.6±6.7	78.1±10.2	79.3±8.3	81.3±10.1
LT+ADE	66.3±6.0	79.3±9.3	83.6±8.2	81.7±11.2	81.4±7.8	84.7±9.1
C4.5-ADE	58.3±15.1	69.2±16.6	75.1±12.0	82.1±12.3	85.3±8.9	88.2±8.2
C4.5+ADE	60.3±14.4	75.0±12.6	83.6±6.5	89.9±5.2	93.0±3.4	95.6±5.1
k-NN-ADE	65.2±9.8	71.0±7.8	75.3±5.3	80.7±3.8	81.0±4.1	82.0±3.8
k-NN+ADE	71.1±9.0	85.9±7.3	92.0±4.6	96.8±3.1	97.3±3.6	98.4±1.7
SC-ADE	66.7±8.2	71.7±6.0	78.7±8.4	84.3±6.5	87.4±6.0	90.6±5.3
SC+ADE	67.7±7.7	87.1±6.4	94.1±4.2	96.6±4.1	97.5±2.6	99.2±1.0

This section presents the results of the experiments carried out to validate this work. Experiments were conducted with *seeker* agents initialised with random models of the policies of *provider* agents. 100 runs were conducted for each case, and tasks were randomly created during each run from 375 possible configurations.

Table 1 illustrates the effectiveness of identifying and learning policies through argumentation-derived evidence using the three machine learning techniques described earlier, and the control condition (lookup table). It shows the average percentage of policies classified correctly and the standard deviations for each of the approaches, namely: Lookup Table without the aid of argumentation-derived evidence (LT-ADE), Lookup Table enhanced with argumentation-derived evidence (LT+ADE), C4.5-ADE, C4.5+ADE, k-NN-ADE, k-NN+ADE, SC-ADE, and SC+ADE. In each case, the model of others' policies is recomputed after each set of 1000 tasks. For all three machine learning techniques considered, the

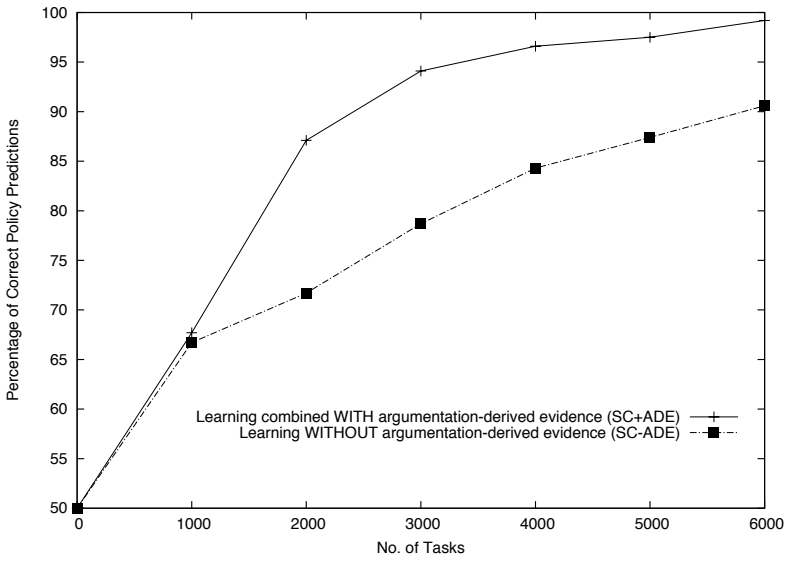

Fig. 10. Graph showing the effectiveness of allowing the exchange of arguments in learning policies.

percentage of policies predicted correctly as a result of exploiting evidence derived from argumentation was consistently and significantly higher than those predicted without such evidence. Figure 10 gives a graphical illustration of the effectiveness of learning policies with the aid of argumentation-derived evidence using rule-based learning technique, for instance. After 3000 tasks, the accuracy of the approach with additional evidence had risen above 94% while the configuration without additional evidence was approaching 79%. It is easy to see that the experiments where additional evidence was combined with machine learning significantly and consistently outperformed those without additional evidence. These results show that the exchange of arguments during practical dialogue enabled agents to learn and build more accurate models of others' policies much faster than scenarios where there was no exchange of arguments.

