



1 ELA—A new Approach for Learning Agents*

2 FABRÍCIO ENEMBRECK

enembrec@hds.utc.fr

4 JEAN-PAUL BARTHÈS

barthes@hds.utc.fr

6 UTC, Université de Technologie de Compiègne, Centre de Recherches Royallieu, 60200 Compiègne, France

7 **Abstract.** In this paper we discuss a new incremental learning approach used to implement adaptive
8 behavior in autonomous agents. Adaptive agents must increase their performance based on experience
9 using some learning approach. Often, incremental learning techniques like memory-based reasoning
10 (MBR) are used. However, traditional MBR algorithms require an adequate (generally complex) measure
11 of similarity, need much data and spend much time for computing similarities between examples. Such
12 problems are unacceptable for autonomous agents that live in very dynamic environments, because they
13 have little time to make decisions. Our approach does not use similarity measures between examples,
14 classifies examples very fast and can compact data. We represent data as a concept graph (CG), each node
15 representing a partition of the data. We propose an algorithm that uses the partitions to classify new
16 examples. We compare our results with other techniques and conclude that the method performs quite
17 well. Finally, we apply the approach to an application of adaptive agents for personalizing web search.

18 **Keywords:** adaptive agents, incremental learning, memory-based reasoning.

19

20 1. Introduction

21 The development of systems capable of increasing their performance using past
22 experiences is a well-known challenge of AI. Such adaptivity seems crucial for au-
23 tonomous systems because it can save resources and lead to more robust systems.
24 Autonomous agents that present such behavior are named *adaptive agents*.

25 Adaptive autonomous agents are capable of adapting their behavior according to
26 changes in the environment. Thus, they can act in unexpected situations and improve
27 their performance. For instance, if we consider a robot similar to that used by Fikes
28 and Nilsson [17] we immediately perceive the utility of learning. The robot must
29 continuously cross ten rooms looking for boxes. It perceives after some iterations
30 that rooms 1 – 3 are always empty. It can then decide to decrease the frequency of
31 visiting such rooms. Thus, the robot increases the speed of execution of its task,
32 spends less energy and visits the other rooms more frequently.

33 In this paper we discuss learning approaches used to implement adaptive behavior
34 in adaptive agents. Thus, we look for a learning method that allows the agent to
35 learn without requiring many calculations. The agent must learn concepts dyna-
36 mically using information received from the environment. Moreover, this method
37 must be sufficiently generic that any agent can use it in any field. To solve this
38 problem, we developed a new memory-based reasoning (MBR) framework, trying to
39 avoid the common problems of MBR methods like similarity measures ambiguity,

*This research was funded by Région Picardie in France. We thank Emerson Paraiso and Cesar Tacla for comments and time spent with the discussions.



computational complexity and data space requirements. In the next section we present an overview of adaptive agents. In Section 3 we discuss our approach, entropy-based learning approach (ELA). Performance evaluation is done in Section 4 using published results and a evaluation using MLC++ is presented in Section 5. Section 6 presents a discussion about related work. We discuss in Section 7 an application based on adaptive agents that use ELA for personalizing results from a web search. We show that ELA can be used successfully in common applications of learning agents, like web search, producing quite good results. Finally, Section 8 concludes the paper.

2. Adaptive agents

In some applications, the environment does not change simply on the level of events, but also on the level of the structures. When this happens, either the agent is coded again to be able to continue to survive in the environment, or an adaptation mechanism is used. Maes [33] did one of the early studies highlighting the use of automatic learning for the autonomous agents. According to the author, learning algorithms for adaptive autonomous agents must have the following characteristics:

- Learning must be incremental;
- Learning must take the noise into account;
- Learning must be unsupervised; and
- The algorithm can possibly use the background knowledge provided by the user and/or the developer of the system.

According to Maes [33] and Brenner [9], three learning techniques are usable to develop adaptive autonomous agents: *reinforcement learning*, *models learning* and *classifier systems*.

2.1. Reinforcement learning

Watkins [50] developed the Q-learning algorithm for solving reinforcement learning problems. In such class of problems, the mechanism consists in assigning rewards (weights) to actions that contribute to the resolution of the problem. Let us suppose that the environment where an agent is present is the state s . At the beginning, the weights corresponding to the various actions are unknown. After applying a given action a , the environment changes to a state $s1$. As a consequence, the agent calculates the reward $Q(s, a)$. The solution of the problem consists in determining for each state s , the value $Q(s,a)$ which maximizes the accumulated reward.

Oliveira [41] uses reinforcement learning to improve coordination between autonomous agents. In his approach, the agents learn the best negotiation model. Vault, Simonin and Ferber [11] use a technique of rewards that deals with the satisfaction of the neighboring agents in an application in which agents must be coordinated for transporting objects.

79 2.2. *Models learning*

80 The agents that use model learning try to find causal relations between their
81 actions and the events occurring in the environment. In general, they use either
82 probabilistic models or logical models. Maes proposed a behavior network for
83 modeling the procedure of action selection for autonomous agents in a dynamic
84 and unforeseeable environment [32]. Her original ideas were later developed by
85 several authors (see [13]). In a behavior network, each node represents a possible
86 behavior (action) as a structure similar to that of a STRIPS operator [17]. Each
87 node is composed of pre-conditions, post-conditions, a list *add* predicates, a list
88 delete predicates (like STRIPS operators) and an activation energy. The goal of the
89 network is to accumulate activation energy on the nodes that represent the most
90 relevant actions. The network also keeps a minimal activation threshold that a
91 node must respect. If the activation of the node is lower than the threshold, then it
92 is removed. After some times the network must preserve only actions that normally
93 are interdependent. The main advantages of such an approach, according to Maes,
94 are that the network can model changes in the objectives of the agents. It also can
95 take into account quantitative environments and multiple actions. However, be-
96 cause of the propagation of activation values, the decision-making tends to take
97 some time.

98 Another adaptation technique for autonomous agents consists in using inductive
99 logic programming (ILP). Benson [5] presents an algorithm capable of learning the
100 action models of agents that are then used for planning. The algorithm represents
101 actions as **Teleo-OPerators** (TOPs) [39] composed by conditions, variables, or ef-
102 fects. The agent takes periodic readings on the environment. During each reading,
103 the agent constructs a set of positive or negative examples according to the actions
104 whose conditions are satisfiable or not. When the conditions of a TOP become true,
105 post-conditions, modeled previously, are used for updating the structure of the TOP.
106 Then, starting from this data set, an ILP algorithm extracts the relationships between
107 the propositional formulas. Such relationships constitute the plans for the agent to
108 achieve its goals more quickly. It seems that the adaptation process of the agent is
109 quite slow, because it requires many observations of the environment. In addition,
110 the algorithm is not incremental.

111 2.3. *Classifier systems*

112 Classification is the most common form of automatic learning. Starting from a given
113 state of the environment, autonomous agents use classification systems for dis-
114 covering the best action to execute, based generally on statistical techniques. We
115 consider in this work only incremental approaches for the reasons detailed in Sec-
116 tion 2. Incremental learning algorithms (ILAs) have no data at the beginning of the
117 learning process. The learning algorithm must produce or update the models with
118 each new example (typically, new states or events from the environment). Thus, the
119 algorithm takes only into account the models of the data and can update them
120 according to the data characteristics.

121 We can distinguish between two main approaches commonly used in autonomous
 122 agents:

- 123 – Rule-based systems;
- 124 – Memory-based learning.

125 **2.3.1. Rule-based systems.** The first classifying systems used for autonomous
 126 agents are systems containing rules, e.g., [22]. In such systems, the developer must
 127 construct a set of rules that describes the behavior of the agent. Each rule connects a
 128 set of conditions or premises to an action. A rule fires when the premises are true in
 129 the current state of the environment. To learn, the agent chooses a set of applicable
 130 rules based on the current state. Then, it uses a measure of quality attached to each
 131 rule (priority) to select the best one. Generally, the agent receives a reward according
 132 to its action, like in reinforcement learning. In this case, it increases the quality of all
 133 the rules selected previously. Consequently, as the time goes by, the priority of the
 134 rules changes. Many approaches were proposed to improve the technique, like ge-
 135 netic algorithms or parallel computing. See for instance [8] and [23]. Unfortunately,
 136 this approach inherits several defects from the reinforcement learning approach like
 137 the processing time for reward calculations and learning. The agent needs an initial
 138 model (a whole set of rules) to start acting, which requires a representation language
 139 and much background knowledge. In spite of the elegance of genetic algorithms and
 140 parallel calculations for exploring the search space, the time needed for adaptation
 141 restricts its practical use to a small number of applications.

142 **2.3.2. Memory-based reasoning.** The first applications using the memory to im-
 143 prove the performance of autonomous agents were developed at the beginning of the
 144 nineties, see [28] and [29]. Such applications used reasoning by memory, better
 145 known under the term MBR. In the field of machine learning, this approach is
 146 known as instance-based learning (IBL).

147 The general idea of MBR is the following. If a given action took place in the past
 148 in a given situation s and gave good results, it will be useful in a new situation s'
 149 similar to s . Thus, the key point in MBR is to find a similarity measure to compare
 150 observations. The traditional methods of MBR use the well-known K-NN algorithm
 151 with traditional similarity measures like the Euclidean distance or the cosine. The
 152 idea of MBR is the same as that of case-based reasoning (CBR) [44]. However, the
 153 CBR is a more complex process using analysis and adaptation of cases. Cunningham
 154 and collaborators [12] discuss the limitations of MBR and the relationship between
 155 MBR and CBR. According to the authors, MBR is different from the CBR because
 156 MBR does not use background knowledge. In CBR, background knowledge is often
 157 used for defining similarity measures and cases adaptation. On the other hand, MBR
 158 is purely empirical, requiring a great amount of data, much processing, and much
 159 effort for structuring cases. The authors argue that MBR must be used in activities
 160 where semantics is not necessary. The problem is that the case structuring often
 161 requires a semantic model of the field. With regard to autonomous agents, the cases
 162 represent the states of the environment, and thus, a minimal knowledge of the field
 163 will be necessary to describe them, generally as vectors of attribute-value pairs.

164 Faltings argues in [15] that similarity measures for attributes and cases are gener-
165 ally inaccurate and very hard to compute. To minimize such problems, the author
166 proposes the usage of tools from qualitative modeling. With such tools the user can
167 give the system accurate models capable of decreasing the ambiguity of MBR
168 measures. Kozierok and Maes [28] and Lieberman [30] have used a similar approach
169 on intelligent interfaces. Kozierok considers user's feedback to increase/decrease the
170 relevancy of predictions made by an MBR system on scheduling meetings. Lie-
171 berman's interface agent uses the interaction with the user to define different weights
172 for attributes used to retrieve data like personalized apartments on rental domains.
173 The comparison between cases is often made with weighted measurements. The
174 major problem is the processing time necessary to update the weights as new in-
175 formation comes in. One of the solutions consists in updating information, not every
176 time new information arrives, but less frequently. It is the case, for example, in the
177 work of Kozierok and Maes [28], in which the model is updated once every night.
178 Consequently, the agent always works shifted with respect to the current model of
179 the world. In our case, we need a learning method that does not require too many
180 calculations. The agent must learn and update dynamically the concepts about the
181 world according to new experiences. Moreover, this method must be sufficiently
182 generic so that any agent can use it in any field. The method we developed, described
183 in Section 3, answers such requirements.

184 3. Entropy-based learning approach—ELA

185 We argue that incremental learning techniques must be used in dynamic applications
186 where the examples arrive continuously, in particular for adaptive autonomous
187 agent applications. In our method the incoming examples are taken into account
188 immediately. The method has no training phase. Learning is done continuously like
189 in an MBR algorithm. Our learning mechanism is divided into two stages: data
190 acquisition/representation and classification. For data acquisition we use a graph
191 structure. Each node of the graph represents a subset (a partition) of the data. For
192 classification, our algorithm quickly looks in the graph for the best partition in terms
193 of data distribution. Finally, the classification algorithm uses the partition found in
194 the previous step for identifying the best class by looking at the number of conditions
195 of the example that are satisfied by the classes. The best class must cover the maximal
196 number of conditions of the example.

197 3.1. Data Representation

198 We developed a structure named graph of concepts (GC). The main goal is to give
199 the learning algorithm a certain number of partitions. On the graph, each node
200 represents a concept (an attribute value, not an attribute) and each arc represents a
201 transition from an attribute to another one. We do not weigh attributes, nor do any
202 post-processing on the graph. The order of the attributes results from the original
203 data representation. Concepts and links are created on the fly. The edge between two
204 concepts A and B contains ids of examples that contain A and B .

Contrary to most approaches that represent data with trees, graphs or rules, the graph presented here is not goal-oriented. More traditional approaches represent data based on predicting attributes and one single goal attribute. In fact, the problem is understood as finding relationships between predicting attributes that allow predicting the value of the goal-attribute. In our approach, the graph (presented here) is a general representation of the data where the goal attribute is unknown. The goal attribute is given before the classification. We discuss this point in Section 6.1. We used here the Quinlan's database to illustrate the approach (Figure 1).

