The *IONWI* Algorithm: Learning when and when not to interrupt

Silvia Schiaffino and Analía Amandi
ISISTAN Research Institute – Facultad de Cs. Exactas – UNCPBA –
Campus Universitario, Paraje Arroyo Seco, Tandil, 7000, Bs As, Argentina
Also CONICET, Consejo Nacional de Investigaciones Científicas y
Técnicas, Argentina
{sschia,amandi}@exa.unicen.edu.ar

Abstract. One of the key issues for an interface agent to succeed at assisting a user is learning when and when not to interrupt him to provide him assistance. Unwanted or irrelevant interruptions hinder the user's work and make him dislike the agent because it is being intrusive and impolite. The *IONWI* algorithm enables interface agents to learn a user's preferences and priorities regarding interruptions. The resulting user profile is then used by the agent to personalize the modality of the assistance, that is, assisting the user with an interruption or without an interruption depending on the user's context. Experiments were conducted in the calendar management domain, obtaining promising results.

Keywords: intelligent agents, user profiling, human-computer interaction

1. Introduction

As intelligent agents take on more complexity, higher degrees of autonomy and more "intelligence", users start to expect them to play by the same rules of other complex, autonomous and intelligent entities in their experience, namely, other humans [16]. Our previous studies [19] demonstrated that the way in which an interface agent assists a user has an impact on the competence of this agent and it can make the interaction between user and agent a success or a failure. This is the

Please use the following format when citing this chapter:

Schiaffino, S., Amandi, A., 2006, in IFIP International Federation for Information Processing, Volume 217, Artificial Intelligence in Theory and Practice, ed. M. Bramer, (Boston: Springer), pp. 21–30.

concern of a recent research area within Human-Computer Interaction (HCI) that studies the "etiquette" of human-computer relationships [3; 17; 18]. We agree with the researchers in this area on that the ability to adapt to the way in which a user wants to interact with the agent is almost as important as the ability to learn the user's preferences in a particular domain.

As pointed out in [14], one of the problems with the interface agents developed thus far is their incorrect estimates of the user's task priorities, which makes information to be introduced at inappropriate times and with unsuitable presentation choices. Although agents are well-intentioned, they do not consider the impact an interruption has on the user. Research has found that interruptions are harmful. They are disruptive to the primary computing task and they decrease users' performance. However, interruptions are necessary in interface agent technology since agents need to communicate important and urgent information to users.

To solve this problem, when the agent detects a (problem) situation relevant to the user it has to correctly decide if it will send him a notification without interrupting the user's work, or if it will interrupt him. On the one hand, the user can choose between paying attention to a notification or not, and he can continue to work in the latter case. On the other hand, he is forced to pay attention to what the agent wants to tell him if it interrupts him abruptly.

Not to disturb the user, the agent has to base its decision on various factors, such as: the relevance and the urgency the situation has for the user; the relationship between the situation to be notified or the assistance to be provided and the user's goals; the relevance the situation underlying the interruption has to the current user tasks; how tolerant the user is of interruptions; and when he does not want to be interrupted no matter how important the message is. In summary, the interface agent has to learn which situations are relevant and which are irrelevant so that no unwanted interruptions occur.

In this work we present a user profiling algorithm named *IONWI* that learns when a user can or should be interrupted by his agent depending on the user's context. In this way, the agent can provide personalized assistance to the user without hindering his work.

This article is organized as follows. Section 2 presents our proposed profiling algorithm. Section 3 shows the results we have obtained when assisting users of a calendar management system. Section 4 describes some related works. Finally, Section 5 presents our conclusions and future work.

2. The *IONWI* Algorithm

In order to assist a user without hindering his work, an interface agent has to learn the user's interruption needs and preferences in different contexts. In this work we propose an algorithm, named *IONWI* (acronym for Interruption Or Notification Without Interruption), capable of learning when to interrupt a user and when not from the observation of the user's interaction with a computer application and with the agent.

The algorithm learns when a situation that may originate an interruption is relevant to the user's needs, preferences and goals, and when it is irrelevant. In addition, this algorithm also considers the relationship and relevance the situation originating the interaction has with the user's current task.

