Memory Based Reasoning and Agents Adaptive Behavior

Ana S. Aguera[1], Alejandro Guerra[2] and Manuel Martínez[2]

[1]Instituto de Investigaciones Eléctricas
Av. Reforma 113
Temixco, Morelos.
México
aaguera@iie.org.mx
[2]Departamento de Inteligencia Artificial
Universidad Veracruzana
Sebastián Camacho # 5
Xalapa, Veracruz
México
Tel (28) 17 29 57 Fax (28) 17 28 55
aguerra@mia.uv.mx, mmartine@mia.uv.mx

Abstract The main purpose of this work is to explore the application of Memory Based Reasoning (MBR) to adaptive behavior in agents. We discuss the design of an interface agent for e-mail management assistance as a prototype for the experimental assesment of an MBR learning algorithm performance. Results are discussed and a brief summary of conclusions follows, as well as a sketch of future work to be done.

1. Introduction

In this paper we explore the possibility of applying Memory Based Reasoning (MBR) to adaptive behavior in interface agents. We start by giving a definition of what is to be understand as an interface agent, followed by a discussion about the importance of learning in order for the agent to adapt to the interface user behavior. Next a detailed description of the MBR learning algorithm is given, followed by discussion of Hermes-II, an interface agent prototype for e-mail management assistance. The final sections are devoted to the presentation of experimental results and a summary of the main conclusions obtained.

Our work is related to Maes and Kozierok agent which learns from three different sources: User observation, user feedback and explicit training from the user [Maes and Kozierok 93]. The agent proposed by Gary Boone [Boone 98] learns concepts extracted from e-mail useful features, and later applies these concepts to learn the user actions.

The method here described does not generate rules or concepts, uses only one source for learning, and as shall be shown, has a high degree of predictability and a high learning rate [Aguera 99].

2. Interface Agents

The agents we consider are part of an embedded system [Kaelbling 93]. An embedded system is a pair agent-world. The *world* or *environment* may be described by a 3-tuple <E,A,T> where E is the set of possible *world states*, A is the set of *actions* the agent can take on the world, and T is the world *transition function* T:EXA→E, which maps each pair (state,action) into a new state. T may be deterministic or stochastic, defined by a family of conditional probability functions. In general, an agent may be defined as a 3-tuple <P,R,B,> in which P is the set of *perceptions* of the agent (or the set of the world states that the agent distinguishes), R is the *reinforcemen*t or *reward* function R:E→R which assigns to each state a real number (R denotes the set of real numbers), and B is the *behavior function* B:(P,R)→A which maps each pair perception-reward into an action (in many applications E and P are identical). R is related to the agent goals. In fact this formalism is related to the so called *PAGE* description (*P*erception, *A*ction, *G*oals, *E*nvironment).

The system then evolves as follows: the agent selects an action a, given that the world is in state e, then the world moves to a new state e'=T(e,a), which has associated a reward r, the agent percieves e' as p' and selects a new action a'=B(p',r), the cycle repeats and the agent is said to behave uninterruptedly.

It is said that the agent exhibit learning behavior if it has a built-in algorithm that modifies function B in order for the agent to acquire (or approximate) maximum reward, usually given by the expected total reward [Kaelbling 94]. This is considered to be the agent´s goal.

An interface agent [Maes 94], [Nwana 96], [Mitchell et al. 94], [Laurel 90] is a software component complementary to a user interface, in which agent and user engage in a colaborative process which includes communication, event monitoring and task ejecution. The agent's world is the interface which in turn is the user-agent interaction medium. The agent acts independently of the interface in benefit of the user´s goals and without his explicit intervention [Lieberman 97]. The metaphore for this kind of interaction is that of a personal assistant [Maes 94].

A personal assistant must have a great deal of autonomy so that it can act independently in the ejecution of specific tasks, without constant external intervention or supervision. It must be robust so that it is less vulnerable to failure, and it should have some knowledge about the user's habits and preferences. Since in general the user's behavior changes in time, so that it is not always possible to predict his future actions by means of fixed rules, the agent must exhibit some kind of adaptive functionality in order to be useful.

