Fuzzy Decision Tree for User Modeling From Human-Computer Interactions

Marc Damez
Pôle IA, LIP6, UPMC
8 rue du Capitaine Scott,
75015, Paris, France
Marc.Damez@lip6.fr

Thanh Ha Dang*
Pôle IA, LIP6, UPMC
8 rue du Capitaine Scott,
75015, Paris, France
ThanhHa.Dang@lip6.fr

Christophe Marsala
Pôle IA, LIP6, UPMC,
8 rue du Capitaine Scott,
75015 Paris, France
Christophe.Marsala@lip6.fr

Bernadette Bouchon-Meunier
Pôle IA, LIP6, CNRS,
UPMC, 8 rue du Capitaine Scott,
75015, Paris, France
Bernadette.Bouchon-Meunier@lip6.fr

*Corresponding author.
Tel: (+ 33) 1 44 27 87 51
Fax: (+ 33) 1 44 27 70 00

Fuzzy Decision Tree for User Modeling

From Human-Computer Interactions

Abstract

Providing helpful analysis of cognitive process is one of the main concerns for user modeling and adaptive hypermedia conception. In this paper, a work is presented that looks very closely to the Human Computer Interactions and exploits these interactions to help novice computer users. We think that all the cognitive actions involved in a task resolution reflect the level of expertise that the user has with the interface [1]. In this paper, we describe an application that learns these cognitive processes by fuzzy decision tree technique and discriminates novice from experimented user automatically. Once the discrimination is done, the system can provide an advice to the novice computer user with a contextual help.

Keywords

Fuzzy decision tree, traces of navigation, user modeling.

I. Introduction

Providing helpful analysis of cognitive process is one of the main concerns for user modeling and adaptive hypermedia conception. In this paper, we present the TAFPA (*Tree Analysis For Providing Advices*) software (called hereafter TAFPA for short). This system uses a learning agent, a fuzzy decision tree builder, which works on traces of human-computer interactions.

As defined in [9], approaches used in several Adaptive Education Hypermedia Systems (AEHS) reported in the literature can be divided in two kinds: *(i)* systems that use the learning style information in order to design the content of instructions, and *(ii)* systems that use the learning style information to adapt to the learners "forms" of cognitive activity. The TAFPA software is based on the first kind as an analysis of the cognitive activity of a user is completed to provide hints. These hints are related to the design of the environment and provide the user tips to use helpful functions of the interface.

TAFPA is thus provided with a learning agent that is trained and that is used to give advices to make easier the use of the given software.

In a first phase, the *training phase*, the learning agent is trained by using a set of interactions that have been stored when well-known computer-skilled users have interacted with the given environment. Thus, this agent learns the cognitive characteristics that these users' interactions could give.

Afterwards, during the *operational phase*, TAFPA handles the interactions of a user with the environment and uses the learned agent to classify the user as experimented or not. If necessary an advice could be suggested by TAFPA to the user to solve better the task he is faced with. This advice is based on techniques more experimented users are familiar with.

This paper is organized as follow, in Section II, the application methodology is presented. In this part, the presentation of the system that collects traces of user/computer interactions is done. In Section III the training algorithm, the construction of fuzzy decision trees, used by the learning agent is presented briefly. In Section IV, an experiment is presented where TAFPA is faced with a concrete application. In Section V, the results of the experiment are presented and commented. Finally, we conclude on this methodology and we present some future works.

II. Application Methodology

As presented in the Introduction, TAFPA processes in two phases: the *training* phase and the *operational* one.

The training phase is concerned with the training of the learning agent of TAFPA. This learning agent is built from data that come from log files (called *logs* for short) of human-computer interactions when using the given environment. These logs contain descriptions of the user interface interactions. By means of these logs, the agent learns the differences between an experimented computer user and a novice computer user.

During the operational phase TAFPA uses the characteristics that have been previously learned during the training process. The framework of TAFPA is organized as proposed in [5].

Recording of traces.

Logs are composed of *traces* of the interactions of the user with the environment. Traces are built from events produced by the Graphic User Interface (GUI) of the environment. As much as possible events should be produced in order to have the best representation of the human computer interactions stored in the logs. These events are highly connected to the components (*widgets*) of the GUI of the interface and their recording takes advantages of the X-Window manager used to execute the environment. For each event, a *set of descriptors* is kept: their target, their container (the context of the event) and of course the date of their occurrence and all the descriptive values that theses widgets could offer (label of the button, description text …).

For instance, if the environment is a web browser, recorded events are the moves of the mouse, the clicks on a button (with its name "File", "Back", "Forward", "Reload",…), the uses of the scrollbar (up/down), the clicks in text area (to write an URL or in a particular text area of a web page), the uses of the keyboard (and set of keys used), the clicks on links, the closing of the browser,…

TAFPA records all these events to make up logs. In these logs, events are represented as a tree by means of a XML structure that has been introduced for this purpose.

For instance, a click on a button labeled "Show first question" and a click into a text area to type the answer can be represented as the XML structure (or XML tree) given in *Figure 1*.

```
- <context role="HTML" uri="file:///D:/Expelip6/Experience_fichiers/question.html">
   - <target role="INPUT" value=" Show first question ">
      - <eventMotif type="Mouse" duration="130" time="51153532">
            <event type="mousedown" time="51153532" />
            <event type="mouseup" time="51153652" />
            <event type="click" time="51153662" />
         </eventMotif>
     </target>
   - <target role="TEXTAREA">
      - <eventMotif type="Mouse" duration="120" time="51156697">
            <event type="mousedown" time="51156697" />
            <event type="mouseup" time="51156817" />
         </eventMotif>
     </target>
...
</context>
```

Figure 1. Example of TAFPA's traces

From such logs, TAFPA automatically builds a graph to represent the whole interactions (see *Figure 10* for such a graph). This graph shows, as an *UML scenario* schema, all the different targets that occur during the interactions.

