

Agent Learning in the Multi-Agent Contracting officer's technical representative System (MACS)

Bonnie Rubenstein-Montano^{a†}, Stuart Lowry^b, Francisco Cantu^c, Kevin Drummey^d, Victoria Yoon^a, Teresa Wilson^a, Jay Liebowitz^a

^aDepartment of Information Systems, UMBC, USA

^bScience Applications International Corporation, Annapolis, MD, USA

^cCenter for Artificial Intelligence, Monterrey Institute of Technology, Mexico

^dNational Security Agency, USA

[†] This work has been supported by External Acquisition Research Program (EARP '00).

Agent Learning in the Multi-Agent Contracting officer's technical representative System (MACS)

Abstract. This paper presents a Bayesian learning approach for a Web-based multiagent system. Of particular interest is the application area for which the system was built — defense contracting. In U.S. defense research contracting, scientists expend significant effort to complete administrative details for contract acquisition, and often rely on the Defense Acquisition Deskbook for assistance. However, the current system requires human experts to respond to queries by scientists. Preliminary work on automating the process has been completed. The research presented in this paper builds on the past, preliminary, work by developing a multiagent system, termed Multi-Agent Contracting officer's technical representative System (MACS). MACS is an intelligent multiagent system with the ability to learn from and adapt to its environment via Bayesian learning. Efficacy of MACS has been determined by analyzing the accuracy and degree of learning in the system. This was accomplished by testing the system against historical data.

1. Introduction

Intelligent agents are entities that possess some degree of autonomy and can act on behalf of a user [2]. They can be characterized by three attributes: agency, mobility, and intelligence [10]. Agency is the degree of autonomy an agent possesses. It is measured by the ability of an agent to act on behalf of a user. Mobility measures an agent's ability to move through a distributed network. Intelligence measures an agent's ability to reason and

learn. It refers to the level of reasoning and learned behavior exhibited by an agent.

Intelligence can be as simple as stating preferences or as complex as learning from and adapting to the external environment [3]. This paper posits that higher levels of intelligence exist when agents can adapt to their environment [7]. Such adaptation involves learning about the user's objectives and resources available to the agent in its environment.

Learning can also be thought of as belief revision [1] [8] [18]. This is the actual mechanism by which adaptation, and thus learning, occurs. For example, if agent A believes that agent B can provide information on contract justification, but in fact agent B cannot, then agent A must revise this belief so that next time it looks for information on contract information it will know not to query agent B.

Emphasis on the intelligence attribute of agents is important because the intelligence, and learning capability, of an agent influences its performance. Gaines [9] and Pendharkar [24] allude to this notion of linking learning and performance. Systems built upon a knowledge base, as are agent systems, tend to degrade significantly as the limits of knowledge are reached [21]. That is, intelligent agent performance can be sensitive to initial knowledge distribution among agents in a multi-agent system [17]. Holland [14] and Smith [29] term this brittleness, and it is highly relevant for intelligent agent systems where agents are assumed to be somewhat autonomous as a result of the knowledge they possess. Since it is not realistic to encode complete knowledge into the intelligent agent system *a priori*, systems must be able to adapt to changing environmental conditions and apply knowledge gained from previous experiences to maintain, and even improve, performance levels [9] [21] [24]. Thus, systems must be able to learn.

Traditional machine learning has developed a wide variety of algorithms for providing single-agent systems with learning capacity [19]. Among the main classes of algorithms for traditional machine learning are induction of trees and rules, learning in neural nets, system classifiers and genetic algorithms, reinforcement learning, Bayesian learning, case-based learning, logic-based learning, and some others. However, these algorithms do not apply directly when used in MAS. Learning in multiagent systems (MAS) opens new challenges and opportunities for researchers.

Agent work on learning has been conducted by such researchers as Ayala and Yano [1], who use agents in the area of computer supported collaborative learning, Vaario and Ueda [30] look at modular learning in multi-agent environments, and Norrie and Gaines [20] who conceptualize learning on the Web through agents. There are also several researchers who have looked specifically at Bayesian learning in MAS.

The objective of this paper is to present a Bayesian learning approach for a Web-based MAS. The application area employed to illustrate agent learning is defense research contracting. In U.S. defense research contracting, scientists expend significant effort to complete administrative details for contract acquisition, and often rely on the Defense Acquisition Deskbook for assistance. However, the current system requires human experts to respond to queries by scientists. Preliminary work on automating the process has been completed by Liebowitz et al. [16], who built a multiagent system to respond to scientist queries. This work builds on past work by extending the system to a multiagent system that can learn from and adapt to its environment via Bayesian learning. The system, termed **Multi-Agent Contracting officer's technical representative System (MACS)**, can be accessed at <http://kmlab-01.ifsm.umbc.edu/macs/>.

Several learning techniques were explored for possible integration with MACS, and Bayesian learning proved to be the most appropriate. A primary distinction from past work on Bayesian learning in MAS is that a negotiation problem is not used to illustrate learning. Instead, we use a cooperative system that does not involve negotiation between agents, MACS, to illustrate our research findings. Efficacy of MACS is determined by analyzing the accuracy of learning in the system.