In this paper we present a learning algorithm based on MBR that makes an agent adapt to user's needs, illustrating its use through the prototype Hermes-II, an e-mail management assistant. The following section describes the basic MBR algorithm used throughout.

3. Memory Based Reasoning

Different MBR algorithms can be found in the literature [Stanfill and Waltz 86], [Kasif et al. 97], each one with its own specific features depending on the application domain.

Nevertheless a general inclusive description of MBR algorithms may be given.

An MBR algorithm acts on the following components:

- *Mem:* a memory containing N examples,
- X : a new example described by the vector $\{x_1,x_2,...,x_m\}$ where $\{x_1,x_2,...,x_{(m-1)}\}$ are the prediction attributes and x_m is the class attribute,
- Y : a memory example $\{y_1,y_2,...,y_m\}$, with the same structure as X,
- V_k : a vector containing the k examples in memory closest to X.

The algorithm works as follows:

1. For each example Y in *Mem* compute the distance $d(X,Y)$ (the distance function employed in our work is defined below).
2. Search for the set V_k (using a specific criterion for selecting the k-closest neighbors to X in *Mem*).
3. For each element in V_k obtain the values of the class attributes $y_{m's}$.
4. Select the class value y_m that best predicts x_m .

Once a prediction has been made, a confidence measure is computed. We used a confidence measure proposed by [Maes 93]:

$$\left(1 - \frac{\frac{d_{predic}}{n_{predic}}}{\frac{d_{other}}{n_{other}}} \right) * \frac{K_{total}}{K}$$

where (vector $\{x_1,x_2,...,x_{(m-1)}\}$ is called the new situation):

- K is the number of cases considered to make the prediction of x_m
- d_{predic} is the distance of the attribute vector $\{y_1,y_2,...,y_{(m-1)}\}$ in V_k closest to $\{x_1,x_2,...,x_{(m-1)}\}$ with the same class attribute y_m as the predicted one.
- n_{predic} is the number of cases closest to the new situation within a treshold value with the same class attribute as the predicted one.
- d_{other} is the distance to the closest example with a different class attribute.
- n_{other} is the number of cases closest to the new situation within a threshold value with class attribute not equal to the predicted one.
- K_{total} i s the sum $n_{predic}+n_{other}$.

If the result is negative, the confidence is 0. If there is no n_{other} value then we set the confidence equal to 1.

We used as distance function the weighted value difference metric (**wvdm**) defined as follows.

Let X be a new attribute vector, Y an example in Mem and $G=\{g_1,g_2,...,g_n\}$ the set of possible values of the class attribute (x_m). The distance between an example Y and the new vector X is

$$d(X,Y) = \sum_{i=1}^{m-1} vdm(x_i, y_i)$$

The value difference between two values x and y of a given attribute is computed by

$$vdm(x, y) = \sum_{g \in G} \left| \frac{f_g(x)}{f_m(x)} - \frac{f_g(y)}{f_m(y)} \right|^2$$

Where

- $f_g(x)$ is the frequency of x within all cases having the g-th class value.
- $f_m(x)$ is the frequency of x in all memory examples.

Since this metric assumes attribute independence, then the ratio $f_g(x)/f_m(x)$ is an estimate of the conditional probability $p(g/x)$ that value g occurs given that the attribute considered takes value x. It is important to weight the relevance of attributes, which can be attained using wieghts:

$$w(x) = \sqrt{\sum_{g \in G} \left| \frac{f_g(x)}{f_m(x)} \right|^2}$$

Thus, the **wdvm** is given by

$$d(X,Y) = \sum_{i=1}^{m-1} vdm(x_i, y_i) w(x_i)$$

4. What is Hermes-II?

Hermes-II is an interface agent in experimental development, its task is to assist a user in e-mail management incorporating MBR to learn the user's behavior and eventually automate repetitive actions of the user. We present the PAGE (Perception,Action,Goals,Environment) description of the agent and discuss what the agent has been able to achieve.

