An Intelligent Inference Approach to User Interaction Modeling in a Generic Agent Based Interface System

Yanguo Jing, Nick Taylor, Keith Brown
Intelligent Systems Laboratory
Department of Computing & Electrical Engineering
Heriot-Watt University, Edinburgh, UK
EH14 4AS
+44 131 4495111 Ext. 4164
{ceeyj, nick, keb }@cee.hw.ac.uk

ABSTRACT
Research has shown that a typical user exhibits a certain pattern when interacting a computer system. This work infers the user interaction habits automatically from the actions that the user and the application produce. The user interaction habits can enable an interface agent to provide assistance to a user or operate the application on the user's behalf. All these actions are automatically segmented into several meaningful action sequences, which are stored in the user's interaction history. An inference algorithm is used to construct the user's interaction model from his/her interaction history. Probabilities of each action's next possible action are calculated. Interface agents then provide assistance based on a user's interaction model. The interaction model is modified automatically as the interface agents collect more actions. This method is embedded within a generic agent based interface system. Information on the prototype system, some experiments and future work are presented.

Keywords
Intelligent interface agents, interaction habits, task, user model, interaction model, inference

INTRODUCTION
The field of intelligent interface agents has emerged during the past few years to address the increasing complexity of current software systems. One of the key problems is how to build up situation-action links. Maes [7] used memory-based reasoning techniques to predict a user's interests, however this method needs domain knowledge to decide the elements of a situation. Bauer [1][2] exploited clustering techniques to learn action hierarchies in automated assistants. His algorithm learns only the required steps to accomplish each top-level goal. Garland and Lesh [3] tried to extend Bauer's work to support hierarchical task models and optional steps. They provided an algorithm for learning recipes, which are used to decompose non-primitive actions into sub-goals. In the APE project, Ruvini and Dony used an algorithm called IDHYS [9] to learn the user's repetitive operations, however the algorithm only targets the user's repetitive tasks. We are more interested in a user's primitive actions and the orders between these actions. We use actions produced by users and applications to infer the user's interaction habits. Research [5] has shown that a typical user exhibits a certain pattern when interacting a computer system. The "bottom up" method presented in this paper can be used to recognize the patterns that are called in a user's interaction model in this paper.

This paper is organized as follows. The next section discusses the details of the user interaction model construction method. After that, we explain how this method is implemented in an agent based interface system for an eye disorders diagnosis system (ARMD). Then some preliminary experimental results are presented. Finally, conclusions and future work are detailed.

THE INTERACTON MODEL CONSTRUCTION
As a user interacts with a system, many events are generated and operations are undertaken. All these events and operations are collected by an Observer. The collected events and operations are segmented into several meaningful sequences of operations, which are called action sequences. These action sequences are stored in the user's interaction history (IH). A user's interaction model (IM) is inferred from all the collected actions to represent his/her interaction habits. At the moment, the interaction model inferring method takes into account the order of just two actions (or events) in an action sequence. There are two reasons for this: Firstly, the order between two actions can be used to infer the order among several (more than two) actions. Secondly, as the number of actions whose order is taken into account increases, the computational effort increases exponentially.

The whole method consists of four algorithms, the action collection algorithm, the action sequences segmentation, the interaction model construction algorithm and the probability calculation algorithm.

Action Collection

The actions are collected by the Observer in our model based intelligent interface agent (MI2A) architecture [7]. The Observer looks over the user's shoulder, collects all the user's actions (through mouse and keyboard), selects meaningful actions and sends them to the user model management agent, where the interaction model is produced and stored.