2.1 Algorithm inputs and outputs

The input for our learning algorithm is a set of user-agent interaction experiences. An interaction experience *Ex* is described by seven arguments <*Sit, Mod, Task, Rel, UF, E, date*>: a problem situation or situation of interest *Sit* is described by a set of features and the values these features take, $Sit=\{(feature_i, value_i)\}$; the modality *Mod* that indicates whether the agent interrupted the user or not to provide him assistance; the *Task* the user was executing when he was interrupted or notified, which is described by a set of features and the values these features take $Task=\{(feature_i, value_i)\}$; the relevance *Rel* the interruption has for the *Task*; the user feedback *UF* (regarding the assistance modality) obtained after assisting the user; an evaluation *E* of the assistance experience (success, failure or undefined); and the *date* when the interaction experience was recorded.

For example, consider that the user is scheduling a meeting with several participants and he is interrupted by his agent to remind him about a business meeting that will take place the next day. The user does not pay attention to the message being notified and presses a button to tell the agent not to interrupt him in these occasions. From this experience the agent learns that reminders of this kind of meetings are not relevant to the user, and it will send him a notification in the future without interrupting him. In this example, the different components of the assistance experience are:

Sit ={(type, event reminder), (event-type, business meeting), (organizer, boss), (participants, [Johnson, Taylor ,Dean]), (topic, project A evolution), (date, Friday), (time, 5p.m.), (place, user's office)}

Mod = interruption

Task = {(application, calendar management system),(task, new event), (event type, meeting), (priority, high), ...}

Rel = irrelevant, unrelated

UF = {(type, explicit), (action, do not interrupt)}

E = {(type, failure), (certainty, 1.00)} (interruption instead of notification)

Date = {(day, 18), (month, December), (year, 2005)}

The output of our algorithm is a set of facts representing the user's interruptions preferences. Each fact indicates whether the user needs an interruption or a notification when a given situation occurs in the system. Facts constitute part of the user profile. These facts may adopt one of the following forms: "in problem situation *Sit* the user should be interrupted", "in situation *Sit* the user should not be interrupted", "in situation *Sit* and if the user is performing the task *T*, he should not be interrupted", "in situation *Sit* and if the user is performing the task *T*, the agent can interrupt him". Each fact *F* is accompanied by a certainty degree *Cer(F)* that indicates how certain the agent is about this preference. Thus, when an interface agent has to decide whether to interrupt the user or not given a certain problem

situation, the agent uses the knowledge it has acquired about a user's interruption preferences to choose the assistance modality it supposes the user expects in that particular instance of a given situation. Once the assistance has been provided, the agent obtains explicit and/or implicit user feedback. This new interaction is recorded as an assistance experience, which will be used in the future to incrementally update the knowledge the agent has about the user.

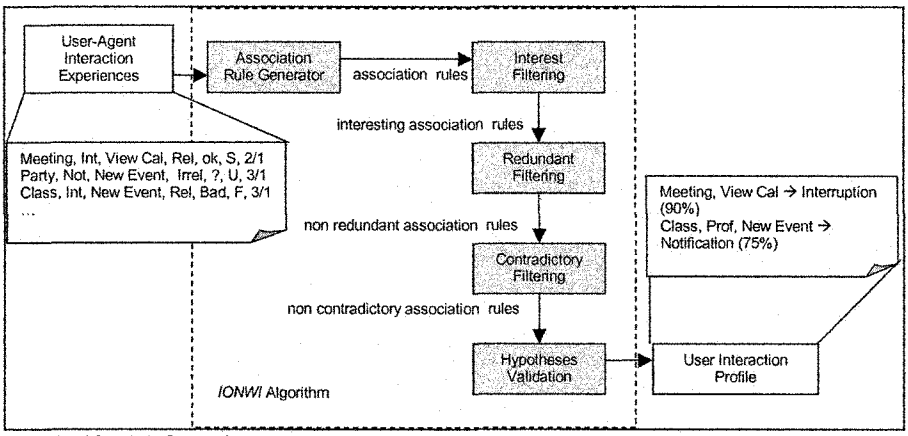

Fig. 1. ***IONWI*** Overview

2.2 *IONWI* Overview

The *IONWI* algorithm uses association rules to obtain the existing relationships between situations, current user tasks and assistance modalities. Classification techniques have been discarded since we cannot always label an interaction as a success or a failure, and we need a group of interactions to draw a conclusion about the user's preferences.