For instance, in Figure 10, the targets of two traces have been placed on the parallel horizontal bars. The arrows represent the path of the user between these targets. The arrow's length projected on the horizontal axis shows the duration between the beginnings of two interactions with the targets on the base and on the end of the arrow.

With such a representation, it can be easier to analyze logs from different users. For instance, it can be found automatically that some users were doing a step when some others were doing another step.

Training phase.

The system has been given the capacity to distinguish the descriptors that are skill dependant and those which are issued from the cognitive process involved in the task resolution. As we place ourselves in a supervised learning system (see Section III for more details), the training of the learning agent in TAFPA should take into account the more convenient variables for the description of the cognitive process of the user.

To be more precise in the description of the cognitive process, TAFPA has been added a "Task File" (see *Figure 2*) that describes the steps that all the users need to accomplish within the interface.

Many papers were published about estimating these tasks automatically, but no real option is more accurate than to use a predefined one [3]. As the aim is to find a correct and precise model of the user's cognitive activity, a big amount of XML traces has been produced that contains all the interactions.

All these traces are stored in a *SQL database* in order to be easily recovered. This SQL database can be organized as trees, where each table defines a level of the tree.

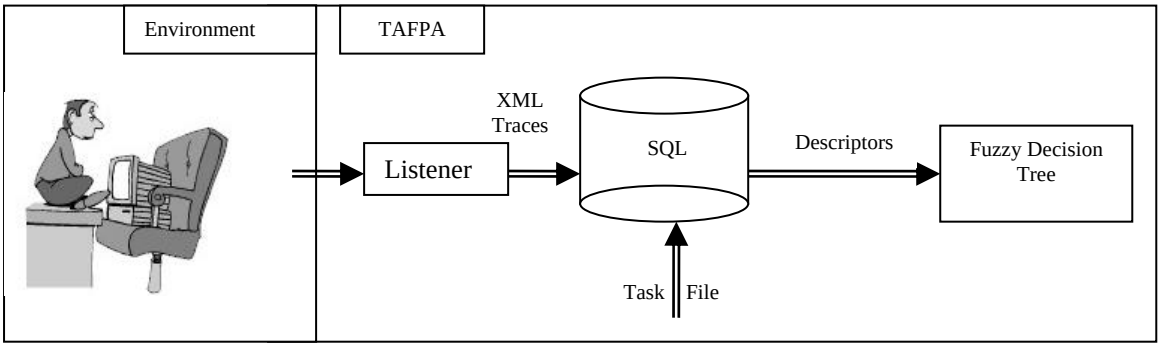

Figure 2. TAFPA's training phase.

For instance, a table of this database contains the more general descriptor of an event: the *container*. In the case of a hypermedia, it is the pages visited by the user. So on, this table is composed of the whole elementary events, at a cognitive level, like "mouse down", "mouse up", "key pressed", etc…
There are two kinds of descriptors: the *global* ones and the *specialized* ones.
The global descriptors are pumped from either experimented or novice users. They will enable the learning agent to classify the user during the operational phase. They are composed of all the characteristics that define cognitive process. For instance, the frequency of switches between the two devices (mouse and keyboard) will tell if the user is comfortable and/or experimented within the interface.
The specialized descriptors contain the hints that will be suggested during the operational phase. So they are actions that the experimented user does easily and that the novice user does not know about or does not use at all. The final aim of TAFPA is to detect automatically these actions but in the experiment presented in this paper they are known variables.

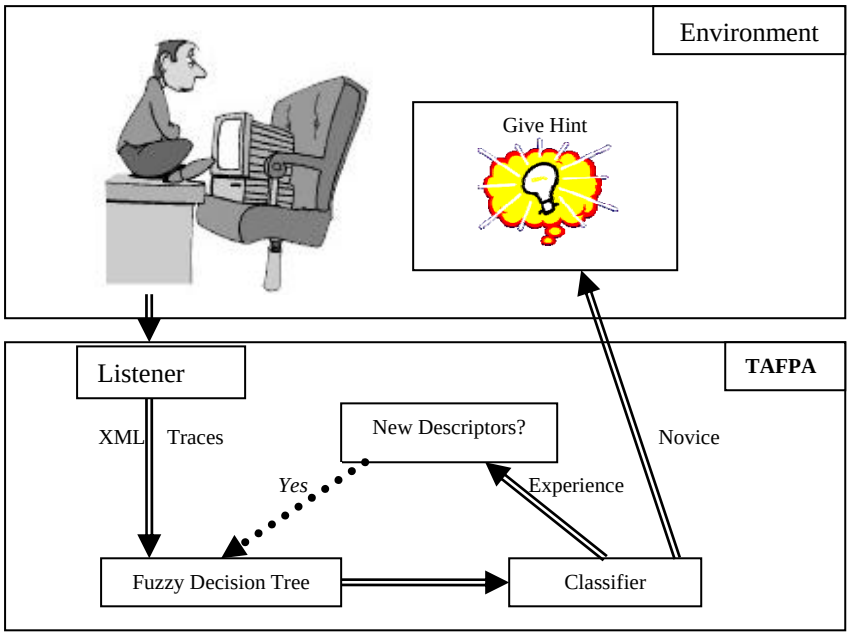

Figure 3. TAFPA's operational phase.