The next section describes the MACS system. Then Section 3 provides an overview of Bayesian learning and how it is implemented in MACS. Section 4 analyzes the effectiveness of Bayesian learning in MACS, and Section 5 draws conclusions from the analysis and suggests future research directions.

2. MACS

2.1 Description of MACS

The MACS system is a MAS developed for procurement and acquisition of defense contracts. Specifically, it is designed to assist Acquisition Request Originators (AROs) and Contracting Officer's Technical Representatives (COTRs) with the pre-award phase of contracting and procurement [16]. The system architecture consists of nine agents — a User agent, a Facilitator agent, a Natural Language agent, a Machine Learning agent, and five specialty agents. The specialty agents are encoded with domain knowledge about the five general areas of expertise required of AROs/COTRs, and the user agent interfaces with AROs/COTRs. Interaction between AROs/COTRs and the system occurs through either keyword searches or natural language queries. As shown in Figure 1, the MACS

architecture implements a typical three-tiered brokered architecture. The Facilitator agent coordinates agent activities and communicates with the agent(s) capable of responding to an incoming query.

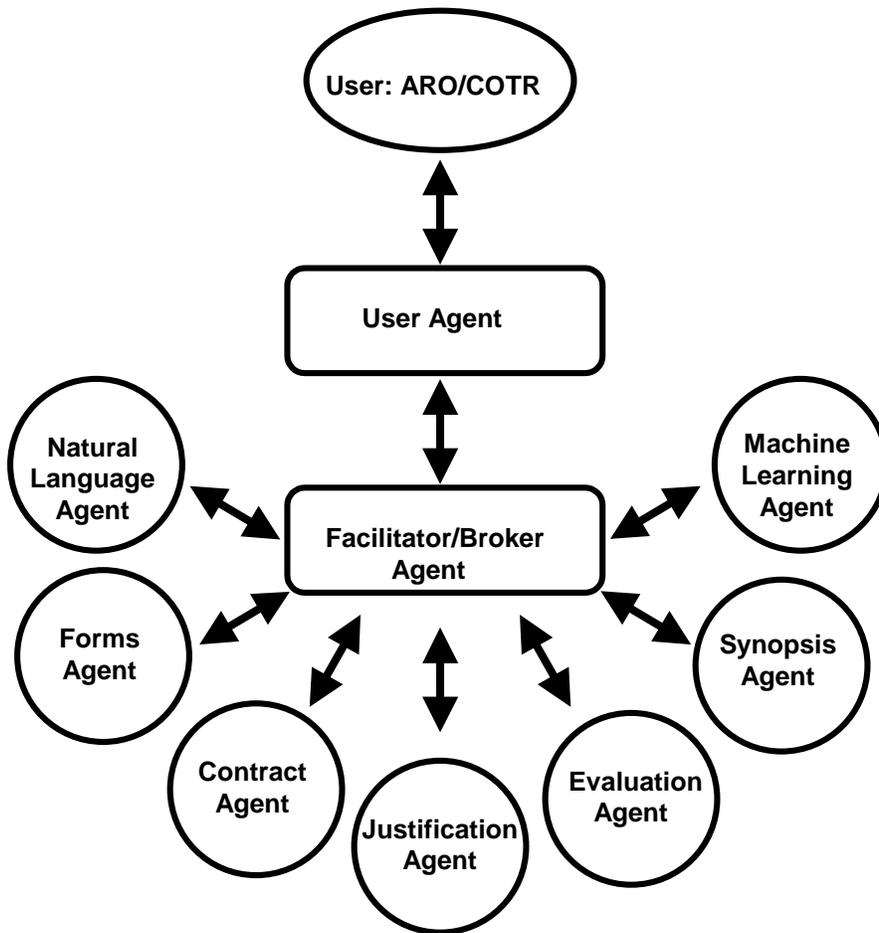


Figure 1. Agent architecture and communication channels

The user agent interacts with the user/ARO/COTR to welcome the user, ask what pre-award questions the user has, and serve as the interface between the user/ARO/COTR and the other agents in the system [16]. Two business logic threads have been designed into

the User agent. One thread supports the Natural Language capability of the system, and the other supports the keyword search capability of the system. The User agent sends incoming user queries to the Facilitator agent, which is responsible for communicating with all of the other agents in the MACS system as illustrated in Figure 1.

Queries submitted by users are forwarded, by the Facilitator agent, to the Machine Learning (ML) agent, the ML agent implements Bayesian learning and creates an action plan, and that plan is then issued back to the Facilitator agent for completion. In MACS, the action plan is essentially a determination of which specialty agent(s) should be contacted to respond to an incoming query. The facilitator completes the action plan by performing the necessary low-level communication between the specialty agents. These communications lead to solutions being sent from a specialty agent (or agents) to the Facilitator agent and then from there to the User agent. As solutions to a query are collected, the ML agent updates its internal tables, making note of which agents responded to which user queries. This information is used to calculate the response plan for similar queries in the future.

The five specialty agents in the system relate to the pre-award phase of a contract and include the *Forms*, *Justification*, *Evaluation*, *Synopsis*, and *Type of Contract* agents. The Forms agent identifies the forms needed to complete a procurement request package. The Justification agent indicates situations where a justification and approval is required to complete a procurement request. The Evaluation agent provides guidelines for evaluating proposals. The Synopsis agent identifies the type of synopsis for a given procurement request. Lastly, the Type of Contracts agent identifies the type and nature of a contract based on conditions such as the source of contract, the nature of the work, etc.

Each agent in MACS contains a rule base and has explicit goals. Its rule-base describes how to achieve the goals under varying circumstances. The specialty agents respond to incoming queries by presenting necessary information and/or requirements for AROs/COTRs. For example, the Evaluation Agent can assist an ARO/COTR with information regarding how to evaluate a project and what criteria or weights to use for evaluation of a contract. If an ARO/COTR has a question regarding "determining weights on evaluation criteria," the Evaluation agent will reply with "You can develop your own weights on technical, qualifications, and cost criteria. Generally speaking, a weight of 40 percent (out of 100%) is given to cost." [16].

The knowledge contained within each specialty agent is independent of the knowledge contained within the other specialty agents. Thus, coordination between the specialty agents is not required for the current implementation. However, each specialty agent does coordinate with the user agent in order to answer queries. In the original system [16], the user agent broadcast messages to all specialty agents. The learning capability that is now part of MACS allows the user agent to learn which specialty agent(s) should receive incoming messages. The user agent asks the ML agent to determine which specialty agent(s) should receive the query. The ML agent makes this determination probabilistically, by means of Bayesian learning. This is explained more fully in Section 3.

2.2 Technical Details of MACS

MACS has been implemented using the Open Agent Architecture (OAA), which is supported by the Artificial Intelligence Lab at the Stanford Research Institute. There are

many facets of the OAA that are worthy of discussion. However, only those capabilities relevant for the MACS system are mentioned here.

The Facilitator agent functions as part of MACS, but it is a specialized server agent that is part of OAA, and it performs many basic functions. The Facilitator agent has the ability to route messages, manage data, and fire registered triggers as well as accept incoming messages. The Facilitator agent delegates incoming messages to the appropriate specialty agent, and specialty agent responses are then relayed back to the requesting agent (the User Agent in the case of MACS).

A process called unification evaluates incoming messages. Unification is a powerful mechanism for determining where the incoming query should be forwarded. For instance, the contracts agent publishes strings such as “contract(Query,Flag,Result)” where the arguments that start with an uppercase letter indicate variables. If a request is received in the form “contracts(‘\$25,000’,’sole source’,X)” it will be routed to the contracts agent. The implementation of the MACS system in OAA allows for message brokering. That is, the facilitator automatically routes incoming queries to specific specialty agents based on the "solvable" each specialty agent registers with the facilitator.

Furthermore, the rules for MACS have been encoded as XML documents. The XML encoding of the rules offers a flexible means for rule modification that was not possible in the earlier version of MACS [16]. Each specialty agent loads the XML document as its rulebase. The rulebase is used to compare against incoming queries to determine if a rule is true or false. The XML rulebases are simply ASCII documents that are served up by a web server. A series of web forms have been designed for modification of the rulebases and general system maintenance.

3. Learning in multiagent systems

3.1. Overview

Learning in MAS can be classified into two main categories: centralized learning and decentralized learning. Centralized learning, also called isolated learning, is performed by a single agent and does not require any interaction with other agents. Decentralized learning, also called interactive learning, is performed through the interaction of several agents who work to achieve the learning goal(s). Issues that must be addressed in MAS learning include the degree of centralization; the level, persistence, frequency and pattern of interaction; the learning method (rote, instruction, examples, analogy, discovery); and the learning feedback (supervised, reinforcement, unsupervised).

The Credit-Assignment Problem (CAP) is the basic problem any learning algorithm is confronted with. The CAP consists of properly assigning credit or blame, for an overall performance change to each of the system activities that contributed to the change. For MAS learning, the CAP problem becomes more challenging because several agents should be considered instead of just one agent, as has been the case in traditional machine learning. The CAP for MAS is decomposed into two subproblems:

- a) *The inter-agent CAP.* The feedback into the system should be distributed among the various agents. This consists of assigning credit or blame to each of the specific agents for an overall performance change.
- b) *The intra-agent CAP.* The feedback for a single agent should be distributed among its internal inferences and decisions. This consists of assigning credit or blame to each internal operation of the agent, which typically are inferences and decisions.

The difference between inter-agent CAP and intra-agent CAP is a conceptual distinction. In practice, it is not always easy to separate the two problems. These two problems are difficult ones for any MAS or traditional learning algorithm.

There are several aspects of learning to be considered for MAS [28]. The first refers to how agents can learn to coordinate their activities by cooperating with other agents in order to achieve overall or individual goals. The second aspect regards how an agent can learn to improve its individual performance. It may happen that an agent's improvement is attained at the expense of other agents. Lastly, the relationship between learning and communication must be assessed.

Agents must be able to learn to coordinate their activities in order to optimize resources or maximize their own profit. Frequently, agents are designed off-line and then put to work without being able to adapt to the environment, to new opportunities, or to the goals of other agents. Then, in order to be effective, agents need to learn about other agents and adapt to a dynamic environment [28]. There are two approaches to this problem. In the first approach, an agent is not aware of the existence of other agents, it acts as if it were the only agent in the system. In the second approach, each agent is aware of the existence of the other agents. In both cases, reinforcement learning is the main technique employed to achieve agent improvement.

Agents improve their own performance by learning about other agents in order to take advantage of opportunities. The objective is to predict the behavior of other agents in order to improve and refine the behavior of the agent. This can be achieved via learning organizational roles, learning to benefit from market conditions, and learning to play better against an opponent.

There are two main research threads for learning and communication: learning to communicate, and communication as learning. In learning to communicate, the idea is to reduce the load of communication among individual agents. It is believed that communication is very slow and expensive; therefore, it should be avoided or at least reduced whenever possible. Enabling the individual agents to acquire and refine knowledge of the other agents' task solving abilities can reduce communication. T. Ohko and colleagues have used case-based reasoning as a learning technique to avoid the need for broadcasting messages when promoting agents bids [22]. In communication as learning, communication is viewed as a method for exchanging information that allows agents to refine their learning activities. Agents that have a limited access to relevant information run the risk of failing in solving a given learning task. Enabling the agents to explicitly exchange information may reduce this risk. Both perspectives on learning and communication address issues of what to communicate, when to communicate, with whom to communicate, and how to communicate.

3.2. Bayesian Learning

3.2.1. Background

Bayesian probabilistic inference has been widely used to represent and reason with uncertain knowledge in intelligent systems [23]. Bayesian inference uses a Bayesian network representation to make inferences about how likely it is that a hypothesis will be true given certain evidence. The nodes of the network represent random variables and the arcs represent dependencies among variables, and the dependencies may represent causal

relationships, where the arrows point from the causes to the effects. Bayesian inference is commonly applied for such things as diagnostic, classification, and prediction tasks, information retrieval and user profiling.

Bayesian networks and inference have been used as knowledge representation and automated reasoning techniques, respectively, for uncertain domains. In one approach to Bayesian modeling, human experts are responsible for designing the network structure by identifying the variables and finding the dependencies among the variables, and also, they are responsible for judging and calculating the conditional probabilities associated with each node. In recent years, attempts have been made to automate both the design of the network structure, as well as the calculation of the conditional probabilities [5] [6] [12]. This automation process is known as *Bayesian learning*. In this case, probabilities are not known *a priori*. Rather, in Bayesian learning, networks (their structure and their conditional probabilities) are learned (or updated) automatically in order to facilitate the design and use of Bayesian networks in various tasks.

3.2.2. Multiagent systems

Essentially, Bayesian learning in MAS allows agents to use new or updated knowledge about other agents in the system to enhance performance of the system as a whole. Recent work on multiagent learning has been conducted by such researchers as Sen and Sekaran [27], Zeng and Sycara [32] [31], Goldman and Rosenchein [11], Bui et al. [4], and Prasad, Lesser, and Lander [25]. Much of this research has been for learning in autonomous negotiations and learning organizational roles (i.e., group dynamics). Our work differs by looking at a problem that does not involve negotiation. Rather than learning

preferences of other agents, the MACS system learns abilities of other agents in order to enhance performance and efficiency of the system. Furthermore, the MACS system applies agent learning to the domain of defense contracting.

Zeng and Sycara [32] present an approach to negotiation in MAS based on the sequential decision making paradigm where a sequence of decisions are dependent on each other, and the decision maker has a chance to update his knowledge after implementing the decision at each step in the sequence via feedback. This allows for more informed decision making as the series of decisions progresses. Their sequential decision making model, Bazaar, learns by explicitly modeling beliefs about the negotiation environment and the participating agents under a probabilistic framework using a Bayesian learning representation and updating mechanism [32].

Bui et al. [4] and Sandolm and Lesser [26] have conducted research related to that of Zeng and Sycara [32]. Bui et al. [4] use probabilistic learning to allow agents in a MAS to learn other agents' preferences during negotiation. A Bayesian classifier augments agent communications by capturing knowledge from past negotiation exchanges to allow agents to update beliefs about other agents and thus adjust behavior for the current exchange. Sandolm and Lesser [26] look at coalition building among agents by learning preferences of other agents in the system. This is useful for negotiations in which there are more than two agents that are negotiating to solve a problem in the MAS.

Prasad, Lesser, and Lander [25] look at a different type of learning in MAS — coordination. Their work looks at how agents can change their behavior as they learn about the MAS of which they are a part. As an agent's knowledge of the system configuration changes, it adjusts its behavior to help meet the common goals of the system and enhance

its overall performance. Learning occurs in the form of each agent learning which part(s) of the common goal it can work to solve. This work uses the TEAM framework presented by Lander and Lesser [15] to explore issues of learning.

The examples detailed above provide an overview of past work on multiagent systems. The Zeng and Sycara [32], Bui et al. [4], and Sandolm and Lesser [26] examples involve deliberate agents that can act, more or less, on their own to solve a problem. In contrast, Prasad, Lesser, and Lander [25] look at reactive agents. In a reactive system, each agent possesses partial knowledge required for successful problem solving so that the agents must work together to complete the task(s) at hand.

3.3. Learning in MACS

In this section, we describe how learning is undertaken in MACS. Bayesian learning is applied in the ML agent so that it can learn which specialty agents should receive incoming messages. The approach is similar to that of Heckerman and Horvitz [13] where user goals are inferred from user queries using a naive Bayesian classifier. However, the Heckerman and Horvitz [13] approach allows for free-text queries whereas MACS is designed for both free-text, Natural language, queries and keyword searches.

Essentially, the user agent employs a Bayesian model to identify which specialty agent(s), a_i , are most likely to respond correctly to a query given the evidence, e , appearing in the query, q . The Bayesian learning procedure is as follows:

1. User enters a query, $q_{incoming}$
2. Use Bayesian reasoning to determine which a_i should receive the query.
 - For each a_i ($i = \text{evaluation, synopsis, justification, forms, type of contracts}$):
 - ◆CALCULATE the percentage of time each keyword appears in all $q_{existing}$ (e.g., evaluation criteria appears in 80% of existing queries sent to the $a_{evaluation}$)
 - ◆CALCULATE probability(evidence/ a_i) by multiplying all percentages calculated in the immediately preceding step that correspond to $q_{incoming}$. Probability(evidence/ a_i) represents the likelihood that a query actually corresponds to the domain knowledge of that a_i . This is a causal relationship from the cause (a_i) to the effect ($q_{incoming}$).
 - ◆USE the Bayesian formula by (a) multiplying the prior probability, $P(a_i)$, by probability(evidence/ a_i), (b) sum all calculations from (a), and (c) divide each individual result from (a) by (b). This will give the probability(a_i /evidence). $P(a_i)$ represents the likelihood $q_{incoming}$ should be sent to a particular a_i given no evidence. At time 0, $P(a_i) = 0.2$ for all i . Probability(a_i /evidence) is the posterior probability distribution of a_i given the evidence. This probability assesses the likelihood that a specialty agent will answer $q_{incoming}$ based on the evidence provided by $q_{incoming}$ itself.
 - ◆DIVIDE probability(a_i /evidence) by $P(a_i)$
 - After completing the calculations for each a_i
 - ◆IDENTIFY the result with the greatest value
 - ◆SEND $q_{incoming}$ to the corresponding a_i
 - ◆ADD $q_{incoming}$ to the list of $q_{existing}$ for the corresponding a_i

The learning model is displayed in Figure 2. This model is simple, it shows a causal relationship between the specialty agents and the evidence in the query. a_i is a random variable with a probability distribution over the five specialty agents. Evidence is a random variable gathered from $q_{incoming}$, with a probability distribution over the attributes associated with each specialty agent. Evidence consists of keywords that are extracted from a user's natural language query and used to probabilistically determine which specialty agent(s) should receive $q_{incoming}$.

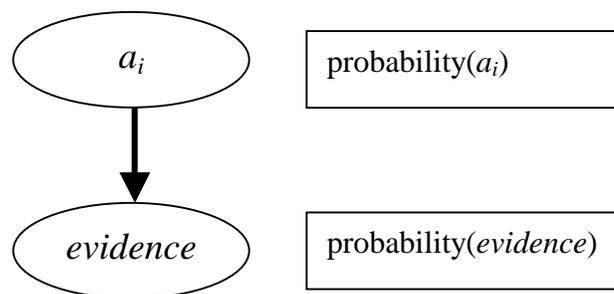


Figure 2. Bayesian network for the user agent

An example of how Bayesian learning is implemented in MACS is provided in subsequent paragraphs. The prior probability for each specialty agent represents the likelihood that agent will respond to an incoming query. The prior probabilities are adapted over a series of runs, and the values displayed below are from a point in time prior to the sample run used in this example. In the example detailed below, the Contracts agent has the greatest probability, when compared to the other specialty agents, of responding to a new query. This indicates that a majority of the incoming queries contain terms that are used by the Contracts agent. It is important to note that all of the prior probabilities sum to one; the probability displayed as 1.0 for the Contracts agent is rounded up and is not actually 1.0. The prior probabilities are given in Table 1.

Table 1. Prior Probabilities for MACS Example

Agent	Prior Probabilities
forms	2.5822858689540867 E-242
contracts	1.0
justification	1.0240324973261967 E-166
evaluation	1.9181461523891187 E-214
synopsis	3.6206243551615097 E-190

The example run includes the following four keyword terms:

1. Not on GSA Schedule
2. Cost plus fixed fee
3. Synopsis format
4. Sole source (non-competitive) procurement

The terms are sent to the Machine Learning agent and the Bayesian formula is applied. At the time of this run, none of these terms submitted were input in previous queries in either the Forms or Evaluation agent. The remaining specialty agents have used some of these terms in previous responses. Table 2 presents calculations from the first step in the Bayesian learning procedure.

Table 2. Step 1 in Bayesian Learning

Agent	Probability (evidence/a _i)
forms	1.0000000000000004E-20
contracts	3.698224852071007E-14
justification	1.0526315789473686E-16
evaluation	1.0000000000000004E-20
synopsis	1.5306122448979592E-12

The probability(evidence/ a_i) values for each specialty agent are then multiplied by their respective prior probabilities and summed as shown in equation [1]. This value is the denominator for the Bayesian formula in subsequent calculations.

$$\begin{aligned}
 &\text{Bayesian denominator:} \\
 &(2.5822858689540867\text{E-}242 * 1.0000000000000004\text{E-}20) + \\
 &(1.0 * 3.698224852071007\text{E-}14) + \\
 &(1.0240324973261967\text{E-}166 * 1.0526315789473686\text{E-}16) + \\
 &(1.9181461523891187\text{E-}214 * 1.0000000000000004\text{E-}20) + \\
 &(3.6206243551615097\text{E-}190 * 1.5306122448979592\text{E-}12) = \\
 &\hspace{15em} 3.698224852071007\text{E-}14 \hspace{10em} [1]
 \end{aligned}$$

The next step is to calculate the updated prior probability values, as well as the probability(a_i /evidence). These values are used to determine the order in which the agents will be contacted to answer the incoming query. The calculations used for the Forms agent are included here in equations [2] and [3]. The same calculations are made for each agent.

$$\begin{aligned}
 &\text{Posterior probability:} \\
 &1.0000000000000004\text{E-}20 * 2.5822858689540867\text{E-}242 / 3.698224852071007\text{E-}14 = \\
 &6.98250098965185\text{E-}249 \hspace{10em} [2]
 \end{aligned}$$

$$\begin{aligned}
 &\text{Score:} \\
 &6.98250098965185\text{E-}249 / 2.5822858689540867\text{E-}242 = 2.704\text{E-}7 \hspace{10em} [3]
 \end{aligned}$$

The posterior probability and score calculations for each specialty agent calculated from equations [2] and [3] are given in Table 3.

Table 3. Posterior Probability and Score Calculations for Bayesian Learning

Specialty Agent	Posterior Probability	Score
Forms	6.98250098965185E-249	2.704E-7
Contracts	1.0	1.0
Justification	2.914719866073721E-169	0.0028463157894736833
Evaluation	5.186667196060177E-221	2.704E-7
Synopsis	1.4984951412790896E-188	41.3877551020408

As is indicated by the Score values, the Machine Learning agent has determined that the Synopsis agent is the most likely agent to have a response for the incoming query. In fact, the Synopsis agent is the correct agent to respond in this example. The posterior probability values are then used to update the prior probability values for each specialty agent for the next iteration of MACS.

4. Analysis of learning in MACS

4.1 Methodology

In order to evaluate the performance of the ML agent, it is tested against historical data. This is accomplished via the current implementation of the User agent that is coded to send incoming queries to all specialty agents in addition to the ML agent. This allows all specialty agents to access historical data in their knowledge domains when responding to queries. The results from the broadcast query are then compared to the results from the ML agent in order to determine how well the ML agent is performing. The level of effort required of the ML agent in order to obtain a correct response from keyword searches in MACS, is the focus of this paper. The methodology for data collection, and how the current system is implemented, can be summarized as follows:

1. A user enters a query, $q_{incoming}$
2. The query is packaged up by the User agent and broadcast to all specialty agents
3. Results from each agent are collected
4. The User agent sends the original $q_{incoming}$ to the ML agent
5. Data on where the ML agent suggested routing $q_{incoming}$ are collected
6. Results from the specialty agents contacted via machine learning are collected
7. Results from both the broadcast and machine learning paths are compared and analyzed

Results are analyzed using descriptive statistics. First is the accuracy of the ML agent. Here, the specialty agent to which the ML agent sends a query is compared to which

specialty agent(s) replied to that same query when the query was broadcast to all. This statistic targets accuracy of the system, but not learning. Learning is handled by the second statistic captured.

The second set of data collected is the number of specialty agents the ML agent sent a query before reaching the *correct* agent. The term *correct* used here refers to the agent with the domain expertise to answer a user's question. This statistic targets the ability of the system to learn. Learning is evaluated over time using this statistic, where messages should be sent to fewer and fewer specialty agents before reaching the correct one over time. That is, the Bayesian probabilities are updated after every new query so that over time the ML agent learns which specialty agents can answer which types of questions.

4.2 Results

4.2.1 Accuracy

Accuracy of MACS is 74.31%. That is, 74.31% of the time, an incoming query was sent to the correct specialty agent and fully resolved. The accuracy rate of 74.31% represents only those cases where the query was fully resolved by a single specialty agent's response. In many cases, multiple specialty agents may have to respond to a query in order to resolve it fully, where each specialty agent provides a different area of domain expertise. 22.94% of the time there was a second specialty agent that should also have received the query, 1.83% of the time there were two additional specialty agents that should also have also received the query, and 0.92% of the time there were three additional specialty agents that should have also received the query.

The current design of MACS only allows the ML agent to designate one specialty agent to receive an incoming query. Thus, the system design will need to be changed before the accuracy rate can be improved above 74.31%. However, it should also be noted that Bayesian learning identified a correct specialty agent in every case. That is, there may have been multiple specialty agents necessary to fully resolve the query, but at least one of those was identified in every case. Thus, queries were not sent to specialty agents with no expertise relevant for the incoming query.