As shown in Figure 1, the first step of our algorithm is generating a set of association rules from the user-agent interaction experiences. Then, the association rules generated are automatically post-processed in order to derive the user profile from them. Post-processing steps include: detecting the most interesting rules according to our goals, eliminating redundant and insignificant rules, pruning out contradictory weak rules, and summarizing the information in order to formulate the hypotheses about a user's preferences more easily. Once a hypothesis is formulated, the algorithm looks for positive evidence supporting the hypothesis and negative evidence rejecting it in order to validate it. The certainty degree of the hypothesis is computed taking into account both the positive and the negative evidence. This calculus is done by using metrics from association rule discovery. Finally, facts are generated from the set of highly supported hypotheses; facts compose the user interaction profile.

The following subsections describe in detail each step of the algorithm.

2.3 Mining Association Rules from User-Agent Interaction Experiences

An association rule is a rule that implies certain association relationship among a set of objects in a database, such as occur together or one implies the other. Association discovery finds rules about items that appear together in an event (called transactions), such as a purchase transaction or a user-agent interaction experience. Association rule mining is commonly stated as follows [1]: Let $I=\{i_1,...,i_n\}$ be a set of items, and D be a set of data cases. Each data case consists of a subset of items in I. 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 antecedent of the rule and Y is the consequent. The support of a rule $X \rightarrow Y$ is the probability of attribute sets X and Y occurring together in the same transaction. The rule has support s in D if s% of the data case in D contains $X \cap Y$. If there are n total transactions in the database, and X and Y occur together in m of them, then the support of the rule $X \rightarrow Y$ is m/n. The rule $X \rightarrow Y$ holds in D with confidence c if c% of data cases in D that contain X also contain Y. The confidence of rule $X \rightarrow Y$ is defined as the probability of occurrence of X and Y together in all transactions in which X already occurs. If there are s transactions in which X occurs, and in exactly t of them X and Y occur together, the confidence of the rule is t/s.

Given a transaction database D, the problem of mining association rules is to find all association rules that satisfy: minimum support (called *minsup*) and minimum confidence (called *minconf*). There has been a lot of research in the area of association rules and, as a result, there are various algorithms to discover association rules in a database. The most popular is the Apriori algorithm [1], which is the one we use to find our association rules.

2.4 Filtering Out Uninteresting and Redundant Rules

In this work, we are interested in those association rules of the form "situation, modality, task → user feedback, evaluation"; "situation, modality → user feedback, evaluation"; "situation, modality, relevance → user feedback, evaluation" and "situation, modality, task, relevance → user feedback, evaluation", having appropriate support and confidence values. We are interested in these rules since they provide us information about the relationships between a situation or problem description and the modality of assistance the user prefers, which have received a positive (negative) evaluation. They also relate a situation and the current user task with an assistance modality, as well as a situation, the current user task and the relevance of the situation to the task with a certain assistance modality. To select these types of rules, we define templates [10] and we insert these templates as restrictions in the association mining algorithm. Thus, only interesting rules are generated (steps 1 and 2 in Figure 1 are then merged).

Once we have filtered out those rules that are not interesting for us, we will still have many rules to process, some of them redundant or insignificant. Many discovered associations are redundant or minor variations of others. Thus, those spurious and insignificant rules should be removed. We can then use a technique that removes those redundant and insignificant associations [13]. For example, consider the following rules:

R1: Sit{(Type, Event Reminder)(Event-Type = doctor))} (Task=View Calendar), (Mod =interruption) → (UF = do not interrupt), (Ev = failure) [sup: 0.4, conf: 0.82]

R2: Sit{(Type, Event Reminder)(Event-Type = doctor)}, (Task=View Calendar), (Event-Priority = high)), (Mod =interruption) → (UF= do not interrupt), (Ev = failure) [sup:0.4, conf: 0.77]

If we know R1, then R2 is insignificant because it gives little extra information. Its slightly higher confidence is more likely due to chance than to true correlation. It thus should be pruned. R1 is more general and simple.