Operational phase.
During this phase (see *Figure 3*), the GUI events collected by TAFPA as "XML Traces" are given as input for the learning agent that uses the Fuzzy Decision Tree (FDT) built in the previous step to classify these events.
In fact, the training phase gives the important cognitive descriptors that are useful for the classification. As soon as the classifier gets an answer of the users categories, TAFPA takes the decision to "Give Hint" or possibly finds a new "Descriptor" to rebuild the FDT.
By doing so, two major properties of Fuzzy Decision Trees are highlighted: the faculty to learn and classify inputs, and the explanation provided when classifying new cases.

Construction of cognitive descriptors.
Many works have been done by interface designers to find out the way of limiting cognitive process. For instance, the GOMS family techniques analyze that process and try to find out the best path of actions to reach a task [7]. In our case we look for some general descriptors categorizing actions. To be more precise in the description of our actions, we propose two methods for cutting the general task in steps in which the actions occurred:

- *With an expert*. In this method an expert has to look at the general task and find the different steps that all the people have to accomplish to reach the goal.

- *Automatically*. This method consists of looking for the Longest Common Sequence (LCS) of events (the lowest level in the log traces). Many approaches have been developed in the literature for such a task. However, in our application, as the log files are in tree format, we propose to use the algorithm of "Extracting recurrent patterns from stratified ordered trees" presented in [4]. The longest pattern with support 1 obviously gives the LCS. Then we can build *n+1* steps, with n equals the length of the LCS. In this case, the descriptors need to be more accurate for each step, but in the same way as previously we look for general attributes categorizing the actions.

When the steps are built, the descriptors will be as much as possible pattern attributes and/or items attributes: frequency, durations, number of iterations, etc…
During the operational phase of TAFPA, the learning agent is asked for a class (novice or expert for example) at each step. This means that one FDT is constructed for each step. At step n, all the descriptors of the data collected at previous steps *(n-1, n-2…, 0)* can be used to construct the FDT. As soon as the answer is enough accurate, the system makes a decision to give or not a hint to the user.

III. Fuzzy Decision Trees

Fuzzy decision trees are a combination of decision tree [10] and fuzzy set theory [14] to improve the performance of decision tree (see Figure 9 for an instance of Fuzzy Decision Tree).
Inductive learning is based on a training set of examples described by means of a number of attributes, each one belongs to a class corresponding for instance to a decision to make. It is made of two steps. The first one is the construction of a so-called decision tree, which corresponds to a ranking of the attributes according to their influence on the belonging to a class. The second step consists in classifying a new example only known through the attributes, in order to determine which class it must be assigned to.
The induction of decision trees is an efficient way of learning from examples. It has been shown to be interpretable, scalable and simple to construct. Many methods have been developed for constructing decision trees. The main method to construct a decision tree is the so-called *Top Down Induction* method. The tree is constructed from its root to its leaves by successive partitioning of the training set into subsets. Each partition is done thanks to the values of a selected attribute which is usually chosen as the most informative with regard to the identification of the class of an example, and leads to the definition of a node of the tree labelled by this attribute. The values of the selected attribute are assigned to the branches of tree. A leaf is constructed from a set that contains only examples with the same class or more generally from a set that is enough homogenous. It is labelled by the name of the major class or by the probability distribution of all examples associated to this leaf.
Usually, the Shannon entropy is used as a measure of selection for the choice of the best attribute by which the training set is partitioned [10].
Most real-world classification problems involve continuous-valued attributes. Decision trees should be extended to deal with such attributes. One solution for overcoming these problems is the introduction of the discretization of numerical attributes before or on the fly. The discretization on the fly is usually considered as a sub-step of the selection of the best attribute. A good discretization technique allows also to exploit the proximity between numerical values and thus to reduce the noises in the data. Many discretization

techniques have been developed such as the families of entropy-based and purity-based methods, unsupervised method (equal width discretization, equal frequency discretization...), statistic-based methods like Chi-merge and CHI2 [8].

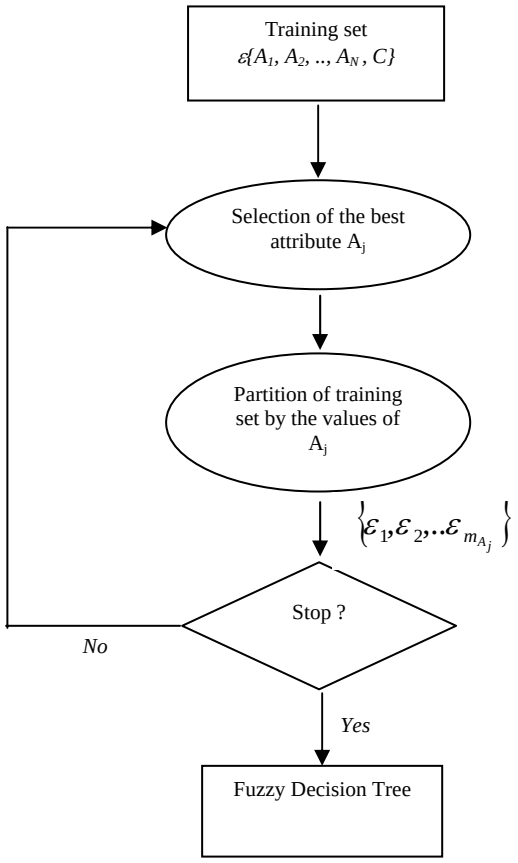

Figure 4. Construction of fuzzy decision tree

Due to the existence of vague and imprecise information in real-world applications, the values of attribute and then the class boundaries may not be defined clearly. The introduction of fuzzy theory in the decision tree technique allows resolving efficiently this kind of problem. It also improves certain qualities of decision tree such as precision, robustness and it is nearer to the human thinking.

Fuzzy Decision trees have been introduced by [11] in order to improve the capabilities of decision trees to handle numerical data. The aim of the introduction of the Fuzzy Sets Theory of Zadeh [14] was to enable the taking into account of imprecise and uncertain values. Several works on fuzzy decision trees have been further done. We can cite for instance [6][13].

A fuzzy decision tree is a generalization of the crisp case. Each node is associated to a fuzzy set of examples and labelled by an attribute. Each branch from the node is associated to a fuzzy set defined on the domain of the attribute (fuzzy value). Each leaf is labelled by a set of labels of classes with different membership degrees instead of the label of one class.

In general, induction of fuzzy decision trees follows the same steps as that of a classical decision tree. Among theses steps, the fuzziness is incorporated in the following:

- the fuzzy discretization of numerical attributes
- the choice of the best attribute by a fuzzy measure of selection of attributes
- the verification of the stopping criteria
- the use of the fuzzy decision tree for classifying new examples

Let $L = \{e_1, e_2, .., e_n\}$ be a set of examples, a so-called training set. Each example *e* is described by means of a set of values for N attributes $\{A_1, A_2, .., A_N\}$ and belongs to a class c from a set of classes $C = \{C_1, C_2, .., C_m\}$. Each attribute A takes its value in the set $\{v_1, v_2, .., v_{m_A}\}$ where the values v_i can be

symbolic, numerical, or fuzzy. The induction of a fuzzy decision tree from the training set enables us to obtain a general law to point out the relation between the values of attributes and the classes in *C*.

Definition of fuzzy probability and Shannon fuzzy entropy

The probability of fuzzy events (or fuzzy probability for short) has been introduced formally by Zadeh[15]. Given a fuzzy set X of examples $X = \{e_1/\mu_1, e_2/\mu_2, .., e_n/\mu_n\}$ with μ_i is the membership degree of example e_i in the set *X*.

Firstly, the probability of fuzzy event C_i in X is defined by: $p^*(C_i) = \dfrac{\sum_{e_j \in C_i} \mu_j}{\sum_1^n \mu_j}$.

If the probability of each example in X are not equal, the following formula should be applied $p^*(C_i) = \dfrac{\sum_{e_j \in c_i} p_j \mu_j}{\sum_1^n p_j \mu_j}$ where p_j is the probability of example e_j.

This formula combines a measure of uncertainty (probability) and a measure of imprecision (fuzziness).
The fuzzy entropy (entropy star) of *X* is an extension of Shannon entropy to fuzzy event then defined by[11]: $I^*(X) = -\sum_{i=1}^{m} p^*(C_i) \log p^*(C_i)$.

For more details on fuzzy entropy, see [11].

Fuzzy discretization of numerical attributes

At each node, before each selection of the best attribute, all numerical attributes should be discretized into several fuzzy intervals (2 intervals in our system). Each cut point is described by a triple of real values *[c-δ, c, c+δ]* generally chosen by maximizing the homogeneity of all fuzzy partitions separated by it. Suppose that at the input of the discretization step of attribute A_k, the sub-training set *L* contains *n* examples: *{e_1/μ_1, e_2/μ_2, .., e_n/μ_n}* with μ_i ($\mu_i > 0$) the membership degree of example e_i in *L*. Initially, the training set contains all examples with membership degree that equals 1 ($L = \varepsilon$). In the next times, the membership degrees may not be 1.
Let v_i be the value of the actual considered attribute A_k of example e_i. Without generalization problem, we can suppose that $v_1 \le v_2 \le .. \le v_n$. The fuzzy cut point *[c-δ, c, c+δ]* is selected among *{[c_i-δ_i, c_i, c_i+δ_i], 0<i<n}* where:

$c_i = \dfrac{v_i + v_{i+1}}{2}$ and $\delta_i = v_{i+1} - v_i$ (δ_i can be parameterized).

This cut point partitions the domain value of the attribute into 2 fuzzy partitions D_L and D_R the membership degrees of which are defined as follows.

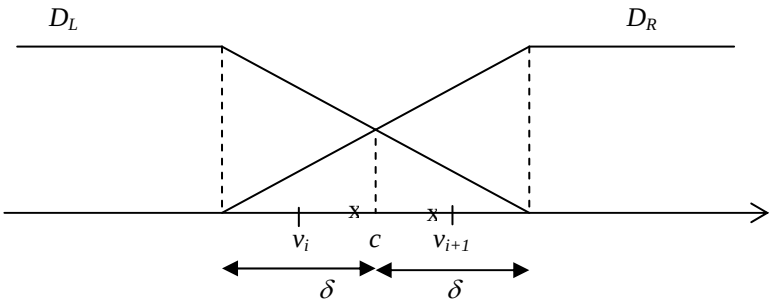

Figure 5. Fuzzy cut point

$$\mu_{D_L}(x)=\begin{cases}1 \text{ if } x \le c-\delta \\ \dfrac{c+\delta-x}{2\delta} \text{ if } c-\delta < x < c+\delta \\ 0 \text{ if } x \ge c+\delta\end{cases} \text{ and } \mu_{D_R}(x)=\begin{cases}0 \text{ if } x \le c-\delta \\ \dfrac{x-c+\delta}{2\delta} \text{ if } c-\delta < x < c+\delta \\ 1 \text{ if } x \ge c+\delta\end{cases}.$$

We have: $\mu_{D_L}(x)+\mu_{D_R}(x)=1$ *for all x*.

The sub-training set is then partitioned into 2 fuzzy-partitions L_L (partition on the *left* of the fuzzy cut point) and L_R (partition on the *right* of the fuzzy cut point). The membership degrees of L_L and L_R are defined as follows: $\mu_L(e_i)=\mu_i * \mu_{D_L}(v_i)$ and $\mu_R(e_i)=\mu_i * \mu_{D_R}(v_i)$.

The selected fuzzy cut point is the one that minimizes: $I^*(L/A_k)=\frac{|L_L|}{|L|}I^*(L_L)+\frac{|L_R|}{|L|}I^*(L_R)$ where $I^*(X)$ is the fuzzy entropy of the set X defined above. This is the entropy of the sub-training set conditioned by the attribute discretized by the fuzzy cut-point. Note that: $\frac{|L_L|}{|L|}+\frac{|L_R|}{|L|}=1$.

Choice of the best attribute by a fuzzy measure of discrimination

After the discretization of all numerical attributes, the gain of information brought out by numerical attribute A_k, is valued: $Gain(A_K)=I^*(L)-I^*(L/A_K)$. For nominal attribute, the gain of information is valued by means of the same formula but $I^*(L/A_k)=\sum_{i=1}^{m_{A_k}}\frac{|L_i|}{|L|}I^*(L_i)$ with L_i is the fuzzy sub-set of all examples from L whose value of A_k is v_i. The best attribute is selected as the most informative one which maximizes the gain of information.

The sub-training set is then partitioned thanks to the value of the selected attribute into 2 fuzzy partitions. Each example may belong to both partitions with different membership degrees. The selected attribute becomes the label of the node associated with the sub-training set L and D_L, D_R are considered as fuzzy values and assigned as labels of the 2 edges from the mentioned node.

The stopping criteria

The partitioning of training at each node should end when the sub-training set is enough homogeneous or its fuzzy cardinal is too small (less than 1). In fact, all examples belonging to the sub-training set with very little membership degree are eliminated before considering to partition.

Use of the fuzzy decision tree for classifying new example

The fuzzy decision tree constructed is used for classifying new examples. It is used as a set of fuzzy rules; each one has the form: "If [A_1 is d_1] and [A_2 is d_2] … then [C is d_c]".

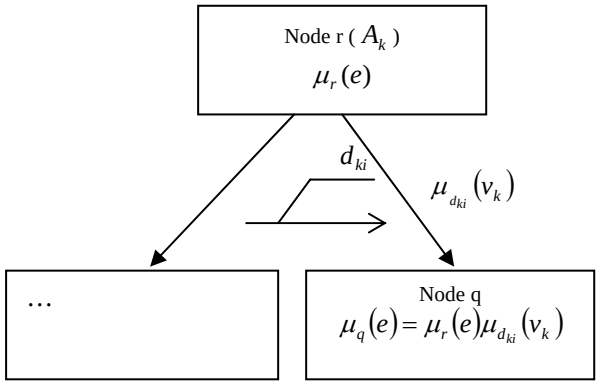

Figure 6. Node membership

As the values associated to the branches of the tree can be fuzzy, at each internal node each example can be transmitted to many edges by different membership degrees. Let the membership degree of an example e in a node r which is labelled by attribute A_k be $\mu_r(e)$. Let the membership degree of the value of attribute A_k of e in a branch of r be $\mu_{d_{ki}}(v_k)$. Then the membership degree of e in the child node q of r corresponding to the branch is valued as: $\mu_q(e) = \mu_r(e)\mu_{d_{ki}}(v_k)$.

Many leaves can be reached with different degrees. Moreover each leaf can be associated to several classes. Therefore an aggregation operator is needed to make a final decision the class of example if needed. In regarding our experiment, we used the aggregation operator called *max-product*: each leaf is labelled by the name of the major class and the class of example is the label of the leaf, to which the example belongs with maximal membership degree.

If in the construction of fuzzy decision trees, the spreading out of fuzzy cut point (fuzzy zone) is always small enough such that all examples belong to each node with membership degree equal to 0 or 1, the form of fuzzy decision tree is identical to the crisp one constructed by the crisp measure of discrimination.

The DTGen (Decision Tree Generator) is a software [12] that has been developed in our research group. It implements a generic TDIDT algorithm for the construction of decision trees (crisp and fuzzy) from a training set and allows using decision trees (one or many) for classifying the new examples. In addition, it enables the selection of different measures of selection: Shannon entropy, other classical entropies and the fuzzy version of these measures. Moreover, it implements the presented algorithm to discretize automatically on the fly a set of numerical values by means of the chosen entropy. DTGen works with symbolic attributes, numerical attributes and fuzzy attributes described in certain forms like triangular or trapezoidal.

