A retraining methodology for enhancing agent intelligence

Andreas L. Symeonidis a,∗, Ioannis N. Athanasiadis b, Pericles A. Mitkas a

a Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki and Laboratory of Intelligent Systems and Software Engineering, Informatics and Telematics Institute/CERTH, 54124 Thessaloniki, Greece
b Istituto Dalle Molle di Studi sull’Intelligenza Artificiale, CH-6928 Manna, Lugano, Switzerland

Received 5 August 2004; accepted 3 June 2006
Available online 17 October 2006

Abstract

Data mining has proven a successful gateway for discovering useful knowledge and for enhancing business intelligence in a range of application fields. Incorporating this knowledge into already deployed applications, though, is highly impractical, since it requires reconfigurable software architectures, as well as human expert consulting. In an attempt to overcome this deficiency, we have developed Agent Academy, an integrated development framework that supports both design and control of multi-agent systems (MAS), as well as “agent training”. We define agent training as the automated incorporation of logic structures generated through data mining into the agents of the system. The increased flexibility and cooperation primitives of MAS, augmented with the training and retraining capabilities of Agent Academy, provide a powerful means for the dynamic exploitation of data mining extracted knowledge. In this paper, we present the methodology and tools for agent retraining. Through experimented results with the Agent Academy platform, we demonstrate how the extracted knowledge can be formulated and how retraining can lead to the improvement – in the long run – of agent intelligence.

Keywords: Data mining; Multi-agent systems; Agent intelligence; Training; Retraining

1. Introduction

In a highly complex and competitive business environment, companies must take swift, yet fit decisions that rely on corporate logic and domain knowledge. Diffusing, however, this knowledge into the software processes of the company is a difficult task, which requires reconfigurable software architectures and human expert involvement. A unified approach for discovering useful corporate knowledge and incorporating it into the company’s software would therefore be highly desirable.

The most dominant solution for discovering non-trivial, implicit, previously unknown and potentially useful [8] knowledge is Data Mining (DM), a technology developed to support the tremendous data outburst and the imperative need for the interpretation and exploitation of massive data volumes. DM issues concerning data normalization, algorithm complexity and scalability, result validation and comprehension have already been successfully dealt with [1,14,25], while numerous approaches have been adopted for the realization of autonomous and versatile DM tools, which foster all the appropriate pre- and post-processing steps that constitute the process of Knowledge Discovery in Databases (KDD) [6,8,20]. The ultimate goal of DM is the extraction of a valid knowledge model (i.e., Decision Rules, Decision Tree, Association Rules, Clusters, etc.) that best describes the trends and patterns that underlie in the data.

On the other hand, despite the support corporate software provides on process coordination and data organization, it often – especially legacy software – lacks advanced capabilities, resulting therefore in decreased company competitiveness. The increasing demand for sophisticated software that comprises of collaborative, yet autonomous, units to regulate, control and organize all distributed activities involved in the company processes, has oriented AI
The coupling of DM and AT principles is therefore expected to provide an efficient gateway for developing highly reconfigurable software approaches that incorporate domain knowledge and provide decision making capabilities. The exploitation of useful knowledge extracted by the use of DM may considerably improve agent infrastructures, while also increasing reusability and minimizing customization costs.

Going briefly through related work, attempts to couple DM and AT already exist. Galitsky and Pampapathi [13] use both inductive (DM) and deductive (AT) approaches, in order to model and process the claims of unsatisfied customers. Deduction is used for describing the behaviors of agents (humans or companies), for which we have complete information, while induction is used to predict the behavior of agents, whose actions are uncertain to us. A more theoretical approach on the way DM extracted knowledge can contribute to AT performance has been presented by Fernandes [10], who attempts to model the notions of data, information and knowledge in purely logical terms, in order to integrate inductive and deductive reasoning into one inference engine. Kero et al. [17], finally, propose a DM model that utilizes both inductive and deductive components. Within the context of their work, they model the discovery of knowledge as an iteration between high-level, user-specified patterns and their elaboration to (deductive) database queries, whereas they define the notion of a meta-query that performs the (inductive) analysis of these queries and their transformation to modified, ready-to-use knowledge.

