A Learning Interface Agent for User Behavior Prediction

Gabriela Şerban, Adriana Tarţa, and Grigoreta Sofia Moldovan

Department of Computer Science
Babeş-Bolyai University,
1, M. Kogălniceanu Street, Cluj-Napoca, Romania
{gabis,adriana,grigo}@cs.ubbcluj.ro

Abstract. Predicting user behavior is an important issue in *Human Computer Interaction* ([5]) research, having an essential role when developing intelligent user interfaces. A possible solution to deal with this challenge is to build an *intelligent interface agent* ([8]) that learns to identify patterns in users behavior. The aim of this paper is to introduce a new agent based approach in predicting users behavior, using a probabilistic model. We propose an intelligent interface agent that uses a supervised learning technique in order to achieve the desired goal. We have used *Aspect Oriented Programming* ([7]) in the development of the agent in order to benefit of the advantages of this paradigm. Based on a newly defined evaluation measure, we have determined the accuracy of the agent's prediction on a case study.

Keywords: user interface, interface agent, supervised learning, aspect oriented programming.

1 Introduction

Nowadays, computers pervade more and more aspects of our lives, that is why an interactive software system has to be able to adapt to its users. Adaptability is possible to achieve by intelligent interface agents that learn by "watching over the shoulder" ([8]) of the user and by detecting regularities in the user's behavior.

Human-Computer Interaction ([5]) is a discipline that deals with improving user interfaces. Graphical user interfaces are still hard to learn for inexperienced users. Intelligent User Interface agents are a way of solving this problem, for example by guiding users through a given task.

Interface agents are autonomous semi-intelligent systems that employ artificial intelligence techniques in order to provide assistance to a user dealing with a particular application. Their autonomy is provided by the capability of learning from their environment.

Aspect Oriented Programming (AOP) is a new programming paradigm that addresses the issues of *crosscutting concerns* ([7]). A crosscutting concern is a feature of a software system whose implementation is spread all over the system. An aspect is a new modularization unit that corresponds to a crosscutting concern. The aspects are integrated into the system using a special tool called *weaver*.

J. Jacko (Ed.): Human-Computer Interaction, Part III, HCII 2007, LNCS 4552, pp. 508–517, 2007.

In this paper we propose a new approach in predicting users behavior (sequences of user actions) using an interface agent, called **LIA** (Learning Interface Agent), that learns by supervision. In the training step, **LIA** agent monitors the behavior of a set of real users, and captures information that will be stored in its knowledge base, using Aspect Oriented Programming. Aspect Oriented Programming is used in order to separate the agent from the software system.

Based on the knowledge acquired in the training step, **LIA** will learn to predict the sequence of actions for a particular user. Finally, this prediction could be used for assisting users in their interaction with a specific system. Currently, we are focusing only on determining accurate predictions. Future improvements of our approach will deal with enhancing **LIA** with assistance capability, too.

The main contributions of this paper are:

- To develop an intelligent interface agent that learns using an original supervised learning method.
- To present a theoretical model on which our approach is based.
- To define an evaluation measure for determining the precision of the agent's prediction.
- To use *Aspect Oriented Programming* in the agent development.

The paper is structured as follows. Section 2 presents our approach in developing a learning interface agent for predicting users behavior. An experimental evaluation on a case study is described in Section 3. Section 4 compares our approach with existing ones. Conclusions and further work are given in Section 5.

2 Our Approach

In this section we present our approach in developing a learning interface agent (**LIA**) for predicting users behavior. Subsection 2.1 introduces the theoretical model needed in order to describe the agent behavior given in Subsection 2.2. The overall architecture of **LIA** agent is proposed in Subsection 2.3.

2.1 Theoretical Model

In the following, we will consider that **LIA** agent monitors the interaction of users with a software application $\mathcal{SA}$, while performing a given task $\mathcal{T}$. We denote by $\mathcal{A}$ the set $\{a_1, a_2, \ldots, a_n\}$ of all possible actions that might appear during the interaction with $\mathcal{SA}$. An action can be: pushing a button, selecting a menu item, filling in a text field, etc.

During the users interaction with $\mathcal{SA}$ in order to perform $\mathcal{T}$, user traces are generated. A *user trace* is a sequence of user actions. We consider a *user trace successful* if the given task is accomplished. Otherwise, we deal with an *unsuccessful* trace.

Currently, in our approach we are focusing only on *successful* traces, that is why we formally define, in the following, this term.