By e-mail management we mean the presentation and administration of e- mail messages. In a first version (Hermes-I, [Aguera and Guerra 98]) the agent's task was simply to retrieve messages and show them to the user. Hermes-II incorporates the ability to learn from the user's behavior how to manage new messages, suggesting or deciding what to do with them (store, delete, answer, etc.) according to user's past preferences. Hermes-II stores its experience in a memory which is used as the basis of the MBR algorithm. Wenever a new mail message arrives Hermes-II tries to select an appropiate action (store, delete, etc.) according to past user´s behavior. To achieve this it selects those examples closer to the new one and chooses the most promising action, in terms of the criteria discussed in the MBR algorithm section. Next, the agent

computes the predictive confidence and, given a threshold, decides either to take an action, make a suggestion to the user, or make no suggestion and wait until it gathers enough information. Hermes-II interacts with the user through a graphic interface.

4.1 PAGE Description of Hermes-II

- **Perceptions**

  Hermes has three sensors to gather information from its environment:

  - User's Information (UI*):* this sensor allows the agent to identify the user and locate his mail box, thus

    $$UI=(server,login,password)$$

  - *New Mail (NM)*: This sensor detects the arrival of new messages, it is a binary sensor:

    $$NM = \begin{cases} 0 \text{ if there are no new messages in the mail box} \\ 1 \text{ if there are new messages in the mail box} \end{cases}$$

  - *User Action*: This sensor recognizes when the user has taken a particular action over an active message, the parameters delivered by this sensor are:

    $$UA = \begin{cases} (message, action) & \text{if a button of the GUI is activated} \\ null & \text{otherwise} \end{cases}$$

Where *message* is a message description and *action* is the action associated with the selected button.

- **Actions**

The set of Hermes-II actions is subdivided in *basic actions* and *high level action*s.

- *Basic actions* are defined as those that the agent can activate on the user e-mail interface: read,delete, answer, forward, store.
- *High level actions* are those defining the behavior of the agent: *reasoning* and *learning*. Reasoning generates the basic actions and learning adapts the results of reasoning about the user's behavior.

- **Goals**

Hermes-II has as a global goal to adapt to the user's behavior in such a way that the user is satisfied, that is that the user rejects the smallest number of suggestions made by the agent. Thus Hermes-II should try to minimize the ratio

$$N_r/N_s$$

computed from the last p suggestions, where N_r is the number of rejections and N_s is the total number of suggestions. To minimize the above ratio is equivalent to maximizing

$$(N_s-N_r)/N_r=1-N_r/N_s$$

which is an estimate of correct prediction probability.

- **Environment**

Hermes-II environment is conformed by the e-mail application, the system which it resides and the user with which it interacts through the GUI. The environmet dynamics is defined by two sources: changes in the environment produced by the system (eg. message arrivals) and the actions taken by the user. The environment is non-deterministic since it is not possible to predict its next state given the present perception of the agent (for example, it is not possible to know what the user's next action will be, or if there will be a new message at all). The figure 1 shows the interaction of Hermes with its environment.

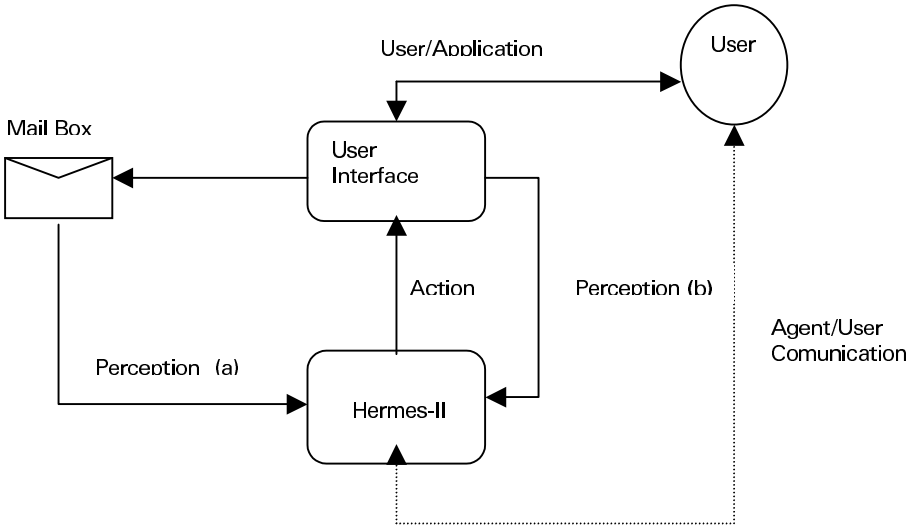


Fig. 1. Hermes-II through its senses can have two types of perceptions: (a) new message arrivals and (b) user actions on the active mesage. Each of these types activates a high level action that eventually triggers a set of basic actions.

