Interactive Interface Agents as Recommender Systems

Michael Fleming and **Robin Cohen**
Department of Computer Science
University of Waterloo
Waterloo, Ontario
N2L 3G1
mwflemin@neumann.uwaterloo.ca, rcohen@watdragon.uwaterloo.ca

Abstract

This paper presents a model for more interactive interface agents. Using learning interface agents is one strategy for designing recommender systems. The more interactive style of agents presented in this paper aims to increase the trust and understanding between the user and the agent, by allowing the agent to solicit further input from the user under certain conditions. We illustrate our design for more interactive interface agents by including some examples in the domain of electronic mail. We then discuss why the model is also applicable to designing recommender systems, in general.

Overview

One strategy for designing recommender systems is to construct a learning interface agent, which interacts with a user in order to produce its recommendations (*e.g.*, (Lieberman 1995)). This approach is one which provides personalized assistance, since the agent examines how a user interacts with the environment and then makes recommendations based on general patterns which are learned.

Previous designs of learning interface agents (*e.g.*, (Maes 1994)) attempt to automate actions on behalf of users, with a minimal amount of participation from the user. Our work has focused instead on developing more interactive agents (still restricted not to bother unduly), which solicit further input from the user, toward improving their overall performance. This kind of design allows for an increase in trust and understanding between the user and agent.

The point of this position paper is that our model of more interactive interface agents would be useful to apply to the design of recommender systems, in order to address the concern with insufficient trust in the recommendations which are provided. Although we have primarily examined individual interface assistance domains such as e-mail and scheduling, we believe that our model can be a useful starting point for the design of an agent to assist with applications such as recommending web pages to users.

Background

In order to develop our model, we have used as a starting point the learning interface agent architecture developed by the Software Agents group at MIT. The following is a very brief description of how these agents operate; see (Maes 1994) for more detail. The MIT agents act primarily by observing their users, and by using a form of learning called memory-based reasoning (Stanfill & Waltz 1986). For each new situation that arises, the agent computes the distance between the current state and each of the past situations it has stored in its memory, using a weighted sum of several relevant features. According to the actions taken by the user in the most similar past situations, the agent selects an action for the current situation, and calculates a corresponding confidence value (Kozierok 1993). According to "do-it" and "tell-me" thresholds established

PRIOR TO OPERATION: The user has set the tell-me and do-it thresholds, has indicated how many past situations the agent should look at during its action selection, etc.

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 completed an action on the user's behalf, has suggested an action, or has decided to do nothing for the current situation.

```
Select action $A$ via learning techniques and assign confidence value $C$.
if $C >$ do-it threshold then
  – perform action $A$ and add it to a list of automated actions for user
    to examine at his own leisure
  – if user indicates that action was incorrect, ask user to adjust priority
    weightings for the various features which contribute to calculations
else if $C >$ tell-me threshold then
  – suggest action $A$
else
  – consult other agents for help, establish suggested action $A'$ and compute new confidence value $C'$.
  – if $C' >$ do-it... (as above)
  – else if $C' >$ tell-me... (as above)
  – else do nothing
```

Figure 1: High-level algorithm for the behaviour of learning interface agents

by the user, the agent determines whether to automate an action on the user's behalf, to suggest an action, or to do nothing at all. Figure 1 shows a high-level description of the behaviour of these learning agents.

More Interactive Interface Agents

While the MIT design has many strong points, several shortcomings can be identified (Fleming 1998). In particular: (i) these agents do not deal very well with situations that are somewhat ambiguous; (ii) the lack of communication between agent and user makes it difficult for a user to understand and to trust such an agent; (iii) memory-based learning can be quite slow because it may require an examination of a large number of previous situations.

We address these issues and others, by presenting a variation on the architecture of the MIT learning agents. This new model allows for agents which are still more or less autonomous, but which recognize opportunities for asking the user for further information,

PRIOR TO OPERATION: The user has set the tell-me, do-it and bother thresholds, has indicated how many past situations the agent should look at during its action selection, *etc.*

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 completed an action on the user's behalf, has suggested an action or has communicated to the user that it can do nothing 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
    $A_1, \dots, A_n$

(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 2: High-level algorithm for our more interactive interface agents

with the goal of improving the agents' overall performance. A very high level algorithm for our semi-autonomous agents is shown in Figure 2. The major points of this algorithm are explained in this paper, illustrated for the domain of assisting users with e-mail.

Ambiguous situations

In our model, we 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"?

Suppose a message with the following feature values has just been read:

Feature	From	Cc	Date	Subject
Value	David Fleming	None	October 26	Hockey pool

Suppose also that the agent has assigned the following weights to each of the relevant fields, based on how well the current situation's value in each of those fields has typically predicted the action taken (as in (Kozierok 1993)).