Definition 3.1(α operator) For each node nd we define the operator α . Such an operator gives all the examples present on the incoming edges of nd . For instance, in Figure 1, $\alpha(\text{"True"}) = [2\ 6\ 7\ 11\ 12\ 14]$, indicating the nodes of the graph that have examples containing "wind."

Definition 3.2(Non-taxonomy principle) A more important point is that child nodes do not represent specializations of parent-nodes. For instance, let B be a child

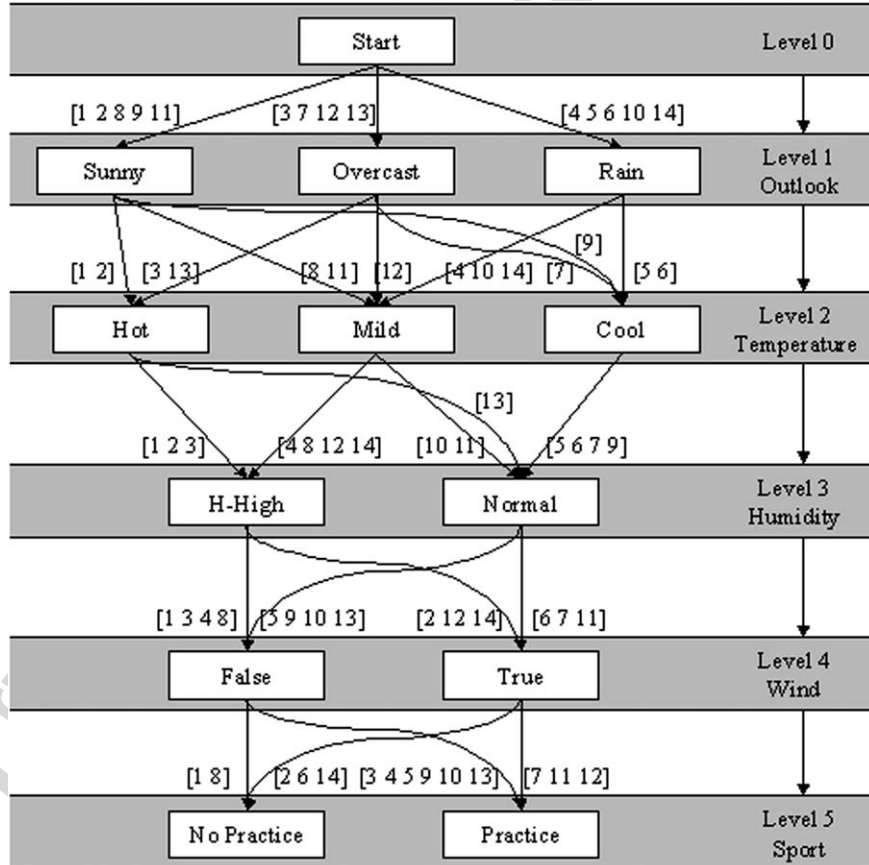


Figure 1. Graph built from Quinlan's database [37].

of A , $\alpha(A) \subset \alpha(B)$ is generally false, implying that attributes are not ordered hierarchically. Although the graph presented here is similar to the *oblivious read-once decision graphs* discussed in [26] and in Section 1, we do not use any algorithm for updating, pruning or changing the structure of the graph during data representation, avoiding the graph specialization for a set of well-defined concepts (in terms of machine learning concepts). For us, such operations are prohibitive because agents generally do not know the entire dataset, but receive information gradually. Consequently the concepts definition (in terms of machine learning concepts) changes over time, which requires a more general structure.

3.2. Classification

The classification process described in this paragraph is presented in Figure 2. We also use the example scratched in Figure 3 for illustrating the technique. First, on the graph we select all the concepts present in the example, e.g., *Rain*, *Cool*, *H-High* and *True* on Figure 2. Second, we apply the operator α over such concepts. At this point, we have a collection of subsets. In Figure 3 these subsets are [4 5 6 10 14], [5 6 7 9], [1 2 3 4 8 12 14] and [2 6 7 11 14]. We consider that a class is a partition of the data and contains a number of examples. The strategy is to predict the class where the maximal number of conditions is covered, e.g., *Outlook = Rain*, *Temperature = Cool*,

```

Input: A concept graph  $G$  and an example  $e$ 
Output: A label for example  $e$ 

Partitions = subset of partitions in  $G$  selected according to the values of the attributes in  $e$ 
 $P$  = best partition in Partitions (partition presenting the lowest entropy)
 $C = P$  split according to the classes
 $Q = \{\}$ 
For each subset  $C'$  in  $C$ 
     $E = \{\}$ 
     $c$  = class of  $C'$ 
    For each  $P'$  in Partitions
         $E = E \cup \{C' \cap P'\}$ 
    End For
    // At this point  $E$  has the following format:
    //  $E = \{E_1, E_2, \dots, E_n\}$  where  $E_i$  is a set of examples
    //  $V$  is the entropy of  $E$  taking into account the cardinality of each  $E_i$ 
    // where  $|E| = \sum_{i=1}^n |E_i|$ 
    
$$V = - \sum_{i=1}^n \frac{|E_i|}{|E|} \times \log_2 \frac{|E_i|}{|E|}$$

    Insert the pair  $(c, V)$  into the priority queue  $Q$  using  $V$  for ordering the queue
End For
// Retrieve the pair containing the class with maximal entropy
 $(\text{class}, \text{Value})$  = the last element of  $Q$ 
Return class

```

Figure 2. Classification algorithm.

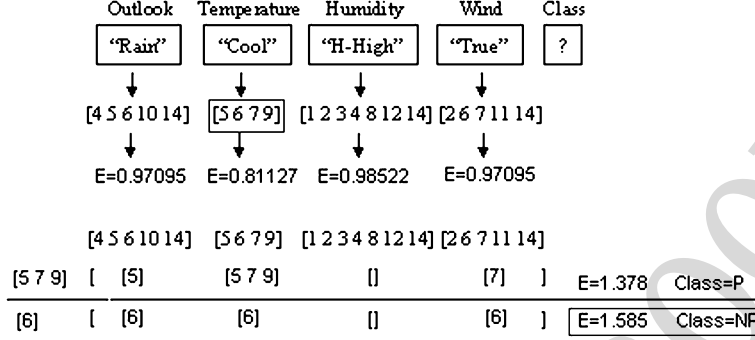


Figure 3. Example of classification.

237 *Humidity = H-High* and *Wind = True*. In other words, we must find for each class
 238 how many intersections are not empty between the examples of the class and the
 239 subsets selected from the graph. The best class must have its examples spread along
 240 the subsets in a homogeneous fashion, covering the maximal number of subsets. This
 241 criterion is exactly the opposite of the entropy measure and other methods for
 242 measuring purity of sets. Thus, the inverse of such criteria can be used to identify the
 243 correct class.

244 An important point is how to define the initial partitioning of the classes, or, in
 245 other words, for each class what examples to consider. We do not use the entire
 246 training set to define the class distribution. In contrast, only the subsets are used to
 247 give the structure of the classes. More precisely, we examine the subsets looking for
 248 the one where the class distribution is the best one using the traditional minimal
 249 entropy criterion. Such a process often gives us a feasible initial partitioning. Thus,
 250 we reduce the classification search space, taking the examples of a single subset.
 251 Figure 3 shows an example. First we use the minimal entropy to discover the best
 252 initial partition ([5 6 7 9]). Next, the maximal value of the entropy is used to classify
 253 the example, based on the intersection sets.

254 According to the Quinlan's rule in [42], **if** (*outlook = overcast*) **and** (*wind = yes*)
 255 **then** *class = NP*. Thus our algorithm classifies the example correctly. This demon-
 256 strates the generalization ability of the technique for predicting the class of test
 257 examples.

258 **3.2.1. Why does it works.** The classification algorithm takes a set of partitions as
 259 an input. The overall set of partitions does not give enough information for dis-
 260 criminating the classes. Thus, we reduce the problem by choosing the best partition
 261 using the minimal entropy criteria and splitting the partition based on the classes.

262 The choice of the class takes into account the quality of the class in relation to all
 263 the partitions. In fact, the optimal class must be present entirely inside the initial
 264 partitions. Obviously when this occurs, the intersection between the class and the
 265 partitions must produce equal size subsets. This is the case in Figure 3 for the class
 266 NP. The intersections produce three sets of size 1 ([6], [6] and [6]). An imperfect class

267 produces intersections of variable size. Therefore, we can use this behavior for
 268 identifying the best class using the maximal entropy.

269 The following sections present the performance of the approach, comparing re-
 270 sults with other learning techniques on publicly available test databases for machine
 271 learning.

272 4. Comparing performance with published results

273 To evaluate the system performance we used some public qualitative databases
 274 obtained from the UCI Repository [7]: Soybean, Zoo, Audiology, Tic-Tac-Toe and
 275 Mushroom. We compare our results with results presented by Eklund [14] and Kasif
 276 [25]. To compare results with Kasif's results, a 10-fold cross-validation is used
 277 (Table 1). Comparison with Eklund's results uses a 5-fold cross-validation (Table 2).
 278 According to the cross validation process, for each iteration i of N , the database
 279 (DB) is divided into training database and test database. The examples are selected
 280 randomly. The size of test database is $(1/N) * |DB|$ and the remainder of the ex-
 281 amples is used for training. The algorithm learns with the training data and its
 282 performance is evaluated on the test database. At the end of the iterations, the
 283 average performance and the standard deviation are calculated.

284 The algorithms presented in Table 1 are Naïve Bayes, Nearest Neighbor (NN) and
 285 Parallel exemplar-based learning system (PEBLs). The algorithms presented in
 286 Table 2 are C4.5 (Quinlan's decision tree algorithm), C4.5 Rules (Quinlan's rule

Table 1. Comparison of results.

Algorithms	ELA	Bayes*	NN*	PEBLs*
Tic-Tac-Toe	76.63 ± 5.70	69.40 ± –	81.70 ± –	94.40 ± –
Soybean	91.00 ± 3.96	81.40 ± –	91.20 ± –	93.20 ± –
Zoology	96.00 ± 4.90	89.10 ± –	96.00 ± –	95.00 ± –
Mushroom	98.50 ± 0.34	99.70 ± –	100.00 ± –	100.00 ± –

– Not given.

* Results from [24].

Table 2. Comparison of results.

Dataset \ Algorithm	ELA	C45*	C45 Rules*	CN2*	LMDT*
Tic-Tac-Toe	74.34 ± 1.19	70.03 ± 7.69	75.95 ± 7.24	75.21 ± 7.20	78.94 ± 8.69
Soybean	92.13 ± 2.41	91.60 ± 5.09	91.58 ± 5.09	91.92 ± 5.95	95.45 ± 4.71
Zoology	95.00 ± 0.00	86.85 ± 0.62	–	70.64 ± 0.34	–
Mushroom	98.49 ± 0.24	71.02 ± 2.10	71.55 ± 3.92	72.19 ± 2.36	73.51 ± 4.30
Audiology	78.00 ± 4.84	77.05 ± 4.77	74.84 ± 6.23	69.53 ± 5.03	81.71 ± 13.28

– Not given.

* Results from [14].

287 induction algorithm), CN2 (rule induction algorithm) and LMDT (linear machine
288 decision tree).

289 The comparison with Kasif's results (Table 1) shows that the ELA performance is
290 similar to that of the NN algorithm. This behavior was expected because the clas-
291 sification algorithm of Section 3.2 considers the number of covered features for
292 selecting the best class. Thus, we benefit from the NN accuracy, which according to
293 results related by Michie et al. [35], outperforms most learning algorithms. Section 3
294 gives a more detailed comparison between NN and ELA. Naïve Bayes presents a
295 poor performance compared to the others algorithms. The results on the Tic-Tac-
296 Toe dataset suggest that the algorithm is less accurate on databases where attributes
297 have similar discrimination power. This happens because the initial partitioning in
298 the classification step has a direct effect over the results. Thus, a poor partitioning
299 decreases the accuracy.

300 The results of Table 2 are very encouraging. In spite of a similar accuracy, the
301 algorithm presents a reduced deviation. Over the Mushroom database, ELA obtains
302 a much better accuracy with a much lower deviation. Over the Audiology database
303 the ELA performance is also quite good. Such a database has a heterogeneous class
304 distribution. Indeed, from 200 examples of the database, a single class has 48 ex-
305 amples and six classes have only one example (the total number of classes is 24).
306 Another problem with the Audiology database is the amount of irrelevant attributes
307 (from 72 features, less than 10 are normally used for classification in a decision tree).
308 This suggests that ELA is relatively robust in terms of unbalanced class distribution
309 and of irrelevant attributes.

310 4.1. Data reduction

311 A well-known problem of MBR techniques concerns the large amount of data ne-
312 cessary for learning. Consequently, such algorithms need much space for storing
313 data and, worse, require much processing time for computing distance among ex-
314 amples. In this section we study how the graph presented in Section 3.1 can be used
315 to significantly reduce the size of the training data without degrading too much the
316 precision of the system.

