Polite Personal Agents

Silvia Schiaffino and Analía Amandi, *ISISTAN Research Institute, Universidad Nacional del Centro de la Provincia de Buenos Aires, and Comisión Nacional de Investigaciones Científicas y Técnicas, Argentina*

To succeed, agents must be aware of both the user's current context and the complexities of human behavior. In this design approach, agents exploit user profiles to provide personalized, context-aware assistance.

Personal agents are computer programs that learn users' interests, preferences, and habits and give them proactive, personalized assistance with a computer application. Such agents are analogous to personal assistants in the real-world work environment.[1] Both agents and assistants gradually learn a target individual's preferences and habits to enhance collaboration and productivity. Although personal agents have been used in various application domains,[1] problems have emerged with their use, including poor guessing about users' goals and needs, inadequate consideration of an action's costs and benefits, poorly timed actions, and a failure to optimize opportunities for users to better guide agent services and refine suboptimal results.[2]

Existing personal agents have concentrated on obtaining user preferences with respect to a computer application. However, as the sidebar, "Related Work on Human-Computer Interaction," briefly describes, they've yet to learn how to properly interact with users so as not to hinder their work. Just as people have unique interactions with their personal assistants, users' interaction with personal agents also varies. Users differ as to the kind of assistant they want and expect, the agent errors they will tolerate, and how they want an agent to provide assistance. Some users, for example, don't object to their personal assistant interrupting them with notifications and suggestions, so they probably won't object to a personal agent doing this. For other users, such behavior would be intolerable in either case. Personal agents who fail to meet user expectations are unlikely to last. For example, Microsoft product managers decided to "kill" Clippit, the Microsoft Office Assistant, because of widespread developer scorn (www.cnn.com/TECH/computing/9810/16/clipdeath.idg). Consequently, learning how to best assist and interact with users is a crucial issue in personal-agent development.

We developed a user-profiling approach that personalizes and enhances the interaction between a user and his or her personal agent. Our goal is to design agents that can provide context-aware assistance and make context-aware interruptions. To achieve this, we capture and model contextual information so the agent can learn how to behave and interact in different situations. To test our approach, we compared it with the more common confidence-based approach in the calendar management domain.

User-agent interaction issues

We empirically studied a set of user-agent interaction issues that might require personalization.[3] Through this study, we discovered several issues that agent developers must address to personalize and improve user-agent interactions:

- the particular assistance required for different contexts,
- when (and when not) to interrupt the user,
- the type of assistant the user wants,
- the user's tolerance for agent errors, and
- the amount of control the user will delegate to the agent.

Developers must also provide a way for users to give the agent simple, useful, and explicit feedback, and control and inspect agent behavior. The agents must also capture as much implicit feedback from users as possible.

Our research tackles the first two problems above: discovering what assistance a user requires in a given context and how to provide this assistance—that is, whether an agent should interrupt the user. To

1541-1672/06/$20.00 © 2006 IEEE
Published by the IEEE Computer Society

achieve this goal, agents must be able to observe and understand user behavior during their interactions. Given the complexities of human behavior, this is a difficult task. Furthermore, agents must be aware of the user's current context before offering assistance. Unfortunately, capturing and modeling context in user-agent interaction is also challenging.[4] Among the factors agents must detect and account for are the user's current task and its priority, and the user's schedule, interests, goals, preferences, habits, personal contacts, and emotional state. Finally, the agent must be able to detect when any of these factors change.

To study different aspects of user-agent interaction that might require personalization, we conducted a survey of 42 users (29 male and 13 female) ranging in age from 21 to 50. All participants had some agent experience, ranging from those people in our research group, who had developed and actively used personal agents in several application domains, to people who had interacted only with agents found in Microsoft Word or Excel.

We gave each participant a survey containing questions about user-agent interaction and user assistance. We asked them to answer the questions in as much detail as possible. Participants from our research group answered the questions in terms of their personal experience with the agents we developed. Users whose only experience was with Microsoft Office assistants answered the questions based both on their experience with the assistants and their expectations about an interface agent's behavior in those cases where they had no practical experience. (More detailed information about our experiments is available elsewhere.[3])

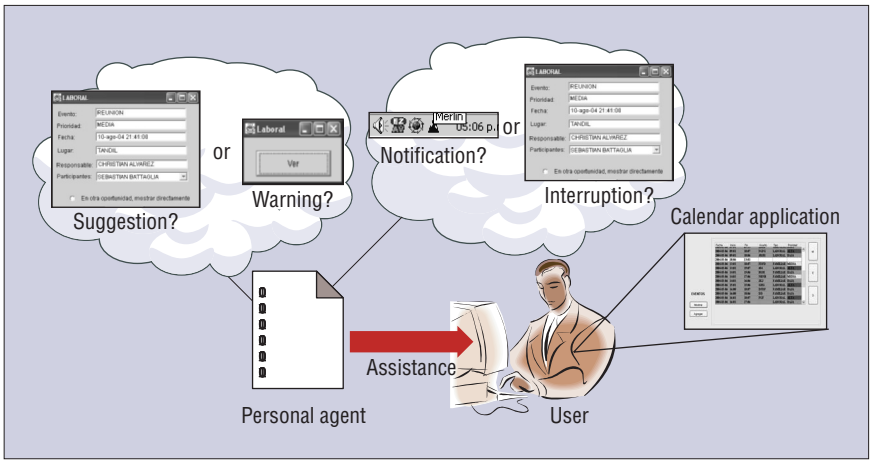

Figure 1. Agent assistance options in helping the user organize a meeting date. Different users will prefer different alternatives.

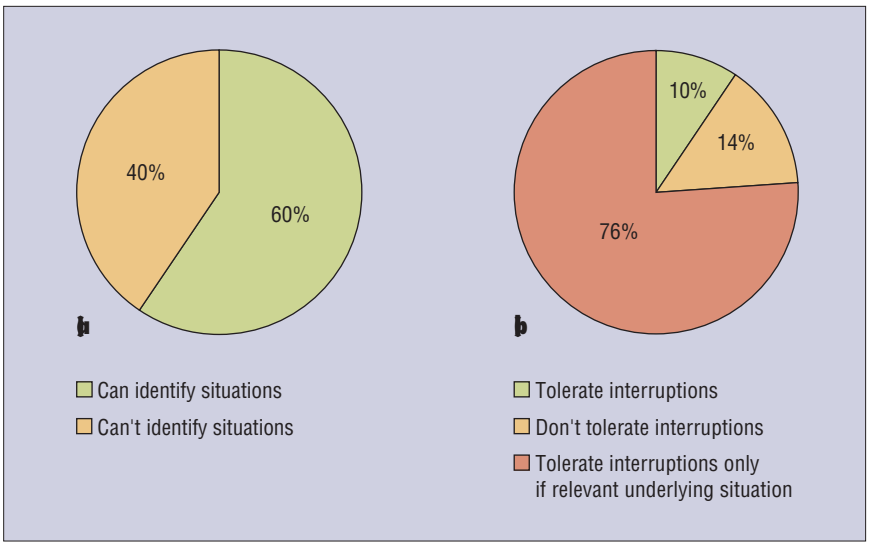

Figure 2. Results of a survey regarding users' (a) ability to identify situations where users need assistance and (b) tolerance of interruptions.

Context-specific assistance

To illustrate the issue of context, we'll consider an agent helping a user, John Smith, organize his calendar. Smith's current task is to schedule a meeting with several participants for the following Saturday in a free time slot. From past experience, the agent knows that one participant will disagree with the meeting date, because he never attends Saturday meetings. As figure 1 shows, the agent can

- warn the user about this problem,
- suggest another meeting date that considers all participant preferences and priorities, or
- do nothing.

In this situation, some users would prefer the simple warning, while others would want suggestions as to an alternative meeting date.

In our experiment, 60 percent of the 42 participants identified some situations in which they would clearly prefer one assistance action over another (see figure 2a). However, 40 percent of the participants couldn't identify such situations. With users in the first group, the agent must learn to provide different context-sensitive assistance based on different user preferences. Typically, the users who couldn't identify their preferences were inexperienced with personal agents. Most likely, inexperience—rather than an actual lack of preferences—prevented them from stating their preferences. So, the agent will have to discover the preferences.

Context and interruption tolerance

As figure 1 shows, when providing user assistance, agents can either interrupt the user's work or not. As some studies have shown, interruptions can be disruptive, both frustrating users and decreasing the efficiency with which they perform ongoing tasks.[5]

As Figure 2b shows, 10 percent of the participants didn't object to agent interruptions, while 14 percent wouldn't tolerate interruption under any circumstances. Most people (76 percent) didn't object to interruption if the underlying reason was both important and personally relevant. The big problem is

Related Work on Human-Computer Interaction

Our work relates to research in two primary areas:

- human-computer relationship etiquette,[1] which entails learning when to interrupt users and how best to assist them, and
- context-aware applications,[2] which use context to provide relevant information and services to users.

Our work is novel because we use context to personalize the interaction between users and personal agents.

Several algorithms exist to help agents discover whether to assist users in a particular situation and how to assist them (that is, which action to execute). These algorithms use different approaches to decide which action to execute—some use decision and utility theory;[3] others use confidence values attached to different actions.[4] However, they don't consider a user's assistance needs and interaction requirements, different types of assistance (warnings versus suggestions), and the particulars of a target situation.

Finally, interruptions have received considerable attention in human-computer interaction research (see www.interruptions.net/literature.htm for a bibliography), but research in personal agent development hasn't considered these studies.

References

1. C. Miller, "Human-Computer Etiquette: Managing Expectations with Intentional Agents," *Comm. ACM*, vol. 47, no. 4, 2004, pp. 31–34.
2. A.K. Dey, G.D. Adowd, and D. Salber, "A Conceptual Framework and Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," *Human-Computer Interaction*, vol. 16, nos. 2–4, 2001, pp. 97–166.
3. E. Horvitz, "Principles of Mixed-Initiative User Interfaces," *Proc. ACM Conf. Human Factors in Computing Systems* (CHI 99), ACM Press, 1999, pp. 159–166.
4. P. Maes, "Agents That Reduce Work and Information Overload," *Comm. ACM*, vol. 37, no. 7, 1994, pp. 31–40.

therefore to discover which situations are relevant or urgent for each user.

Discussion

As we expected, different users have different preferences with respect to our three assistance actions (warning, suggestion, and acting on the user's behalf). Even individual users prefer different actions depending on the problem and context. On the other hand, to avoid hindering a user's work, agents must discover whether the user objects to being interrupted or not, and in what context (if any). So, our main problem is to detect context-specific factors from a user's behavior and yield to different user preferences with respect to the agent's assistance actions and modalities.

Solution: Designing a user profile

To achieve our goals, personal agents must know users in the same way that personal assistants know them. Such knowledge typically resides in the user profile, which is a description containing the most important or interesting facts about the user. To personalize user-agent interactions, we need new user profile components that represent a user's assistance and interruption requirements. We therefore propose a definition for a user interaction profile that models a user's interaction preferences, requirements, and habits.

Personal agents can use this user interaction profile to personalize interactions with each user and adapt their behavior accordingly. Our profiling algorithm builds the user interaction profile using association rules based on the user's interaction with the agent, and the implicit and explicit feedback the agent obtains from this interaction.

User interaction profile

A user profile typically contains two types of user information. *Application-independent* information is mainly personal information about the user. *Application-dependent* information includes a user's interests, preferences, goals, working habits, behavioral patterns, knowledge, needs, priorities, and commitments regarding a particular domain.

However, such information isn't enough to personalize interaction with a user. In our approach, a user profile also needs information about user-agent interactions. We therefore add to the user profile information about the situations or contexts in which the user

- requires a suggestion to deal with a problem,
- needs only a warning about a problem,
- accepts an interruption from the agent,
- expects an action on his or her behalf, or
- wants a notification rather than an interruption.

So, we define a user interaction profile as a set of user-assistance requirements and a set of user-interruption preferences:

User Interaction Profile =
User Assistance Requirements
∪ *User Interruption Preferences*

We define the assistance requirements as a set of problem situations (or situations of interest) with the required assistance action and a parameter (certainty) indicating how sure the agent is about the user wanting that assistance action in that particular situation:

User Assistance Requirements =
{*User Assistance Requirement*}
User Assistance Requirement =
<*Situation*, *Agent Action*, *Certainty*>

We define the interruption preferences as a set of situations with the preferred assistance modality (interruption or no interruption). It might also contain the current user task, the notification's relevance to the current task, and the type of assistance action to execute. A parameter indicates how certain the agent is about this user preference:

User Interruption Preferences =
{*User Interruption Preference*}
User Interruption Preference =
<*Situation, [User Task],*
[Task Relevance],
[Assistance Action],
Assistance Modality,
Certainty>

Following are two examples of user interaction profile components in a calendar management application. The first indicates that, when two events overlap and one of them is a meeting organized by the user, the user requires a warning about the problem:

```
assist-req(situation(type(event overlapping),
features(event-one((event-type, meeting), (host,
```

user)), event-two, Monday 16th, 10am), action(warning), certainty(0.7)))

The second one expresses that, when an email arrives asking the user to attend a meeting organized by the user's boss, the user wants to be interrupted:

int-pref(situation(type(new email), features(sender(boss), topic(meeting))), task, relevance, action(warning), modality(interruption), certainty(0.8))

Profiling approach

Personal agents typically gather information by observing users and directly or indirectly soliciting information from them. We used these same methods to learn about a user's assistance requirements and interaction preferences.

Specifically, agents build user profiles by recording

- information about the actions a user requests from the agent,
- assistance actions the agent performs, and
- feedback the user provides on these assistance actions.

This feedback can be explicit or implicit, as well as positive or negative. Explicit feedback comes when the user directly evaluates the agent's actions through some user interface mechanism. The user can also offer explicit feedback by selecting interaction and interruption preferences. Implicit feedback is gathered when the agent observes the user's actions following an assistance action.

We record the information obtained from observation as a set of user-agent interaction experiences. We describe an interaction experience as Ex = <Sit, Act, Mod, Task, Rel, UF, E, Date>, where

- Sit is a situation that originates an interaction,
- Act is the assistance action the agent executed to deal with the situation (warning, suggestion, or action on the user's behalf),
- Mod is the modality that indicates whether the agent interrupted the user to provide the assistance,
- Task is the task the user was carrying out,
- Rel is the interaction situation's relevance to the user's task,
- UF is the user feedback obtained after assisting the user,
- E is an evaluation of the assistance experience (success, failure, or undefined), and
- Date indicates when the interaction took place.

Input: A set *Ex* of user–agent interaction experiences
Output: A set *F* of facts representing the user interaction profile
1: $AR \leftarrow$ Call association rule mining algorithm (Apriori) with *Ex*, *minsup*, and *minconf*
2: $AR_1 \leftarrow$ Filter out uninteresting rules from AR
3: $AR_2 \leftarrow$ Eliminate redundant and insignificant rules from AR_1
4: $AR_3 \leftarrow$ Eliminate contradictory rules from AR_2
5: $H \leftarrow$ Transform rules in AR_3 into hypotheses
6: **for** $i = 1$ to size of H **do**
7: Find evidence for (E^+) and against $(E^-)H_i$
8: $Cer(H_i) \leftarrow$ compute certainty degree of H_i considering E^+ and E^-
9: **if** $Cer(H_i) \geq \delta$ **then**
10: $F \leftarrow F \cup H_i$
11: **endif**
12: **endfor**

Figure 3. Algorithm 1 uses association rules to create the user interaction profile.

Say, for example, that our agent is assisting Smith with scheduling a new event: a meeting with his employees (Johnson, Taylor, and Dean) to discuss project A's evolution. The event is being scheduled for Friday at 5 p.m. in Smith's office. By observing Smith's previous actions and schedules, the agent has learned that Dean never schedules Friday evening meetings and therefore will likely disagree about the meeting date and time. The agent thus decides to warn Smith about the problem. In reply to this warning, Smith asks the agent to suggest another date for the event.

In this example, the assistance experience comprises these parts:

- Sit = {(type, new-event), (event-type, business meeting), (organizer, user), (participants, [Johnson,Taylor,Dean]), (topic, project A evolution), (date, Friday), (time, 5 p.m.), (place, user's office)}
- Act = {(type, warning), (message, Dean does not like meetings on Friday evenings)}
- Mod = {(type, notification)}
- Task = {(type, new-event)}
- Rel = relevant
- UF = {(type, explicit), (action, asks for a suggestion)}
- E = {(type, failure), (certainty, 1.00)} (suggestion instead of warning)
- Date = 18/02/04

Our profiling approach takes as input the set of user-agent interaction experiences and from these learns when the user requires a suggestion or warning about a situation and when the user wants the agent to perform a task on his or her behalf. This process also determines when the agent can interrupt the user's work to provide assistance and when it can send notifications without interrupting the user.

The algorithm's outputs constitute the user interaction profile. The decision-making algorithm uses user profile information to decide which assistance action to execute and how best to execute it to assist the user. As with previous personal agents, our agent obtains its assistance action's content from the user's preferences and interests.

Building a profile

We use the WATSON (*W*arning, *Act*ion, *S*uggestion, *or N*othing) algorithm to build a user interaction profile from the set of user-agent interaction experiences. We use association rules as our machine learning technique; figure 3 shows the main steps. Association rules imply a relationship among a set of items in a given domain.[6] We use association rules to discover the existing relationships between problem situations or situations of interest and the assistance actions a user requires to deal with them, as well as the relationships between situations, the primary user task, and the assistance modality required.