Action Segmentation Algorithm

All the actions sent to the user model management agent need to be segmented into action sequences. A meaningful action sequence consists of several user actions, which are taken by the user to achieve particular tasks. Rich and Sidner [8] used a recipe tree to help the agent target this problem. Our approach is to distinguish each action's function according to the task model. Detailed information about the task model is beyond the scope of this paper, please refer to [6]. The task model is structured within a graph. Each user action is associated with one specific task node in the task model. Figure 1 shows an example of a task model. In order to achieve task T6, task T5 needs to be achieved; in order to achieve T5, either T3 or T4 need to be achieved; in order to achieve T4, either T1 or T2 need to be achieved; in order to achieve T3, T1 needs to be achieved. Tasks T1, T2, T3, T4, T5 are called T6's *pre-conditions*, task T5 is called task T6's *direct pre-condition*. Tasks T3, T5, T6 are called T1's *post-conditions*, Task T3 is called T1's *direct post-condition*.

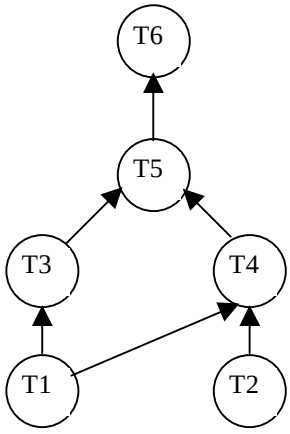

Figure 1 A Task Model Example

Let *task(a)* represent a function which returns an associated task of action *a*. The segmentation algorithm is given in Figure 2:

1 Set action sequence S_{temp} to empty.

2 For each action a_i in the collected action lists - L, do
{

3 If action a_i is included in S_{temp} or
task(a_i)!=task(a_j) or
task(a_i) is not the direct post-condition of task(a_j), then
{

4 Add S_{temp} to the interaction history

5 Set action sequence S_{temp} to empty

6 Add a_i to S_{temp}

7 $a_j = a_i$, a_j is the last added action in S_{temp}
}
Else
{

8 Add action a_i to S_{temp} as the last added action in it.

9 $a_j = a_i$, a_j is the last added action in S_{temp}
} }

Figure 2 The Segmentation Algorithm

An action sequence is terminated in one of the following ways 1. When the new-collected action has already appears in the action sequence. 2. When the new-collected action corresponds to a different task to the last added action in the action sequence. 3. When the new collected action corresponds to a task that is not the direct post-condition of the last added action's corresponding task.

Consider the following example. Suppose the Observer collects the action list { a_{21}, a_{22}, a_{41}, a_{42}, a_{21}, a_{22}, a_{41}, a_{42}, a_{51}, a_{52}, a_{61}, a_{62}, a_{11}, a_{12} }. Now let a_{21}, a_{22} be associated with task T2; a_{41}, a_{42} be associated with T4, a_{51}, a_{52} be associated with T5; a_{61}, a_{62} be associated with T6; a_{11}, a_{12} be associated with T1. Using the segmentation algorithm, we get three action sequences { a_{21}, a_{22}, a_{41}, a_{42} }, { a_{21}, a_{22}, a_{41}, a_{42}, a_{51}, a_{52}, a_{61}, a_{62} } and { a_{11}, a_{12} }.

Interaction Model Construction

The interaction model construction algorithm is shown in Figure 3. As previously explained, this method only takes into account the order of two actions. These two actions constitute an ordered action pair.

In the first step, we select ordered action pairs from the interaction history. (a1, a2) is an ordered action pair,

which means that action a1 is performed before action a2 is performed. Action a1 and a2 are not necessarily contiguous actions in the action sequence in this "candidate collecting" stage, but when we do the probability calculation, only the contiguous action pairs are taken into account.

In the second step, the final action pairs are selected, based on a statistically derived threshold from the ordered action pairs. The threshold is a percentage decided by the user (default = 100%), which is used to determine whether a particular pair of actions occurs sufficiently often in a given order to be included in the final action pairs. For example, if the threshold is set to 80% and action a1 is performed before action a2 in 85% cases and performed after action a2 in 15% cases in the interaction history, then the ordered action pair (a1, a2) is added to the final action pairs (because 85% > 80%).