Definition 1. ***Successful user trace***
Let us consider a software application $\mathcal{SA}$ and a given task $\mathcal{T}$ that can be performed using $\mathcal{SA}$. A sequence $t = < x_1, x_2, \ldots, x_{k_t} >$, where

- $k_t \in \mathbb{N}$, *and*
- $x_j \in \mathcal{A}, \quad \forall 1 \le j \le k_t$

which accomplishes the task $\mathcal{T}$ is called a ***successful user trace****.*

In Definition 1, we have denoted by k_t the number of user actions in trace t. We denote by $\mathcal{ST}$ the set of all *successful user traces*.

In order to predict the user behavior, **LIA** agent stores a collection of *successful user traces* during the training step. In our view, this collection represents the *knowledge base* of the agent.

Definition 2. ***LIA's Knowledge base – $\mathcal{KB}$***
Let us consider a software application $\mathcal{SA}$ and a given task $\mathcal{T}$ that can be performed using $\mathcal{SA}$. A collection $\mathcal{KB} = \{t_1, t_2, \ldots, t_m\}$ of successful user traces, where

- $t_i \in \mathcal{ST}, \forall 1 \le i \le m$,
- $t_i = < x_1^i, x_2^i, \ldots, x_{k_i}^i >, \forall x_j^i \in \mathcal{A}, \ 1 \le j \le k_i$

represents the ***knowledge base*** *of* ***LIA*** *agent.*
We mention that m represents the cardinality of $\mathcal{KB}$ and k_i represents the number of actions in trace t_i $(\forall 1 \le i \le m)$.

Definition 3. ***Subtrace of a user trace***
Let $t = < s_1, s_2, \ldots, s_k >$ be a trace in the knowledge base $\mathcal{KB}$. We say that $sub_t(s_i, s_j) = < s_i, s_{i+1}, \ldots, s_j >$ $(i \le j)$ is a ***subtrace*** *of t starting from action s_i and ending with action s_j.*

In the following we will denote by $\|t\|$ the number of actions (length) of (sub) trace t. We mention that for two given actions s_i and s_j $(i \ne j)$ there can be many subtraces in trace t starting from s_i and ending with s_j. We will denote by $\mathcal{SUB}(s_i, s_j)$ the set of all these subtraces.

2.2 LIA Agent Behavior

The goal is to make **LIA** agent capable to predict, at a given moment, the appropriate action that a user should perform in order to accomplish $\mathcal{T}$.

In order to provide **LIA** with the above-mentioned behavior, we propose a supervised learning technique that consists of two steps:

1. Training Step
During this step, **LIA** agent monitors the interaction of a set of real users while performing task $\mathcal{T}$ using application $\mathcal{SA}$ and builds its knowledge base $\mathcal{KB}$ (Definition 2). The interaction is monitored using AOP.

In a more general approach, two knowledge bases could be built during the training step: one for the successful user traces and the second for the unsuccessful ones.

2. Prediction Step

The goal of this step is to predict the behavior of a new user *U,* based on the data acquired during the training step, using a probabilistic model. After each action *act* performed by *U*, excepting his/her starting action, **LIA** will predict the next action, $a_r\,(1 \le r \le n)$, to be performed, with a given probability $P(act, a_r)$, using $\mathcal{KB}$.

The probability $P(act, a_r)$ is given by Equation (1).

$$P(act, a_r) = \max\{P(act, a_i), 1 \le i \le n\}. \tag{1}$$

In order to compute these probabilities, we introduce the concept of *scores* between two actions. The score between actions a_i and a_j, denoted by *score*(a_i, a_j) indicates the degree to which a_j must follow a_i in a successful performance of $\mathcal{T}$. This means that the value of *score*(a_i, a_j) is the greatest when a_j should immediately follow a_i in a successful task performance.

The score between a given action *act* of a user and an action a_q, $1 \le q \le n$, *score*(*act*, a_q), is computed as in Equation (2).

$$score(act, a_q) = \max\left\{\frac{1}{dist(t_i,\ act,\ a_q)},\ 1 \le i \le m\right\}, \tag{2}$$

where $dist(t_i,\ act,\ a_q)$ represents, in our view, the *distance* between two actions *act* and a_q in a trace t_i, computed based on $\mathcal{KB}$.

$$dist(t_i,\ act,\ a_q) = \begin{cases} length(t_i, act, a_q) - 1 & if\ \exists sub_{t_i}(act, a_q) \\ \infty & otherwise \end{cases}. \tag{3}$$

$length(t_i, act, a_q)$ defines the minimum distance between *act* and a_q in trace t_i.

$$length(t_i,\ act,\ a_q) = min\{\|s\| \mid s \in \mathcal{SUB}_{t_i}(act,\ a_q)\}. \tag{4}$$

In our view, $length(t_i,\ act,\ a_q)$ represents the minimum number of actions performed by the user *U* in trace t_i, in order to get from action *act* to action a_q, i.e., the minimum length of all possible subtraces $sub_{t_i}(act, a_q)$.