4.2 Environment States and Transition Function

Since the environment is nondeterministic it is not possible to make a precise description of the transition function, although it is feasible to assume a probabilistic transition function, such as we did for the simulation study as described later.

4.3 Learning Elements

Hermes-II learning element is the MBR algorithm described earlier. Hermes-II selects actions by *recalling* past situations recorded in its memory. The figure 2 shows the interaction between the agent's perceptions and the learning algorithm.

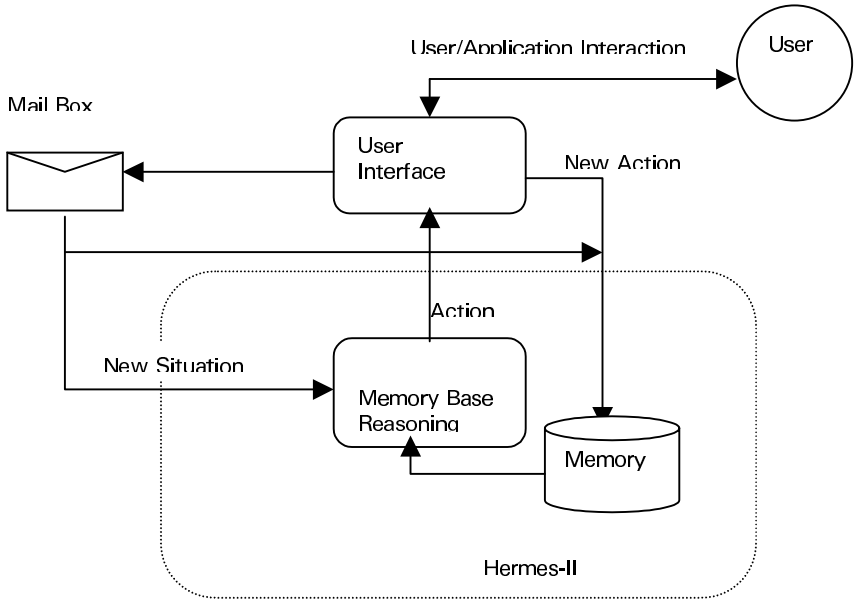


Fig. 2. The arrival of a new message activates the reasoning mechanism to generate a basic action. A new action by the user together with the new situation updates Hermes-II memory.

5. Results and Conclusions

There is a user model underlying the MBR algorithm [Stanfill and Waltz 86]. This model assumes that the attributes are conditionally independent given the user's action; that is the user's behavior can be modelled by a simple bayesian network. Using this fact we were able to simulate several users with different behavior patterns by making use of a bayesian network simulator [Aguirre 98]. We also applied Hermes-II to two real users. As a measure of performance we used the *predictive confidence*, as earlier defined, the *percentage of correct predictions* defined by

*100*Number of correct predictions/Total number of predictions*,

and the *accepted suggestions (among the last p) proportion* given by

(1-Nr/Ns)

The two first measures are related directly with the global agent's behavior, while the last one relates the predictive confidence with the agent's ability to make a right guess.

We now present two examples and the results obtained. The first user is a simulated user, the second one is a real user.

5.1 Simulated User

In our simulations we assumed that the user shows regular patterns of behavior only for some attribute values, for example whenever she or he receives an e-mail which is known to be a comercial advertisement (say by looking at the *sender*) then he or she immediately deletes it without even opening it. In other cases he or she receives an e-mail from a colleague, then depending on the subject he or she will either to read it immediately or store it for later reference. We call the examples belonging to this class *structured examples*. There some other cases in which the user does not follow any particular behavior pattern, we call these cases *unstructured examples*. For real users it is not possible to distinguish a priori these two classes, but anyway the agent learns the stuctured behavior and has a good overall performance.