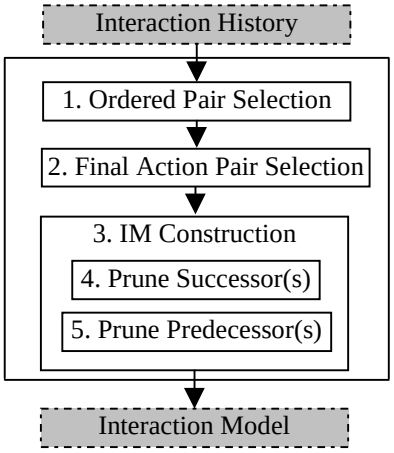

Figure 3 Overview of the Interaction Model Design Method

The selected final action pairs are added to a graph. Pruning algorithms are used to simplify the graph. This graph is the final representation of the Interaction Model.

Here is a simple abstract example: Suppose we have the following interaction history (IH), which consists of two action sequences:

Sequence 1: a1 -> a2 -> a3 -> a4.
Sequence 2: a3 -> a1 -> a2 -> a4.

Using the method shown in Figure 4, the interaction model is inferred in the following steps.

Step 1: We get the following ordered action pairs from Sequence 1: (a3, a4), (a2, a4), (a1, a4), (a2, a3), (a1, a3), (a1, a2) and the following ordered action pairs from Sequence 2: (a2, a4), (a1, a4), (a3, a4), (a3, a2), (a1, a2), (a3, a1).

Step 2: We set the statistical threshold to 100%, then we get the following final action pairs: (a3, a4), (a2, a4), (a1, a4), (a1, a2).

Step 3: The action pairs we get from the final action pairs are added to a graph one by one (we use an arrow to connect two actions in an action pair with the arrow pointing to the second action in the action pair), Step 4 and step 5 are used to prune the graph. For example when action pair (a1, a2) is added to the tree, we get the graph shown in Figure 4, but it still needs to be simplified, the leaf a1 has two paths to a4. Using Step 4 - Prune Successor, we remove the arrow between a1 and a4. The reason why we don't remove the arrow between a1 and a2 is that if we do this, we will lose the order information between a1 and a2, but if we remove the arrow between a1 and a4, we still retain the information that a4 is a successor to a1. This is exactly the information the Predictor needs from a user's interaction model.

Note: The interaction model is designed to represent a user's interaction habits and the interaction model in Figure 5 means that actions a2 and a4 are likely to be taken after action a1. It is not necessary to follow the a1->a2->a4 order. Figure 4 represents the same meaning, so this is a valid transformation.

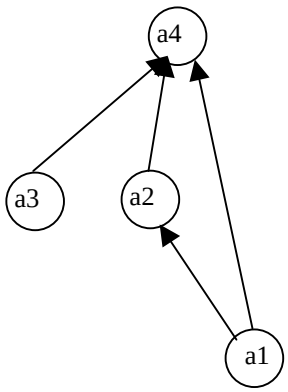

Figure 4 An Interaction Model Example at Step3

After step 3, step 4 and step 5, we get the tree shown in Figure 5, which is the interaction model for this user, representing his/her interaction habits.

In Figure 5, we call action a1 a direct pre-condition of action a2; we call action a4 a direct post-condition of action a2. Given an action pair, the Prune Successor algorithm is used to remove the first action in this action pair from the pre-condition of all the second action's post-conditions. Given an action pair, a Prune Predecessor algorithm is used to remove the second action from the post-condition of all the first action's pre-conditions.

Note: An action's next possible actions include not only the node's direct post-conditions, but also all other post-conditions. In Figure 5, action a1's possible next actions include a2 and a4. The example in Figure 5 is a very simple one, which only includes four actions. As the user actions collected by the Observer increase, the user interaction model will grow as well. The interaction model given in the Prototype and Experiments section is such an example.

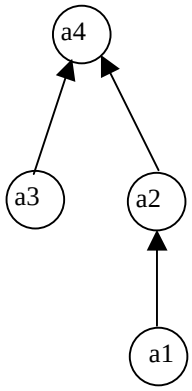

Figure 5 A Final Interaction Model Example

Probability Calculation

A similar algorithm to IOLA is used to calculate the probability of each node's next possible action. IOLA is proposed by B. D. Davision and H. Hirsh in [3]. IOLA makes an assumption that each command depends only on the previous commands. History data is used to count the number of times that each command pair has occurred followed each other and thus calculate the probability of a future command. An n-by-n table is used to show the likelihood of going from one command to the next. There are some limitations to this algorithm.