In addition, we have to analyze certain combinations of attributes in order to determine if two rules are telling us the same thing. For example, a rule containing the pair "interruption, failure" and another containing the pair "notification, success" are redundant provided that they refer to the same problem situation and they have similar confidence values. As well as analyzing redundant rules, we have to check if there are any contradictory rules. We define that two rules are contradictory if for the same situation and, eventually for the same user task, they express that the user wants both an interruption and a notification without interruption.

2.5 Building Facts from Hypotheses

The association rules that have survived the pruning processes described above are those the *IONWI* algorithm uses to build hypotheses about a user's interruption preferences. A hypothesis is obtained from a set of association rules that are related because they refer to the same problem situation but are somewhat different: a "main" association rule; some redundant association rules with regards to the main rule, which could not be pruned out because they did not fulfill the similar confidence restriction; and some contradictory rules with regards to the main rule, which could be not pruned away because they did not meet the different confidence requirement. The main rule is chosen by selecting from the rule set the rule that has the greatest support value, whose antecedent is the most general, and whose consequent is the most specific.

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

Equation 1

Once the *IONWI* algorithm has formulated a set of hypotheses it has to validate them. The certainty degree of a hypothesis H is computed as a function of the supports of the rule originating the hypothesis and the rules considered as positive and negative evidence of H. The function we use to compute certainty degrees is shown in Equation 1, where: α, β and γ are the weights of the terms in the equation (we use α=0.8, β=0.1 and γ=0.1), *Sup(AR)* is the support of the rule originating H, $Sup(E^+)$ is the support of the rules being positive evidence, $Sup(E^-)$ is the support of the rules being negative evidence, *Sup(E)* is the support value of an association rule

taken as evidence (positive or negative), r is the amount of positive evidence and t is the amount of negative evidence.

2.6 Incremental Learning

The database containing interaction experiences is not static, because updates are constantly being applied to it. On the one hand, new interaction experiences are added since the agent keeps observing a user's behaviour. On the other hand, old experiences are deleted because they become obsolete. In consequence, new hypotheses about a user's interruption preferences may appear and some of the learned hypotheses may become invalid.

We address this problem from the association rule point of view, that is, as the database changes new association rules may appear and at the same time, some existing association rules may become invalid. The incremental version of *IONWI* uses the FUP2 algorithm [5] to update the association rules and the DELI algorithm [12] to determine when it is necessary to update the rules. The DELI algorithm uses a sampling technique to estimate the difference between the old and new association rules. This estimate is used as an indicator for whether the FUP2 algorithm should be applied to the database to accurately find out the new association rules. If the estimated difference is large enough (with respect to some user specified threshold), the algorithm signals the need of an update operation, which can be accomplished by using the FUP2 algorithm. If the estimated difference is small, then we do not run FUP2 immediately and we can take the old rules as an approximation of the new rules. Hence, we wait until more changes are made to the database and then re-apply the DELI algorithm.

3. Experimental Results

We tested our algorithm with a set of 26 datasets[1] containing user-agent interactions in the calendar management domain. Each database is composed of the attributes that describe the problem situation or situation of interest originating the interaction, the primary user task, the modality of the assistance, the relationship between the situation and the user task, the user feedback, and the evaluation of the interaction experience. The sizes of the datasets vary from 30 to 120 interactions.

To evaluate the performance of an agent using our learning algorithm we used one of the metrics defined in [4]. The precision metric measures an interface agent's ability to accurately provide assistance to a user. As shown in Equation 2, we can define our precision metric as the ratio of the number of correct interruption preferences to the total number of interruption preferences generated by *IONWI*. Similarly, as shown in Equation 3, we can define the recall metric (i.e. what the agent could not learn) as the ratio of the number of correct interruption preferences to the number of preferences indicated by the user.