IV. Experiment

Browsing a hypertext is subject to a lot of cognitive processes [2]. As the aim is to model these cognitive actions, the experiment is based on the use of a web-browser, and users have to accomplish some tasks on a given web page and to answer a set of questions. The screen of this hypertext is given in *Figure 7*.

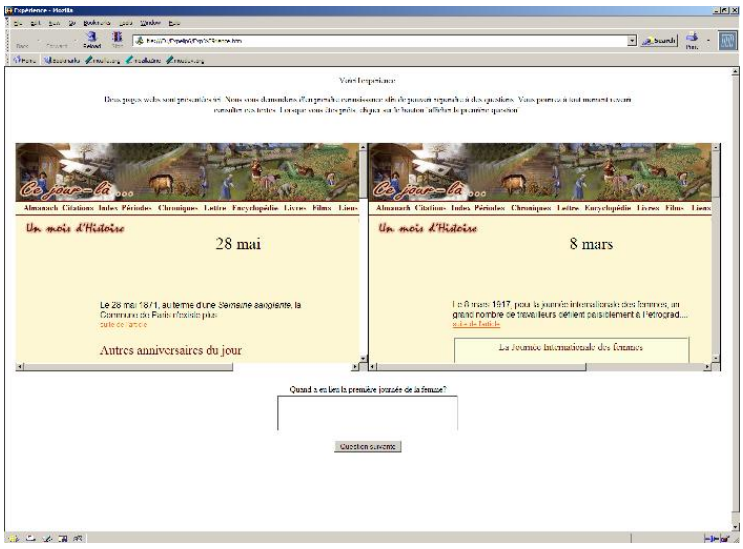

Figure 7. The web page of the experiment.

The first part of the webpage presents the task to the user. It asks him to read the text given within two parallel frames and to answer some questions. The user is also told that the two frames will stay on the interface so that he could return further to have a look at it. So the task was to read or just look at the two texts and click on the button “Show First Question”. Then a question, a text area for the answer, and a button “Next Question” appears. So on, four questions are asked.

This experiment is conducted on two sets of users: a set of experimented users well-adapted to the use of browser, and a set of novice users which are not graphical interface friendly. This skill difference is used by the learning agent as the class to recognize (*experimented* / *novice*).

For this experiment, the two kinds of decomposition described in Section II have been studied, and the results are presented in the next Section.
For the decomposition with an expert, a step is delimited by means of a click on the button “Next Question”. That way, four steps of the same kind (answering a question) have been built that can be analyzed with the same descriptors as the actions. A preliminary step is introduced that corresponds to the reading of texts that ends with the click on the button “Show First Question”.
For the decomposition with the LCS, the learnt sequence is presented in the Next part.

V. Results

Nearly 30 people have been doing the experiment. Less than half of them were real novices and the others were colleagues experimented with graphical interface. The learning agent has been trained with the same proportion of novice and experimented user traces, and the remaining traces have been used for the validation.
Finding the LCS of all traces shows three more steps that correspond to the click on the text area where the answer is typed. Four new steps were expected as four questions were asked, but that made us notice that one user missed a question by double-clicking accidentally on the button “Show next question”. As we previously said, it can lead to some mistakes to use the LCS, but as the Figure 8 shows, the results are much better.

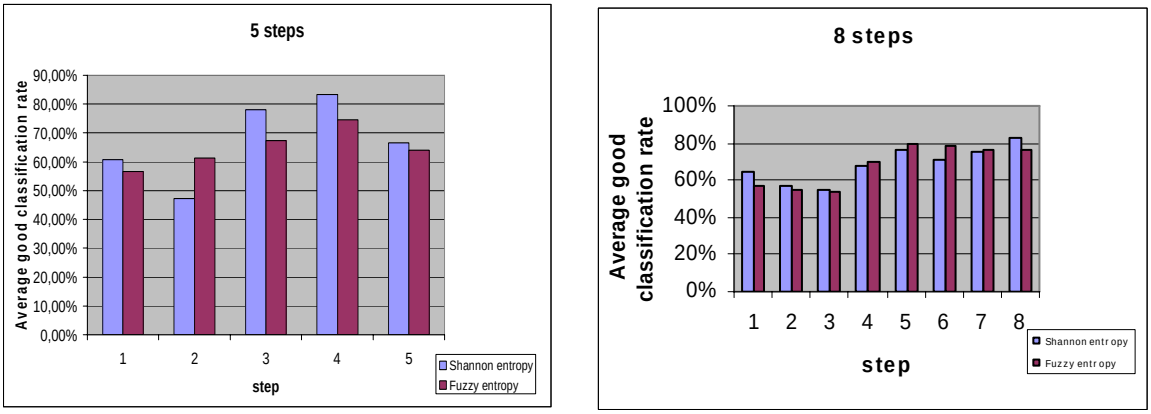

Figure 8. Results.

