A retraining methodology for enhancing agent intelligence

Article in Knowledge-Based Systems · January 2005
DOI: 10.1016/j.knosys.2006.06.003 · Source: DBLP

3 authors:

Andreas Symeonidis
Aristotle University of Thessaloniki
95 PUBLICATIONS 413 CITATIONS
SEE PROFILE

Ioannis N. Athanasiadis
Wageningen University & Research
124 PUBLICATIONS 1,216 CITATIONS
SEE PROFILE

Pericles A. Mitkas
Aristotle University of Thessaloniki
272 PUBLICATIONS 1,635 CITATIONS
SEE PROFILE

All content following this page was uploaded by Andreas Symeonidis on 08 January 2014.
The user has requested enhancement of the downloaded file.
A retraining methodology for enhancing agent intelligence

Andreas L. Symeonidis *, Ioannis N. Athanasiadis, Pericles A. Mitkas

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

Received 5 August 2004; accepted 3 June 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 [1,14,26], 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
researchers towards the employment of Agent Technology (AT) in a variety of disciplines [15,27]. The versatility and generic nature of the multi-agent technology paradigm has indicated that problems which are inherently distributed or require the synergy of a number of distributed elements for their solution can be efficiently implemented as a multi-agent system (MAS) [9].

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 of AA, a DM suite that expands the Waikato Environment for Knowledge Analysis (WEKA) tool [26]. 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 $A_g$, $i = 1, \ldots, k$ of application agents, we use a subset of the application dataset, containing the attributes that are relevant to this specific type. We therefore define $D_{Iag} \subseteq D_{IT}$, where $D_{Iag}$ is the initial training dataset for agent type $A_g$, and $D_{IT}$ is the initial application dataset. In most cases $D_{IT} = D$. For each $A_g$, we perform data mining on the corresponding dataset $D_{Iag}$ in order to extract a useful knowledge model $KM_A(o = 1, \ldots, p)$ and incorporate it into all $A_g(j)$, $j = 1, \ldots, m$, the $Ag$-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
170 \( D_{TF} \), a new non-agent dataset \( D_{N,Ag} \), and all the datasets
171 \( D_{Ag}(j) \), each containing the tuples representing the actions
172 (decisions) taken by the respective agent. It must be denot-
173 ed that \( D_{Ag} = D_{Ag}(1) \oplus D_{Ag}(2) \oplus \cdots \oplus D_{Ag}(m) \). The
174 symbol \( \oplus \) represents the concatenation of two datasets,
175 an operation that preserves multiple copies of tuples. There
176 are five different options of agent retraining, with respect to
177 the datasets used:

178 (A) \( D_{TAgi} \oplus D_{N,Ag} \). Retrain the agent using the initial
179 dataset along with a new, non-agent dataset \( D_{N,Ag} \).
180 (B) \( D_{N,Ag} \oplus D_{Ag} \). Retrain the agent using a non-agent
181 dataset \( D_{N,Ag} \) along with \( D_{Ag} \), a dataset generated
182 by all the \( Ag \)-type agents of the application. AA
183 agents are monitored and their actions are recorded,
184 in order to construct the \( D_{Ag} \) dataset.
185 (C) \( D_{TAgi} \oplus D_{N,Ag} \oplus D_{Ag} \). Retrain the agent using all the
186 available datasets.
187 (D) \( D_{TAgi} \oplus D_{Ag} \). Use the initial dataset \( D_{TAgi} \) along with
188 the agent generated data.
189 (E) \( D_{TAgi} \oplus D_{Ag}(j) \). Use the initial dataset \( D_{TAgi} \) along
190 with \( D_{Ag}(j) \), the generated data of the \( j \)th agent.

A schematic representation of the training and retrain-
191 ing procedure is given in Fig. 1:

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

3. The training and retraining mechanism

In order to enable the incorporation of knowledge into
197 agents, we have implemented Data Miner as an agent-orien-
198ted tool. It is a DM suite that supports the application of
199 a variety of Classification, Clustering and Association Rule
200 Extraction algorithms on application-specific and agent-
201 behavior-specific data (Table 1). Data Miner can also
202 incorporate the extracted decision models into the AA pro-
203 duced agents, augmenting that way their intelligence.
204 Apart from being a core component of the AA platform,
205 the Data Miner can also function as a standalone DM tool.
206 The mechanism for embedding rule-based reasoning
207 capabilities into agents is illustrated in Fig. 2.