[1] The datasets can be found at http://www.exa.unicen.edu.ar/~sschia

Figure 2 presents the results we have obtained. The graph in Figure 2(a) plots the percentage of interruption preferences correctly identified by the *IONWI* algorithm (with respect to the total number of preferences obtained); the number of incorrect interruption preferences; and the number of "hidden" preferences, that is those preferences that were not explicitly stated by the user but are correct. Each figure shows the percentage values obtained when averaging the results we got with the different users. The graph in Figure 2(b) shows the percentage of correct interruption preferences (with respect to the number of preferences specified by the user) and the percentage of missing interruption preferences, that is those that the algorithm could not detect. Each graphic shows the average percentage values of the results obtained with the different datasets.

$$IONWI_{precision} = \frac{number\ of\ correct\ preferences}{number\ of\ preferences}$$

Equation 2

$$IONWI_{recall} = \frac{number\ of\ correct\ preferences}{number\ of\ preferences\ for\ user}$$

Equation 3

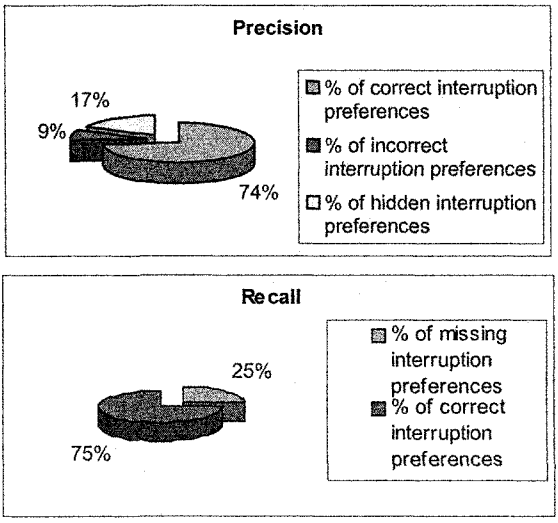

Fig. 2. *IONWI* Precision (a) and Recall (b)

We can observe in the figures that the percentage of incorrect interruption preferences is small (9% in average), and that the percentage of correct preferences

plus the percentage of hidden preferences is considerably high. The percentage of correct interruption preferences plus the percentage of hidden preferences can be considered as the precision of the algorithm. This value is approximately 91%. Thus, we can state that the learning capability of the *IONWI* algorithm is good.

Regarding the algorithm recall, 25% of the interruption preferences specified by the user were not discovered by our algorithm. Although not observable in the graphic, this value was smaller for those datasets containing more than 50 records.

4. Related Work

Interruptions have been widely studied in the HCI area[2], but they have not been considered in personal agent development. These studies revealed that the disruptiveness of an interruption is related to several factors, including complexity of the primary task and/or interrupting task, similarity of the two tasks [8], whether the interruption is relevant to the primary task [6], stage of the primary task when the interruption occurs [7], management strategies for handling interruptions [15], and modalities of the primary task and the interruption [2, 11].

People at Microsoft Research have deeply studied the effects of instant messaging (IM) in users, mainly on ongoing computing tasks [6, 7, 9]. These authors found that IM that were relevant to ongoing tasks were less disruptive than those that were irrelevant. This influence of relevance was found to hold for both notifications viewing and task resumption times, suggesting that notifications that were unrelated to ongoing tasks took longer to process.

As we have already said, related studies on interruptions come from different research areas in which interface agents are not included. Nevertheless, the results of these studies can be taken into account by interface agents to provide assistance to users without affecting users' performance in a negative way and, thus, diminishing the disruptiveness of interruptions. None of the related works we have discussed has considered the relevance of interruptions to users, or the relevance the situation originating the interruption has for the user. This issue and the relevance of interruptions to user tasks are two aspects of interruptions that our learning algorithm considers.

5. Conclusions and Future Work

We have presented a profiling algorithm that learns when and when not to interrupt a user, in order to provide him assistance. We have evaluated our proposal in the calendar management domain and the results we have obtained are quite promising. Experiments with personal agents assisting users with our approach in other domains are currently being carried out.

As a future work, we are planning to enhance the representation of a user's context in order to take other aspects into account.

[2] Bibliography on this topic: http://www.interruptions.net/literature.htm

References

[1] Agrawal, R., Srikant, R. - Fast Algorithms for Mining Association Rules – In Proc. 20th Int. