"6th IASTED International Conference, Artificial Intelligence and Soft Computing (ASC 2002)",
17-19 July 2002, Banff, Alberta, Canada

MINING PATTERNS AND RULES FOR IMPROVING AGENT INTELLIGENCE THROUGH AN INTEGRATED MULTI-AGENT PLATFORM

Andreas L. Symeonidis, Pericles A. Mitkas and Dionisis D. Kechagias
Department of Electrical and Computer Engineering
Aristotle University of Thessaloniki
& Laboratory of Intelligent Systems and Software Engineering,
Informatics and Telematics Institute
CERTH
Thessaloniki, Greece

Andreas L. Symeonidis
Tel. +30-31-99-6349
Fax +30-31-99-6398
email: asymeon@ee.auth.gr

Pericles A. Mitkas
Tel. +30-31-99-6390
Fax +30-31-99-6398
email:mitkas@eng.auth.gr

Dionisis D. Kechagias
Tel. +30-31-99-6349
Fax +30-31-99-6396
email:diok@iti.gr

Keywords: Data Mining, Multi-Agent Systems, Agent Training, Classification, Decision Trees, Association Rules

Abstract

The integration of data mining techniques with multi agent systems to assist in dealing with information overload has received some attention during the last years. Agent Academy, a platform for training agents, introduces a whole new perspective on the improvement of agent intelligence. Data mining techniques are used in order to extract useful patterns on real high-risk and time-efficient applications, and to provide the platform with rules, decisions and classes on test case data. These metadata are embedded into agents in order to improve their existing intelligence. This paper describes the Agent Academy platform, and focuses on the issues and challenges its development has revealed through the prism of data mining.

1. Introduction

The last few years Intelligent Agent (IA) technology has started to regain its glow, introducing an explosion of brand new computer-based services. The use of IAs promises the transformation of computers into personal collaborators that can provide active assistance and even take the initiative in decision-making processes. This new approach can facilitate "smart" solutions in SMEs concerning management, resource allocation and remote administration. An agent can be a) created with enough initial intelligence, b) pre-trained to an acceptable competence level, or c) sent out untrained to learn on its own. Taking the middle ground, a rather simple agent can be created and then trained to learn, adapt, and get smarter and more efficient, through its experience from former transactions with other agents.

On the other hand, data mining is by definition the process of discovering patterns in large databases [1]. Mining information and knowledge has been recognized by many researchers as a key research topic in database systems and machine learning.

Collecting, therefore, the content of agent experience in a federated information repository, has given us the ability to "mine" this data, in order to discover new agent behavioral patterns, and to apply these patterns to existing agents. The aforementioned procedure of monitoring agents, discovering knowledge on their behavior and embedding it back to them, is realized through our proposed framework, the Agent Academy.

The main goal of the **Agent Academy** (AA) framework is to develop a platform for enhancing agents' functionality and intelligence through the use of **Data Mining** (DM) techniques. This particular framework is concerned with two major issues: the first one focuses on the way the AA trains the agents, in order to provide them with the necessary skills to successfully accomplish their assigned tasks. This mechanism combines the use of DM techniques to improve the decision making process in Intelligent Agents (IA), the training of the newly "born" agents, as well as the way the AA deals with different types of agents. The second major issue involves techniques by which the AA obtains information on the way it should train its own agents.

1.1 Functional Description of the Agent Academy

Agent Academy forms an integrated framework that receives input from its users and the Web. The main block diagram of AA is illustrated in Figure 1 and shows the communication of its internal components.. AA operates as a multi agent system, which can train new agents or retrain its own agents in a recursive mode. A user issues a request for a new agent as a set of functional specifications. The Agent Factory, a module responsible for selecting the most appropriate agent type and supplying the base code for it, handles the request. A newly created untrained agent (UA) comprises a minimal degree of intelligence, defined by the software designer. This agent enters the *Agent-Training Module*, where its world perception increases substantially during a virtual interactive session with an agent master (AM). Based on the encapsulated knowledge, acquired in the knowledge extraction phase, an AM can take part in long agent-to-agent (A2A) transactions with the UA. This process may include modifications in the agent's decision path traversal and application of augmented adaptivity in real transaction environments.

The core of AA is the *Agent Use Repository* (AUR), which is a collection of data on prior agent behavior and experience. It is on the contents of AUR, where data mining techniques, such as extraction of association rules for the decision making process, are applied in order to augment the intelligence of the AM in the training module. Building AUR is a continuous process performed by a large number of mobile agents that are controlled by the *Data Acquisition Module*. A large part of an agent's intelligence handles the knowledge acquired by the agent since the beginning of its social life through the interaction with the environment it acts upon.