317 **4.1.1. Data reduction with ELA.** An additional criterion to evaluate MBR algo-
318 rithms is the number of cases the algorithm needs for learning. Since the ELA does
319 not compact data we decided to evaluate the capacity of data reduction of the system
320 and the impact in terms of the loss of precision, comparing it to other well-known
321 approaches.

322 The technique employed is basically the same employed in algorithms IBL of Aha,
323 see [2]. When a new case arrives, the algorithm tries to classify it. If it classifies the
324 example correctly, then the example is ignored, if not (the algorithm finds another
325 class) then it is stored into the database.

326 **4.1.2. Comparison method.** We recovered the results presented by Wilson and
327 Martinez [52]. The authors evaluated 16 MBR algorithms over 30 bases available
328 from the database of the University of California Irvine [7]. We also recovered tests

done by Brighton and Mellish [10]. All tests were accomplished with a cross validation of 10 interactions. The results are presented in Table 3.

Table 3 compares results with the following algorithms: Concept Graph (ELA) without data reduction (same results as in Table 1) plus Audiology database; the Concept Graph with data reduction (ELAr); the average performance of the 16 algorithms compared by Wilson and Martinez [52]; the result of the algorithm RT3 [52] presented by Brighton and Mellish [10]. We present in Table 3 the accuracies and the percentage of training examples used for learning (Space).

4.1.3. ELA, ELAr and RT3. We note that there is a dramatic degradation of performance on the soybean base (40%) and an improved performance on the zoology base (2%) when ELA and ELAr are compared. Concerning the Mushroom database the result is surprising: from approximately the 7313 cases used for training, the algorithm stored only 66 cases (0.86%). In spite of that, the performance is only 5% worse. This performance is much better than the performance of RT3 that needs to store 402 cases. For the Audiology database the results are also encouraging because approximately 35% of the data were stored during the graph construction without a great loss of precision.

4.1.4. ELAr and Average16. Table 3 shows that the capacity of data reduction of ELAr is much higher than the average of the 16 algorithms presented by Wilson and Martinez [52]. However, in spite of this large difference in terms of the number of stored cases, the precision of the classification is similar.

4.1.5. ELA and NN. As mentioned previously, the ELA uses the number of conditions satisfied to classify an example indirectly, like NN. Here we outline the advantages of ELA over NN algorithms.

Quite often, NN algorithms require a different number of neighbors, K, for reaching high performance. Thus, for such K-NN algorithms, the estimation of K is not a simple task when learning is incremental. The simplest strategy consists of

Table 3. Performance results with data reduction.

Dataset\Algorithm	ELA	ELAr	Average16*	RT3**
Soybean	91.00 ± 3.96	50.33 ± 11.78	84.81 ± —	—
Soybean (Space)	100%	28.48%	38.31%	—
Zoology	96.00 ± 4.90	98.00 ± 4.00	91.05 ± —	87.08 ± —
Zoology (Space)	100%	14.07%	34.69%	25.13%
Mushroom	98.50 ± 0.34	93.93 ± 1.47	—	98.89 ± —
Mushroom (Space)	100%	0.86%	—	5.50%
Audiology	77.27 ± 9.75	73.18 ± 10.65	—	—
Audiology (Space)	100%	34.07%	—	—

— Not given.

* Results from [14].

** Results from [10].

356 using different values of K on validation datasets and to choose the one that max-
 357 imizes the accuracy. In ELA such a problem does not exist.

358 Another advantage of ELA over K -NN and probabilistic algorithms is the clas-
 359 sification time. In ELA only part of the data is used to classify a new example. In
 360 addition, the computation of intersections between subsets is quite fast. K -NN and
 361 probabilistic algorithms are generally very slow because they use the entire training
 362 database to compute similarity measures and probabilities, which is problematic for
 363 large databases.

364 **4.1.6. General comments on data reduction.** Starting from the results obtained
 365 with the data reduction, we can conclude that the algorithm has a significant po-
 366 tential for reduction. We believe that the good results are mainly due to the
 367 property of entropy. Because the class distribution does not change, the value of
 368 entropy also does not change. For example, if we have the following distributions of
 369 classes: (4500 450) and (10 1) the entropy value remains the same. Consequently, we
 370 can represent a problem with a very reduced number of instances while maintaining
 371 the same distribution of the data. We discuss more rigorous tests on the Section 5.

372 5. Evaluating using MLC++

373 Comparing with results of published work can be problematic because quite often
 374 different implementations lead to non-comparable results. To define a more accurate
 375 set of trials we used the MLC++ [27] binary files available from <http://www.sgi.com/tech/mlc/> in the experiments. We compared the performance of ELA and ELAr
 376 with four algorithms (C45, LazyDT, IB1 and IB3) on 23 benchmark datasets.

377 The C45 algorithm is a well-known non-incremental decision tree algorithm.
 378 LazyDT is an incremental decision tree, and IB1 and IB3 are the standard 1 and 3
 379 NN algorithms. A short discussion about the algorithms can be found in [27]. For us,
 380 it is important to compare results with incremental algorithms because learning in
 381 autonomous agents must be incremental; however we verify if the results are ac-
 382 ceptable using also the C45 algorithm. Table 4 presents some information about the
 383 databases and the test methodology (cross validation or test examples given). Ta-
 384 ble 5 presents the respective accuracies obtained with the algorithms. Since ELA and
 385 ELAr work only with nominal data, continuous attributes are discretized using the
 386 standard entropy-based discretization method available on MLC++. Consequently
 387 all algorithms use identical input data.

388
 389 Figures 4 and 5 present the difference in the precision and deviations, comparing
 390 algorithms in pairs. When the difference is greater than zero, the performance of
 391 ELA or ELAr is better. Comparing ELA and C45 in Figure 4, we observe a quite
 392 close performance; however, C45 presents a better performance on 14 of the 23
 393 databases used in the trials. Since we are looking for incremental methods, and cross-
 394 validation does not take into account the order of the training examples (which is
 395 very important for incremental methods) choosing examples randomly, we think the
 396 performance of ELA is acceptable. Results comparing ELA and other incremental
 397 methods are better. ELA and LazyDT presented a better performance on an equal

Table 4. Information about the datasets.

N.	Database	N. Features	N. Classes	N. Exs.	Test
1	Echocardiogram	12 (7 cont.)	3	132	CV-10
2	Soybean Large	35	19	307	CV-10
3	Anneal	38 (6 cont.)	6	798	CV-10
4	Flags	27 (10 cont.)	6	194	CV-10
5	Wine	13 cont.	3	178	CV-10
6	Hepatitis	19 (6 cont.)	2	155	CV-10
7	Audiology	69	24	226	CV-10
8	Zoo	16	7	101	CV-10
9	Mushroom	22	2	8124	CV-10
10	Image	19 cont.	7	210	CV-10
11	Monk3	6	2	122	432
12	Heart	13 (5 cont.)	5	270	CV-10
13	B. Cancer Wdbc	9 cont.	2	569	CV-10
14	Vote	16	2	435	CV-10
15	Ionosphere	34	2	351	CV-10
16	Credit	15 (6 cont.)	2	690	CV-10
17	Monk1	6	2	124	432
18	Thyroid	29 (7 cont.)	5	2800	CV-10
19	Monk2	6	2	169	432
20	Hayes	4	3	132	CV-10
21	B. Cancer Wisc	9 cont.	2	699	CV-10
22	Iris	4 cont.	3	150	CV-10
23	Balance	4	5	625	CV-10

number of databases; however, taking the databases where ELA wins, we observe a greater precision, while LazyDT presents accuracy slightly higher when ELA loses. Comparing the results with instance-based methods (IB1 and IB3), ELA happens to be much better, presenting a higher precision in most cases.

As we expected, ELAr presents a sensible loss of precision when compared with the C45 algorithm, because the algorithm eliminates many training examples when the graph is built. Table 6 presents the percentage of reduction for the databases. However, we can observe a very close performance when ELAr is compared to incremental methods. Even with the data reduction, ELAr outperforms LazyDT in 11 of the 21 databases (see Figure 5). Performance of ELAr is really very close to that of IB1 and IB3.

Comparing ELAr and ELA, we can note a serious degradation of performance on 6 databases (2-Soybean, 4-Flags, 5-Wine, 10-Image, 11-Monk3 and 19-Monk2). In three of these databases (2-Soybean, 5-Wine and 10-Image), examples are ordered according to the classes. Thus, when an example of a new class arrives, ELAr always add it to the graph (because the class is unknown) but we cannot guarantee that this example is important for the classification or that it is a member of the boundary of the class. When this is not the case, such examples can provoke mistakes, showing the dependence of the algorithm on the initial data organization. We expect in the future to improve the data reduction criterion by not only controlling the input of new examples but also by removing examples of the graph probably using distances

Table 5. Comparing results using MLC++.

N.	ELA	ELAr	C45	LazyDT	IB1	IB3
1	63.85 ± 11.94	63.85 ± 9.76	58.44 ± 11.99	60.00 ± 10.77	63.85 ± 14.62	75.38 ± 10.77
2	91.00 ± 3.96	57.00 ± 11.78	86.34 ± 4.07		91.95 ± 3.67	85.45 ± 5.67
3	97.21 ± 1.47	96.20 ± 2.12	94.04 ± 1.26	93.28 ± 3.64	98.05 ± 1.39	95.97 ± 2.64
4	65.26 ± 4.20	52.63 ± 10.26	61.58 ± 10.00	51.58 ± 10.21	52.11 ± 12.11	52.63 ± 11.29
5	92.94 ± 6.34	74.12 ± 10.91	90.58 ± 4.69	88.24 ± 6.96	95.29 ± 4.40	95.29 ± 4.40
6	80.67 ± 10.52	83.33 ± 10.85	79.34 ± 5.54	84.00 ± 6.11	80.00 ± 8.94	82.00 ± 10.35
7	77.27 ± 9.75	73.18 ± 10.65	76.36 ± 6.03	58.99 ± 7.54	70.38 ± 9.42	51.77 ± 6.46
8	96.00 ± 4.90	98.00 ± 4.00	96.00 ± 6.63	89.67 ± 13.01	89.67 ± 17.16	89.67 ± 17.16
9	98.50 ± 0.34	93.93 ± 1.47	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.91 ± 0.08
10	82.38 ± 4.79	70.00 ± 9.54	83.81 ± 5.72		79.51 ± 13.88	76.93 ± 11.55
11	92.50 ± 5.83	68.33 ± 14.34	95.84 ± 5.59	90.83 ± 4.49	80.83 ± 9.17	77.50 ± 7.50
12	28.55 ± 9.99	33.29 ± 10.71	32.03 ± 13.06	27.65 ± 11.25	26.26 ± 7.51	29.48 ± 11.79
13	90.72 ± 2.08	94.29 ± 3.55	95.17 ± 2.76	92.86 ± 2.88	95.18 ± 2.12	97.68 ± 1.79
14	93.02 ± 3.89	85.58 ± 6.39	96.98 ± 3.61	93.33 ± 4.23	92.05 ± 3.87	92.99 ± 5.59
15	86.86 ± 6.54	88.57 ± 7.88	91.15 ± 3.91	76.24 ± 7.51	89.96 ± 4.69	87.95 ± 3.86
16	78.55 ± 3.29	77.68 ± 2.07	84.08 ± 2.05	82.17 ± 2.90	82.17 ± 3.67	83.91 ± 2.71
17	83.33 ± 7.45	80.83 ± 18.65	88.33 ± 13.03	77.5 ± 17.5	74.17 ± 12.61	64.17 ± 11.21
18	92.86 ± 1.81	95.29 ± 2.43	98.22 ± 0.77	92.75 ± 3.70	89.92 ± 5.12	90.29 ± 5.06
19	61.88 ± 9.04	50.00 ± 13.11	67.47 ± 11.11	71.25 ± 14.03	52.50 ± 14.84	48.13 ± 10.48
20	74.62 ± 9.76	73.85 ± 10.99	79.21 ± 8.47	85.67 ± 22.71	44.83 ± 24.23	66.33 ± 21.32
21	86.38 ± 3.68	95.22 ± 1.30	93.51 ± 2.27	91.71 ± 2.78	94.34 ± 2.74	96.24 ± 2.55
22	85.33 ± 8.84	82.67 ± 9.04	94.66 ± 6.53	96.62 ± 6.17	97.95 ± 3.13	97.95 ± 3.13
23	65.81 ± 5.53	61.77 ± 6.61	74.36 ± 4.80	80.97 ± 5.14	63.06 ± 3.85	54.19 ± 5.11

metrics like in [10] for discovering disjoint examples, thus producing an hybrid approach for data reduction.

The performance of ELA has been improved with ELAr on 7 databases (6-Hepatitis, 8-Zoo, 12-Heart, 13-Breast Cancer Wdbc, 15-Ionosphere, 18-Thyroid and 20-Breast Cancer Wisc) without increasing the deviation seriously, showing that the algorithm can be used on a large number of applications.

