Designing More Interactive Interface Agents

Michael Fleming and **Robin Cohen**
Department of Computer Science
University of Waterloo
Waterloo, Ontario
N2L 5G7
mwflemin@neumann.uwaterloo.ca, rcohen@watdragon.uwaterloo.ca

This extended abstract describes our work on the interactivity of interface agents. In particular, it focuses on the development of a model for designing agents which are more or less autonomous, but which recognize opportunities for initiating communication with their users in order to garner further useful information. This design can be applied to many different domains. The workshop call for papers indicates two main themes around which the workshop discussions will be organized - the work below fits into the individual agent (rather than multi-agent) theme, and describes how a future requirement of making agents less autonomous and more trustworthy may be addressed.

In recent years, the area of intelligent agents has been one of the most prevalent fields of research in the AI community. This paper deals with one specific type of agent, the interface agent, which is a program that acts as a personal assistant to a user dealing with a particular computer-based application, and which is able to "view" and act upon the application interface just as a human user might. Previous designs of interface agents can be broadly classified into two categories: autonomous agents (*e.g.*, (Maes 1994)), which attempt to automate certain actions on behalf of the user, and collaborative agents (*e.g.*, (Rich & Sidner 1997)), which are more equal partners with their users, working together on a joint plan and participating in a dialogue in order to determine an appropriate course of action.

We argue that there is a middle ground to be covered. Using autonomous learning interface agents as a starting point, we propose a model which makes these agents more interactive, allowing them to take the initiative to solicit further input from the user, toward improving their overall performance. A very high-level algorithm for our semi-autonomous agents is shown in Figure 1. The major points of this algorithm will be explained throughout this abstract.

The first novel aspect of our algorithm, as compared to the learning interface agents developed at MIT (see (Maes 1994), for example), is its incorporation of truly hard-and-fast rules into the agent's behaviour. An example of such a rule, from the e-mail domain, might be "If a message arrives with subject line 'Make money fast', then delete it." Rules can either be programmed by the user, or developed and proposed by the agent when it has high confidence in a prediction (as in Step 4 of Figure 1). We believe that the incorporation of rules is a necessary addition for two main reasons: (1) it will speed up the agent's performance in situations where it can simply apply a rule, rather than going through a series of complex calculations involved in the agent's learning algorithm; (2) it helps to provide the user with a better understanding of, more trust in, and a better sense of control over, the agent's behaviour.

We also address the problem of ambiguous situations: ones in which the agent, via its learning methods, is unable to select one course of action as being a clear winner. (See steps 6-10 in the algorithm.) For example, in the e-mail domain, suppose an agent has successfully learned that all messages from David Fleming should be re-filed in the *David* folder and that all messages with subject "Hockey pool" should be filed in the *Hockey* folder. What will the agent do with a message from David Fleming with subject "Hockey pool"?

In such ambiguous situations, MIT's *Maxims* (Metral 1993) e-mail agent will likely do nothing, due to a low confidence level in its predicted action. Our more interactive agent, on the other hand, would examine the same situation and recognize that two (or more) candidate actions have similar scores. Based on how close together the scores are, along with a number of other factors (including how "important" the agent considers the candidate actions to be[1] and how often the user has been bothered recently), the agent will compute a *clarification factor*. This is then compared to a user-defined *bother threshold* to determine whether or not to initiate a clarification dialogue with the user, to find out which action is most appropriate in this situation and to attempt to generalize this into a rule. Even in cases in which the user is not immediately bothered by the agent, the user can be made aware of the agent's difficulty (*e.g.*, using an additional column in an e-mail program such as Eudora). The user may then decide to initiate a dialogue with the agent, at a convenient time. Our agents also allow for agent-user communication in

[1]based on the do-it thresholds (Maes 1994) established by the user for those actions

the event of conflicts occurring in the actual **rules** programmed by the user (Step 0). This communication is not through natural language, but rather via dialogue boxes, menus and buttons in a graphical user interface.

The goal of interface agents is to help the user deal with a particular computer-based application, and to off-load some of the tedious or repetitive work. Our work looks at the degree to which such systems communicate with a human user. There is a definite trade-off involved here: both the agent and user can benefit a great deal from increased interaction; however, an agent which constantly interrupts with questions and explanations is bound to become an annoyance. The model which we propose aims to provide improved performance over strictly learning interface agents, allowing users to be more aware (and trusting) of their agents' activities, while keeping the bother level to a minimum.

Moreover, our work has something to offer to the growing mixed-initiative research area ((Allen 1994), (Burstein & McDermott 1996)). In providing opportunities for both agents and users to take the initiative, we have presented one approach for designing mixed-initiative interface agents.

Although this discussion has focused on e-mail agents, the model is applicable to a broad range of application domains.

Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Research Council.

References

Allen, J. 1994. Mixed-initiative planning: Position paper. Presented at the ARPA/Rome Labs Planning Initiative Workshop. Available on the World Wide Web at http://www.cs.rochester.edu/research/trains/mip.

Burstein, M., and McDermott, D. 1996. Issues in the development of human-computer mixed-initiative planning. In Gorayska, B., and Mey, J., eds., *In Search of a Humane Interface.* Elsevier Science B.V. 285–303.

Kozierok, R. 1993. A learning approach to knowledge acquisition for intelligent interface agents. Master of science thesis, Massachusetts Institute of Technology, Cambridge MA.

Maes, P. 1994. Agents that reduce work and information overload. *Communications of the ACM* 37(7):31–40.

Metral, M. 1993. Design of a generic learning interface agent. Bachelor of science thesis, Massachusetts Institute of Technology, Cambridge MA.

Rich, C., and Sidner, C. 1997. Collagen: When agents collaborate with people. In *First International Conference on Autonomous Agents*, 284–291.

INPUT: A signal that there exists a new situation to be addressed (*e.g.*, in the e-mail domain: a new mail message arrives, the user has just finished reading a message, *etc.*)

OUTPUT: The agent has either completed an action on behalf of the user, or has suggested an action to the user, for the current situation.

```
(0) Consult rule database for applicable rules previously created by the user (with or without the
    agent's help). If a single rule is found to apply, then use that rule. If two or more conflicting
    rules are found, initiate rule conflict dialogue with user. If no rules are found to apply, then
    proceed with step 1.

(1) Use learning techniques to get possible actions $A_1, ..., A_n$

(2) if choice of action $A$ is clear[a] then

    (3) Compute confidence value $C$ (as in the MIT agents - see (Kozierok 1993), for example)

    (4) if $C >$ do-it threshold then perform action $A$ and indicate that there is a proposed rule
        for the user to approve/reject/edit

    (5) else if $C >$ tell-me threshold then suggest action $A$

(6) else //choice unclear because two or more actions have similar scores

    (7) if peer agents exist and are able to provide trustworthy advice then automate/suggest
        recommended action

    (8) else // choice still unclear

        (9) Compute clarification factor $CF$.

        (10) if $CF >$ user-defined bother threshold then initiate dialogue with user.
```

[a]The choice is considered clear if the score computed for the highest-scoring action exceeds the score of the next best choice by a constant difference threshold (say, 10%).

Figure 1: High-level algorithm for our more interactive interface agents