An Efficient Behavior Classifier based on Distributions of Relevant Events

Jose Antonio Iglesias and **Agapito Ledezma** and **Araceli Sanchis**[1] and **Gal Kaminka**[2]

1 Introduction

Recognizing the behavior of others is a significant aspect of many different human tasks. In order to make a good decision, humans usually try to predict the behavior of others. We present an approach for creating automatically the model of the behavior of agents (software agents, robots or humans). Because of the sequence learning is a common form of human and animal learning, the observations of an agent are transformed into a sequence of atomic behaviors which is statistical analyzed to find out its corresponding behavior model.

Before recognizing a behavior, it needs to be modeled. Different techniques have been used in agent modeling in different areas: opponent-modeling in soccer domain simulation [6], intelligent user interface [7], and virtual environment for training [8]. However, although lot of research focus on agent modeling in an specific environment, it is not clear that they can be used in other environments.

The aim of this research is to provide a *general* framework which can represent and classify different agent behaviors in a wide range of domains. Also, as the actions performed by an agent are usually influenced by his past experiences, the automated sequence learning is used for behavior classification.

2 ABCD: Agent Behavior Classifier based on Distributions of relevant events

Any behavior has a sequential aspect and this sequentiality should be considered in the modeling process. Our approach classifies an observed agent behavior into the classes (behaviors) stored previously in a library. Therefore, this process is divided in the following 2 parts:

2.1 Construction of Behavior Models

1. **Obtaining Atomic Behavior Sequences:** Useful features are extracted from the stream of observations of the environment and an ordered sequence of *events* is obtained. An *event* is an atomic behavior that occurs during a particular interval of time and defines an specific agent act. The type of events is domain-dependent.
2. **Creating the behavior model:** The temporal dependencies are very significant and to get the most representative set of sequential events (subsequences) from the acquired sequence, the data structure *trie* [2] is used as in [3, 4]. The construction of a *trie* from a single sequence of events is processed in three steps:
   a) Segmentation of the sequence: This segmentation can be done by using some environment characteristic that separates the sequence in several subsequences of uninterrupted events or by obtaining every possible ordered subsequence of a defined length.
   b) Storage of the subsequences in a *trie*: The subsequences of events are stored in a *trie*, in which every node represents an event, and the node's children represent the events that have appeared following this event. Each node keeps track of the number of times an event has been inserted on to it. The subsequence suffixes (subsequences that extend to the end of the sequence) are also inserted.
   c) Creation of the behavior model: The *trie* is traversed to calculate the relevance of each subsequence. For this purpose, frequency-based methods are used and the relative frequency or support of a subsequence is calculated. Then, an agent behavior model is represented by the distribution of its subsequences.
3. **Storing the model in the Library:** Once a behavior model (distribution of relevant subsequences) is created, it is stored in *Library of Behavior Models (LibBM)* (similar to plan-libraries used in plan recognition). This model is stored (with an identification name) as a *trie* for a good and effective handling (Figure 1a).

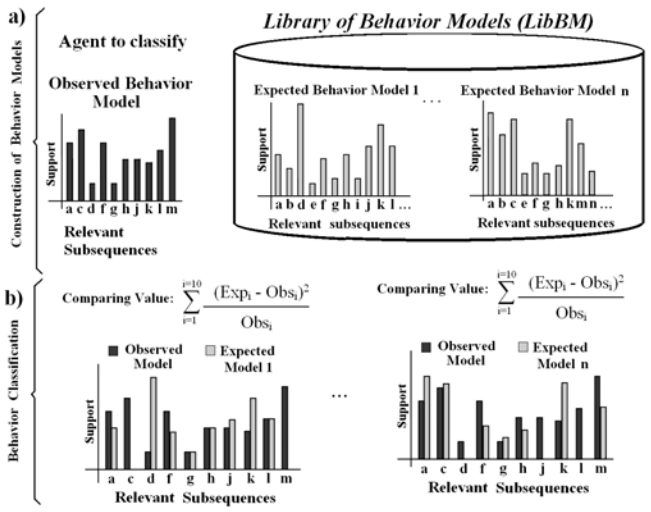


Figure 1. Agent Behavior Classification Process

2.2 Behavior Classification

The observations of the agent to classify are collected and the corresponding behavior model (represented by a distribution of events) is created. Then, it is matched with all the behavior models stored in *LibBM*. As both models are represented by a distribution of events, an statistical test is applied for matching these distributions.