6. Related work

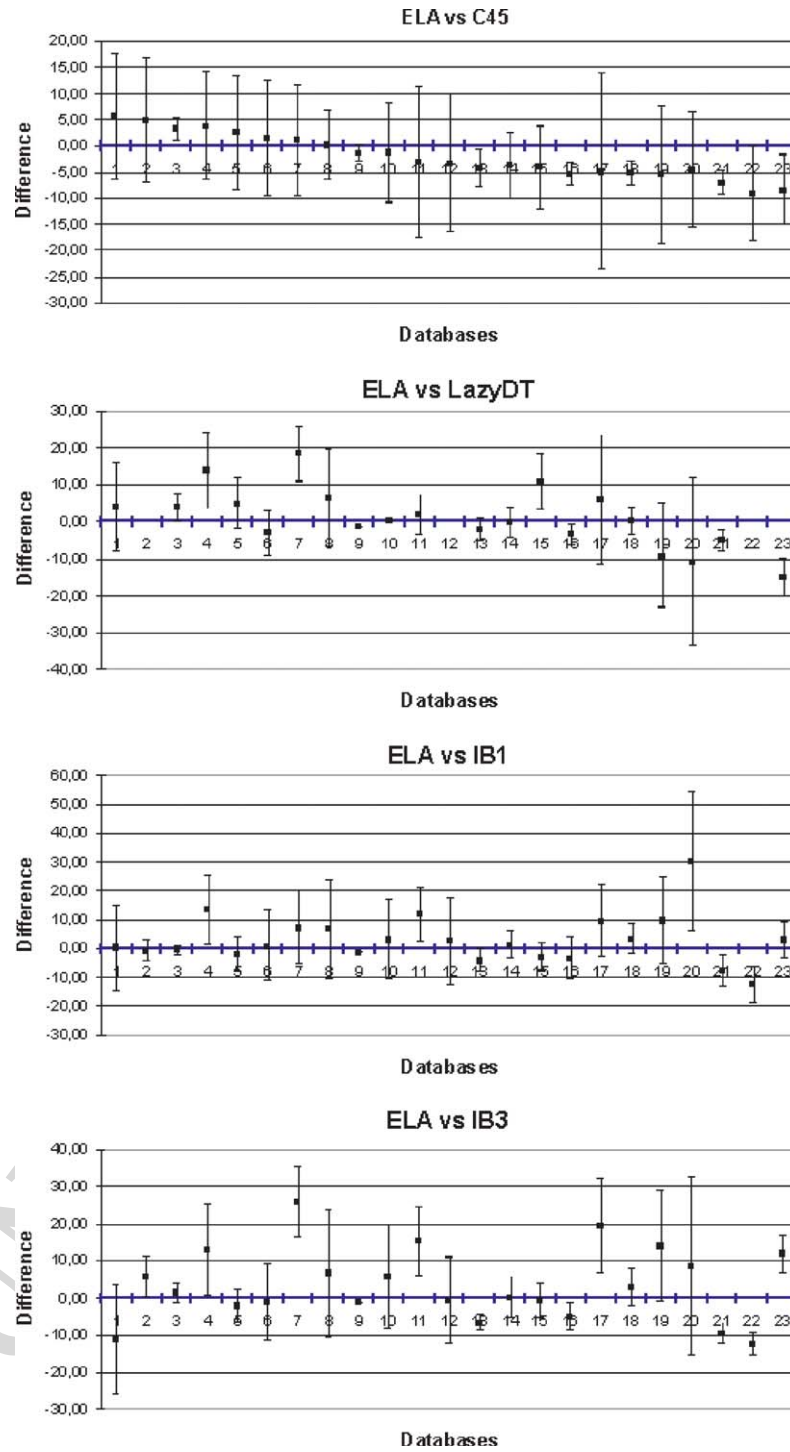
In this section we discuss related works based in two main aspects: *the graph construction and graph algorithms* and *the graph usage*.

6.1. Graph construction and graph algorithms

Kohavi proposes to represent data on oblivious **read-once** decision graphs in [26]. The term read-once refers to the fact that each variable occurs at most once along any computation path. A **levelled** graph is one where the nodes are

Figure 4. Comparing ELA with C45 (a), LazyDT (b), IB1 (c) and IB3 (d).

COLOUR FIG.



COLOUR FIG.

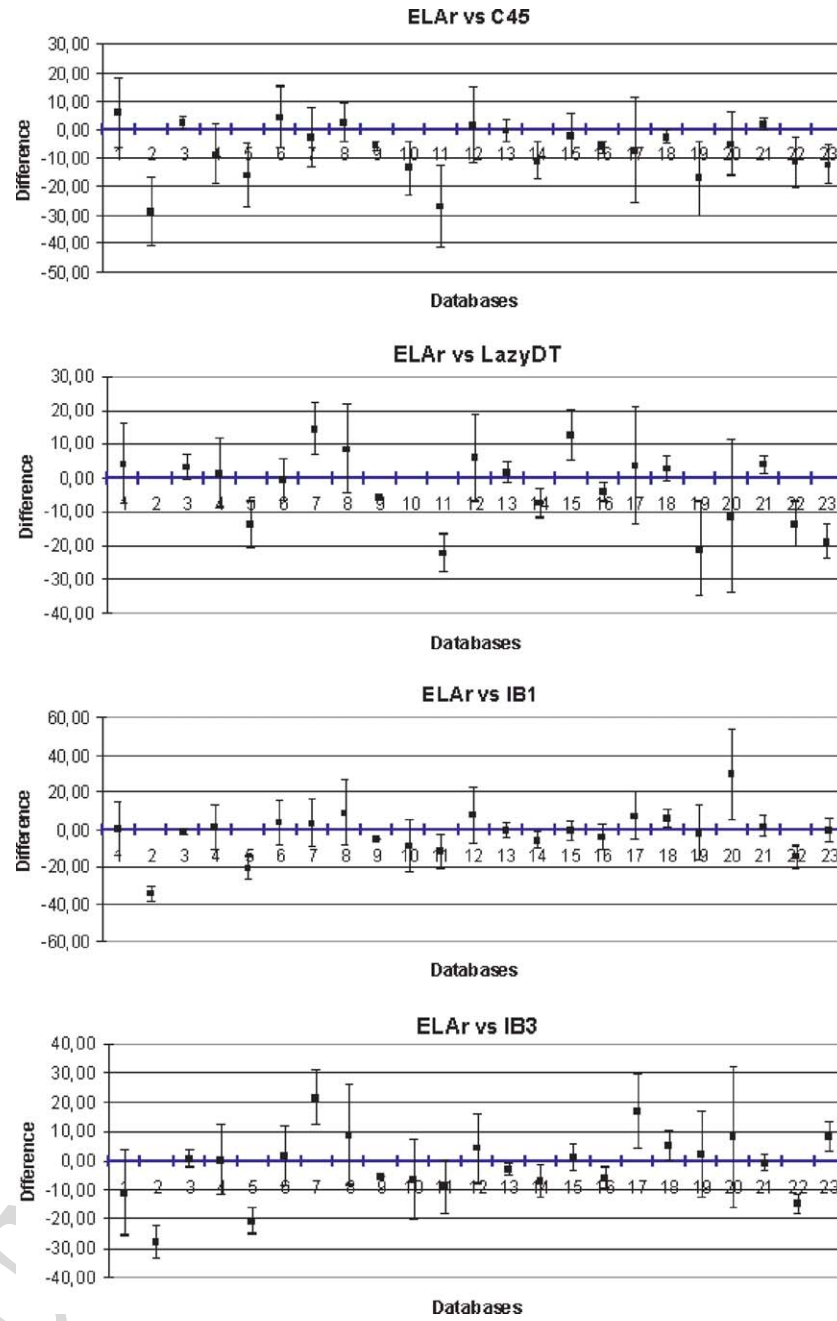


Figure 5. Comparing ELAr with C45 (a), LazyDT (b), IB1 (c) and IB3 (d).

Table 6. Percentage of training examples used for learning by ELAr.

N.	Database	%	N.	Database	%
1	Echo	34.29	13	Breast-wdbc	7.80
2	Soybean	28.48	14	Vote	14.59
3	Anneal	8.05	15	Ionosphere	12.47
4	Flags	44.69	16	Credit	14.27
5	Wine	4.97	17	Monk1	17.50
6	Hepatitis	20.36	18	Thyroid	3.13
7	Audiology	34.07	19	Monk2	41.07
8	Zoo	14.07	20	Hayes	41.21
9	Mushroom	0.86	21	Breast-Wisc	7.13
10	Images	8.78	22	Iris	4.96
11	Monk3	21.97	23	Balance	32.86
12	Heart	72.61			
	Average of Reduction:	23.07			

partitioned into disjoint sets. An **oblivious decision graph** is a labelled graph where each level of the graph is labelled by the same variable. Thus, we can conclude that the graph introduced in Section 1 is an *oblivious read-once decision graph*.

In [42, 43] Quinlan proposes to represent data with decision trees. Decision trees are not oblivious graphs because a given level of the tree can be labeled by different variables. Both Quinlan's and Kohavi's algorithms are used on non-incremental learning. Most recent works [19, 49] extended such techniques for allowing incremental learning but they tend to be complex and difficult to compute. We believe that incremental learning from rules [46] presents the same problems. The main difference observed in relation to the work presented here, concerns of the graph. Such branching programs use feature selection criteria and/or split and join operations to produce reduced graphs. Thereafter, the classification algorithm is trivial. Here, the most complex phase is the classification task, not the graph construction. Our algorithm selects only one attribute for finding the initial partitioning. Next, the classification algorithm, through intersections over subsets, determines the class. Thus, we do not order nor select attributes many times, which improves the speed of the algorithm.

The techniques discussed in this section are goal-oriented, meaning that the structures used to represent the data refer to one single variable. Often, agents have to make decisions based on different variables, for instance, to predict the receiver of a message, to predict the value for a service or to predict when to send a message. Thus, the technique presented in Section 3 can be used to predict values for any variable in the database.

Gonzalez [21] discusses the SubdueCL graph based learning system. He discusses three techniques based on ILP: ILP Systems (FOIL and Progol), Conceptual Graphs and Galois Lattice. Next, he compares some results with FOIL and Progol systems. Both systems however are not incremental.

460 6.2. Graph usage

461 The algorithm of Section 3.2 uses the graph to find combinations of conditions that
 462 are important to classify a new example, where the class of the example is another
 463 condition. This is a problem similar to association rules defined by Agrawal in [1].
 464 For instance, the following association rule can be found based on the example in
 465 Figure 3:

if((*Outlook* = *Rain*) **and** (*Temperature* = *Cool*)) **then** (*Class* = *Non – Practice*)

467 The main problem with association rules is generally the time for computing and
 468 for evaluating the rules. For instance, for n conditions (attributes), there are $2^n - 1$
 469 possible rules for each class. The *Apriori* algorithm of Agrawal [1] proposes the
 470 identification of 1-condition rules, 2-condition rules, etc. Next, the algorithm prunes
 471 rules that do not satisfy a support value and combines only interesting rules. The
 472 STUCCO algorithm of Bay [4] uses the concepts of [1] but represents rules as a tree.
 473 Thus, the algorithm represents rules more efficiently and can decrease the time
 474 needed for learning rules. None of the techniques are used for incremental learning.
 475 In this work, we are not interested in general association rules for two main reasons:
 476 (i) in our applications, the system has little time for learning, and (ii) the relation-
 477 ships among the variables do not need to be represented, i.e., we are not looking for
 478 knowledge, but for an accurate classifier. For instance, a personal assistant for
 479 personalized search on the web needs to find interesting documents based on the user
 480 profile, but it does not need to justify why it believes that such documents are
 481 interesting.

482 7. Using ELA for searching the web — MAIS

483 Learning agents are often used for searching the web. In this section we discuss an
 484 adaptive application, multi-agent based Internet search (MAIS), for personalization
 485 of web searches. In such an application, adaptative agents are implemented using
 486 ELA, the learning method introduced in previous paragraphs. We demonstrate that
 487 ELA can be successfully used for the dynamic learning of users' interests in order to
 488 personalize results of searches on the Internet.

489 Web search is a widely studied problem that has no well-defined solution. Search
 490 engines often give very different results based on the same query. This problem is due
 491 to two main factors: (i) search engines have different strategies to evaluate the quality
 492 of the sites in relation to user's queries, and (ii) search engines databases contain
 493 different sets of indexed pages.

494 We think that the generic strategies of search engines for ranking pages may work for
 495 many users but cannot work for most them, essentially because they do not consider
 496 the user's profile nor background in order to obtain better, personalized results. Thus,
 497 web search must be improved with agents capable of adapting the results, taking into
 498 account user's preferences and centers of interest. In this application we implemented

499 adaptive agents using the ELA approach introduced in the previous section for
 500 learning the user model and classifying the results of the searches.

501 Search engines are often incapable of covering all the web space and maintaining
 502 consistent data (ii). In this case we believe that a distributed approach for re-
 503 presenting and sharing information can improve the covering of the web space. In
 504 this application we use specific agents to make meta-search on top of well-known
 505 search engines.