Advancing on earlier research efforts to couple the two technologies, we have developed Agent Academy [19,22], an integrated platform for developing MAS architectures and for enhancing their functionality and intelligence through the use of DM techniques.

Agent Academy (AA) agents are developed over the Java Agent Development Framework (JADE) [5], which conforms to the FIPA specifications [11]. The MAS ontologies are developed through the Agent Factory module (AF) of AA. Data to be mined are imported to AA in XML format and are forwarded to the Data Miner module (AF) of AA, a DM suite that expands the Waikato Environment for Knowledge Analysis (WEKA) tool [25]. The extracted knowledge structures are represented in PMML (Predictive Model Markup Language), a language that efficiently describes clustering, classification and association rule knowledge models [7]. The resulting knowledge is then incorporated into the agents of the MAS by the use of the Agent Training Module (ATM) of AA. All necessary data files (application data, agent behavior data, knowledge structures, agent ontologies) are stored into AA’s main database, the Agent Use Repository (AUR). Agents can be periodically recalled for retraining, since appropriate agent tracking tools have been incorporated into Agent Academy, in order to monitor agent activity after their deployment.

It is through retraining that we intent to prove certain DM techniques can be used to augment agent intelligence and therefore improve MAS overall performance. The rest of the paper is organized as follows: Section 2 determines the formal model for training and retraining agents through Agent Academy and specifies all the necessary notations. Section 3 outlines the already developed mechanism for training and retraining, while Section 4 describes the various training and retraining options for the improvement of agent intelligence and presents some indicative experimental results. Finally, Section 5 summarizes and concludes the paper.

2. Formal model for agent (re)training

When a MAS application is deployed by the use of Agent Academy, the developer has to follow a certain methodology. These steps are:

(a) Create the application ontology;
(b) Create agent behaviors;
(c) Create agent types, realizing the created behaviors;
(d) Perform data mining on agent type-specific datasets;
(e) Generate knowledge models for each agent type;
(f) Create the agents of the application (of the different agent types);
(g) Incorporate the extracted knowledge models into the corresponding agents;
(h) Instantiate the MAS;
(i) Monitor agents;
(j) Periodically retrain the agents of the MAS.

Let \( O \) be the ontology of the MAS. Let \( A = \{A_1, A_2, \ldots, A_n\} \) be the set of attributes described in \( O \) and defined on \( D \), the application data domain. Let \( D \subseteq D \) be a set of application data, where each dataset tuple is a vector \( t = \{t_1, t_2, \ldots, t_n\} \), and \( t_i, i = 1, \ldots, n \) is a value for the corresponding attribute \( A_i \). Missing values are allowed within \( t \).

In order to initially train a certain type \( A_{gi}, i = 1, \ldots, k \) of application agents, we use a subset of the application dataset, containing the attributes that are relevant to this specific type. We therefore define \( D_{IAT} \subseteq D_{IT} \), where \( D_{IAT} \) is the initial training dataset for agent type \( A_{gi} \), and \( D_{IT} \) is the initial application dataset. In most cases \( D_{IT} = D \). For each \( A_{gi} \), we perform data mining on the corresponding dataset \( D_{IAT} \) in order to extract a useful knowledge model \( KM_{j}(o = 1, \ldots, p) \) and incorporate it into all \( A_{gj}, j = 1, \ldots, n \), the \( A_{gj} \) type agents of the MAS. We then instantiate the MAS and monitor its agents.

In the retraining phase, each agent can be retrained individually. The available datasets include: the initial dataset
$D_{IT}$, a new non-agent dataset\(^1\) $D_{NAgi}$, and all the datasets $D_{Agj}(j)$, each containing the tuples representing the actions (decisions) taken by the respective agent. It must be denoted that $D_{Agj} = D_{Agj}(1) \oplus D_{Agj}(2) \oplus \ldots \oplus D_{Agj}(m)$.

The symbol $\oplus$ represents the concatenation of two datasets, an operation that preserves multiple copies of tuples. There are five different options of agent retraining, with respect to the datasets used:

(A) $D_{lAgj} \oplus D_{NAgi}$. Retrain the agent using the initial dataset along with a new, non-agent dataset $D_{NAgi}$.