Figure 11 captures the effectiveness of the three machine learning techniques described earlier, and a simple memorisation technique (a lookup table) in learning policies. The result shows that both instance-based learning (k-NN+ADE) and rule-based learning (SC+ADE) constantly and consistently outperform the control condition (LT+ADE) throughout the experiment. It is interesting to see that, with relatively small training set, the control condition performed better than the decision tree learner (C4.5+ADE). This is, we believe, because the model built by the decision tree learner overfit the data. The tree was pruned after each set of 1000 tasks and after 3000 tasks the accuracy of the C4.5+ADE model rose to about 83% to tie with the control condition and from then the

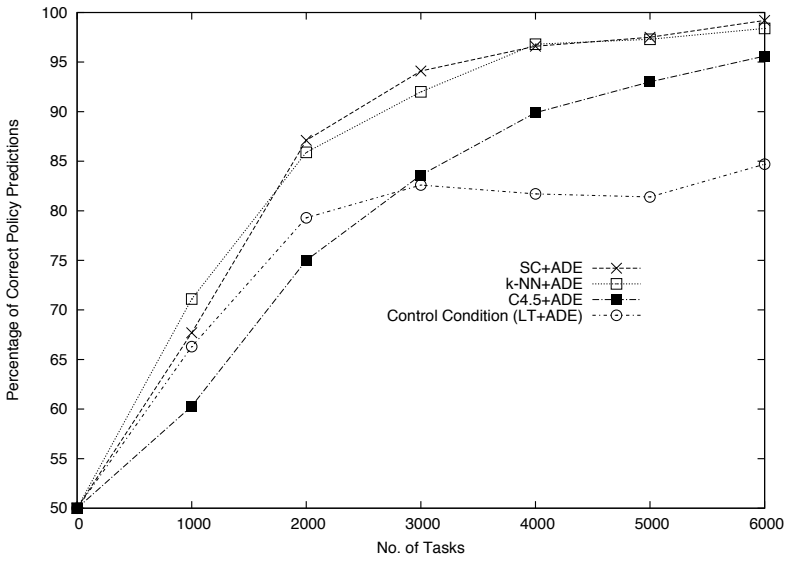

Fig. 11. Graph showing the effectiveness of learning policies with the aid of argumentation-derived evidence using various techniques (LT+ADE, C4.5+ADE, k-NN+ADE & SC+ADE).

decision tree learner performed better than the control condition. The performance of the control condition dropped to about 81% after 4000 tasks. After 6000 tasks the accuracy of the decision tree learner had risen above 95% while that of the control condition was just over 84%.

Tests of statistical significance were applied to the results. The standard deviations of the results were analysed and the trend line plotted. (See Figure 12). Using linear regression, the analysis of variance (ANOVA) shows that as the number of tasks increases, each of the three machine learning techniques (with or without argumentation-derived evidence) consistently converges with a 95% confidence interval. Furthermore, for all the pairwise comparisons, the scenarios where argumentation-derived evidence was combined with machine learning techniques consistently yielded higher rates of convergence ($p < 0.02$) than those without additional evidence. Specifically, the decision tree learner enhanced with argumentation-derived evidence (C4.5+ADE) converges ($y = 15.3944 - 0.0022x$) with a F value of 15.66 and significance $p = 0.0167$. The k-NN+ADE converges ($y = 9.7983 - 0.0014x$) with a F value of 38.58 and significance $p = 0.0034$, and the SC+ADE ($y = 8.819 - 0.0013x$) converges with a F value of 136.45 and significance $p = 0.0003$. On the other hand, with a significance $p = 0.3957$, there is no statistical significance as to whether LT+ADE converges or not. These results confirm our hypotheses.