506 7.1. *Adaptive agents and the web*

507 Agents are often used in the implementation of more intelligent Internet services.
 508 Such services can aid the user to personalize news, find papers, search sites or filter
 509 information. Since the nineties, a large wave of applications has been developed with
 510 such features. What most such applications have in common is the usage of an
 511 accurate user model for improving and personalizing results. Middleton [36] dis-
 512 cusses more than sixty applications. The author organized the applications according
 513 the different ways of implementing the user model:

- 514 – **Supervising the user behavior:** the agents represent users' actions on unsupervised
 515 databases then, an incremental learning method (for instance, behavior networks
 516 or memory-based algorithms) is used for identifying the sequences of actions
 517 adapted to a given situation;
- 518 – **Using explicit feedback:** the agent ask the user for information for correcting errors
 519 and refining the answers to the problems; Lieberman [31] calls this approach
 520 *Profiling by Example*;
- 521 – **Learning from a base of experiences:** the agent constructs a database of experiences
 522 and a non-incremental learning algorithm produces a user model;
- 523 – **Allowing the user to program the agent:** the user can program the agent for im-
 524 proving its performance;
- 525 – **Using a knowledge base:** the agent can use predefined user models developed
 526 previously. The user can fill forms and give answers to general questions. Based
 527 on this information the user constructs a knowledge base.

528 MAIS uses the first technique for modeling the user. The user model is updated
 529 dynamically, based on the user's actions. For the web search, the user's actions are
 530 limited to adding and removing sites to and from favorites and changing the centers
 531 of interest. Additionally, the user can give feedback, adding sites presented in the
 532 result of the searches as favorites.

533 7.2. *MAIS architecture*

534 Our web search system has four types of agents: personal assistant (PA), library
 535 agent (LA), filter agent (FA) and search agent (SA). These agents are presented in
 536 Figure 6. Agents exchange labelled messages (Request, Answer and Call-For-Bids
 537 "CFB") for dynamic task allocation, containing documents, document vector

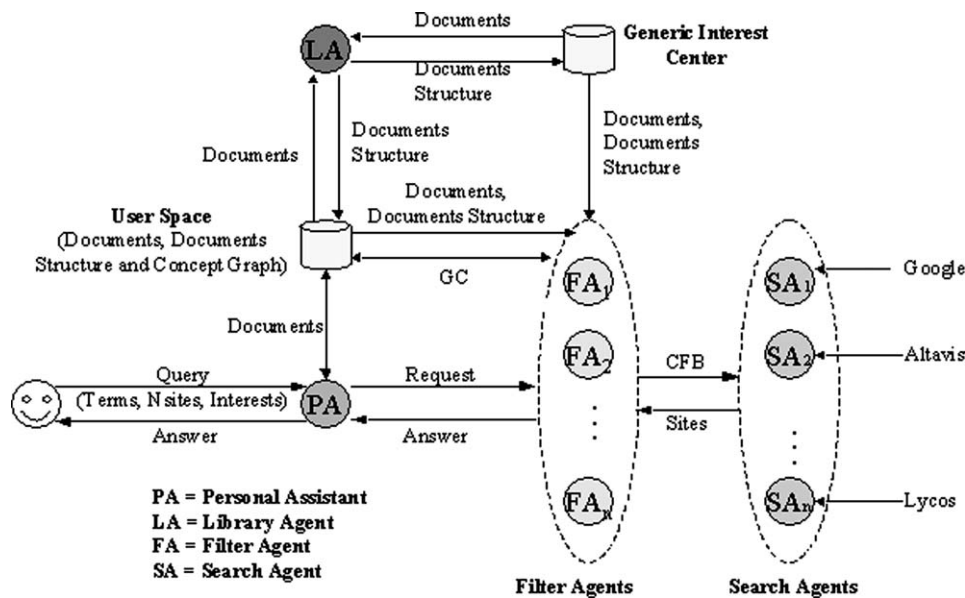


Figure 6. MAIS architecture.

models or Concept Graphs (CG) used by the learning algorithm for classifying and scoring the results of the queries.

The user interacts with the system using a PA interface. The PA provides two services: management of the favorite sites and management of the the search.

7.3. Management of favorite sites

Favorite sites are stored in the user workspace inside folders as text documents. Next, the LA analyzes the documents and constructs a general dictionary used to represent the overall user's documents. The resulting structure contains the most important terms based on the collection of personal documents. No agent interacts directly with the LA. In fact, the LA can be programmed for updating the structure of the users' documents automatically and regularly or receive an explicitly request of the system administrator. Similarly, an LA is used to construct the dictionary of the public interest centers available in the system.

The user can improve the performance of the searches, selecting public interest centers. These interest centers are models made from documents for specific domains. Experts have chosen such documents. Each interest center is composed of a fixed collection of documents and the dictionary learned with LA. For instance, *Artificial-Intelligence-Adaptive-Agents*, *Groupware-Knowledge-Management* and *Artificial-Intelligence-MBR* are examples of interest centers present in the system. Such interest centers are common to any user, but can be personnalized progressively adding new web sites to the profile. The documents in an interest center are copied

559 inside the user space when the topic is selected. Then the user can update this interest
560 center. Currently, the number of available interest centers is small but we expect to
561 build a collection of interest centers similar to the directories organized in categories
562 commonly available in web search engines.

563 7.4. *Search management*

564 The user starts a search by passing information to the PA, i.e., a list of terms, the
565 maximal number of sites wanted in the answer and the interest centers. Then, the PA
566 sends a broadcast message and contracts an FA to start the search.

567 The FA plays a central role in the system. The FA engaged in a contract must
568 retrieve sites from SAs. The FA sends a CFB passing the user's terms, the
569 number of sites and a deadline to SA. FA waits for all answers to arrive before a
570 time limit. In the next step, it starts the analysis of the sites for personalizing the
571 final result.

572 In order to personalize results, the FA orders the list of sites received from SAs
573 according to the user's preferences. Each page is submitted to classification, which
574 gives a degree of interest to the page. The page classes are the favorite folder names
575 and the interest centers of the user. The sites are then ranked and the best ones are
576 selected. The following sections give more details on how each agent has been im-
577 plemented.

578 7.5. *The personal assistant*

579 A user interface and a base of actions compose the PA. The user interface allows
580 starting the search, viewing results and managing favorites. It is the client side of the
581 PA. The base of actions aggregates a set of LISP functions coded on an HTTP
582 server. The functions are executed remotely through a remote interface. Thus, any
583 user connected to the Internet can use the LISP interface. Figure 7 presents the
584 interface of the PA. The user can also select the centers of interest using this interface
585 (see Figure 8).

586 Because the search process can take quite some time, the searches are started
587 asynchronously. Consequently, the user can start several searches and later analyze
588 the results.

589 To make the system interoperable, we have developed a small language used to
590 make communication between the LISP interface and the HTTP server. Similarly,
591 this language allows to communicate with agent platforms other than OMAS [3],
592 used in this work. We tested interoperability of our OMAS agents with another
593 FIPA [16] compliant platform, namely, JADE [24].

594 Figure 9 shows how searches can be started without using the LISP interface
595 but from JADE agents. In the left handside of the window we have information
596 about our platform. The right handside the window is used to send a request for
597 sites. The "searcher@OMAS" agent plays the role of the assistant and starts the
598 search process. "START-REQUEST" is a term from the content language we
599 developed. The complete language specification it is out of the scope of this
600 paper.

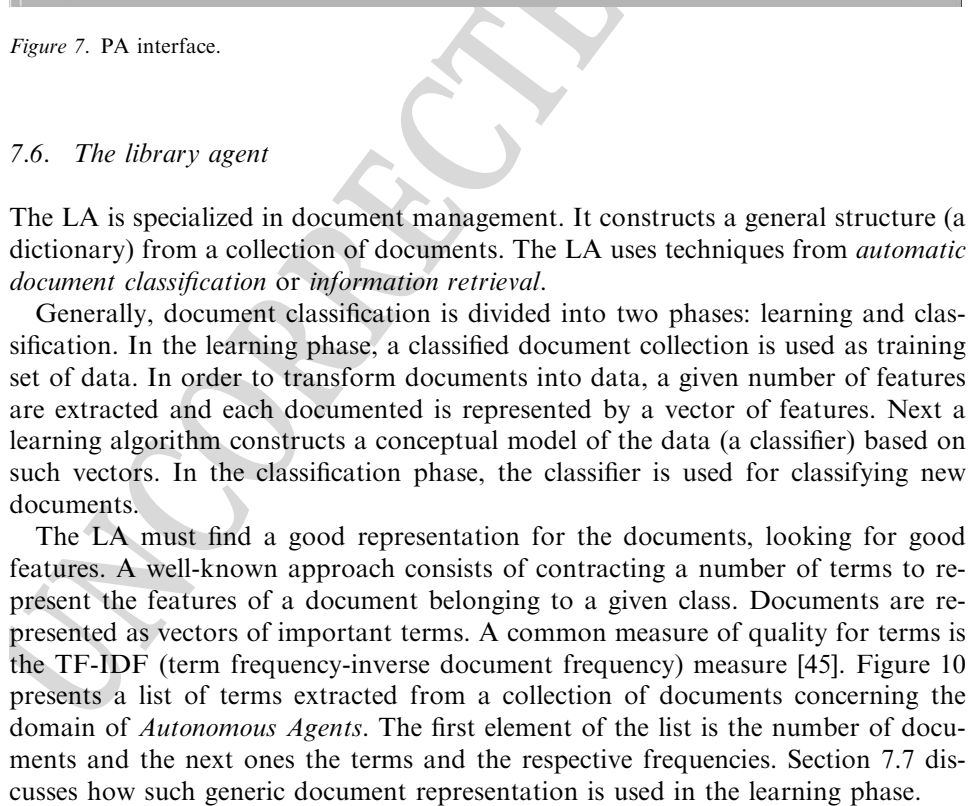


Figure 7. PA interface.

601 7.6. *The library agent*

The LA is specialized in document management. It constructs a general structure (a dictionary) from a collection of documents. The LA uses techniques from *automatic document classification* or *information retrieval*.

Generally, document classification is divided into two phases: learning and classification. In the learning phase, a classified document collection is used as training set of data. In order to transform documents into data, a given number of features are extracted and each document is represented by a vector of features. Next a learning algorithm constructs a conceptual model of the data (a classifier) based on such vectors. In the classification phase, the classifier is used for classifying new documents.

The LA must find a good representation for the documents, looking for good features. A well-known approach consists of contracting a number of terms to represent the features of a document belonging to a given class. Documents are represented as vectors of important terms. A common measure of quality for terms is the TF-IDF (term frequency-inverse document frequency) measure [45]. Figure 10 presents a list of terms extracted from a collection of documents concerning the domain of *Autonomous Agents*. The first element of the list is the number of documents and the next ones the terms and the respective frequencies. Section 7.7 discusses how such generic document representation is used in the learning phase.

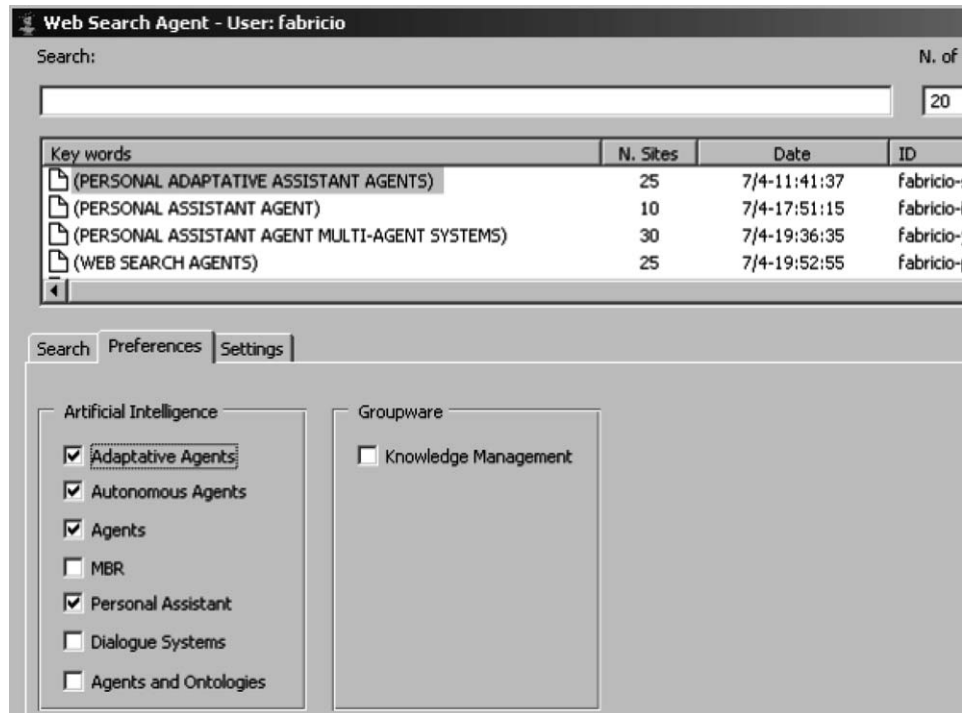


Figure 8. Public interest centers.

621 7.7. The filter agent

622 The FA coordinates the dialog among the agents of the system. It is responsible for
 623 personalizing the list of sites received from SAs based on the user profile. The FA
 624 uses ELA for classifying the sites.

625 The classification of a site returns the class of the site and a classification rate. To
 626 us, the classification rate is the value computed in the ELA algorithm, for instance,
 627 the value 1.585 in Figure 3. Before creating the concept graph we need (i) to compute
 628 a general structure for the user model and (ii) to construct a training set.

629 7.8. The general user model structure

630 The general structure of the user model takes into account the representation of the
 631 user favorite documents and the representation of each center of interest. We merge
 632 such structures, ordering the terms according to their TF-IDF values and limiting
 633 the size of the vector to 1000. With this approach we guarantee that terms, important
 634 for all the classes, are present in the general structure. Figure 11 gives an example of
 635 the process.

636 Thus, the general user model representation consists of a list of terms selected
 637 from $(n + 1)$ vectors representing document collections: 1 for the user favorites and n

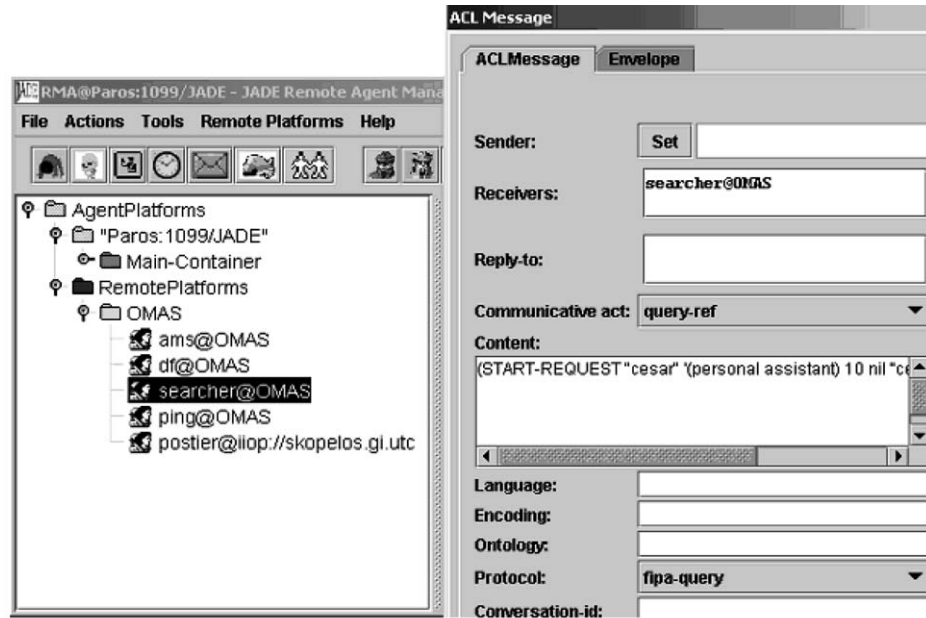


Figure 9. Executing searches remotely with JADE agents.

```
(7
  ({"AGENT" 265 6 6) ("ACTION" 207 7 7) ("SUBSTITUTION" 44 1 1)
  ("EVENT" 91 4 4) ("SYSTEM" 164 7 7) ("GOAL" 145 7 6) ("ROBOT" 79 4 4)
  ("NODE" 84 5 5) ("CONDITION" 97 6 6) ("CIRCUITRY" 30 1 1)
  ("GEORGEFF" 76 5 5) ("BLOCK" 29 1 1) ("INTENTION" 59 4 3)
  ("MODEL" 104 7 7) ("WORLD" 95 7 7) ("CONTROL" 87 7 7)
  ("ENVIRONMENT" 84 7 7) ("SUBSYMBOLIC" 22 1 1) ("FIGURE" 56 5 5)
  ("PLANNER" 37 3 3) ("ARCHITECTURE" 82 7 7) ("TREE" 81 7 5)
  ("AIRCRAFT" 28 2 2) ("ACHIEVE" 64 6 6) ("FIRST" 78 7 7)
  ("EXECUTION" 76 7 7) ("SYMBOLIC" 34 3 3) ("SEQUENCE" 60 6 6)
  ("EXAMPLE" 73 7 7) ("NILSSON" 49 5 4) ("EXECUTINGPLAN" 19 1 1)
  ("PREIMAGE" 19 1 1) ("TRUE" 59 6 5) ("OPTIONAL" 18 1 1)
  ("SYMBOL" 24 2 2) ("TELEOREACTIVE" 24 2 2) ("BENSON" 24 2 2)
  ("PROBLEM" 53 6 6) ("EXECUTINGBRANCH" 17 1 1) ("FUNCTION" 64 7 7)
  ("SPECIFICATION" 35 4 4) ("SIDE" 28 3 3) ("POSSIBLE" 62 7 7)
  ("COMMITMENT" 21 2 2) ("DIFFERENT" 58 7 7) ("GENETIC" 15 1 1)
  ("REGRESSION" 20 2 2) ("INTELLIGENCE" 56 7 7) ("ARTIFICIAL" 55 7 7)
  ("CONCEPT" 24 3 3) ("COMMUNICATION" 14 1 1) ("CURRENT" 52 7 7)
  ("FORMULA" 23 3 3) ("MOTOR" 13 1 1) ("THEOREM" 17 2 2)
  ("INTERNAL" 39 6 6) ("BELIEF" 26 4 4) ("MENTAL" 21 3 3)
  ("BRATMAN" 21 3 3) ("LIBRARY" 21 3 3) ("BOTWORLD" 21 3 2)...))
```

Figure 10. A list of terms selected from the *Autonomous Agents* domain using the TF-IDF measure.

638 for the number of interest centers selected previously. In this process, the system does
 639 not analyze the content of the documents, but simply uses the vector model of each
 640 document collection, saving much processing time.

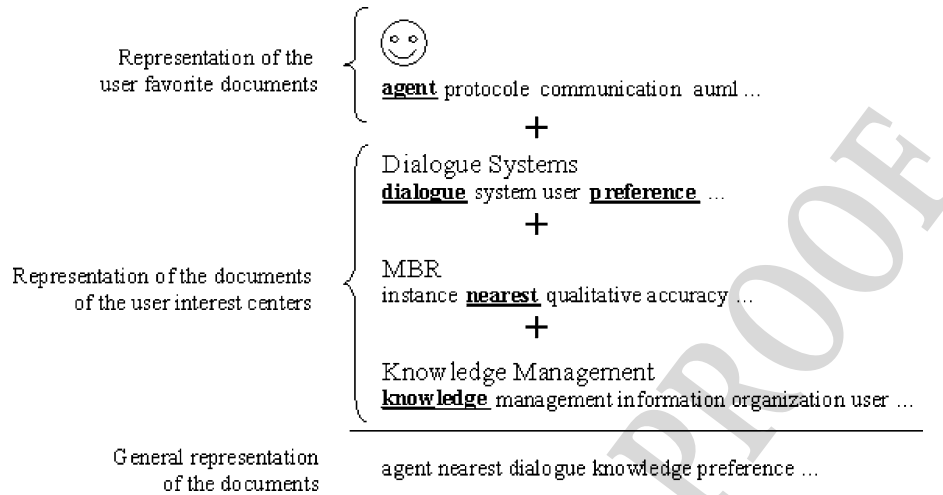


Figure 11. Combination of the document structures in a global document representation.

641 7.9. The training set

642 We use the favorite document vectors and the interest center document vectors to
643 construct the training set for the ELA algorithm. We stress that the class for each
644 vector is known (the name of the favorite folder or the name of the interest center).

645 Since the current version of ELA works strictly with symbolic data, the content of
646 vectors corresponds to binary values indicating the presence or absence of the term
647 in the document. When the FA receives a request for a new search, it looks inside the
648 user's workspace for a graph representation. If the graph is not present, it creates it
649 using the ELA method. Then, it classifies all the sites received from the SA, prunes
650 the list and saves the answer into the user's workspace. The answers can be viewed
651 through the LISP interface. We can see in Figure 7 the result of the classification and
652 the classification rate.

653 7.10. The adaptive behavior of the FA

654 In the web search application the FA adapts the user model according to the user's
655 preferences. The concept graph of the ELA method is used to represent the generic
656 user model. The adaptation of the graph can occur in two situations:

- 657 – The user adds or removes a document to or from favorites;
- 658 – The user demands a search with different interest centers.

659 In the first situation the incremental aspect of the graph makes the modification of
660 the favorites possible with little effort. This means that the user can add or remove
661 documents, and the profile is updated dynamically. Consequently, the searches al-
662 ways take into account a *good* model of the data and the system does not take a long
663 time for learning. For instance, the removing of a document requires only the update

664 of few links of the graph. Here, we consider that the structure of the documents (the
665 dictionary) changes only when the user is not working with the system (Section 7.6).

666 In the second case, we reconstruct the graph. Each modification in the interest
667 centers corresponds to several documents. Thus, it is easier to reconstruct the graph
668 with the favorite document vectors and the updated interest center vectors than to
669 add or to remove each document individually. Obviously, the involved processing
670 depends on the number of interest centers concerned and on the number of docu-
671 ments in the favorites. However, the simplicity of the graph-creating algorithm al-
672 lows a large number of documents to be represented very quickly.

673 7.11. *Search agent*

674 The simplest agents of the system are the SAs. Each agent is specialized in a specific
675 search engine, like Google, Altavista or Lycos. As the agents work in an open
676 environment, new SAs can be added to the system anytime. The search for sites is
677 made in parallel, saving time for the retrieval of sites. To make sure that the space of
678 documents analyzed by the FA is large enough, each SA must return $2N$ sites where
679 N is the number of sites specified by the user. Thus, if the system has n SAs, the FA
680 will analyze up to $2nN$ sites. Obviously, this list can be smaller than $2nN$ because SAs
681 can retrieve identical sites.

682 7.12. *MAIS construction*

683 OMAS agents [3] have been used in the construction of the system. OMAS agents
684 live in an open environment where agents communicate with KQML messages
685 transmitted using a UDP protocol. The agents always communicate in broadcast
686 mode. The content language is generally coded in LISP structures and the interaction
687 protocol is Contract Net [48].

688 A transfer agent (TA) guarantees interoperability between OMAS platforms and
689 others FIPA compliant platforms in a transparent way. The TA implements some
690 basic FIPA services like agent manager system, directory facilitator and agent
691 communication channel. The TA can translate OMAS messages in SL-0 messages
692 and make OMAS and FIPA protocols compatible. Consequently, the TA is the
693 bridge between OMAS agents and remote platforms. Figure 12 presents the archi-
694 tecture of the system.

695 From a technical point of view, the PA is composed of a LISP interface, capable to
696 send messages to the TA that works on a remote HTTP server. Then, the TA sends
697 the messages to the internal platform. We use Allegro Common Lisp as the HTTP
698 server in this application. The application has been entirely coded in LISP. Because
699 the web server is a LISP program, the interpretation of KQML and SL-0 structures
700 is extremely easy. Figure 13 presents the resources used in the application.

701 7.13. *Evaluating MAIS*

702 Evaluation of web search engines is always a difficult task. Common measures used
703 in systems based on information retrieval, like recall and precision, are not adapted

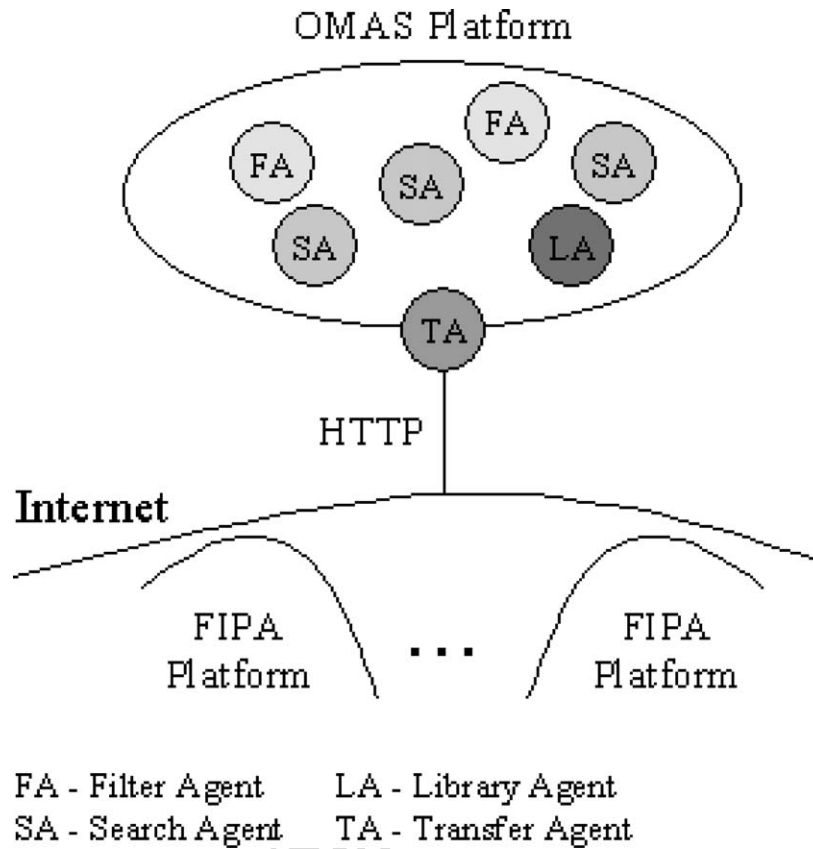


Figure 12. Communication between the OMAS platform and FIPA platforms.