1. The IOLA method is proposed for an ideal situation where each command depends only on the previous command, which is not the case in most applications. In our experiences, when trying to achieve a specific task, a user's actions are most dependent on the previous actions involved in that task, however, when a new task is attempted, it is largely independent of the actions in other tasks. This is the reason why we segment the collected actions into meaningful action sequences, each of which is supposed to achieve a specific task.

2. The IOLA method considers all of the user actions to be potential next action candidate. We only take the actions in post-conditions, which saves a great deal of calculation time and memory space. Each node in our interaction model has post-conditions associated with it, along with the probability that each post-condition is the next action. Figure 6 shows the probability calculation algorithm. We adopt IOLA's method for increase the likelihood of the most recent post_condition. IOLA used a constant α, which reduced the probabilities of non-recent-observed post-conditions by multiplying them by α and increase the probability of the recent-observed post-condition to α*original probability+(1-α).

```
1 Update (An action sequence  AS)
{
2   For every node N in the interaction history
  {
3       Let SET be the set that consists of all N's
post_conditions.
4   For every node M in SET
    {
5       if action pair (N, M) is an immediate successor action
pair appears in AS, then
6         UpdateNode(N, M);
    }  }}
```

```
1  UpdateNode(node N, node M)
{
2     if node N's probability has not been initialized then,
      {
3       Set all probability to 0.
      }
4     Multiply of each post-condition's probability by α
5     add (1-α) to the probability of M
  }}
```

Figure 6 The probability calculation algorithm

How To Use The User Interaction Model To Give Assistance

The assistance given is very straightforward. Once the Observer agent observes a user doing a particular action a_i, it will send to the user modeling management agent, who will inform the Predictor that a_i is just observed, the Predictor will select the action that has the highest probability in a_i's post-conditions to be the predicting action based on the user's interaction model (IM).

PROTOTYPE AND EXPERIMENTS

We implemented this method in an eye disorders diagnosis system called ARMD [6][7] using Java, Jess and Jade. The main function of this system is to identify indicators of age-related macular degeneration (ARMD). A camera is used to obtain a raw image of a retina. A validated artificial neural network (ANN) is used to diagnose a patient's eye disorder. A patient and a patient's medical record should be created or selected before a diagnosis is attempted. To diagnose with the validated ANN, a validated ANN must first be selected and then the diagnosis can be performed. If the ANN is not good enough (in terms of its accuracy), it should be trained. After the ANN is trained, it should be assessed to evaluate its accuracy.

As an experiment, a user played a doctor's role to use this software to diagnose eye disorders. 102 user actions were collected. These were segmented into 32 action sequences in the interaction history (IH). Using the interaction model construction methods introduced in this paper, we obtained the interaction model shown in Figure 7. This interaction model is used to represent the user's interaction habits. For example, when the doctor selects "Open a Patient's retina image" operation from the menu, the Observer will collect this action and translate it into a form that can be understood by other interface agents. The Predictor will use the prediction algorithm to predict the next user action. Figure 7 shows that the Predictor predicts the next action is "using the ANN to diagnose".