The proposed non-parametric test applied for matching two behaviors is a modification of *Chi-Square Test* for two samples. The behavior model to classify is considered as an observed sample and all the behavior models stored in *LibBM* are considered as expected

[1] Carlos III University of Madrid, Spain, {jiglesia, ledezma, masm}@inf.uc3m.es
[2] Bar-Ilan University, Israel, galk@cs.biu.ac.il

samples. This test compares the observed distribution with all the expected distributions objectively and evaluates if a deviation appears.

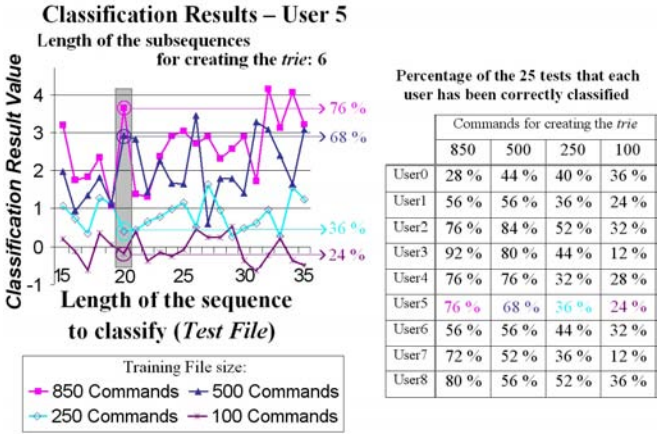
The proposed test is the comparison of two sets of support values in which *Chi-Square* is the sum of the terms $\frac{(Exp-Obs)^2}{Obs}$ (Figure 1b). With this comparison, a value (*comparing value*) that indicates the difference (deviation) between the two distributions is obtained. The lower the value, the closer the similarity between the two behaviors. This comparison test is applied once for each behavior model stored in *LibBM*. The model which obtains the lowest deviation is considered as the most similar one. An advantage of the proposed test is its rapidity because only the observed subsequences are evaluated. However, there is no penalty for the expected relevant subsequences which do not appear in the observed distribution.

3 Experiments

3.1 UNIX User Classification

In this domain, the behavior of a user is represented by the sequence of UNIX commands he/she typed during a period of time. We use 9 sets of preprocessed user data drawn from the command histories of 9 UNIX computer users [1]. Each *UNIX user file* is divided in: **1.Training Files:** created with a small and random part of consecutive commands (100, 250, 500 and 850 commands) taking from the corresponding *User file* and creating 4 different *LibBMs*. These results are calculated using subsequences of size 6. **2. Testing Files:** Obtained from the other part of each given *user file*. 20 *Testing files* with different amount of commands (from 15 to 35) are evaluated.

For evaluating the results, a value (*Classification Result Value*) is calculated from the ranking list obtained for each classification. If the classification is done correctly, this value is the difference (positive) between the lowest and the second lowest value. If the classification is done incorrectly; for evaluating how far the obtained result is from the correct one, this value is calculated by comparing the lowest value with the obtained value (obtaining a negative value).



Percentage of the 25 tests that each user has been correctly classified

	Commands for creating the *trie*			
	850	500	250	100
User0	28 %	44 %	40 %	36 %
User1	56 %	56 %	36 %	24 %
User2	76 %	84 %	52 %	32 %
User3	92 %	80 %	44 %	12 %
User4	76 %	76 %	32 %	28 %
User5	76 %	68 %	36 %	24 %
User6	56 %	56 %	44 %	32 %
User7	72 %	52 %	36 %	12 %
User8	80 %	56 %	52 %	36 %

Figure 2. Classification Results - User 5

Figure 2 shows the classification results of 20 different commands of a UNIX user. X-axis: length of the sequence to classify (from 15 to 35 commands). Y-axis: *classification result value* obtained by applying ABCD. The 4 lines show the results by using 4 different sizes of *Training Files* to create the *Tries* of the *LibBM*: 100, 250, 500 and 850 commands. Each graph point is the average value of 25 different test conducted. Although this average determines that a sequence is correctly classified most of the tests, the classification of the 25 tests is not always correct. The percentages of the 25 tests correctly classified using *testing file* of 20 commands are shown in Figure 2.