In this example we simulated a user with the following structured (probabilistic) behavior:

Sender	**Day**	**Time**	**Subject**	**Action**	**Probaility**
Aguerra	*	*	*Hola *	Read	0.9
Aguerra	*	*	*Hola *	Reply	0.1
Aguerra	*	*	*Saludos *	Read	0.9
Aguerra	*	*	*Saludos *	Reply	0.1
Aguerra	*	*	*Tesis *	Save	1.0
Aguerra	*	*	*Avance *	Save	1.0
Aguerra	*	*	*Agentes *	Save	0.7
Aguerra	*	*	*Agentes *	Forward	0.3
Mmartine	*	*	* Avance *	Reply	1.0
Mmartine	*	*	* Tesis *	Reply	1.0
Mmartine	*	*	* Curso *	Read	0.8
mmartine	*	*	* Curso *	Save	0.1
mmartine	*	*	* Curso *	Reply	0.1
mmartine	*	*	* Conferencia *	Read	0.8
mmartine	*	*	* Conferencia *	Save	0.1
mmartine	*	*	* Conferencia *	Reply	0.1
mmartine	*	*	* Bienvenida *	Read	0.8
mmartine	*	*	* Bienvenida *	Reply	0.2
mmartine	*	*	* Chiapas *	Read	0.7
mmartine	*	*	* Chiapas *	Save	0.2
mmartine	*	*	* Chiapas *	Forward	0.1
mmartine	*	*	* Interés *	Read	0.7
mmartine	*	*	* Interés *	Save	0.2
mmartine	*	*	* Interés *	Forward	0.1
agonzale	*	*	* Biblioteca *	Delete	0.8
agonzale	*	*	* Biblioteca *	Read	0.2
nejacome	*	*	* Hola *	Read	1.0
nejacome	*	*	* ondas *	Read	1.0
ceuribe	*	*	*	Read	0.8
ceuribe	*	*	*	Reply	0.2
e-Alert	*	*	*	Delete	1.0
psj	*	*	* Saludos *	Read	1.0
psj	*	*	* Hola *	Read	1.0
psj	*	*	* Ondas *	Read	1.0
psj	*	*	* casa *	Read	0.8
psj	*	*	* casa *	Reply	0.2
psj	*	*	* house *	Read	0.8
psj	*	*	* house*	Reply	0.2
psj	*	*	* Barbara *	Read	0.8
psj	*	*	* Barbara *	Reply	0.2

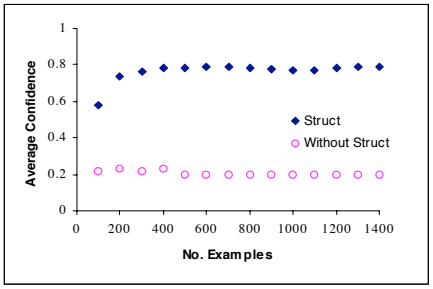
jgutierr	*	*	*	Reply	1.0
cperez	*	*	Ricardo *	Reply	1.0
cperez	*	*	* Tesis *	Reply	1.0
cperez	*	*	* Solaris*	Reply	1.0
cperez	*	*	* Hola *	Read	0.7
cperez	*	*	* Hola *	Reply	0.2
cperez	*	*	* Hola *	Forward	0.1
cperez	*	*	* Saludos *	Read	0.7
cperez	*	*	* Saludos *	Reply	0.3
jgonzale	*	*	* Tesis *	Save	0.7
jgonzale	*	*	* Tesis *	Reply	0.3
jgonzale	*	*	* Avance *	Save	0.7
jgonzale	*	*	* Avance *	Reply	0.3

The probability column shows conditional probabilities of the action given the attribut values, for example, the probability at the end of the first row is

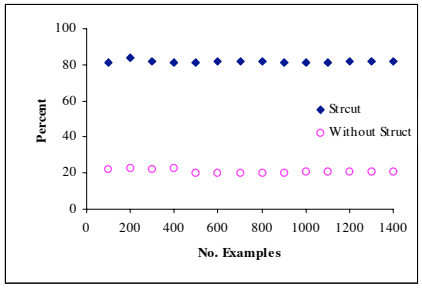
Prob(Action=read/Sender=aguerra,Day=*,Time=*,Subject=Hola)=0.90

"*" means that the attribute value is irrevelant, it is a dummy value. In our example, it means that the day of the week, and the time (hour) that the mesage is received are not relevant for the action that is taken. The above reprsentation also means that in the same situation with probability 0.10 the user will take some other action (with uniform probability) distinct from *read*. For all other attribute combinations not shown in the table, it is assumed that the user takes an action using a uniform probability distribution over the set of possible actions (these cases belong to the unstructured examples).

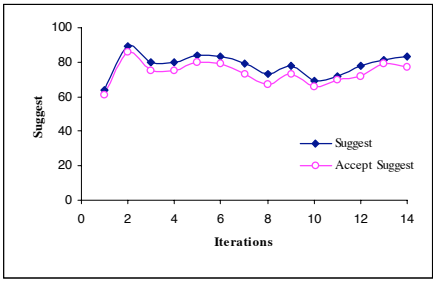
The results of the simulation are shown below.



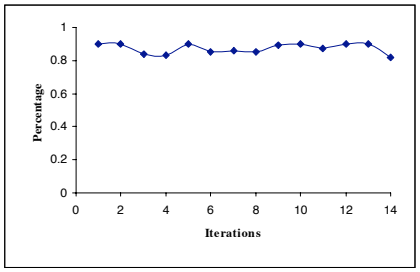
Accumulated predictive confidence



Accumulated percentage of correct predictions



Suggestions/Accepted Suggestions

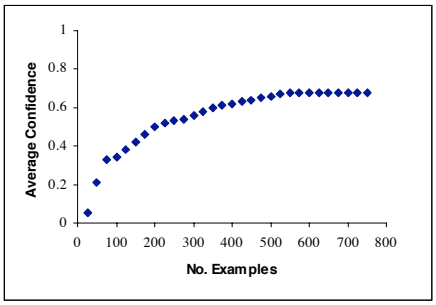


Percentage of accepted suggestions mean

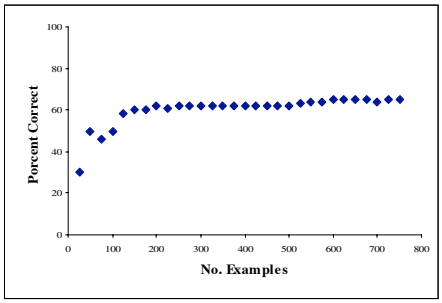
As expected the agent learns very quickly the user's behavior for the structured class, and it will learn nothing within the unstructured examples class, since the actions in this case are selected randomly. Notice from the last two graphs that the overall behavior performance measured in terms of accepted suggestions is quite acceptable, being very close to one. This is so because when the agent is confronted with unstructered examples it will make no suggestion since no matter how many of these are stored in memory, the predicted confidence of them will seldom exceed the threshold.

5.2 Real User

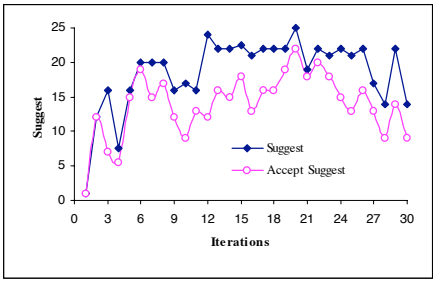
We used files generated during several months for two real users and then aply on them Hermes-II. The results obtained for one of this users are summarized in the following graphs.



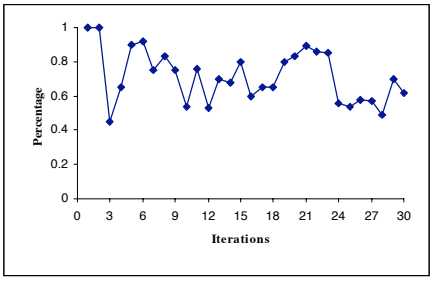
Accumulated predictive confidence



Accumulated percentage of correct predictions



Suggestions/Accepted Suggestions



Percentage of accepted suggestions mean

As can be noted the performance measures converge more slowly than in the simulated users case. This is so since a real user shows less regularities in his behavior than a simulated one.

The graphs show that confidence prediction increases with memory size, reaching a value of 0.7 for 450 examples. Percentage of correct predictions shows a similar behavior. The mean of accepted suggestions is very low at the beginning, but from the 10-th iteration on it starts to be in an acceptable level. From this point on the algorithm begins detection of regularities in the user's behavior.

From this results we can point out the following conclusions:

- MBR algorithms prove to be useful in adaptive agents
- It is feasible to use MBR with value difference metric to make interface agents learn from users behavior.
- In general MBR may be quite useful when applied to systems that generate huge data bases
- A limitation of MBR as here implemented is that it assumes conditional independence of attributes. It is possible to incorporate to the agent's structure a preprocessing module that "discovers" the agent's behavior structure , for example through the use of bayesian networks learning algorithms [Sucar and Martínez 98], which will make the learning algorithm more efficient.
- Another limitation of Hermes-II is its inability to forget, improvement in the agent's performance in the long run may be achieved by adding a forgetness module to Hermes-II.
- Hermes-II architecture, and the learning MBR algorithm are not restricted to interface agents, we can think of many other applications which involve classification and prediction, such as medical or educational applications.
- If the number of attributes is large, as memory size increases the algorithm may turn out to be quite slow; a possible solution is to think in parallel processing.

6. Bibliography

[Aguera 99] Aguera, A. S. *Razonamiento Basado en la Memoria y Comportamiento Adaptivo de Agentes*, Tesis para obtener el grado de Maestra en Inteligencia Artificial, Universidad Veracruzana, 1999.

[Aguera and Guerra 98] Aguera, A.S., and Guerra A. *Aprendizaje en Agentes Autonomos*, XIV Seminario Internacional de Ingeniería y Ciencias Básicas, Apizaco, Tlax. México,1998.

[Aguirre 98] Aguirre, A.E., *Bayes Versión 1.0*, Manual del Usuario, Maestría en Inteligencia Artificial, México, 1998.

[Boone 98] Boone, G., *Concept Features in Re: Agent, an Intelligent Email Agent*, to appear in the Second International Conference on Autonomous Agents (Agents 98), Minneapolis/ St Paul, May 10-13, 1998.

[Kaelbling 94] Kaelbling, L.P., *Learning in Embedded Systems*, Cambridge Masachussets: The MIT Press., 1993.

[Kasif et al 97] Kasif S, Salzberg S., WaltzD., Rachlin J., Aha D., *Towards a Framework for Memory-Based Reasoning*, 1997.

[Laurel 90] Laurel B., *Interface Agents: methaphors with character*, in: The Art Human Computer Interface Design, editor Brenda Laurel, Ed. Addison-Wesley, United States, 1990.

[Lieberman 97] Lieberman H., *Autonomous Interface Agents*, MIT Media Lab, 1997.

[Maes and Kozierok 93] Maes P., and Kozierok R., *Learning Interface Agents*, in: Proc. Of the Eleventh National Conf. On Artificial Intelligence, 459-465, Whashington, D.C., 1993, MIT Press.

[Maes 94] Maes P., *Agents that Reduce Work and Information Overload*, Communications of the ACM, 37(7), 31-40, 1994.

[Mitchell et al, 94] Mitchell T., Caruana R., Freitag D., and Zabowski D., *Experience with a Learning Personal Assistan*t, Communications of the ACM, 37 (7), 81-91, 1994.

[Nwana 96] Nwana H.S., *Software Agents: An Overview*, Knolowdge Enginnering Review, 11 (3), 205-244, 1996.

[Stanfill and Waltz 86] Stanfill C. and Waltz D, *Toward Memory-Based Reasoning*, Communications of the ACM, vol 29, num 8, December 1986.