(B) $D_{NAgi} \oplus D_{Agj}$. Retrain the agent using a non-agent dataset $D_{NAgi}$ along with $D_{Agj}$, a dataset generated by all the $Agj$-type agents of the application. AA agents are monitored and their actions are recorded, in order to construct the $D_{Agj}$ dataset.

(C) $D_{lAgj} \oplus D_{NAgi} \oplus D_{Agj}$. Retrain the agent using all the available datasets.

(D) $D_{lAgj} \oplus D_{Agj}$. Use the initial dataset $D_{lAgj}$ along with the agent generated data.

(E) $D_{lAgj} \oplus D_{Agj}(j)$. Use the initial dataset $D_{lAgj}$ along with $D_{Agj}(j)$, the generated data of the $j$th agent.

A schematic representation of the training and retraining procedure is given in Fig. 1.

Through AA and its training/retraining capabilities the user can formulate and augment agents’ intelligence. AA supports a variety of both supervised (classification) and unsupervised learning (clustering, association rule extraction) DM techniques, shown in Table 1.

### 3. The training and retraining mechanism

In order to enable the incorporation of knowledge into agents, we have implemented Data Miner as an agent-oriented tool. It is a DM suite that supports the application of a variety of Classification, Clustering and Association Rule Extraction algorithms on application-specific and agent-behavior-specific data (Table 1). Data Miner can also incorporate the extracted decision models into the AF produced agents, augmenting that way their intelligence. Apart from being a core component of the AA platform, the Data Miner can also function as a standalone DM tool.

The mechanism for embedding rule-based reasoning capabilities into agents is illustrated in Fig. 2.

Data, either application-specific or agent-behavior-specific, enter the module in XML format. Each data file contains information on the name of the agent the file belongs to and on the decision structure of the agent it will be applied on. The XML file is then inserted into the Preprocessing Unit of the Data Miner, where all the necessary data selection and data cleaning tasks take place. Next, data are forwarded to the Miner, where the user decides on the DM technique, as well as on the specific algorithm to employ. After DM is performed, the results are sent to the Evaluator, which is responsible for the validation and visualization of the extracted model. If the user accepts the constructed model, a PMML document describing the knowledge model is generated. This document expresses the referencing mechanism of the agent we intend to train. The resulting decision model is then translated to a set of facts executed by a rule engine. The implementation of the rule engine is realized through the Java Expert System Shell (JESS) [12], which is a robust mechanism for executing rule-based agent reasoning. The execution of the rule engine transforms the Data Miner extracted knowledge into a living part of the agent’s behavior.

After the MAS has been instantiated, the user has the ability to monitor AA agents and their decisions. These decisions are stored into the AUR. For agent $j$, data stored in the AUR constitute the $D_{Agj}(j)$ dataset. The user can then decide, as mentioned in Section 2, on the dataset s/he would like to perform retraining on.

### 4. Augmenting agent intelligence

#### 4.1. Different retraining approaches

Retraining is performed in order to either increase or refine agent intelligence. By reapplying data mining on a new or more complete dataset, the user expects to derive more accurate patterns and more efficient associations.

The five retraining options with respect to the available datasets, can be classified into two main approaches: a) the type-oriented, which deals with the augmentation of intelligence of $Ag$, all the type-i agents (options A–D) and, b) the agent-oriented, which focuses on the refinement of intelligence of an individual agent $Ag(i)$, the $i$th agent of type $i$ (option E).

It should also be denoted that we differentiate on the way we define “intelligence improvement”, since AA provides both supervised and unsupervised learning DM techniques. In the case of classification, improvement can be measured by evaluating the knowledge model extracted metrics (mean-square error, accuracy, etc.), while in the case of clustering and association rule extraction intelligence augmentation is determined by external evaluation functions.

The classification algorithms provided by the AA platform are decision tree (DT) extraction algorithms. The basic prerequisites for the proper application of a DT construction algorithm are the existence of a distinct set of classes and the availability of training data. All the DT algorithms supported by the AA platform are criterion gain algorithms, i.e., algorithms that decide on the construction of the DT, according to the minimization (or maximization) of a certain criterion. In the case of ID3 and C4.5, this criterion is the information gain [21], in the case of CLS, it is record sorting [14], and in the case of FLR, the criterion is the inclusion measure [16].