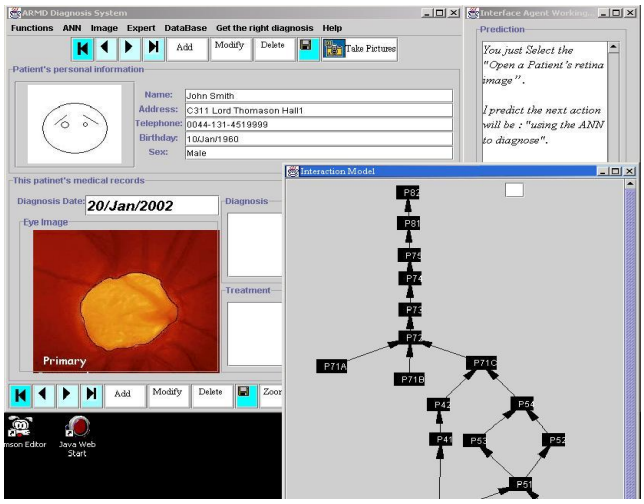

Figure 7 The ARMD System

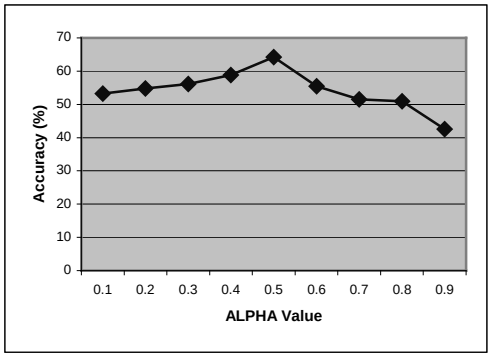

Figure 8: A range of ALPHA values that affects the predictive accuracy of the method.

Figure 8 shows a range of ALPHA values that affect the predictive accuracy of the method. Figure 8 also shows that the highest accuracy we achieved in this experiment was 64.2% (when ALPHA=0.5). This is only a preliminary result, more additional experiments with more users are needed for the further evaluation of the system. As we mentioned previously, although we only take into account the order of two actions, we can still infer the order of many actions from the interaction model and the primary interactions are retained in the interaction history.

CONCLUSIONS

We have presented a user interaction model inference algorithm and its implementation. A prototype system has been presented using the methods described. The methods presented in this paper seem to produce good results based on our initial tests. This method is embedded in a generic agent based interface system [7]. Although this architecture is designed for the ARMD system, we believe that it can also be used in other systems requiring intelligent interface agents. More experiments and evaluations are planned for the future. Future work also includes building up a general and robust segmentation method to segment the actions into meaningful action sequences.

ACKNOWLEGEMENTS

This work is supported by a Heriot-Watt University Scholarship and Computing & Electrical Engineering departmental funds. We want to thank the anonymous reviewers for providing wonderful feedbacks.

REFERENCES

[1] M. Bauer, Acquisition of Abstract Plan Descriptions for Plan Recognition, AAAI'98, pp. 936 – 941, 1998.

[2] M. Bauer, From Interaction Data to Plan Libraries: A Clustering Approach, IJCAI'99, pp. 962 – 967, 1999.

[3] B. D. Davision, H. Hirsh, Predicting Sequences of User Actions, the proceedings of AAAI'98, 1998.

[4] A. Garland, N. Lesh, Learning Hierarchical Task Models by Demonstration, the proceedings of 11th Int. Joint Conf. on AI, 2001.

[5] H. Hirsh, C. Basu, B. D. Davison, Learning to personalize, Communications of the ACM, Vol. 43, N. 8, August, 2000.

[6] Y. Jing, K. Brown, N. Taylor, A Model Based Architecture for Intelligent Interface Agents, proceedings of the UK workshop on Computational Intelligence, pp. 169 – 174, 2000.

[7] Y. Jing, K. Brown, N. Taylor, Intelligent Interface Agents for a System to Diagnose Eye Disorders, to appear in the proceedings of the first joint conference of Autonomous Agent & MultiAgent Systems (AAMAS), 2002.

[8] P. Maes, "Agents that reduce work and information overload", *CACM*, Vol. 37, No. 7, pp. 30-40, 1994.

[9] C. Rich, C. L. Sidner, Segmented Interaction History in a Collaborative Interface Agent, proceedings of the international conference on Intelligent user interfaces, pp. 23 – 30, 1997.

[10] J, Ruvini, C. Dong, The "APE" Project: target the problem of decrease the burden of entering these sequences of commands, proceedings of International conference of Intelligent User Interfaces, 2000, pp. 229 – 232, 2000.