In this Figure, we recall that, in the “5 steps” graphic, each step is associated with a question of the questionnaire. For each step, the decision tree, fuzzy or not, performs a high good classification rate. Moreover, it performs better for more complex questions (4 and 5) where being experimented is an advantage.
We recall also that, in the “8 steps” graphic, seminal steps are defined as events between 2 clicks on button or text area. Here again, we can observe the performance of the trees when the user faced with complex question.
Even tough, the advice could be given sooner. In our experience, advices that could have been displayed automatically are the “Ctrl-F” function for users accurate with the keyboard, and the “find” function in the menu “Edition” that have been use by some mouse friendly experimented users. These two actions have been added in the cognitive descriptors as the answer to deliver was referring to a name to find in the pages. Their utility is verified in the graph on step 4 (with the LCS) or step 3 (with the expert decomposition) where the jump in the classification rate is observed.
For these experiments, a 10-fold cross-validation on the entire dataset was performed. DTGen constructs a fuzzy decision tree from training sets by using fuzzy entropy. The test sets is then classified by the fuzzy decision tree described (see Figure 9 for one example).
Using a Fuzzy Decision Tree (Fuzzy Entropy) instead of a classic ID3 Decision Tree (Shannon entropy) can seem unjustified in this case. But it is very important to notice that, in this experiment, the learning machine needs to be trained with more examples.
Another justification is that the classes are too well separated. Expert and novice users are really using differently the interface and the devices, and their cognitive actions are very different. The use of the FDT

is really justified regarding the categorization of cognitive actions. It allows a more flexible classification and seems naturally more appropriate. If a lot of data are incorporated during the training phase, it is expected to have better result with the fuzzy entropy.

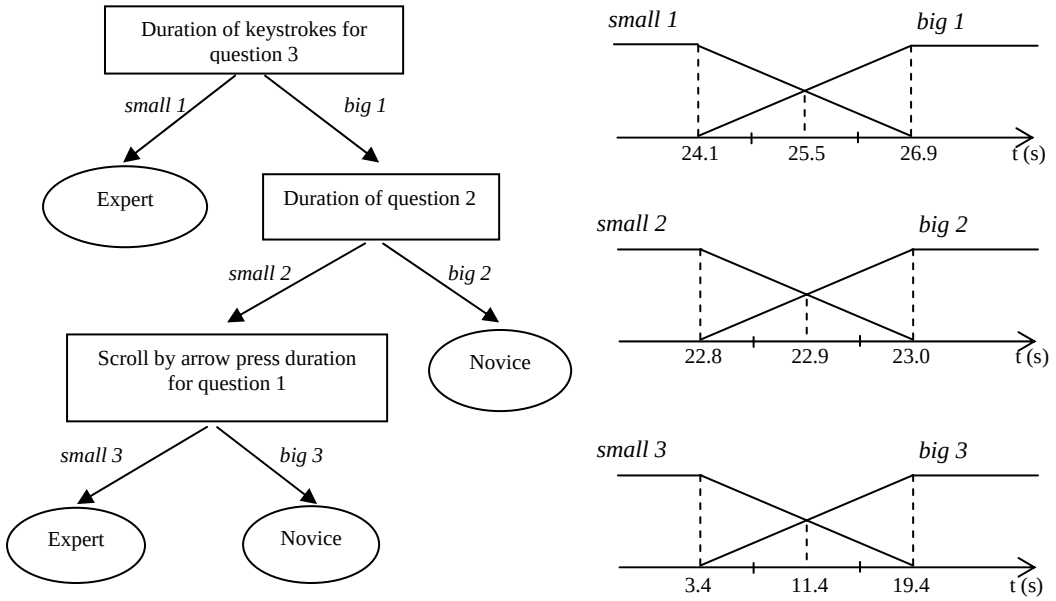

Figure 9. Example of fuzzy decision tree with definition of fuzzy sets.

VI. Conclusion

Providing helpful analysis of cognitive process is one of the main concerns for user modeling and adaptive hypermedia conception. In this paper, we presented the TAFPA (*Tree Analysis For Providing Advices*) system. This software uses a learning agent, a fuzzy decision tree, which works on traces of human-computer interactions. It can provide advices that come from people who have the same cognitive behavior. This methodology can be applied to Adaptive Hypermedia Educational Systems. It is domain independent and can be used for providing help to teachers as it builds a cognitive model of the student.
Future work will be oriented on incorporating more cognitive descriptors and especially to cognitive sequence descriptors. Making a clustering on these attributes could help to build an unsupervised learning system.

VII. References

[1] Brusilovsky P. [2001]. *Adaptive Hypermedia*, In *User Modeling and User-Adapted Interaction*, vol. 11, pp. 87-110.

[2] Carmel E. & Crawford S. & Chen H. [1992]. *Browsing in Hypertext: A Cognitive Study*. In *IEEE transaction on systems, Man and Cybernetics*, vol. 22(5), pp. 865-883.

[3] Dragunov A.N. & Dietterich T.G. & Johnsrude K. & McLaughlin M. & Li L. & Herlocker J. L. [2005]. TaskTracer*: A Desktop Environment to Support Multi-tasking Knowledge Workers.* In *International Conference on Intelligent User Interfaces*. San Diego, California, USA, pp. 75-82.

[4] Ganascia J.G. [2001]. *Extraction of recurrent patterns from stratified ordered trees*. In *European Conference on Machine Learning*, pp. 167-178.

[5] Hilbert D. & Redmiles D. [2000]. *Extracting usability information from user interface events*. In *ACM computing surveys*, Vol 32, Issue 4, pp 384-421.

[6] Janikow C.Z. [1998]. *Fuzzy decision trees: Issues and methods*, In *IEEE Transactions on Systems, Man, and Cybernetics* (28), pp. 1-14.

[7] John B.E. & Kieras D.E. [1996]. *The GOMS family of User Interface Analysis techniques: Comparison and Contrast*. In *ACM Transactions on Computer-Human Interaction*, pp. 320-351.