Data, either application-specific or agent-behavior-spe-
208 cific, enter the module in XML format. Each data file con-
209 tains information on the name of the agent the file belongs
210 to and on the decision structure of the agent it will be
211 applied on. The XML file is then inserted into the Prepro-
212 cessing Unit of the Data Miner, where all the necessary
213 data selection and data cleaning tasks take place. Next, the
214 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 \( D_{Ag}(j) \) dataset. The user can
then decide, as mentioned in Section 2, on the dataset s/235
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 intelli-
genence of \( Ag_{j} \), all the type-\( i \)-agents (options A-D) and, b) the
agent-oriented, which focuses on the refinement of intelli-
genence of an individual agent \( Ag_{j}(j) \), the \( j \)th agent of type \( i \)
(option E).

It should also be denoted that we differentiate on the
way we define “intelligence improvement”, since AA pro-
vides both supervised and unsupervised learning DM tech-
niques. 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 Intelli-
genence augmentation is determined by external evaluation
functions.

The classification algorithms provided by the AA plat-
form are decision tree (DT) extraction algorithms. The
basic prerequisites for the proper application of a DT con-
struction 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 maximiza-
tion) 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].

We define a non-agent dataset, as the dataset that contains informa-
tion 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

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

Table 1  
DM provided techniques and algorithms

<table>
<thead>
<tr>
<th>DM technique</th>
<th>DM technique</th>
<th>DM technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Association rules</td>
<td>Clustering</td>
</tr>
<tr>
<td>ID 3</td>
<td>Apriori</td>
<td>$K$-means</td>
</tr>
<tr>
<td>C 4.5</td>
<td>DHJ</td>
<td>PAM</td>
</tr>
<tr>
<td>CLS</td>
<td>DIC</td>
<td>EM</td>
</tr>
<tr>
<td>DLR</td>
<td>$K$-Profile</td>
<td></td>
</tr>
</tbody>
</table>

*a The FLR and $K$-Profile algorithms are novel algorithms, developed within the context of Agent Academy. More information on these algorithms can be found at (Kaburlasos et al. 2003, Athanasiadis et al. 2003) [2].

Fig. 1. Training and retraining the agents of a MAS.
4.2.1. Initial training

When training takes place, classification is performed on $D_{I_{Ag_i}}$, 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 $D_{I_{Ag_i}}$, $D_{N_{Ag_i}}$ and $D_{Ag_i}$. Retraining option $C$ ($D_{I_{Ag_i}} \oplus D_{N_{Ag_i}} \oplus D_{Ag_i}$) 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 $D_{I_{Ag_i}}$ and $D_{N_{Ag_i}}$, the user may choose among the different retraining options illustrated in Table 2:

<table>
<thead>
<tr>
<th>Retraining options when the $D_{N_{Ag_i}} \oplus D_{Ag_i}$ dataset is selected</th>
<th>Causality</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{I_{Ag_i}}$</td>
<td>Initial model validation</td>
</tr>
<tr>
<td>$D_{N_{Ag_i}}$</td>
<td>Model investigation on data independency</td>
</tr>
<tr>
<td>$D_{Ag_i}$</td>
<td>New knowledge model discovery</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Retraining options for $D_{N_{Ag_i}} \oplus D_{Ag_i}$ dataset</th>
<th>Causality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concatenation and cross-validation</td>
<td>Indirect initial model validation</td>
</tr>
<tr>
<td>Concatenation and cross-validation</td>
<td>New knowledge model discovery</td>
</tr>
</tbody>
</table>

![Diagram of the agent training/retraining mechanism](image.png)

Fig. 2. The agent training/retraining mechanism.

$$Info(D) = -\sum_{i=1}^{o} p(i) \log_2 p(i)$$

$$Info(D, A_i) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \cdot Info(D_j)$$

$Info(D)$ is the information needed to classify $D$ with respect to the predefined distinct classes $C_i$ (for $i=1, \ldots, o$), and is given by

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

$Info(D, A_i)$ is the information needed in order to classify $D$, after its partitioning into subsets $D_j$, $j=1, \ldots, v$, with respect to the attribute $A_i$, $Info(D, A_i)$, which is also denoted as the Entropy of $A_i$, is given by

$Info(D, A_i)$ 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 $D_{I_{Ag_i}}$ and $D_{N_{Ag_i}}$, the user may choose among the different retraining options illustrated in Table 2:

<table>
<thead>
<tr>
<th>Retraining options when the $D_{N_{Ag_i}} \oplus D_{Ag_i}$ dataset is selected</th>
<th>Causality</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{I_{Ag_i}}$</td>
<td>Initial model validation</td>
</tr>
<tr>
<td>$D_{N_{Ag_i}}$</td>
<td>Model investigation on data independency</td>
</tr>
<tr>
<td>$D_{Ag_i}$</td>
<td>New knowledge model discovery</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Retraining options for $D_{N_{Ag_i}} \oplus D_{Ag_i}$ dataset</th>
<th>Causality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concatenation and cross-validation</td>
<td>Indirect initial model validation</td>
</tr>
<tr>
<td>Concatenation and cross-validation</td>
<td>New knowledge model discovery</td>
</tr>
</tbody>
</table>
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 $D_{T_Agi}$ 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 $D_{T_Agi}$. 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 $D_{T_Agi}$, 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 $Ag_{tf}$-type agents.

Table 4

<table>
<thead>
<tr>
<th>Retraining options for $D_{T_Agi} \oplus D_{Ag_i}$</th>
<th>Dataset</th>
<th>Causality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concatenation and cross-validation</td>
<td>$D_{T_Agi}$</td>
<td>$D_{Ag_i}$</td>
</tr>
<tr>
<td>More application-efficient knowledge model</td>
<td>$D_{T_Agi}$</td>
<td>$D_{Ag_i}$</td>
</tr>
</tbody>
</table>

Table 5

<table>
<thead>
<tr>
<th>Retraining options for $D_{T_Agi} \oplus D_{N_Agi} \oplus D_{Ag_i}$</th>
<th>Dataset</th>
<th>Causality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concatenation and training</td>
<td>$D_{T_Agi}$</td>
<td>$D_{Ag_i}$</td>
</tr>
<tr>
<td>Concatenation and cross-validation</td>
<td>$D_{T_Agi}$</td>
<td>$D_{N_Agi}$</td>
</tr>
<tr>
<td>Testing</td>
<td>$D_{Ag_i}$</td>
<td>$D_{Ag_i}$</td>
</tr>
</tbody>
</table>

Please cite this article in press as: A.L. Symeonidis et al., A retraining methodology for enhancing agent intelligence, Knowl. Based Syst. (2006), doi:10.1016/j.knosys.2006.06.003
439 4.3.2. Retraining by clustering
440 Clustering results are in most cases indirectly applied to
441 the deployed MAS. In practice, some kind of an external
442 exploitation function is developed, which somehow fires
443 different agent actions in the case of different clusters. All
444 the available datasets  \( \{D_{\text{Ag}_1}, D_{\text{Ag}_2}, D_{\text{Ag}_3}\} \) can
445 therefore be used for both training and testing for Initial
446 model validation, Model Data dependency investigation
447 and New Knowledge Model discovery. A larger training
448 dataset and more thorough testing can lead to more accu-
449 rate clustering. Often retraining can result in the dynamic
450 updating and encapsulation of dataset trends (i.e., in the
451 case of customer segmentation). Retraining \( A_j \) can there-
452 fore be defined as a “case-specific” instance of retraining,
453 where data provided by agent \( j, D_{\text{Ag}_j} \), are used for own
454 improvement.

455 4.3.3. Retraining by association rule extraction
456 The ARE technique does not provide training and test-
457 ing options. The whole input dataset is used for the extrac-
458 tion of the strongest association rules. Consequently, all
459 available datasets  \( \{D_{\text{Ag}_1}, D_{\text{Ag}_2}, D_{\text{Ag}_3}\} \) are con-
460 catenated before DM is performed. This unified approach
461 for retraining has a sole goal: to discover the strongest
462 association rules between the items \( t \) of \( D \). In a similar to
463 the clustering case manner, retraining \( A_j \) can be viewed
464 as a “case-specific” instance of retraining.

5. Experimental results
5.1. Intelligent environmental monitoring system
5.2. Speech recognition agents
5.2.1. The experiment was performed for the O\(_3\)RTAA
5.2.2. Slovenian-based intelligent environmental monitoring
5.2.3. system developed for assessing ambient air-quality [4].
5.2.4. A community of software agents is assigned to monitor
5.2.5. and validate multi-sensor data, to assess air-quality, and, and,
5.2.6. finally, to fire alarms to appropriate recipients, when need-
5.2.7. ed. Data mining techniques have been used for adding
5.2.8. data-driven, customized intelligence into agents with suc-
5.2.9. cessful results [16].
5.2.10. In this work we focused on the Diagnosis Agent Type.
5.2.11. Agents of this type are responsible for monitoring various
5.2.12. air quality attributes including pollutants’ emissions and
5.2.13. meteorological attributes. Each one of the Diagnosis Agent
5.2.14. instances is assigned to monitor one attribute through the
5.2.15. corresponding field sensor. In the case of sensor break-
5.2.16. down, Diagnosis Agents take control and perform an esti-
5.2.17. mation of the missing sensor values using a data-driven
5.2.18. Reasoning Engine, which exploits DM techniques.

Table 6
Classification accuracies for the Diagnosis Agent

<table>
<thead>
<tr>
<th>Dataset</th>
<th>( D_{\text{Ag}_1} )</th>
<th>( D_{\text{Ag}_2} )</th>
<th>( D_{\text{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>74.66%</td>
</tr>
</tbody>
</table>

One of the Diagnosis Agents is responsible for estimating missing ozone measurement values. This task is accom-
plished 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 \( C2Onda01 \) and supplied by
CEAM, which contained data from a meteorological station
in the district of Valencia, Spain. Several meteorolog-
ical 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 (\( D_{\text{IAg}_1} \)), a sec-
ond subset for agent testing (\( D_{\text{Ag}_1} \)) and another subset for
validation (\( D_{\text{Val}} \)) containing around 40%, 35% and 25%
of the data, respectively.

The initial training of the Diagnosis Agent was conduct-
ed using Quinlan’s C4.5 [21] algorithm for decision tree
induction, using the \( D_{\text{IAg}_1} \) subset. This decision tree was
embedded in the Diagnosis Agent and the agent used it
for deciding on the records of the \( D_{\text{Ag}_1} \) subset. Agent deci-
sions along with the initial application data were used for
retraining the Diagnosis Agent (Option D: \( D_{\text{IAg}_1} \subset D_{\text{Ag}_1} \)).
Finally, the Diagnosis Agent with the updated decision tree
was used for deciding on the cases of the last subset (\( D_{\text{Val}} \)).

The retrained Diagnosis Agent performed much better
compared to the initial training model, as shown in
Table 6. The use of agent decisions included in \( D_{\text{Ag}_1} \) has
e nhanced the Diagnosis Agent performance on the \( D_{\text{Val}} \)
subset by 3.65%.

5.2. Speech recognition agents

This experiment was based on the “vowel” dataset of
the UCI repository [25]. The problem in this case is to rec-
ognize a vowel spoken by an arbitrary speaker. This data-
set 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 \( Ag_1 \) and \( Ag_2 \) were deployed to recog-
nize vowels. Although of the same type, the two
Again, retraining with the $D_{\text{Int}} \oplus D_{\text{Ag}}$ 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 ($D_{\text{Int}}$, $D_{\text{Ag}}$, and $D_{\text{Ag}}(j)$), 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.

7. Uncited reference

[23].

Acknowledgement

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

References


5.3. The iris recommendation agent

In order to investigate retraining in the case of clustering, we used the Iris UCI Dataset [25], a dataset widely used in pattern recognition literature. It has four numeric attributes describing the iris plant and one nominal attribute describing its class. The 150 records of the set were split into two subsets: one subset (75%) for initial training ($D_{\text{Int}}$) and a second subset (25%) for agent testing ($D_{\text{Ag}}$). Classes-to-clusters evaluation was performed on $D_{\text{Int}}$ and $D_{\text{Int}} \oplus D_{\text{Ag}}$ (Option D) and the performance of the resulted clusters was compared on the number of correctly classified instances of the dataset (Table 8).