4.2.2 Learning

The data on how many attempts the ML agent made before reaching the correct agent to whom queries should be sent suggest the MACS system is in fact learning. Figure 3 illustrates learning over time. As the trend line shows, over time, fewer tries were needed before the correct specialty agent was identified, indicating the ML agent sufficiently learned specialty agents to which incoming queries should be sent. The y-axis identifies the number of tries the ML agent made before learning the correct agent.

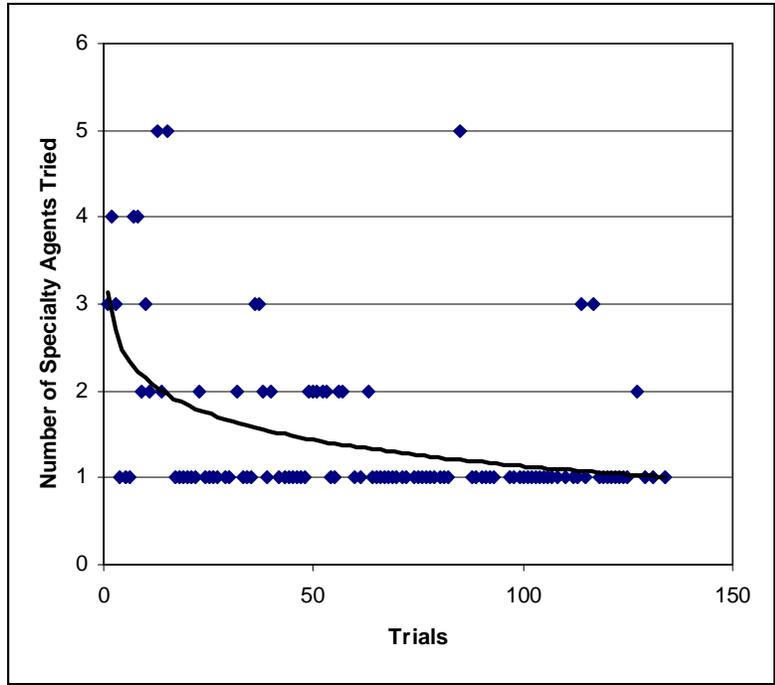


Figure 3. Learning in MACS Over Time

5. Conclusions and Future Directions

The multiagent system, MACS, presented in this paper illustrates how Bayesian learning can be built into a MAS, and how a human-based process can transition into an agent-based process. A real-world problem in the area of defense contracting is used to illustrate the usefulness of the system design and learning techniques built into the system. MACS has been extended beyond the system presented by the authors [16] in previous research in that it now possesses the ability to learn from and adapt to its environment. Furthermore, the system has been migrated to OAA and uses XML coding. There is also a natural language agent that is in the process of being developed and expanded.

The system has been tested against known, historical data in order to show definitively that it is learning. The data presented in Section 4 support the authors' claims

that Bayesian learning is a meaningful approach, and that learning is in fact occurring in the system. However, the authors believe the results would be far more dramatic in a larger system where there are many more specialty agents that may need to respond to queries. In such systems, performance enhancements from Bayesian learning will be more pronounced because resources allocated to agent communication increase as system size increase. Clearly, as the number of agents increases, the value of targeted brokering versus communicating with all agents in the system increases.

While the research presented in this paper contributes to the existing MAS literature by building learning into a Web based MAS, it also lays a foundation upon which future work can build. The primary direction for future work is in the area of more comprehensive learning than what is currently achieved by the MACS system. This will occur in two dimensions. First, future research will look at a MAS in which the specialty agents cooperate with each other to more completely answer user queries. This is in contrast with the existing system where specialty agents communicate with a user agent, but not with each other. Second, reinforcement learning will be built into the system so that MACS can learn from user feedback.

6. References

- [1] G. Ayala, Y. Yano, "A collaborative learning environment based on intelligent agents," *Expert Systems with Applications* vol. 14 pp. 129-137, 1998.
- [2] W. Baker, P. R. Witmer, "Intelligent agents go to work for management accountants," *Management Accounting* vol. 78 p. 32, 1997.
- [3] J. M. Bradshaw, *Software Agents*, AAAI Press: Menlo Park, California, 1997.
- [4] H.H. Bui, D.H. Kieronska, and S. Venkatesh, "Learning other agents' preferences in multiagent negotiation," in: *Proc. of the 13th Nat. Conf. on Artif. Intell. and 8th Innovative Applications of Artif. Intell. Conf.*, The MIT Press, Cambridge, MA, 1996, pp. 114-119.
- [5] W.L. Buntine, "Operations for learning with graphical models," *Journal of Artificial Intelligence Research* vol. 2 pp. 159-225, 1994.
- [6] G.F. Cooper and E. Herskovitz, "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning* vol. 9 pp. 309-349, 1992.
- [7] G. Elofson, M. Beranek, P. Thomas, "An intelligent agent community approach to knowledge sharing," *Decision Support Systems* vol. 20 pp. 83-98, 1997.
- [8] J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley: Harlow, England, 1999.
- [9] B. R. Gaines, "Knowledge management in societies of intelligent adaptive agents," *Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies* vol. 9(3) pp. 277-298, 1997.
- [10] D. Gilbert, M. Aparicio, B. Atkinson, S. Brady, J. Ciccarino, B. Grosz, P. O'Connor, D. Osisek, S. Pritko, R. Spagna, and L. Wilson. "IBM Intelligent Agent Strategy," IBM Corporation, 1995.
- [11] C.V. Goldman and J.S. Rosenschein, "Mutually supervised learning in multiagent systems," in G. Weiss and S. Sen (eds.), *Adaptation and Learning in Multiagent Systems*, Springer-Verlag, New York, 1996, pp. 85-96.
- [12] D. Heckerman, D. Geiger, and D.M. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Machine Learning* vol. 20 pp. 197-243, 1995.
- [13] D. Heckerman and E. Horvitz, "Inferring informational goals from text-free queries: A Bayesian approach," in *Proc. of the 14th Conf. on Uncertainty in Artif. Intell.*,

- Morgan Kaufmann, San Francisco, CA, 1998, pp. 230-237.
- [14] J. H. Holland, "Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems," in R. S. Michalski, J. G. Carbonnell and T. M. Mitchell (eds.), *Machine Learning: an Artificial Intelligence Approach II*, Tioga, Palo Alto, CA, 1986, pp. 593-623.
 - [15] S.E. Lander and V.R. Lesser, "Sharing metainformation to guide cooperative search among heterogeneous reusable agents," *IEEE Transactions on Knowledge and Data Engineering* vol. 9(2) pp. 193-208, 1997.
 - [16] J. Liebowitz, M. Adya, B. Rubenstein-Montano, J. Buchwalter, and M. Imhoff, "MACS: Multi-Agent COTR System for Defense Contracting," *Knowledge-Based Systems Journal* vol. 13(3), in press.
 - [17] J. MacIntosh, S. E. Conry, and R. A. Meyer, "Distributed automated reasoning: Issues in coordination, cooperation, and performance," *IEEE Transactions on Systems, Man, and Cybernetics* vol. 21(6) pp. 1307-1316, 1991.
 - [18] J. N. Martínez and P. P. González, "Net of multi-agent expert systems with emergent control," *Expert Systems with Applications* vol. 14 pp. 109-116, 1998.
 - [19] T. Mitchell, *Machine Learning*, McGraw-Hill: New York, 1997.
 - [20] D. H. Norrie and B. R. Gaines, "The learning web: A system view and agent-oriented model," *International Journal of Educational Telecommunications* vol. 1(1) pp. 23-41, 1995.
 - [21] M. O. Odetayo, "Knowledge acquisition and adaptation: A genetic approach," *Expert Systems* vol. 12(1) pp. 3-13, 1995.
 - [22] T. Ohko, K. Hiraki, and Y. Anzai, Y, "Addressee learning and message interception for communication lead reduction in multiple robot environments," in G. Weiss (ed.) *LNAI 1221*, Springer-Verlag, Berlin, 1997.
 - [23] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann: San Mateo, California, 1988.
 - [24] P. C. Pendharkar, "A computational study on design and performance issues of multi-agent intelligent systems for dynamic scheduling environments," *Expert Systems with Applications* vol. 16(2) pp. 121-133, 1999.
 - [25] M.V.N. Prasad, V.R. Lesser, S.E. Lander, "Learning organizational roles in a heterogeneous multiagent system," in *ICMAS-96 Proceedings, Second Annual Conf. on Multi-agent Systems*, AAAI Press, Menlo Park, CA, 1997, pp. 291-298.

- [26] T.W. Sandholm and V.R. Lesser, "Coalition formation among bounded rational agents," in Proc. of the 14th Int. Joint. Conf. on Artif. Intell., Morgan Kaufmann, San Mateo, CA, 1995, pp. 662-669.
- [27] S. Sen and M. Sekaran, "Multiagent coordination with learning classifier systems," in Adaption and Learning in Multi-Agent Systems, IJCAI 95 Workshop, Springer-Verlag, Berlin, Germany, 1996, pp. 218-233.
- [28] S. Sen, and G. Weiss, "Learning in multiagent systems," in Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press, Cambridge, Massachusetts, 1999, pp. 259-298.
- [29] S. F. Smith, "Adaptive learning systems," in R. Forsyth (ed.) Expert Systems Principles and Case Studies, Chapman and Hall, London, 1994.
- [30] J. Vaario and K. Ueda, "Modular learning in a multiagent environment," in C. H. Dagli, M. Akay, C. L. P. Chen, B. R. Fernandez and J. Gosh (eds.) Intelligent Engineering Systems Through Artificial Neural Networks, ANNIE 96 Proc., ASME Press, New York, 1996.
- [31] D. Zeng and K. Sycara, "Bayesian learning in negotiation," AAAI Spring Symposium, Working Notes, Stanford University, 1996.
- [32] D. Zeng and K. Sycara, "Bayesian learning in negotiation," International Journal of Human Computer Studies vol. 48(1) pp. 125-141, 1998.