704 due the dynamic behavior of the web corpus. Thereby, most works use a limited
 705 corpus of web pages. A common strategy is to score web pages using a similarity
 706 measure between results and training pages. For instance, Menczer et al. [34] used
 707 reference web pages organized in 100 categories proposed by Yahoo for the com-
 708 parison of three search engines. Nevertheless, the page datasets used for learning and
 709 test are small and static and do not represent a good sample of the Internet data. To
 710 avoid such problems, Gasparetti and Micarelli [20] used recall and precision mea-
 711 sures on the “.GOV” database provided by TREC (Text REtrieval Conference).
 712 This database contains 1.2 million pages “.gov” collected until 2002. According to
 713 the authors, the database is large enough to represent a feasible sample of web data.
 714 The authors evaluated a web page classification algorithm using three queries:
 715 “exchange office,” “educational department” and “children foundation.” The
 716 average number of results for each query is respectively 373,712, 445,964 and
 717 117,036. Such performance measures are usefull for retrieving much information
 718 used for automatic processing but, for us, a system for personalized information
 719 retrieval must take into account the human limits spent in the analysis of the results.

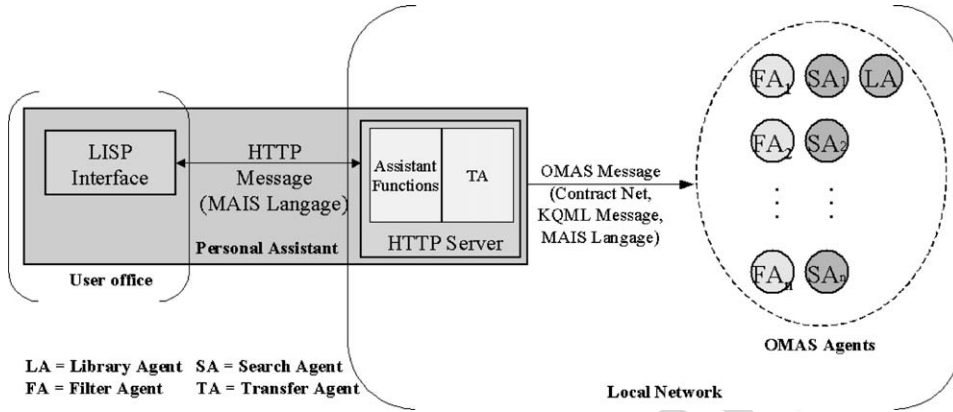


Figure 13. Software resources of the system.

720 Indeed, the user will analyze only the first results [47]. In our evaluation approach,
 721 we use the feedback of three users for estimating the quality of the results obtained
 722 directly from the web. Thus, we do not use simulate databases. The users analyze the
 723 first 20 or 30 web pages, telling how many pages are interesting to them. All users
 724 search for information about multi-agent systems and a site is considered interesting
 725 if it is related to the multi-agent research domain. We selected queries on the same
 726 domain in order to minimize ambiguities in terms of identification of interesting sites
 727 and consequently to estimate the importance of interest centers and favorite sites for
 728 the results accurately. Each user selected a number of reference web pages before the
 729 experiments considering personnel interests. We analyzed the results without con-
 730 sidering the identity of the different search engines used in the experiment, avoiding
 731 possible biases. Table 7 gives more details about the users.

732 Each user sent the following queries: “agent,” “autonomy agent,” “personal as-
 733 sistant” and “personal assistant agent.” The queries are intentionally ambiguous.
 734 The goal of the method of evaluation is to establish the importance of the users’

Table 7. Information about the users.

User	Expertise	Number of reference web pages	Interest Centers
A	Multi-agent systems	58	Adaptative agents, autono- mous agents, agents, personal assistant
B	Artificial intelligence	8	Adaptative agents, autono- mous agents, agents, personal assistant
C	None	1	—
C-int	None	1	Adaptative agents, autono- mous agents, agents, personal assistant

735 expertise, used as a reference for filtering the information. We also measured the
 736 influence of public interest centers proposed by the system through user C. User C
 737 was invited to send the queries first with no support from interest centers, and, later,
 738 using the interest centers (C-int).

739 The curves presented in Figure 14 show the performance of three widely used
 740 search engines (All-the-Web, Altavista and Google) and the results from users A (A-
 741 55) and B (B-8) with MAIS. As expected, user A obtained the best results, showing
 742 that the system is capable of taking into account the user context represented as
 743 favorite sites for personalizing results. User B also benefited from this advantage.

744 Two observations seem clear, based on the results of Figure 14:

- 745 – The first one is the limit on the performance of MAIS in relation to the quality of
- 746 the search engines;
- 747 – The second one is the variability of search engines.

748 Because MAIS does meta-searches, if all search engines used in MAIS give poor
 749 results, so does MAIS. This can be observed with the query “agent,” which is quite
 750 ambiguous. Some search engines present good performances for specific queries but
 751 poor performance for different queries. We observed for instance that All-the-Web
 752 produces good results for the query “Autonomy Agent” and poor results for the
 753 query “Personal Assistant.”

754 When the first 30 web pages have been analyzed, MAIS gave still the best results
 755 for user A (Figure 15). However, other search engines have outperformed the per-
 756 formance regarding user B in three out of four queries. The system cannot support
 757 the user when the user profile is not similar with the information he is looking for.
 758 This can be solved if the user informs the system not to use her profile for the search
 759 but only the public interest centers.

760 In order to measure the importance of the public interest centers for obtaining the
 761 information, user C (whose profile is almost empty) initially sends queries without

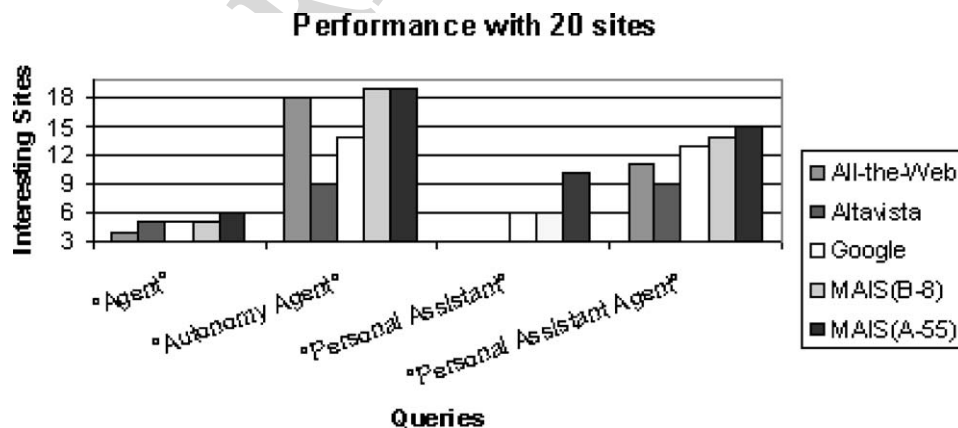


Figure 14. Performance with the first 20 sites.

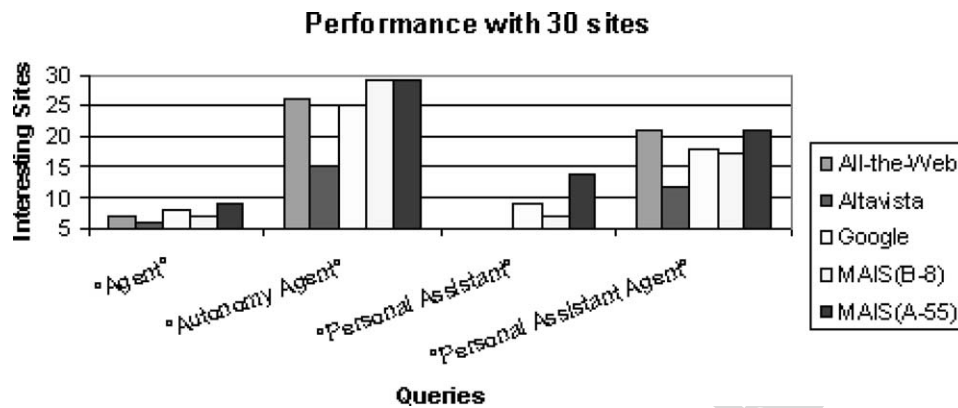


Figure 15. Performance with the first 30 sites.

762 using the interest centers (MAIS(C)), and, later, queries using the interest centers
 763 (MAIS-int). Results are shown in Figure 16. For the query "Agent," interest centers
 764 are useless because the query is actually quite ambiguous. For the query "Autonomy
 765 Agent" the interest centers improved the performance significantly when both the
 766 first 20 and 30 sites are used. For the last two queries, the performance of the system
 767 was improved when the first 30 sites are used. When only 20 sites are used, interest
 768 centers had not a high importance for the queries "Personal Assistant" and "Per-
 769 sonal Assistant Agent." This behavior demonstrates that even for reasonably accu-
 770 rate queries like "Personal Assistant Agent," search engines used in MAIS are not
 771 capable of clustering the best pages and of presenting them at the top of the list.

772 Average performance is presented Table 8. Analyzing the first 20 and 30 sites,
 773 MAIS outperforms all search engines. It shows that the system can benefit from the
 774 advantages of all specific search engines, producing a more accurate result. Sur-
 775 prisingly, the winner (C-int) contains an almost empty profile (only one personal
 776 home page), but uses public interest centers. Such results indicate that the sites

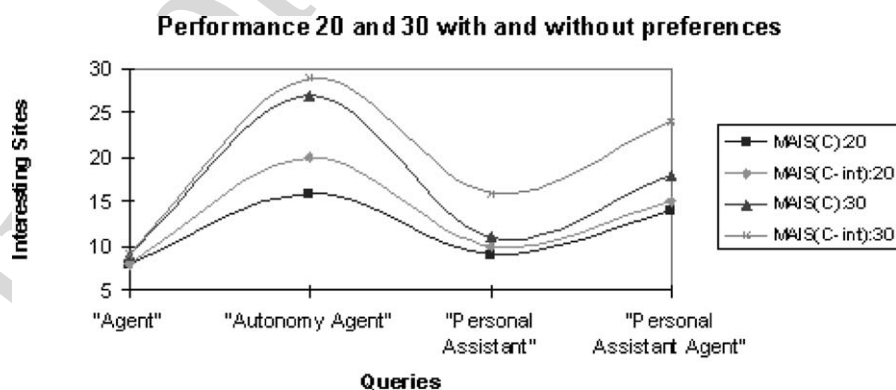


Figure 16. Results with and without interest centers.

Table 8. Average results regarding the number of sites in the results and the number of interesting sites in the results for the four queries.

With 20 sites		With 30 sites	
MAIS(C-int)	13.25	MAIS(C-int)	19.50
MAIS(A-55)	12.50	MAIS(A-55)	18.25
MAIS(C)	11.75	MAIS(C)	16.25
MAIS(B-8)	11.00	MAIS(B-8)	15.00
Google	9.50	Google	15.00
All-the-Web	8.25	All-the-Web	13.50
Altavista	6.50	Altavista	9.25

judged interesting by the users do not add important information to the system. In fact, some web pages can even decrease the accuracy of the system because they do not contain enough text and introduce noise. It explains for example why the user MAIS-C, who does not use the public centers of interest and refers only to his own personal page in his favorites can have a better result than user MAIS(B-8). Considering the results for MAIS(B-8), the system considered as interesting several professional sites related to finance, jobs, personal trainers and health. Such sites quite often used terms like “intelligent,” “time,” “personal,” “assistants,” “personal assistants,” “training,” “learning,” “planning,” “organization,” “life” or “service,” leading the system to be deceptive. In addition, when the user profile is empty or incomplete (this is the case for MAIS-C), i.e., has not a minimal number of terms (fixed empirically to 200) required for the creation of the preference models, the system ranks the sites using a voting strategy, increasing the importance of a document if several search engines point to it. Such a technique is similar to the CombMNZ method, proposed by Fox and Shall [18]. However, fusion approaches are out of the scope of this paper, because they very often are not capable of personalizing results. The system was more accurate when users selected the public interest centers, probably because they are formed of documents selected by experts, the documents are representative of the field and contain much textual information. Thus, we can conclude that the users will have to carefully analyze the sites they think appropriate before adding them to their profile.

7.14. Comparing MAIS and other multi-agent based tools

Widyantoro and coworkers [51] discuss different representations of user profiles for personalization in information systems. The authors stress the importance of using long term and short term user profiles. They introduce a technique based on explicit user feedback, where users indicate positive and negative examples. Some other works are also based exclusively on user feedback: Amalthea [6, 37], Webnaut [40] and WAIR (Web Agents for Internet Research) [53]. In such systems the user must provide a significant amount of feedback, increasing the human workload and the dependency of the system. The main difference with MAIS is how the user feedback is used for updating the user profile.