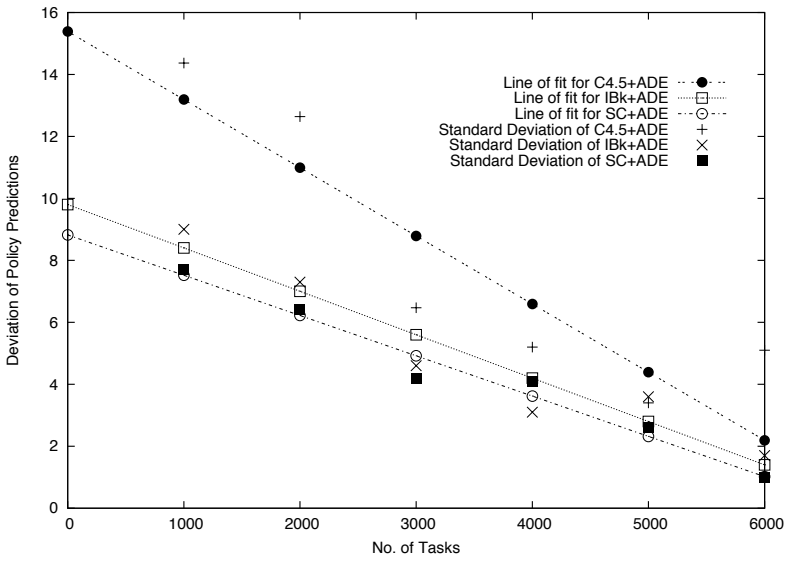

Fig. 12. Graph showing the rate of convergence of the three techniques enhanced with ADE in learning policies (C4.5+ADE, k-NN+ADE, & SC+ADE).

6 Discussion and Related Work

The research presented in this paper represents the first model for using evidence derived from argumentation to learn underlying social characteristics (e.g. policies/norms) of others. There is, however, some prior research in combining machine learning and argumentation, and in using argument structures for machine learning. In that research, Možina et al. [9] propose a novel induction-based machine learning mechanism using argumentation. The work implemented an argument-based extension of CN2 rule learning (ABCN2) and showed that ABCN2 out-performed CN2 in most tasks. However, the framework developed in that research will struggle to disambiguate between constraints that may produce similar outcome/effect, which is the main issue we are addressing in our work. Also, the authors assume that the agent knows and has access to the arguments required to improve the prediction accuracy, but we argue that it is not always the case. As a result, we employ information-seeking dialogue to tease out evidence that could be used to improve performance.

In related research, Rovatsos et al. [15] use hierarchical reinforcement learning in modifying symbolic constructs (*interaction frames*) that regulate agent *conversation patterns*, and argue that their approach could improve an agent's conversation strategy. In our work, we used information-seeking dialogue to obtain evidence from the interaction and learned the entire sequence as against a segment (frame) of the interaction [15]. We have demonstrated the effectiveness of using argumentation-derived evidence to learn underlying social characteris-

tics (e.g. policies) without assuming that those underlying features are public knowledge.

In recent research, Sycara et al. [16] investigate agent support for human teams in which software agents aid the decision making of team members during collaborative planning. One area of support that was identified as important in this context is guidance in making policy-compliant decisions. This prior research focuses on giving guidance to humans regarding their own policies. An important and open question, however, is how can agents support human decision makers in developing models of others' policies and using these in guiding the decision maker? Our work is aimed at bridging this gap (a preliminary version was presented in [5]). We employ a novel combination of techniques in identifying, learning and building accurate models of others' policies, with a view to exploiting these in supporting human decision making.

In our future work, we plan to develop strategies for advising human decision makers on how a plan may be resourced and who to talk to on the basis of policy and resource availability constraints learned [10]. Parsons et al. [12] investigated the properties of argumentation-based dialogues and examined how different classes of protocols can have different outcomes. Furthermore, we plan to explore ideas from this work to see which class of protocol will yield the "best" result in this kind of task. We are hoping that some of these ideas will drive the work on developing strategies for choosing who to talk to.