From Equation (2), we have that *score*(*act,* a_q) $\in$ [0,1] and the value of *score*(*act,* a_q) increases as the *distance* between *act* and a_q in traces from $\mathcal{KB}$ decreases.

Based on the above *scores*, $P(act, a_i), 1 \le i \le n$, is computed as follows:

$$P(act, a_i) = \frac{score(act,\ a_i)}{\max\{score(act, a_j) \mid 1 \le j \le n\}}. \tag{5}$$

In our view, based on Equation (5), higher probabilities are assigned to actions that are the most appropriate to be executed.

The result of the agent's prediction is the action a_r that satisfies Equation (1). We mention that in a non-deterministic case (when there are more actions having the same maximum probability P) an additional selection technique can be used.

2.3 LIA Agent Architecture

In Fig. 1 we present the architecture of **LIA** agent having the behavior described in Section 2.2.

In the current version of our approach, the predictions of **LIA** are sent to an *Evaluation Module* that evaluates the accuracy of the results (Fig. 1). We intend to improve our work in order to transform the agent in a personal assistant of the user. In this case the result of the agent's prediction will be sent directly to the user.

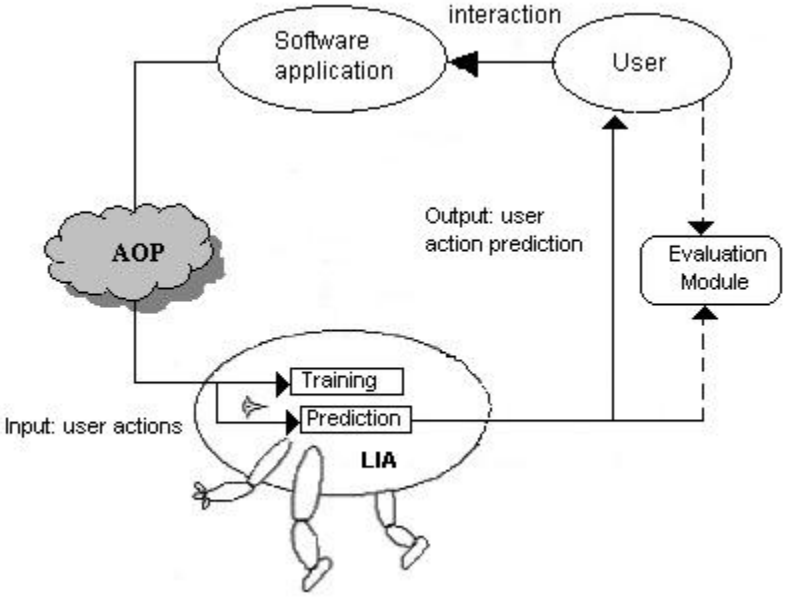

Fig. 1. LIA agent architecture

The agent uses AOP in order to gather information about its environment. The AOP module is used for capturing user's actions: mouse clicking, text entering, menu choosing, etc. These actions are received by **LIA** agent and are used both in the training step (to build the knowledge base $\mathcal{KB}$) and in the prediction step (to determine the most probable next user action).

We have decided to use AOP in developing the learning agent in order to take advantage of the following:

- Clear separation between the software system $\mathcal{SA}$ and the agent.
- The agent can be easily adapted and integrated with other software systems.

- The software system $\mathcal{SA}$ does not need to be modified in order to obtain the user input.
- The source code corresponding to input actions gathering is not spread all over the system, it appears in only one place, the *aspect*.
- If new information about the user software system interaction is required, only the corresponding aspect has to be modified.

3 Experimental Evaluation

In order to evaluate **LIA's** prediction accuracy, we compare the sequence of actions performed by the user U with the sequence of actions predicted by the agent. We consider an action prediction accurate if the probability of the prediction is greater than a given threshold.

For this purpose, we have defined a quality measure, $ACC(\mathbf{LIA}, U)$, called *ACCuracy*. The evaluation will be made on a case study and the results will be presented in Subsection 3.3.