Association rule mining is commonly stated as, Let $I = i_1, \ldots, i_n$ be a set of items and D be a set of transactions, each consisting of a subset X of items in I.[6] An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$. X is the rule's antecedent and Y is the consequent. The rule has support s in D if s percent of D's transactions contains $X \cap Y$. The rule $X \rightarrow Y$ holds in D with confidence c if c percent of D's transactions that contain X also contain Y. Given a

```
1. modality=notification --> primarytask=newevent,primarytaskrelevance=relevant sup(1) conf(1)
2. primarytask=newevent --> modality=notification,primarytaskrelevance=relevant sup(1) conf(1)
3. primarytaskrelevance=relevant --> modality=notification,primarytask=newevent sup(1) conf(1)
4. modality=notification,primarytask=newevent --> primarytaskrelevance=relevant sup(1) conf(1)
5. modality=notification,primarytaskrelevance=relevant --> primarytask=newevent sup(1) conf(1)
6. primarytask=newevent,primarytaskrelevance=relevant --> modality=notification sup(1) conf(1)
7. primarytask=newevent --> primarytaskrelevance=relevant sup(1) conf(1)
8. primarytaskrelevance=relevant --> primarytask=newevent sup(1) conf(1)
9. modality=notification --> primarytaskrelevance=relevant sup(1) conf(1)
10. primarytaskrelevance=relevant --> modality=notification sup(1) conf(1)
11. modality=notification --> primarytask=newevent sup(1) conf(1)
12. primarytask=newevent --> modality=notification sup(1) conf(1)
13. agentaction=warning --> modality=notification,primarytask=newevent,primarytaskrelevance=relevant sup(0.9) conf(1)
14. agentaction=warning,modality=notification --> primarytask=newevent,primarytaskrelevance=relevant sup(0.9) conf(1)
15. agentaction=warning,primarytask=newevent --> modality=notification,primarytaskrelevance=relevant sup(0.9) conf(1)
16. agentaction=warning,primarytaskrelevance=relevant --> modality=notification,primarytask=newevent sup(0.9) conf(1)
17. agentaction=warning,modality=notification,primarytask=newevent --> primarytaskrelevance=relevant sup(0.9) conf(1)
18. agentaction=warning,modality=notification,primarytaskrelevance=relevant --> primarytask=newevent sup(0.9) conf(1)
19. agentaction=warning,primarytask=newevent,primarytaskrelevance=relevant --> modality=notification sup(0.9) conf(1)
20. agentaction=warning --> primarytask=newevent,primarytaskrelevance=relevant sup(0.9) conf(1)
21. agentaction=warning,primarytask=newevent --> primarytaskrelevance=relevant sup(0.9) conf(1)
22. agentaction=warning,primarytaskrelevance=relevant --> primarytask=newevent sup(0.9) conf(1)
23. agentaction=warning --> modality=notification,primarytaskrelevance=relevant sup(0.9) conf(1)
24. agentaction=warning,modality=notification --> primarytaskrelevance=relevant sup(0.9) conf(1)
25. agentaction=warning,primarytaskrelevance=relevant --> modality=notification sup(0.9) conf(1)
26. agentaction=warning --> modality=notification,primarytask=newevent sup(0.9) conf(1)
27. agentaction=warning,modality=notification --> primarytask=newevent sup(0.9) conf(1)
28. agentaction=warning,primarytask=newevent --> modality=notification sup(0.9) conf(1)
29. agentaction=warning --> primarytaskrelevance=relevant sup(0.9) conf(1)
30. agentaction=warning --> primarytask=newevent sup(0.9) conf(1)
31. agentaction=warning --> modality=notification sup(0.9) conf(1)
```

Figure 4. The algorithm generated this subset of association rules using the data set of user-agent interactions. The minimum support (minsup) value is 0.1 and the minimum confidence (minconf) value is 0.8.

transaction database D, the problem of mining association rules is to find all association rules that satisfy minimum support (**minsup**) and minimum confidence (**minconf**).

We use the Apriori algorithm[6] to generate association rules from a set of user-agent interaction experiences. We automatically postprocess the rules Apriori generates so that we can derive useful hypotheses from them. Postprocessing includes detecting the most interesting rules according to our goals, eliminating redundant and insignificant rules, eliminating contradictory weak rules, and summarizing the information to formulate the hypotheses more easily.

To filter rules, we use templates to express and select relevant rules. For example, we are interested in those association rules of the forms **situation, assistance action → user feedback, evaluation** and **situation, modality, [user task], [relevance], [assistance action] → user feedback, evaluation** (where brackets mean that the attributes are optional). To eliminate redundant rules, we use a subset of the pruning rules that Devavrat Shah and his colleagues proposed.[7] We define a contradictory rule as one indicating a different assistance action (modality) for the same situation and having a small confidence value with respect to the rule being compared.

After pruning, we group rules by similarity and generate a hypothesis that considers a main rule, positive evidence (redundant rules that couldn't be eliminated), and negative evidence (contradictory rules that couldn't be eliminated). Once we have a hypothesis, the algorithm computes its certainty degree by taking into account the main rule's support values and the positive and negative evidence. To compute certainty degrees, we use this function:

$$Cer(H) = \alpha Sup(AR) + \beta \frac{\sum_{k=1}^{r} Sup(E^+)}{\sum_{k=1}^{r+t} Sup(E)} - \gamma \frac{\sum_{k=1}^{t} Sup(E^-)}{\sum_{k=1}^{r+t} Sup(E)} \quad (1)$$

where α, β, and γ are the weights of the equation's terms, $Sup(AR)$ is the main rule support, $Sup(E^+)$ is the positive evidence support, $Sup(E^-)$ is the negative evidence support, $Sup(E)$ is the support of a rule taken as evidence (positive or negative), r is the amount of positive evidence, and t is the amount of negative evidence. Finally, we generate facts from the set of highly supported hypotheses (those with certainty greater than δ).

We set $\alpha = 0.7$, $\beta = 0.15$, and $\gamma = 0.15$ because we consider the positive and negative evidence to be equally important but consider the main rule's support to be more important in the certainty calculus. We experimentally set δ to 0.2. Finally, our algorithm is incremental—we generate a new profile when a certain amount of new interaction experiences have accumulated. To this end, we use the FUP2 (Fast Update 2) incremental association-rule-mining algorithm and, to determine when to update, the DELI (Difference Estimations for Large Item sets) algorithm.[8]

An example profile build

As an example of profile building, let's assume Smith has the following interaction preferences. First, if overlap exists between a meeting Smith organizes and an event organized by his boss or the professor in charge of object-oriented programming, Smith wants the agent to interrupt him. In other cases of scheduling conflicts, Smith wants the agent to notify him without interrupting.

Our algorithm first generates association rules from the user-agent interaction experiences. Figure 4 shows a subset of these association rules, which the algorithm generated from the data set of interactions between Smith and his agent. We set **minsup** at 0.1 and **minconf** at 0.8.

From all the association rules generated, the algorithm filters out the irrelevant rules (using the templates we described earlier). The following rules survive (the number "1" refers to the first event):

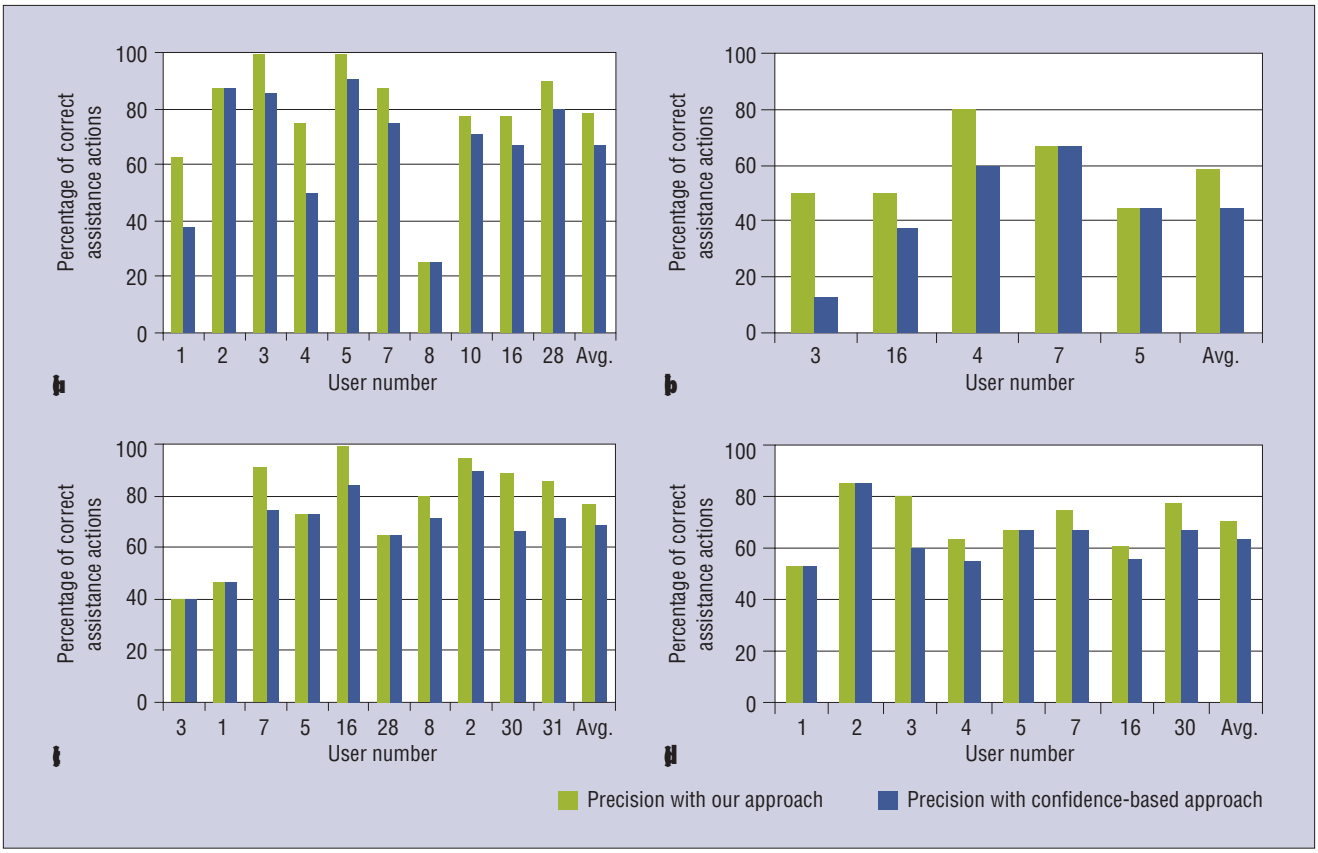

Figure 5. Our approach's precision versus that of the confidence-based approach for the (a) holiday, (b) new-event, (c) overlapping, and (d) time data sets.

1. event type (1) = meeting, place (1) = office, agent action = warning, modality = notification, primary task = new event, primary task relevance = relevant → user reaction = interrupt next time, evaluation = failure; sup(0.6), conf(1)
2. place (1) = office, agent action = warning, modality = notification, primary task = new event, primary task relevance = relevant, user reaction = interrupt next time → evaluation = failure; sup(0.6), conf(1)
3. event type (1) = meeting, agent action = warning, modality = notification, primary task = new event, primary task relevance = relevant → user reaction = interrupt next time, evaluation = failure; sup(0.6), conf(1)
4. event type (1) = meeting, place (1) = office, modality = notification, primary task = new event, primary task relevance = relevant → user reaction = interrupt next time, evaluation = failure; sup(0.6), conf(1)
5. event type (1) = meeting, place (1) = office, agent action = warning, modality = notification, primary task relevance = relevant → user reaction = interrupt next time, evaluation = failure; sup(0.6), conf(1)
6. event type (1) = meeting, place (1) = office, agent action = warning, modality = notification, primary task = new event → user reaction = interrupt next time, evaluation = failure; sup(0.6), conf(1)

In the next step, the algorithm eliminates redundant rules.[7] In this example, all the rules survive this step. The algorithm then searches for contradictory rules, and again finds none. Given this, the algorithm considers all the rules to build the hypotheses. For example, one of our hypotheses has these components:

Main Rule: event type (1) = meeting, place (1) = office, agent action = warning, modality = notification, primary task = new event → user reaction = interrupt next time, evaluation = failure; sup(0.6), conf(1).
Positive Evidence: event type (1) = meeting, place (1) = office, agent action = warning, modality = notification, primary task = new event, primary task relevance = relevant → user reaction = interrupt next time, evaluation = failure; sup(0.6), conf(1).
Negative Evidence: none.
Certainty: 0.57

We compute the certainty degree of this hypothesis as

$$Cer(H) = 0.7 * 0.6 + 0.15 * \frac{0.60}{0.60} - 0.15 * 0 = 0.42 + 0.15 = 0.57$$

After we summarize the information in this hypothesis, the corresponding user interruption preference is

situation: event type = meeting, place = office, agent action = warning; primary task = new event; modality of assistance: interruption

This interruption preference directly reflects one of Smith's stated preferences. As for decision making, if the target situation is the one in the user profile, the agent will interrupt Smith to provide assistance. If the target situation isn't in the user profile, the agent will choose the assistance action on the basis of its confidence value (as with most standard personal agents).

The Authors

Silvia Schiaffino is a professor in the Computer Science Department at the Universidad Nacional del Centro de la Provincia de Buenos Aires and a research assistant at the university's ISISTAN Research Institute. Her main research interests are intelligent agents, personalization, and human-computer interaction. She received her PhD in computer science from UNCPBA. Contact her at ISISTAN Research Inst., Facultad de Ciencias Exactas, Univ. Nacional del Centro de la Provincia de Buenos Aires, Campus Universitario—Paraje Arroyo Seco, Tandil, 7000, Buenos Aires, Argentina; sschia@exa.unicen.edu.ar.

Analía Amandi is a professor in the Computer Science Department at Universidad Nacional del Centro de la Provincia de Buenos Aires, where she leads the ISISTAN Research Institute's Intelligent Agents Group. Her research interests include intelligent agents and knowledge management. She received her PhD in computer science from the Universidad Federal do Rio Grande do Sul. Contact her at the ISISTAN Research Inst., Facultad de Ciencias Exactas, Univ. Nacional del Centro de la Provincia de Buenos Aires, Campus Universitario-Paraje Arroyo Seco, Tandil, 7000, Buenos Aires, Argentina; amandi@exa.unicen.edu.ar.

Experimental results

To test our approach against standard personal-agent approaches, we compared our algorithm's user assistance precision with that of a confidence-based decision-making algorithm.[1] To do this, we used a precision metric that measures a personal agent's ability to accurately assist a user:[9]

$$Precision = \frac{number\ of\ correct\ assistance\ actions}{number\ of\ assistance\ actions} \quad (2)$$

We used this metric to evaluate the agent's performance in deciding between

- a warning, a suggestion, or an action on the user's behalf; and
- an interruption or a notification.

For each problem situation, we compared the number of correct assistance actions against the total number of assistance actions the agent executed. (We considered an assistance action correct if it was the action the user expected in a given situation.)

To carry out the experiments, we used 39 data sets containing user-agent interaction experiences in the calendar management domain (the data sets are available at www.exa.unicen.edu.ar/~sschia). Each database record contains attributes that describe the problem situation (or the situation originating the interaction), the assistance action the agent executed, the user feedback, and the user's evaluation of the interaction experience. We built the data sets according to real user profiles—that is, using a set of interaction preferences established by real users. We obtained the profiles by interviewing 12 of the 42 users who participated in our experiments. The data sets contained anywhere from 30 to 125 user-agent interaction experiences.

We analyzed four calendar-management situations:

- *New event*: The user is scheduling a new event, and the agent has information about the event's potential time, place, or duration.
- *Overlapping*: The user is scheduling an event that overlaps with a previously scheduled event.
- *Time*: Not enough time exists to travel between the proposed event and the event scheduled to follow it.
- *Holiday*: The user is scheduling a business (or work-related) event for a holiday.

The attributes describing the overlapping and time data sets are event type, host, topic, participants, and place (for both events); agent action; assistance modality; user task; task relevance; user feedback; and evaluation. The attributes involved in the new-event and holiday data sets describe an event—namely, event type, topic, participants, host, and place.

Figure 5 shows the two algorithms' precision for different data sets and users. Each bar graph plots the percentage of correct assistance actions with respect to the total number of agent assistances. To determine an assist's correctness and modality, we consulted the profile description containing a user's interruption preferences and assistance requirements. The last pair of bars in each graph plots the average precision for different users on a given data set. For example, the second pair of bars in figure 5b indicates that for user 16, our approach correctly assisted the user 50 percent of the time, while the confidence-based approach was correct 37.5 percent of the time. (We used percentage values because the number of user-agent interactions varied from one user to another.)

On average, as the figure shows, our approach had a higher overall percentage of correct assistance actions, assistance interactions, or interactions (70 percent) than the confidence-based algorithm (60 percent). A 10 percent increase in precision is an important achievement. Enhancing an agent's ability to learn what users want is even more important. The difference between the two algorithms is clear when, for example, the user requires only warning about a frequent problem. In such cases, the confidence-based algorithm proposes a suggestion or an action. Our algorithm issues only a warning, even though the confidence on the assistance content is high. When the problem is infrequent and warnings are required, the two algorithms behave similarly.

Our work makes two main contributions to the areas of intelligent agents and human-computer interaction. First, it increases understanding of personalization in user-agent interaction. Second, our approach enhances a personal agent's actual capabilities. Agent developers can use our study's results to build interface agents that can adapt to users' expectations and preferences regarding user-agent interaction. Our approach enables agents to personalize their user interactions by learning the type of assistance users need in different contexts and by learning how to provide this assistance without annoying users.

Building on our promising results in the calendar management task, we're carrying out further experiments in this and other domains. In our future work, we plan to address the other user-agent interaction issues that require personalization. Also, we currently consider only a subset of factors to characterize the user context. In the future, we plan to enrich the contextual information the agent considers when assisting the user. ■

References

1. P. Maes, "Agents That Reduce Work and Information Overload," *Comm. ACM*, vol. 37, no. 7, 1994, pp. 31–40.
2. E. Horvitz, "Principles of Mixed-Initiative User Interfaces," *Proc. ACM Conf. Human Factors in Computing Systems* (CHI 99), ACM Press, 1999, pp. 159–166.
3. S. Schiaffino and A. Amandi, "User-Interface Agent Interaction: Personalization Issues," *Int'l J. Human-Computer Studies*, vol. 60, no. 1, 2004, pp. 129–148.
4. A.K. Dey, G.D. Adowd, and D. Salber, "A Conceptual Framework and Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," *Human-Computer Interaction*, vol. 16, nos. 2–4, 2001, pp. 97–166.
5. D.C. McFarlane and K.A. Latorella, "The Scope and Importance of Human Interruption in Human-Computer Interaction Design," *Human-Computer Interaction*, vol. 17, no. 1, 2002, pp. 1–61.
6. R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. 20th Int'l Conf. Very Large Data Bases* (VLDB 94), Morgan Kaufman, 1994, pp. 487–499.
7. D. Shah et al., "Interestingness and Pruning of Mined Patterns," *Proc. Workshop Research Issues in Data Mining and Knowledge Discovery*, ACM Press, 1999.
8. S.D. Lee and D. Cheung, "Maintenance of Discovered Association Rules: When to Update?" *Proc. Workshop Research Issues in Data Mining and Knowledge Discovery*, ACM Press, 1997.
9. S. Brown and E. Santos, "Using Explicit Requirements and Metrics for Interface Agent User Model Correction," *Proc. 2nd Int'l Conf. Autonomous Agents*, ACM Press, 1998, pp. 1–7.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.