[8] Liu H. & Hussain F. & Tan C.L. & Dash M. [2002]. *Discretization: An Enabling Technique*, In *Journal of Data Mining and Knowledge Discovery*, Vol. 6(4), pp. 393-423.

[9] Papanikolaou K.A. & Grigoriadou M. [2004]. *Accommodating learning style characteristics in adaptive educational hypermedia systems*. In *Proceedings of the Workshop on Individual Differences in Adaptive Hypermedia in AH2004*. Eindhoven, Nederland, pp. 79-88.

[10] Quinlan J.R. [1986]. *Induction of decision trees*, In *Machine Learning 1 (1)*, pp. 81-106.

[11] Ramdani M. [1994]. *Système d'induction formelle à base de connaissances imprécises*, PhD thesis, University Paris VI, France.

[12] Stermann F. & Longuet N. [2003]. *Document technique de DTGen*, Internal Report, LIP6, France.

[13] Yuan Y. & Shaw M.J. [1995]., *Induction of fuzzy decision trees*, In *Fuzzy Sets and Systems*, vol 69, pp. 125-139.

[14] Zadeh L. A. [1965]. *Fuzzy sets*, In *Information and Control*, vol. 8, pp. 338-353.

[15] Zadeh L. A. [1968]. *Probability measures of fuzzy events*, In *Journal Math. Anal. Appl.*, vol 23, pp. 421-427.

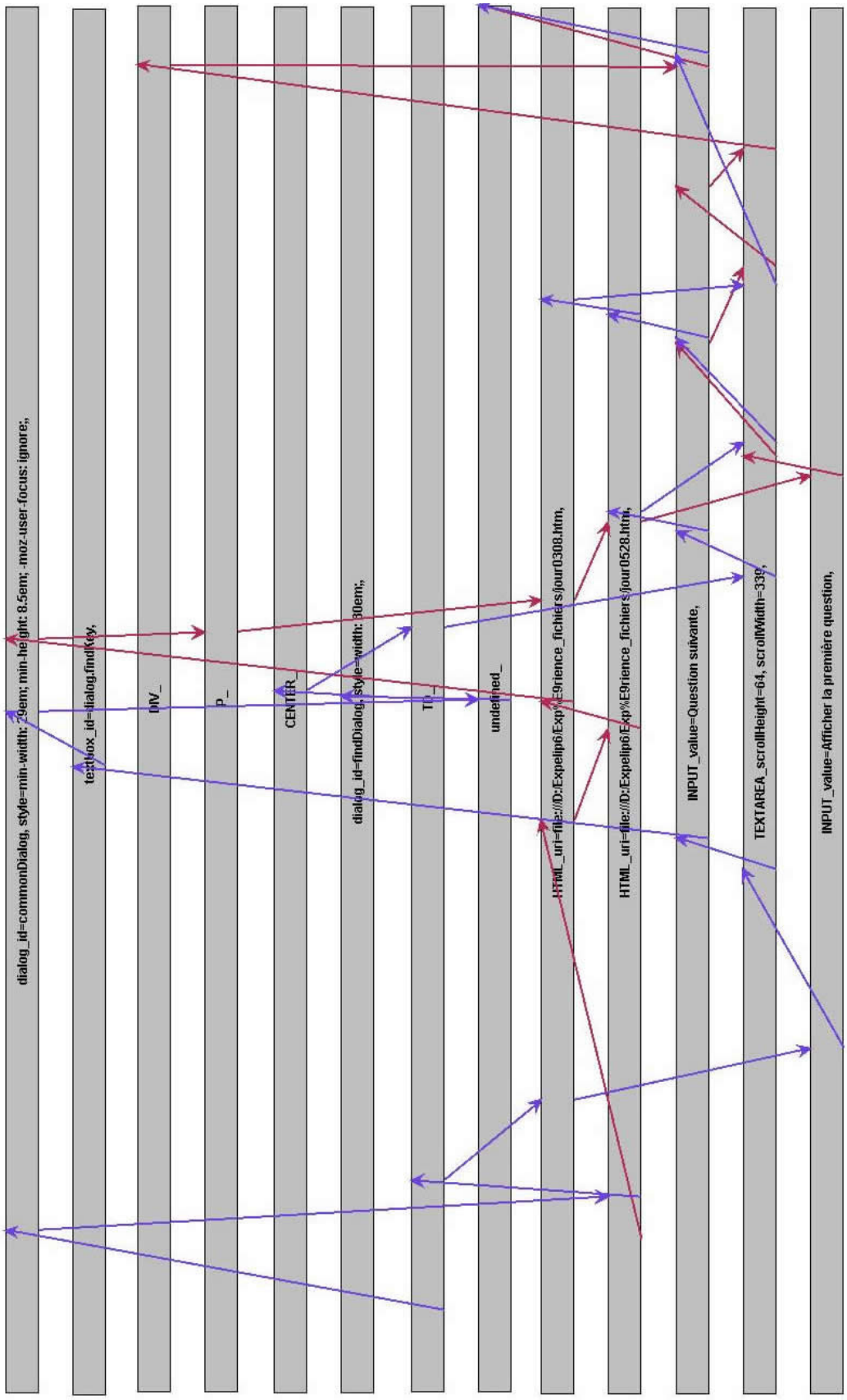

Figure 10. A representation of the sequences of targets in the traces