3.1 Evaluation Measure

In the following we will consider that the training step for **LIA** agent was completed. We are focusing on evaluating how accurate are the agent's predictions during the interaction between a given user U and the software application $\mathcal{SA}$. Let us consider that the user trace is $t_U =< y_1^U, y_2^U, \ldots, y_{k_U}^U >$ and the trace corresponding to the agent's prediction is the following: $t_{LIA}(t_U) =< z_2^U, \ldots, z_{k_U}^U >$. For each $2 \le j \le k_U$, **LIA** agent predicts the most probable next user action, z_j^U, with the probability $P(y_{j-1}^U, z_j^U)$ (Section 2.2). The following definition evaluates the accuracy of **LIA** agent's prediction with respect to the user trace t_U.

Definition 4. ***ACCuracy of LIA agent prediction - ACC***
The accuracy of the prediction with respect to the user trace t_U *is given by* Equation (6).

$$ACC(t_U) = \frac{\sum_{j=2}^{k_U} acc(z_j^U, y_j^U)}{k_U - 1}, \tag{6}$$

where

$$acc(z_j^U, y_j^U) = \begin{cases} 1 & \text{if } z_j^U = y_j^U \text{ and } P(y_{j-1}^U, z_j^U) > \alpha \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

In our view, $acc(z_j^U, y_j^U)$ indicates if the prediction z_j^U was made with a probability greater than a given threshold α, with respect to the user's action y_{j-1}^U. Consequently, $ACC(t_U)$ estimates the overall precision of the agent's prediction regarding the user trace t_U.

Based on Definition 4 it can be proved that $ACC(t_U)$ takes values in [0, 1]. Larger values for *ACC* indicate better predictions.

We mention that the accuracy measure can be extended in order to illustrate the precision of **LIA**'s prediction for multiple users, as given in Definition 5.

Definition 5. ***ACCuracy of LIA agent prediction for Multiple users - ACCM***
Let us consider a set of users, $\mathcal{U}=\{U_1,\ldots,U_l\}$. *Let us denote by* $\mathcal{UT}=\{t_{U_1}, t_{U_2},\ldots,t_{U_l}\}$ *the set of successful user traces corresponding to the users from* $\mathcal{U}$. *The accuracy of the prediction with respect to the user* U_i *and his/her trace* t_{U_i} *is given by* Equation (8):

$$ACCM(\mathcal{UT})=\frac{\sum_{i=1}^{l} ACC(t_{U_i})}{l}. \tag{8}$$

where $ACC(t_{U_i})$ *is the prediction accuracy for user trace* t_{U_i} *given in* Equation (6).

3.2 Case Study

In this subsection we describe a case study that is used for evaluating **LIA** predictions, based on the evaluation measure introduced in Subsection 3.1.

We have chosen for evaluation a medium size interactive software system developed for faculty admission. The main functionalities of the system are:

- Recording admission applications (filling in personal data, grades, options, particular situations, etc.).
- Recording fee payments.
- Generating admission results.
- Generating reports and statistics.

For this case study, the set of possible actions $\mathcal{A}$ consists of around **50** elements, i.e., $n \approx 50$. Some of the possible actions are: filling in text fields (like first name, surname, grades, etc.), choosing options, selecting an option from an options list, pressing a button (save, modify, cancel, etc.), printing registration forms and reports.

The task $\mathcal{T}$ that we intend to accomplish is to complete the registration of a student. We have trained **LIA** on different training sets and we have evaluated the results for different users that have successfully accomplished task $\mathcal{T}$.

3.3 Results

We mention that, for our evaluation we have used the value **0.75** for the threshold α. For each pair (training set, testing set) we have computed *ACC* measure as given in Equation (6). In Table 1 we present the results obtained for our case study. We mention that we have chosen **20** user traces in the testing set. We have obtained accuracy values around **0.96**.

Table 1. Case study results

Training dimension	ACCM
67	0.987142
63	0.987142
60	0.987142
50	0.982857
42	0.96

As shown in Table 1, the accuracy of the prediction grows with the size of the training set. The influence of the training set dimension on the accuracy is illustrated in Fig. 2.