---

\(^1\) We define a non-agent dataset, as the dataset that contains information on the actions of agents, but has not been produced by them (probably data come from a pre-stored application dataset).
The clustering algorithms provided by AA are partitioning algorithms (PAs). The objective of PA algorithms is the grouping of the data provided into discrete clusters. Data must have high intra-cluster and low inter-cluster similarity. PA algorithms' splitting criterion is the Euclidean distance between data [18].

Finally, the association rule extraction (ARE) algorithms provided by AA are mainly focused on transactional datasets. AREs attempt to discover, as their name implies, associations between items. In order for these algorithms to decide on the strongest associations, two metrics are considered: support and confidence [3].

4.2. Training and retraining in the case of supervised learning

Although the splitting criteria are different, all of the above mentioned classification algorithms are applied in a similar manner. We may focus on the information gain criterion that is employed by the C4.5 and ID3 algorithms, nevertheless the approach followed can be easily adjusted to other classification algorithms of the platform.

The information gain expected when splitting dataset \( D \) with respect to attribute \( A_i \), \( A_i \in A \) is given by

\[
\text{Gain}(D, A_i) = \text{Info}(D) - \text{Info}(D, A_i)
\]  

(1)
**4.2.1. Initial training**

When training takes place, classification is performed on $D_{IAgi}$, the initial dataset for the specific agent type. The user can decide to split the dataset into a training and a testing (and/or validation) dataset or to perform $n$-fold cross-validation. To evaluate the success of the applied classification scheme, a number of statistical measures are calculated, i.e., classification accuracy, mean absolute error and confusion matrix. If extracted knowledge model is deemed satisfactory, the user may accept it and store it, for incorporation into the corresponding $Ag_i$-type agents.

**4.2.2. Retraining $Ag_i$**

In the case of retraining agent-type $Ag_i$, the relevant datasets are $D_{IAgi}$, $D_{NAgi}$ and $D_{Ag_i}$. Retraining option $C$ ($D_{IAgi} \oplus D_{NAgi} \oplus D_{Ag_i}$) is the most general, containing all the available data for the specific agent type, while options $A$ and $D$ are subsets of option $C$. They are differentiated, however, since option $D$ is particularly interesting and deserves special attention.

When using datasets $D_{IAgi}$ and $D_{NAgi}$, the user may choose among the different retraining options illustrated in Table 2.

The user decides on which knowledge model to accept, based on its performance. Nevertheless, in the $D_{IAgi} \oplus D_{NAgi}$ case, best model performance is usually observed when option 3 is selected. The inductive nature of classification dictates that the use of larger training datasets leads to more efficient knowledge models.

The retraining options when the $D_{NAgi} \oplus D_{Ag_i}$ dataset is selected are illustrated in Table 3.

When retraining an agent with the $D_{NAgi} \oplus D_{Ag_i}$ dataset, it is important to notice that the only information we have on the training dataset $D_{IAgi}$ is indirect, since $D_{Ag_i}$ is formatted based on the knowledge model the agents follow, a model inducted by the $D_{IAgi}$ dataset. This is why the val-

\[ Info(D) = - \sum_{i=1}^{o} p(i) \log_2 p(i) \]  \hspace{1cm} (2)

with $p(i)$ the ratio of $D$ tuples that belong to class $C_i$.

\[ Info(D, A_i) = \sum_{j=1}^{v} \left( \frac{|D_j|}{|D|} \times Info(D_j) \right) \]  \hspace{1cm} (3)

Splitting is conducted on the attribute that yields the maximum information gain.

![Diagram of the agent training/retraining mechanism](image-url)
and have established these agents in different cities in Greece (Athens, Thessaloniki, Patra, Chania, etc.). Although all these agents rely initially on a common knowledge model, weather conditions in Thessaloniki differ from those in Chania enough to justify refined knowledge models.