After the training is complete, the now intelligent agent, armed with tools for reporting its behavior to the AA, is released to the world. The AA is continuously modified, as it receives feedback from mobile agents roaming the web, updating its agent use repository and refining its data mining and agent training techniques.

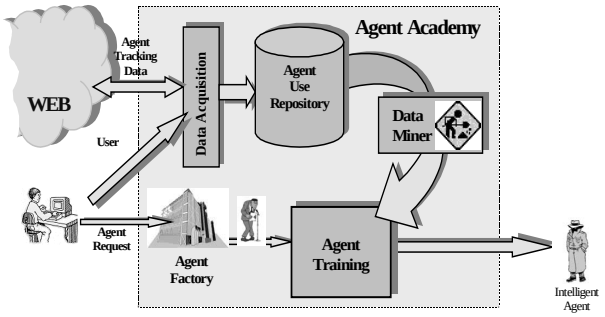

Figure 1. The Agent Academy and its environment.

1.2 Specifications on technologies

A variety of choices were available for the implementation of the AA framework. The integration of the AA components into one fully functional platform gave rise to issues on multi-agent system architecture, on agent development and agent communication, on interoperability and on data mining techniques and algorithms. A survey on the state-of-the-art technologies has led to accurate decisions on these issues.

JADE agents [2] are produced through the AF, as it is of great concern that the AA platform is FIPA compliant, adopting therefore all the architectural specifications, limitation as well as capabilities FIPA organization defines. In order to fulfill all data mining requirements for the successful extraction of knowledge, a MOF (Meta-object Facility) repository [3] is developed to store agent data. Association rule extraction, classification and clustering are the selected data mining techniques, whereas the algorithms used are dependent on the application the agent monitors. The RMI (Remote Method Invocation) communication protocol ensures interoperability between mobile objects, whereas JESS [4] is used to represent knowledge extracted from the DM. Finally, FIPA-ACL is the communication language of agents.

2. Components of Agent Academy

The Agent Academy framework consists of four basic components. These are: a) the Agent Factory (AF), b) the Agent Use Repository (AUR), c) the Data Miner (DM) and, d) the Agent Training component. Figure 2 is the block diagram of AA that illustrates the interconnections between these distinct yet coupled software modules. The operations of these modules, except from the Data Miner, are outlined below. Due to our desire to focus on its internal architecture and the issues its development has generated, Data Miner is presented on the next section of the paper.

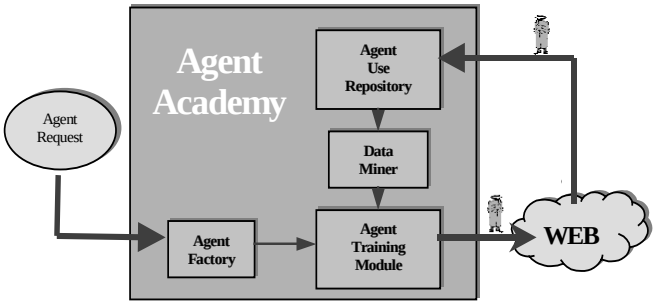

Figure 2. The components of the AA

2.1 The Agent Factory

The Agent Factory utilizes JADE in order to:

i. create new agents on user demand, upon his/her preferences and special request, via a user interface.
ii. supply the user with a generic agent toolkit which supports the generation of certain types of agents with special characteristics
iii. provide a middleware interface between JADE and the AA platform in order to ensure interoperability and compatibility between the two platforms.

2.2 The Agent Use Repository

The Agent Use Repository consists of two subcomponents: the Data Acquisition module and the Repository. MOF and JMI (Java Metadata Interface) are used in order to:

i. develop the required agent tracking tools, which will be able to support the monitoring of Agent-to-Agent (A2A) transactions and audit certain specific attributes in a communication procedure between two or more agents
ii. store data of former A2A transactions into the host repository
iii. provide a formal description of stored data using a certain meta-data formalism

2.3 The Agent Training module

The Agent Training module uses JESS rules and FIPA-ACL in order to:

i. support interoperability with the AF
ii. implement software entities to support the manipulation of data derived from the Data Miner
iii. develop techniques to enable the ability of learning to an agent
iv. develop mechanism to support the modification of an agent's source code in order to improve its intelligence

3. The Data Miner

Data Mining is performed on two levels of abstraction:

i. Association rules are extracted from the message exchange between agents cooperating on a certain application. Usual behavior patterns are identified resulting to the formation of association rule trees (with a pre-specified confidence and support)
ii. Classification is performed on data concerning the overall performance of the certain application. Decision trees on the values of significant control attributes are realized, leading to intelligent decision making on behalf of the agents

3.1 Subcomponents

The process of performing data mining on the contents of the Agent Use Repository is divided into three major phases, each one of which includes a number of essential tasks according to the knowledge discovery in databases paradigm [1], [5]. These three phases are substantiated through equal subcomponents in the Data Miner. These three components are the *preprocessor*, the *performer* and the *evaluator*.