3.2 RoboCup Soccer Coach Simulation

The goal in this domain is to observe a game and recognize the behavior models (previously analyzed and stored in *LibBM*) followed by the opponent team members. For these experiments, we have used the rules from the *RoboCup 2006 Coach Competition*. The construction of models is done considering only the behavior followed by a few players (*player behavior*). However, the behavior to classify is the sum of several *player behaviors* (*team behavior*). The construction of models is done by analyzing several game *log files* (*Training files*) in which different *player behaviors* are activated. The procedure to identify high-level events in a soccer game described by Kuhlmann et al. [5] is used. Then, a new game in which several *player behaviors* are activated at the same time (*team behavior*) is observed and the *player behaviors* activated must be recognized.

In these experiments, 17 *player behaviors* are analyzed (download from *RoboCup 2006 Coach Competition* web page) and stored in *LibBM*. The ranking list obtained (with the most likely *player behaviors*) is evaluated. Table 1 shows the first 10 elements of the ranking lists obtained for the 3 iterations of the first round. The number of player behaviors activated in each iteration is indicated in square brackets. The *player behaviors* are identified with a number (from 00 to 16) and the *player behaviors* activated are marked with an asterisk.

Table 1. Results for the *RoboCup Coach Competition*. Round1

	Ranking list reported (most likely *player behaviors*)
Iter1 [4]	**04(*)**, 16, **00(*)**, 12, **15(*)**, 03, 09, 05, 01, 06
Iter2 [5]	**16(*)**, **01(*)**, 00, **13(*)**, 05, 09, **07(*)**, 03, 10, **08(*)**
Iter3 [5]	**04(*)**, **02(*)**, 13, 05, 12, **00(*)**, 01, **06(*)**, 03, 10

4 Conclusions and Future Works

A *general* approach which can represent and handle different behaviors in a wide range of domains is provided and it is generalizable using behaviors represented by a sequence of events. The experiments show that a system based on *ABCD* is very effective for classifying a UNIX user. For areas such as computer intrusion detection, these results are very encouraging. In the real-time and multi-agent domain; the results depend of the kind of behavior to recognize, however the obtained results are satisfactory.

As many agents change their behavior and their preferences over time, their models should be frequently revised to keep it up to date. This aspect could be solved by using *Evolving Systems*. Also, the use of the classification results for carrying out effective actions in the environment is considered in our future work[3].

REFERENCES

[1] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[2] E. Fredkin, 'Trie memory', *Comm. A.C.M.*, **3**(9), 490–499, (1960).

[3] José Antonio Iglesias, Agapito Ledezma, and Araceli Sanchis, 'A comparing method of two team behaviours in the simulation coach competition', in *MDAI*, volume 3885 of *LNCS*, pp. 117–128. Springer, (2006).

[4] Jose Antonio Iglesias, Agapito Ledezma, and Araceli Sanchis, 'Sequence classification using statistical pattern recognition', in *IDA*, pp. 207–218, (2007).

[5] Gregory Kuhlmann, Peter Stone, and Justin Lallinger, 'UT Austin Villa 2003 simulator online coach team', in *RoboCup2003*, (2004).

[6] Agapito Ledezma, Ricardo Aler, Araceli Sanchis, and Daniel Borrajo, 'Predicting opponent actions by observation.', in *RoboCup*, (2004).

[7] Neal Lesh, Charles Rich, and Candace L. Sidner, 'Using plan recognition in human-computer collaboration', in *UM99*, pp. 23–32, (1999).

[8] M. Tambe and P. S. Rosenbloom, 'Resc: An approach for dynamic, real-time agent tracking', in *IJCAI-95*, Montreal, Canada, (1995).

[3] **Acknowledgments.** This work has been supported by the Spanish Ministry of Education and Science under project TRA-2007-67374-C02-02.