In this case, we have the options to perform agent-type retraining. By the use of the $D_{IAg_i} \ominus D_{Ag_i}$ dataset, it is possible to refine the intelligence of the $j$th agent of type $i$. High frequency occurrence of a certain value $t_i$ of attribute $A_i$ (i.e., “High” humidity in Thessaloniki, “Sunny” outlook in Chania) may produce a more “case-specific” knowledge model. In a similar to the $D_{IAg_i} \ominus D_{Ag_i}$ manner, it can be seen that an increase of $Info(D, A_i)$ can lead to a different knowledge model, which incorporates instance-specific information.

The analysis of different retraining options in the case of Classification indicates that there exist concrete success metrics that can be used to evaluate the extracted knowledge models and, thus, may ensure the improvement of agent intelligence.

### 4.3. Training and retraining in the case of unsupervised learning

In the case of unsupervised learning, training and retraining success cannot be determined quantitatively. A more qualitative approach must be followed, to determine the efficiency of the extracted knowledge model, with respect to the overall goals of the deployed MAS.

#### 4.3.1. Initial training

To perform clustering, the user can either split the $D_{IAg_i}$ dataset into a training and a testing subset or perform a classes-to-clusters evaluation, by testing the extracted clusters with respect to a class attribute defined in $D_{IAg_i}$. In order to evaluate the success of the clustering scheme, the mean square error and standard deviation of each cluster center are calculated. One the other hand, if the user decides to perform ARE on $D_{IAg_i}$ no training options are provided. Only the algorithm-specific metrics are specified and ARE is performed. In a similar to classification manner, if the extracted knowledge model (clusters, association rules) is favorably evaluated, it is stored and incorporated into the corresponding $Ag_i$-type agents.

### Table 5

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Causality</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{IAg_i}$</td>
<td>$D_{Ag_i}$</td>
</tr>
<tr>
<td>Option C-1</td>
<td>Training</td>
</tr>
<tr>
<td>Option C-2</td>
<td>Testing</td>
</tr>
<tr>
<td>Option C-3</td>
<td>Concatenation and training</td>
</tr>
<tr>
<td>Option C-4</td>
<td>Concatenation and cross-validation</td>
</tr>
</tbody>
</table>
4.3.2. Retraining by clustering

Clustering results are in most cases indirectly applied to the deployed MAS. In practice, some kind of an external exploitation function is developed, which somehow fires different agent actions in the case of different clusters. All the available datasets \((D_{Agi}, D_{Nagi}, D_{Agj}, \text{and } D_{Agj}(j))\) can therefore be used for both training and testing for Initial model validation, Model Data dependency investigation and New Knowledge Model discovery. A larger training dataset and more thorough testing can lead to more accurate clustering. Often retraining can result in the dynamic updating and encapsulation of dataset trends (i.e., in the case of customer segmentation). Retraining \(A_i(j)\) can therefore be defined as a “case-specific” instance of retraining, where data provided by agent \(j\), \(D_{Agj}(j)\), are used for own improvement.

4.3.3. Retraining by association rule extraction

The ARE technique does not provide training and testing options. The whole input dataset is used for the extraction of the strongest association rules. Consequently, all available datasets \((D_{Agi}, D_{Nagi}, D_{Agj}, \text{and } D_{Agj}(j))\) are concatenated before DM is performed. This unified approach for retraining has a sole goal: to discover the strongest association rules between the items \(t\) of \(D\). In a similar to the clustering case manner, retraining \(A_i(j)\) can be viewed as a “case-specific” instance of retraining.

5. Experimental results

In order to prove the added value of agent retraining, a number of experiments on Classification, Clustering and ARE were conducted. In this section, three representatives cases are discussed. These experiments are focused mainly on retraining by the use of the \(D_{Agi}\) and \(D_{Agj}(j)\) datasets and illustrate the enhancement of agent intelligence.

5.1. Intelligent environmental monitoring system

The first experiment was performed for the O3 RTAA System, an agent-based intelligent environmental monitoring system developed for assessing ambient air-quality [4]. A community of software agents is assigned to monitor and validate multi-sensor data, to assess air-quality, and, finally, to fire alarms to appropriate recipients, when needed. Data mining techniques have been used for adding data-driven, customized intelligence into agents with successful results [16].

In this work we focused on the Diagnosis Agent Type. Agents of this type are responsible for monitoring various air quality attributes including pollutants’ emissions and meteorological attributes. Each one of the Diagnosis Agent instances is assigned to monitor one attribute through the corresponding field sensor. In the case of sensor breakdown, Diagnosis Agents take control and perform an estimation of the missing sensor values using a data-driven Reasoning Engine, which exploits DM techniques.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>(D_{Agi})</th>
<th>(D_{Agj})</th>
<th>(D_{Val})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of instances</td>
<td>11,641</td>
<td>10,000</td>
<td>7414</td>
</tr>
<tr>
<td>Initial training</td>
<td>Used</td>
<td>73.58%</td>
<td>71.89%</td>
</tr>
<tr>
<td>Retraining</td>
<td>Used</td>
<td>74.66%</td>
<td></td>
</tr>
</tbody>
</table>

One of the Diagnosis Agents is responsible for estimating missing ozone measurement values. This task is accomplished using a predictive model comprised of the predictors and the response. For the estimation of missing ozone values the predictors are the current values measured by the rest of the sensors, while the response is the level of the missing value (Low, Medium, or High). In this way, the problem has been formed as a classification task.

For training and retraining the Ozone Diagnosis Agent we used a dataset, labeled \(C2ONDAM01\) and supplied by CEAM, which contained data from a meteorological station in the district of Valencia, Spain. Several meteorological attributes and air-pollutant values were recorded on a quarter-hourly basis during the year 2001. There are approximately 35,000 records, with ten attributes per record plus the class attribute. The dataset was split into three subsets: one subset for initial training \((D_{Agi})\), a second subset for agent testing \((D_{Agj})\) and another subset for validation \((D_{Val})\) containing around 40%, 35% and 25% of the data, respectively.

The initial training of the Diagnosis Agent was conducted using Quinlan’s C4.5 [21] algorithm for decision tree induction, using the \(D_{Agi}\) subset. This decision tree was embedded in the Diagnosis Agent and the agent used it for deciding on the records of the \(D_{Agj}\) Subset. Agent decisions along with the initial application data were used for retraining the Diagnosis Agent (Option D: \(D_{Agi} \odot D_{Agj}\)). Finally, the Diagnosis Agent with the updated decision tree was used for deciding on the cases of the last subset \((D_{Val})\).

The retrained Diagnosis Agent performed much better compared to the initial training model, are shown in Table 6. The use of agent decisions included in \(D_{Agj}\) has enhanced the Diagnosis Agent performance on the \(D_{Val}\) subset by 3.65%.

5.2. Speech recognition agents

This experiment was based on the “vowel” dataset of the UCI repository [24]. The problem in this case is to recognize a vowel spoken by an arbitrary speaker. This dataset is comprised of ten continuous primary features (derived from spectral data) and two discrete contextual features (the speaker’s identity and sex) and contains records for 15 speakers. The observations fall into eleven classes (eleven different vowels).

The vowel problem was assigned to an agent community to solve. Two agents \(Ag_i(1)\) and \(Ag_i(2)\) were deployed to recognize vowels. Although of the same type, the two
Again, retraining with the $D_{IAgj} \oplus D_{Agj}$ dataset leads to the improvement of clustering results.

The new knowledge models obtained with the above retraining options can be easily incorporated into agents following the already implemented training/retraining mechanism, which is described next.

### 6. Conclusions

Work presented in this paper explains how DM techniques can be successfully coupled with AT, leading to dynamically created agent intelligence. Moreover, the concepts of training and retraining are formulated and special focus is given on retraining, the recursive process of “recalling” an agent for posterior training. Through this procedure, where DM is performed on new datasets ($D_{NAgi}$, $D_{Agj}$ and $D_{Agj}(\overline{j})$), refined knowledge is extracted and dynamically embedded into the agents. The different retraining options in the cases of Supervised and Unsupervised Learning are outlined in this paper and experimental results on different types of retraining are provided. Finally, the training and retraining mechanism is presented. Based on our research work we strongly believe that data mining extracted knowledge could and should be coupled with agent technology, and that training and retraining can indeed lead to more intelligent agents.

### Acknowledgement

Work presented here has been partially supported by the European Commission through the IST initiative (IST project No. 2000-31050).

### References