The preprocessor is a Java interface that collects data from both AUR databases, the communication and the application repository (the aforementioned distinct levels of abstraction). Different types of agents provide data to different data pools, according to the ontology they belong. Next, the String type records of the repository are refined. Noisy data are removed and missing values are filled with the most probable tuple, by using a decision tree induction. Finally, the preprocessed data are formatted into .arff data set files, according to the requirements of the Waikato Environment for Knowledge Analysis (WEKA)– [6] and are exported to the Performer.

In turn, the Performer performs, as its name implies, data mining on the .arff-format instances of data imported from the Preprocessor. ID 3 [7], [8] and C4.5 [9], [10] algorithms are applied for application data. Information on the attribute the data set will be classified on is included in the AUR records. On the other hand, Apriori and Partitioning algorithms [8], [11] are applied on communication data. All data mining algorithms result to decision tree-like structures, which are exploited in order to augment agent intelligence.

Finally, the Evaluator receives the extracted knowledge (decision trees, decision rules, association rules) and creates the knowledge files that include the extracted intelligence. These files lack supplementary data mining information provided by WEKA, in order to simplify the integration of the Data Miner and the Agent Training module.

3.2 AUR-DM interoperability

The agent retains information, in the form of an ACL message. This message provides the AUR with all the essential data to create the data records corresponding to the particular agent. An example of such an ACL message for an agent named Alpha asking Beta if a certain machine functions properly is provided in figure 3. Data in ACL messages is of String type, so the Preprocessor is forced to manipulate the ACL-message string and produce the .arff file, in the correct format.

```
Request (sender (Alpha)
        Receiver (Beta)
        Ontology pp
        :content (action(Beta, normalFunc)))
```

Figure 3. An ACL-message

From each such record (communication or application record), the ontology of the agent can be determined, as well as the content it exchanged with the other agents or the values of the attributes it monitors and a tuple can be added in the data mining training file.

3.3 DM-ATM interaction

Figure 4 demonstrates the interaction between the Data Miner and the Agent Training Module. The extracted data mining knowledge is represented through JESS rules into BDI (Belief Desire Intention) rules. This way knowledge is posted to the agent that must be in suspend mode to accept its upgraded "intelligence".

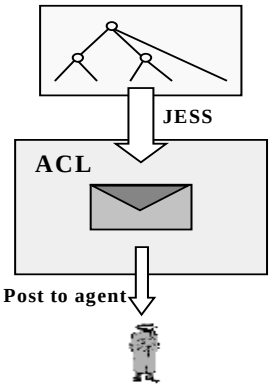

Figure 4. DM-AT interaction

3.4 A working example

In order to make the procedure of data mining comprehensible, let us present a simple example. One of the three test cases of the Agent Academy is the production planning scenario. In this scenario, agents control all the variables that influence production planning in a factory. At top of the hierarchy lays the Service Resource Agent (SRA), an agent that decides if production will initiate or not, according to the values of the attributes it controls. SRA exchanges messages with other agents and collects information that stores as an ACL message. The ACL message of the SRA is something like:

```
(Inform-Values      (ProductionLine Out of order)
                    (MaterialStock Adequate)
                    (HumanResources Yes)
                    (Weekday Mo)
                    (BeginProduction No)
                    )
```

where control information on the attributes is denoted into the ACL message in the form of:

```
(Inform-Values-Control
        (BeginProduction Yes No Recheck)
        (ProductionLine OutofOrder Occupied Available)
        (HumanResources Yes No)
        (MaterialStock Adequate Critical Poor)
        (Weekday Mo Tu We Th Fr Sa Su))

                    )
```

The Preprocessor parses the ACL messages from all SRA records in the application repository of the AUR and creates an .arff file that looks like:

```
@relation BeginProduction
@attributes ProductionLine, MaterialStock, HumanResources,
            Weekday
@attribute MaterialStock {Poor, Critical, Available}
@attribute HumanResources Boolean
@attribute Weekday {Mo Tu We Th Fr Sa Su}
@attribute BeginProduction {Yes, No, Recheck}

@data
OutofOrder, Adequate, Yes, Mo, No;
Available, Adequate, Yes, Su, Yes;
Occupied, Adequate, No, Tu, Recheck;
Available, Critical, Yes, Tu, Recheck;
                ……..
                ……..
                ……..
```

After having applied the ID3 through WEKA, the resulting decision tree is the one shown in figure 5:

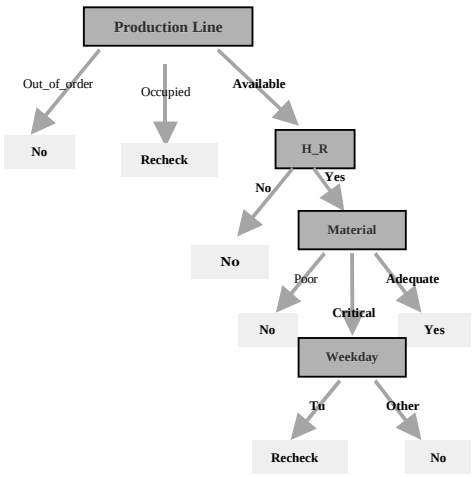

Figure 5. The resulting decision tree

As the extracted knowledge must be embedded into the SRA source code, a string version of the decision tree is realized. The representation of the tree in an ACL message is:

Inform-DT	(ProductionLine No OutofOrder) (ProductionLine Recheck Occupied) (ProductionLine HumanResources Available) (HumanResources No No) (HumanResources Material Yes) (Material No Poor) (Material Weekday Critical) (Material Yes Adequate) (Weekday Recheck Tuesday) (Weekday No Other))

This message is passed to the behavior of the SRA, improving therefore its intelligence.

4. Conclusions

Agent Academy attempts to develop a framework that supports the efficient and effective generation of network and system management applications using novel middleware technologies such as intelligent agents. The use of intelligent agent technology will make these applications dynamically adjustable to a changing environment. In contrast to existing management systems from commercial vendors, the new framework enables the development of lean, customizable, user-oriented management applications that do not incur the overhead and heavy burden of general-purpose solutions.

5. Acknowledgements

This work has been partially supported by the European Commission through the IST project No 2000-31050.

6. References

[1] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth "Knowledge Discovery and Data Mining: Towards a unifying framework".

[2] http://jade.cselt.it/

[3] http://www.omg.org/ technology/cwm/

[4] http://herzberg.ca.sandia.gov/jess/

[5] Usama Fayyad, "Mining Databases: Towards Algorithms for Knowledge Discovery", *Bulletin of the Technical Committee on Data Engineering*, Vol.21, No. 1, March 1998, pp. 39-48

[6] http://www.cs.waikato.ac.nz/

[7] Amihood Amir, Ronen Feldman, and Reuven Kashi, "A New and Versatile Method for Association Generation", Information Systems, Vol. 22, No. 6/7, 1997, pp. 333-347.

[8] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishman, "Mining Very Large Databases", *Computer Magazine*, August 1999, pp. 38-45.

[9] Quinlan, J. R. C4.5: Programs for Machine Learning, Morgan Kaufmann pub. (1993).

[10] Quinlan, J. R. Learning decision tree classifiers. In ACM Computing Surveys, Vol. 28, No.1 (1996) 71-72.

[11] Tzung-Pei Hong, Chan-Sheng Kuo, and Sheng-Chai Chi, "Mining Association Rules from Quantitative Data", *Intelligent Data Analysis*, Vo. 3, Issue 5, November 1999, pp. 363-376.

[12] Jeffrey M. Bradshaw, An Introduction to Software Agents, Software Agents, Jeffrey Bradshaw (editor), AAAI/MIT Press, 1997.

[13] Soumen Chakrabati, Byron E. Dom, S. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins, David Gibson, and Jon Kleinberg, "Mining The Web's Link Structure", *Computer Magazine*, August 1999, pp. 60-67.

[14] Decker K. S. Sycara K. P., "Intelligent Adaptive Information Agents", *Journal of Intelligent Information Systems, 9:239--260, 1997*

[15] Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, May 2001

[16] Pattie Maes, "Agents that Reduce Work and Information Overload", *Communication of the ACM July 1994/Vol. 37, No. 7, 31-40. (p. 811).*

[17] Dejan S. Milojicic, "Trend Wars: Mobile agent applications", *IEEE Concurrency, 7(3): 80-90, July – September 1999 issue*

[18] Hyacinth Nwana, "Software Agents: An Overview", *Knowledge Engineering Review Journal, 11(3), November, 1996*

[19] Hyacinth Nwana and Divine Ndumu, "An Introduction to Agent Technology", *BT Technology Journal 14(4), 1996, pp 55-67.*

[20] Ian H. Witten and Eibe Frank, Data Mining: Practical machine learning tools and techniques with Java implementations, October 2000

[21] http://www.kdnuggets.com

[22] http://www.dwinfocenter.org/ datamine.html

[23] http://kdd.ics.uci.edu/

[24] http://www.rulequest.com/