Feature	From	Cc	Date	Subject
Weight	0.90	0.08	0.01	0.88

Finally, suppose that the following four messages were found to be the most similar to the current situation, with the distance between the value in the current situation and the corresponding value in the past situation shown in the third row. The overall distance between two situations (shown in the fourth row) is computed by taking the sum of the products $d_i w_i$, where d_i is the distance between the values of field i and w_i is the weight assigned to field i.

Feature	From	Cc	Date	Subject
Value	David Fleming	None	October 11	Habs
Distance	0	0	0.90	0.98
$\Delta(s_{new}, s_1)$	0.8714			
Action	File under David			

Feature	From	Cc	Date	Subject
Value	David Fleming	None	October 3	Hi
Distance	0	0	0.92	1
$\Delta(s_{new}, s_2)$	0.8892			
Action	File under David			

Feature	From	Cc	Date	Subject
Value	Owen Barnhill	None	October 7	Hockey pool
Distance	1	0	0.86	0
$\Delta(s_{new}, s_3)$	0.9086			
Action	File under Hockey			

Feature	From	Cc	Date	Subject
Value	S. Fillmore	None	October 23	Hockey pool
Distance	1	0	0.90	0
$\Delta(s_{new}, s_4)$	0.9090			
Action	File under Hockey			

In such a situation, MIT's *Maxims* (Metral 1993) e-mail agent would compute scores for each of the two candidate actions (*File under David* and *File under Hockey*), would choose the action with the higher score and would calculate a confidence value. In this case, the scores for the two actions would be very close together; the agent would choose to file the message in the *David* folder but would have a very low confidence value. As a result, this agent would likely do nothing in such a situation. It would be the responsibility of the user to realize that nothing had been done, and to perform an appropriate action himself.

Our more interactive agent, on the other hand, would examine the same situation and recognize that two candidate actions have similar scores. Based on how close together the scores are, along with a number of other factors,[1] the agent will compute a *clarification factor.* This clarification factor is then compared to a user-defined *bother threshold* to determine whether or not to initiate a clarification dialogue with the user. The goal of such a dialogue is to find out which action is most appropriate in this situation and to attempt to generalize this into a rule. An example screen is presented below:

Situation: The following message has just been read

From	Cc	Date	Subject	...
David Fleming	None	Oct. 27	Hockey pool	...

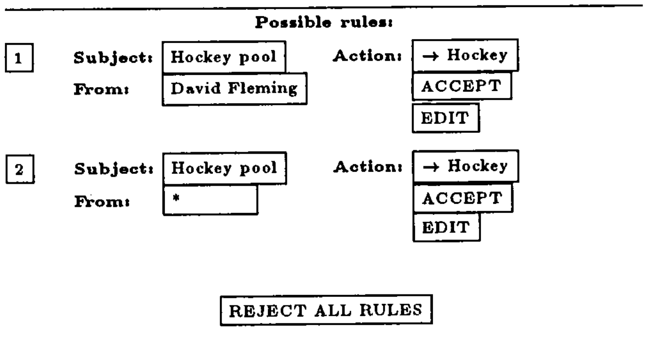
Possible actions:

Action	Score	Explanation
File under David	2.272	In past situations in which the sender was David Fleming, the action taken was *File under David* in 95% of cases.
File under Hockey	2.201	In past situations in which the subject was "Hockey pool", the action taken was *File under Hockey* in 100% of cases.

Please click on the action you wish to choose, or click Cancel to conclude this interaction.

If the user were to choose the action *File under Hockey*, for example, the agent would proceed to propose two rules, as seen in Figure 3. The first states specifically that when the subject line is "Hockey pool" and the message sender is David Fleming, the message should be filed in the *Hockey* folder. The second rule is more general, and states that any messages with subject line "Hockey pool", regardless of the sender, should be filed in the *Hockey* folder. The user has the option of accepting or editing either of these rules, or of cancelling the interaction entirely if neither rule is appropriate.

Even in cases in which the user is not immediately bothered by the agent (*i.e.*, the clarification factor does *not* exceed the bother threshold), the agent can indicate that it has a question for the user without actually requiring the user to deal with it immediately. To achieve this interaction, we propose having the agent maintain a


Possible rules:

1 Subject: Hockey pool From: David Fleming Action: → Hockey ACCEPT EDIT

2 Subject: Hockey pool From: * Action: → Hockey ACCEPT EDIT

REJECT ALL RULES

Figure 3: Agent's proposal of possible rules

"question box" where it would store information about situations with which it could benefit from the user's help, but for which it chose not to interrupt the user immediately due to a low clarification factor. This question box would appear in the interface as a small box in the lower left corner of the screen, indicating how many questions the agent currently had. The user could choose to click on this box at his own convenience, in order to initiate dialogues of the form presented earlier.

Rule base

Another novel aspect of our algorithm, as compared to the learning interface agents developed at MIT, 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 2). Although the MIT group does provide "rules" for its agents, these rules are simply represented as hypothetical situations, and are treated just as though they were situations the agent had actually observed in the past. In any new situation, an agent would still have to examine each past situation in its memory and go through a series of calculations. Our proposal is for the agent to maintain an entirely separate database of rules, which can be fired immediately whenever an appropriate situation is encountered.

We believe that the incorporation of rules is a necessary addition for two main reasons: (1) it will likely speed up the agent's performance[2] 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) because rules are more explicit and concrete than the calculations involved in learning techniques, having a separate rule base which is

[1]These factors include how "important" the agent considers the candidate actions to be (based on the do-it thresholds (Maes 1994) established by the user for those actions) and how often the user has been bothered recently. We omit the presentation of the actual formula in this short paper.

[2]Note that, in practice, the actual gain in performance by using a rule-based approach would depend strongly on the size of the rule base and on the format used to represent rules.

always available to inspect would help to provide the user with a better understanding of, more trust in, and a better sense of control over, the agent's behaviour. Our agents also allow for agent-user communication in 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. (Fleming 1998) presents examples illustrating dialogues to address such rule conflicts.

Self-monitoring and explanations

We also provide means by which agents can monitor their own performance, by comparing their predictions to the actions actually taken by the user, and can propose appropriate times for thresholds to be manipulated. This would reduce the amount of work the user has to do, in terms of watching the agent, determining how well it is performing, and deciding when to give the agent more or less autonomy.

Furthermore, we propose a more general technique for providing users with explanations of agent actions. Rather than simply referring the user to situations from the past which are considered similar to the current situation, the agent should be able to generalize what it has seen, in order to present it to the user in an English sentence. Details of each of these mechanisms can be found in (Fleming 1998).

Reflecting on Initiative

Our work also has something to offer to the growing mixed-initiative research area ((Allen 1994), (Burstein & McDermott 1996)). Several important issues have been identified, which must be addressed when designing mixed-initiative systems, including:

- specification of when exactly the system and user should communicate, and what that communication should look like;
- registration of context when one party interrupts the other;
- ensuring that both parties share the responsibilities involved in the task, and are fully aware of the responsibilities of each party.

For the particular application of interface agent design, our model addresses each of these issues.

An algorithm is presented for determining when an agent should choose to initiate communication with the user, and details are given about the format of this interaction. Registration of context is also taken into consideration in the model. Whenever the agent interrupts the user, it must take care to set the stage for what exactly it wishes to ask the user. For instance, in the example presented earlier, the agent registers the context by establishing that the user has just finished reading a message which the agent does not know how to treat, and by providing the user with the exact features of that particular message. In our model, the agent and user share responsibilities quite well, and should always be aware of who is responsible for what tasks. Upon encountering any new situation, it is understood that the agent will attempt to do whatever it can to perform an action for the user (or to make a suggestion) using the knowledge it has previously acquired. If it has insufficient information to do anything, it will still be able to inform the user by adding messages to the question box discussed earlier.[3] The design decisions made for interface agents may be useful to motivate the design of other kinds of mixed-initiative systems.

Recommender Systems

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 tradeoff 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.

The algorithm we have proposed identifies some general opportunities for initiating clarification dialogues with users, providing a mechanism for agents to interact with users, without bothering them excessively. The particular decisions which we have made, with regard to specific details such as the exact form of agent-user communication and some of the formulas proposed in (Fleming 1998), are certainly flexible, and would depend strongly on the application area. The algorithm we present can be used as a starting point for any researcher wishing to develop a more interactive type of interface agent, because it identifies opportunities for both the agent and the user to initiate further communication. (Fleming 1998) also provides a generalized description of the algorithm, which does not rely on the underlying learning techniques from the MIT agent designs.

With a clear algorithm for interacting with users when there is ambiguity, making users aware of what an agent is contemplating and allowing users to take the initiative to direct an agent, we feel that we have provided a good framework for assistance in recommender system environments, where there is indeed some uncertainty and a clear role for users to make their preferences known. Because recommender systems attempt to make recommendations to their users, they can indeed be modelled as personal assistants. The tradeoff

[3] (Fleming 1998) discusses other methods for communicating with the user as well. For example, it is possible to use a separate "communication column" in the display of all e-mail messages in a mailbox, which records the current status of that message with respect to the agent's processing.

that has been identified for interface agents must be addressed for all recommender systems: how to request clarifying information from time to time, without subjecting the user to unnecessary interruptions.

Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

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.

Fleming, M. 1998. Designing more interactive interface agents. Master of mathematics thesis, University of Waterloo, Waterloo, Ontario.

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

Lieberman, H. 1995. Letizia: An agent that assists Web browsing. In *Proceedings of IJCAI '95.* AAAI Press.

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.

Stanfill, C., and Waltz, D. 1986. Toward memory-based reasoning. *Communications of the ACM* 29(12):1213–1228.