7 Conclusions

In this paper, we have presented a technique that combines machine learning and argumentation for learning policies in a team of collaborating agents engaging in joint activities. We believe, to the best of our knowledge, that this is the first study into learning models of other agents using argumentation-derived evidence. The results of our empirical investigations show that evidence derived from argumentation can have a statistically significant positive impact on identifying, learning and modeling others' policies during collaborative activities. The results also demonstrate that through the use of appropriate machine learning techniques more accurate and stable models of others' policies can be derived more rapidly than with simple memorisation of past experiences. Accurate policy models can inform strategies for advising human decision makers on how a plan may be resourced and who to talk to [16], and may aid in the development of more effective strategies for agents [10]. Our results demonstrate that significant improvements can be achieved by combining machine learning techniques with argumentation-derived evidence. Having shown that accurate models of others' policies could be learned through argumentation-derived evidence, we conjecture that one could, in principle, learn accurate models of other agents' properties (e.g. priorities, preferences, and so on).

Acknowledgements This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under

Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. Bench-Capon, T.J.M., Freeman, J.B., Hohmann, H., Prakken, H.: Computational models, argumentation theories and legal practice. In: Reed, C., Norman, T.J. (eds.) Argumentation Machines. New Frontiers in Argument and Computation. pp. 85–120. Kluwer Academic Publishers, Dordrecht, The Netherlands (2003)
2. Cendrowska, J.: Prism: An algorithm for inducing modular rules. International Journal of Man-Machine Studies 27(4), 349–370 (1987)
3. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transaction on Information Theory 13(1), 21–27 (1967)
4. Emele, C.D., Norman, T.J., Guerin, F., Parsons, S.: Learning policy constraints through dialogue. In: Proc. of the AAAI Fall Symposium on The Uses of Computational Argumentation. pp. 20–26. USA (2009)
5. Emele, C.D., Norman, T.J., Guerin, F., Parsons, S.: Learning policies through argumentation-derived evidence (extended abstract). In: van der Hoek, Lesprance, Kaminka, Luck, Sen (eds.) Proc. of 9th Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010). Toronto, Canada (2010)
6. Friedman-Hill, E.: Jess in Action. Manning (2003)
7. Kelemen, A., Liang, Y., Franklin, S.: A comparative study of different machine learning approaches for decision making. In: Mastorakis, N.E. (ed.) Recent Advances in Simulation, Computational Methods and Soft Computing. pp. 181–186. WSEAS Press, Piraeus, Greece (2002)
8. Mitchell, T.M.: Machine Learning. McGraw Hill (1997)
9. Možina, M., Žabkar, J., Bratko, I.: Argument based machine learning. Artificial Intelligence 171(10-15), 922–937 (2007)
10. Oren, N., Norman, T.J., Preece, A.: Loose lips sink ships: A heuristic for argumentation. In: Proc. of the 3rd Int'l Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2006). pp. 121–134 (2006)
11. Parsons, S., Jennings, N.R.: Negotiation through argumentation-A preliminary report. In: Proc. of the 2nd Int'l Conference Multi-Agent Systems (ICMAS'96). pp. 267–274. Kyoto, Japan (1996)
12. Parsons, S., Wooldridge, M., Amgoud, L.: Properties and complexities of some formal inter-agent dialogues. Journal of Logic and Comp. 13(3), 347–376 (2003)
13. Perrussel, L., Doutre, S., Thévenin, J.M., McBurney, P.: A persuasion dialog for gaining access to information. In: Rahwan, I., Parsons, S., Reed, C. (eds.) Argumentation in Multi-Agent Systems, Lecture Notes in Computer Science, vol. 4946, pp. 63–79. Springer Berlin / Heidelberg (2008)
14. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)

15. Rovatsos, M., Rahwan, I., Fischer, F., Weiss, G.: Practical strategic reasoning and adaptation in rational argument-based negotiation. In: Argumentation in Multi-Agent Systems. LNCS, vol. 4049, pp. 122–137. Springer-Berlin (2005)
16. Sycara, K., Norman, T.J., Giampapa, J.A., Kollingbaum, M.J., Burnett, C., Masato, D., McCallum, M., Strub, M.H.: Agent support for policy-driven collaborative mission planning. The Computer Journal 53(1), 528–540 (2009)
17. Utgoff, P.E.: Incremental induction of decision trees. Machine Learning 4(2), 161–186 (1989)
18. Walton, D.N., Krabbe, E.C.W.: Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning. SUNY Press, USA (1995)
19. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco, 2nd edn. (2005)