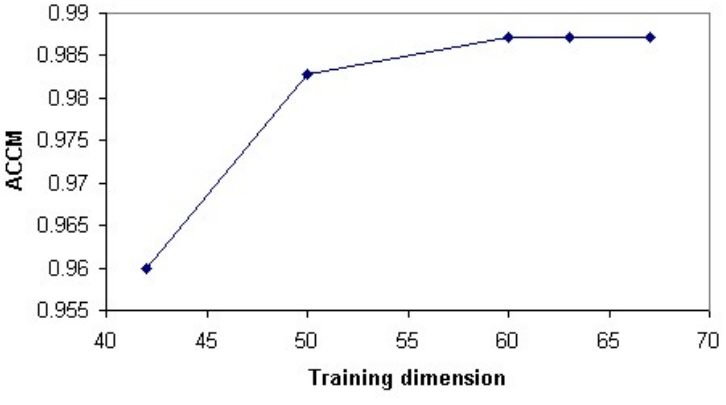

Fig. 2. Influence of the training set dimension on the accuracy

4 Related Work

There are some approaches in the literature that address the problem of predicting user behavior. The following works approach the issue of user action prediction, but without using intelligent interface agents and AOP.

The authors of [1-3] present a simple predictive method for determining the next user command from a sequence of Unix commands, based on the Markov assumption that each command depends only on the previous command. The paper [6] presents an approach similar to [3] taking into consideration the time between two commands.

Our approach differs from [3] and [6] in the following ways: we are focusing on desktop applications (while [3] and [6] focus on predicting Unix commands) and we have proposed a theoretical model and evaluation measures for our approach.

Techniques from machine learning (neural nets and inductive learning) have already been applied to user traces analysis in [4], but these are limited to fixed size patterns.

In [10] another approach for predicting user behaviors on a Web site is presented. It is based on Web server log files processing and focuses on predicting the page that a user will access next, when navigating through a Web site. The prediction is made using a training set of user logs and the evaluation is made by applying two measures. Comparing with this approach, we use a probabilistic model for prediction, meaning that a prediction is always made.

5 Conclusions and Further Work

We have presented in this paper an agent-based approach for predicting users behavior. We have proposed a theoretical model on which the prediction is based and we have evaluated our approach on a case study. Aspect Oriented Programming was used in the development of our agent.

We are currently working on evaluating the accuracy of our approach on a more complex case study.

We intend to extend our approach towards:

- Considering more than one task that can be performed by a user.
- Adding in the training step a second knowledge base for unsuccessful executions and adapting correspondingly the proposed model.
- Identifying suitable values for the threshold α .
- Adapting our approach for Web applications.
- Applying other supervised learning techniques (neural networks, decision trees, etc.) ([9]) for our approach and comparing them.
- Extending our approach to a multiagent system.

Acknowledgments. This work was supported by grant TP2/2006 from Babeş-Bolyai University, Cluj-Napoca, Romania.

References

1. Davison, B.D., Hirsh, H.: Experiments in UNIX Command Prediction. In: Proceedings of the Fourteenth National Conference on Artificial Intelligence, Providence, RI, p. 827. AAAI Press, California (1997)
2. Davison, B.D., Hirsh, H.: Toward an Adaptive Command Line Interface. In: Proceedings of the Seventh International Conference on Human Computer Interaction, pp. 505–508 (1997)
3. Davison, B.D., Hirsh, H.: Predicting Sequences of User Actions. In: Predicting the Future: AI Approaches to Time-Series Problems, pp. 5–12, Madison, WI, July 1998, AAAI Press, California. In: Proceedings of AAAI-98/ICML-98 Workshop, published as Technical Report WS-98–07 (1998)

4. Dix, A., Finlay, J., Beale, R.: Analysis of User Behaviour as Time Series. In: Proceedings of HCI'92: People and Computers VII, pp. 429–444. Cambridge University Press, Cambridge (1992)
5. Dix, A., Finlay, J., Abowd, G., Beale, R.: Human-Computer Interaction, 2nd edn. Prentice-Hall, Inc, Englewood Cliffs (1998)
6. Jacobs, N., Blockeel, H.: Sequence Prediction with Mixed Order Markov Chains. In: Proceedings of the Belgian/Dutch Conference on Artificial Intelligence (2003)
7. Kiczales, G., Lamping, J., Menhdhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., Irwin, J.: Aspect-Oriented Programming. In: Aksit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997)
8. Maes, P.: Social Interface Agents: Acquiring Competence by Learning from Users and Other Agents. In: Etzioni, O. (ed.) Software Agents — Papers from the 1994 Spring Symposium (Technical Report SS-94-03), pp. 71–78. AAAI Press, California (1994)
9. Russell, S., Norvig, P.: Artificial Intelligence - A Modern Approach. Prentice-Hall, Inc., Englewood Cliffs (1995)
10. Trousse, B.: Evaluation of the Prediction Capability of a User Behaviour Mining Approach for Adaptive Web Sites. In: Proceedings of the 6th RIAO Conference — Content-Based Multimedia Information Access, Paris, France (2000)