In our approach we divide the user profile in $n + 1$ parts, where n is the number of interest centers selected by the user plus the user favorites model. For us, the user favorite model represents the long term model of the user profile. The short term model is built dynamically, when the user explicitly changes of interest centers selecting new public centers of interest. We think that our approach reduces the need for user feedback because the user does not need to classify each document, but only to provide some interest centers. In addition, the system behavior does not depend exclusively on the user feedback because the system can have already many feasible models based on collections of high quality documents.

8. Conclusions

In this paper we presented a new technique of incremental learning and applied it to searching the web with a multi-agent system. Our agents need to learn from methods that represent data efficiently, that classify examples quickly and are generic (without parameters like similarity measures between examples, number of neighbors, etc). To achieve this task we represent data as a graph that reduces the space of examples. The graph is built dynamically, as the examples arrive. No treatment (pruning, attribute selection, conjunction or division of nodes) is made on the graph. To predict the class of an example, the algorithm selects an initial partition of the data and tries to find the class capable of satisfying the higher number of conditions.

Our results, when compared with other works, are very encouraging. We made several tests and could note that the algorithm presents a good performance and, moreover, can significantly reduce the amount of data necessary for learning on certain databases.

Then, we used the ELA method in a multi-agent system capable of personalizing the search for information on the web. In this application, the characteristics of ELA like simplicity for representing data, fast classification and precision were very important.

The learning method presented in this paper can reduce the problems introduced by MBR methods in particular related to the processing time or the amount of data. This leads us to believe that the method can be used in most applications where hundreds of autonomous entities exist. For instance, our method could improve the behavior of autonomous entities running in an automobile traffic simulation or in applications implementing general coordination activities.

References

1. R. Agrawal, and R. Srikant, "Mining sequential patterns," Philip S. Yu and Arbee L. P. Chen (eds.), *Proceedings of the 11th International Conference on Data Engineering - ICDE*, IEEE Press, 6 October, 3-14 pp 1995. (ISBN 0-8186-6910-1).
2. D. W. Aha, D. Kibler, and M. K. Albert, Instance-based learning algorithms. *Machine Learning*, vol. 6, pp. 37-66, 1991.
3. J.-P. Barthès, "OMAS v 1.0. Memo UTC/GI/DI/N 151, Université de Technologie de Compiègne," *Département de Génie Informatique*, January, 2002.

- 849 4. S. D. Bay, and M. J. Pazzani, "Detecting change in categorical data: Mining contrast sets," *Knowledge*
850 *Discovery and Data Mining*, pp. 302–306, 1999.
- 851 5. S. Benson, "Inductive learning of reactive action models," *ICML '95*, pp. 47–54, 1995.
- 852 6. D. Billsus, and M. Pazzani, A hybrid user model for news story classification. in *Proceedings of the*
853 *Seventh International Conference on User Modeling (UM '99)*, Banff, Canada. 1999.
- 854 7. C. Blake, C. J. Merz, "UCI Repository of machine learning databases – [www.ics.uci.edu/mllearn/](http://www.ics.uci.edu/mllearn/MLRepository.html)
855 [MLRepository.html](http://www.ics.uci.edu/mllearn/MLRepository.html)." Irvine, CA: University of California, Department of Information and Com-
856 puter Science.
- 857 8. L. B. Brooker, D. E. Goldberg, J. H. Holland, "Classifier systems and genetic algorithms," *Artificial*
858 *Intelligence*. no. 40, pp. 235–282, 1989.
- 859 9. W. Brenner, R. Zarnekow, H. Wittig, "Intelligent software agents," *Springer-Verlag*. 1998. ISBN
860 3-540-63411-8
- 861 10. H. Brighton, C. Mellish, "Advances in instance selection for instance-based learning algorithms,"
862 *Data Mining and Knowledge Discovery. Kluwer Academic Publishers*. no. 6, pp. 153–172, 2002.
- 863 11. J. Chapelle, O. Simonin, J. Ferber, "How situated agents can learn to cooperate by monitoring their
864 neighbors'satisfaction," F. van Harmelen (ed.), in *ECAI'02*, IOS Press, pp. 68–72, 2002. ISBN 1-58603-
865 257-7.
- 866 12. P. Cunningham, B. Smyth, and T. Veale, "On the limitations of memory based reasoning," in *Pro-*
867 *ceedings of the 2nd European Workshop on Case-Based Reasoning*, Paris, 1994.
- 868 13. K. Dorer, "Behavior networks for continuous domains using situation-dependent motivations," in
869 *Proceedings of the IJCAI'99*, pp. 1233–1238, 1999. <http://citeseer.nj.nec.com/dorer99behavior.html>.
- 870 14. P. W. Eklund, A. Hoang, "A performance survey of public domain supervised machine learning
871 algorithms," in *Technical Report of Knowledge Visualisation and Ordering*, [http://www.kvocentral.](http://www.kvocentral.com/kvopapers/7paper.pdf)
872 [com/kvopapers/7paper.pdf](http://www.kvopapers/7paper.pdf).
- 873 15. B. Faltings, "Qualitative models as indices for memory-based prediction," *IEEE Expert*, vol. may–
874 june, pp. 47–53, 1997.
- 875 16. Foundations for intelligent physical agents <http://www.fipa.org>.
- 876 17. R. E. Fikes, N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to
877 problem solving," *Artificial Intelligence*, vols. 3–4, no. 2, pp. 189–208, 1971.
- 878 18. E. A. Fox, and J. A. Shaw, "Combination of multiple searches. The second text retrieval conference
879 (TREC-2)," in *Government Printing Office Washington D.C.*, Gaithersburg, MD, USA, March, 1994.
- 880 19. J. H. Friedman, R. Kohavi, Y. Yun, "Lazy decision trees," in S. Howard and S. Ted (eds.), *Proc. of the*
881 *Thirteenth National Conference on Artificial Intelligence*, AAAI Press pp. 717–724, 1996.
- 882 20. F. Gasparetti, and A. Micarelli, "Adaptive web based on a colony of cooperative distributed agents,"
883 in M. Klusch, A. Omicini, S. Ossowski and H. Laamane (eds.), *Proceedings of the 7th Cooperative*
884 *Information Agents*. Springer-Verlag. LNAI 2782., pp. 168–183 Helsinki, August, 2003. ISBN 3-540-
885 40798-7.
- 886 21. J. A. Gonzales, L. B. Holder, and D. J. Cook, "Graph-based concept learning," *Proceedings of the*
887 *Fourteenth Annual Florida AI Research Symposium*, pp. 377–381, 2001.
- 888 22. J. H. Holland, "Escaping brittleness: The possibilities of general-purpose learning algorithms applied
889 to parallel rule-based systems," In R. S. Michalski, J. G. Carbonell, T. M. Mitchell (eds.), *Machine*
890 *Learning, an Artificial Intelligence Approach*, Morgan Kaufman, vol 2, 1986.
- 891 23. J. H. Holmes, P. L. Lanzi, W. Stolzman, and S. W. Wilson, "Learning classifier systems: New models,
892 successful applications," *Information Processing Letters*, vol 2, ed. 1, pp. 23–30, 2002.
- 893 24. Java agent development framework, <http://sharon.cselt.it/projects/jade/>
- 894 25. S. Kasif, S. Salzberg, and D. Waltz, J. Rachlin, and David, Aha, "A probabilistic framework for
895 memory-based reasoning," *Artificial Intelligence*, vol. 104, no. 1–2, pp. 287–311, 1998.
- 896 26. R. Kohavi, "Bottom-up induction of oblivious read-once decision graphs: strengths and limitations,"
897 *Proceedings of the National Conference on Artificial Intelligence*, pp. 613–618, 1994.
- 898 27. R. Kohavi, D. Sommerfield, J. Dougherty, "Data Mining using MLC++," in *IEEE Tools With*
899 *Artificial Intelligence Best Paper Award*, 1996. <http://robotics.stanford.edu/ronnyk/mlc96.ps.Z>
- 900 28. R. Kozierok, and P. Maes, "A learning interface agent for scheduling meetings," In *Proceedings of the*
901 *Fourteenth International Workshop on Intelligent User Interfaces*, pp. 81–88, Orlando, USA, 1993.

- 902 29. Y. Lashkari, M. Metral, and P. Maes, "Collaborative interface agents," *Proceedings of the Twelfth*
903 *National Conference on Artificial Intelligence*, vol. 1, AAAI Press, Seattle, WA, 1994.
- 904 30. H. Lieberman, "Autonomous interface agents," *Proceedings of the ACM Conference on Computer and*
905 *Human Interface*, Atlanta, March, 1997.
- 906 31. H. Lieberman, "Intelligent Profiling by Example," *Proceedings of the International Conference on*
907 *Intelligent user Interfaces (IUI 2000)*, Santa Fé, January 14–17, 2001.
- 908 32. P. Maes, "Learning behavior networks from experience," *Proceedings of the First Conference on*
909 *Artificial Life*, F. J. Varela and P. Bourguine (in eds.) MIT Press/Bradford Books, 1992.
- 910 33. P. Maes, "Modeling adaptative autonomous agents," *Artificial Life Journal*, no. 1, vol. 1–2, MIT
911 Press, pp. 135–162, 1994.
- 912 34. F. Menczer, G. Pant, P. Srinivasan, and M. E. Ruiz, "Evaluating topic-driven web crawlers," *Pro-*
913 *ceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in*
914 *Information Retrieval*, pp. 241–249, 2001.
- 915 35. D. Michie, D. J. Spiegelhalter, and C.C. Taylor, "Machine learning, neural and statistical classifica-
916 tion," *Ellis Horwood*, 1994, ISBN 0-13-106360-X.
- 917 36. S. E. Middleton, "Interface agents: A review of the field," *Technical Report ECSTR-IAM01-001*,
918 University of Southampton, UK, 2001.
- 919 37. A. Moukas "Amalthaea: information discovery and filtering using a multiagent evolving ecosystem,"
920 *Applied Artificial Intelligence: An International Journal*, vol. 11, n. 5, pp. 437–457, 1997.
- 921 38. D. Maulsby, and I. H. Witten, "Teaching agents to learn: from user study to implementation," *IEEE*
922 *Computer*, vol. 30, no. 11, pp. 36–44, 1997.
- 923 39. N. J. Nilsson, "Teleo-reactive programs for agent control," *Journal of Artificial Intelligence Research*,
924 vol. 1, pp. 139–158, 1994.
- 925 40. Z. Z. Nick, and P. Themis, "Web search using a genetic algorithm," *IEEE Internet Computing*, pp. 18–
926 26, March–April, 2001.
- 927 41. E. Oliveira, "Agent, advanced features for negotiation and coordination," M. Luck (ed.), *ACAI 2001*,
928 *LNAI 2086*, In: Springer-Verlag, pp. 173–186, 2001.
- 929 42. J. R. Quinlan, "Induction of decision trees," *C4.5 : Programs for Machine Learning*, Kluwer Academic
930 Publishers, pp. 81–106, Netherlands, 1986.
- 931 43. J. R. Quinlan, C4.5: "Programs for machine learning," *Morgan Kauffman Publishers*, California, 1993.
- 932 44. C. K. Riesbeck, R. C. Schank, "Inside case-based reasoning," *Lawrence Erlbaum Associates*, ISBN 0-
933 89859-767-6, 423 EUA, 1989.
- 934 45. G. Salton, "Automatic text processing: The transformations, analysis and retrieval of information by
935 computer," *Addison-Wesley*, 1989.
- 936 46. M. Shoenauer, and M. Sebag, "Incremental learning of rules and meta-rules," In: B. W. Porter and R.
937 Mooney (eds.), *Machine Learning*, Morgan Kaufmann, Los Altos/Palo Alto/San Francisco, pp. 49–57,
938 1990.
- 939 47. C. Silverstein, M. R. Henzinger, J. Marais, and M. Moricz, "Analysis of a very large altavista query
940 log," *ACM SIGIR*, no. 33, pp. 6–12, 1999.
- 941 48. R. G. Smith, "A framework for distributed problem solving," *Proceedings of the 6th International joint*
942 *Conference on Machine Learning*, pp. 836–841, Tokyo, Japan, 1979.
- 943 49. P. E. Utgoff, "An improved algorithm for incremental induction of decision trees," *International*
944 *Conference on Machine Learning*, pp. 318–325, 1994.
- 945 50. C. Watkins, "Learning from deployed rewards," *PhD Thesis, King's College*, Cambridge, 1989.
- 946 51. D.H. Widyantoro, T.R. Ioerger, and J. Yen, "Learning user interest dynamics with a three-descriptor
947 representation," *Journal of the American Society for Information Science*, vol. 52, no. 3, pp. 212–225,
948 2001.
- 949 52. D. R. Wilson, A. R. Martinez, "Instance pruning techniques," *Machine Learning*, vol. 38, no. 3,
950 pp. 257–286, 2000.
- 951 53. B. Zhang, and Y. Seo, "Personalized webdocument filtering using reinforcement learning," *Applied*
952 *Artificial Intelligence*, no. 15, pp. 665–685, 2001.
- 953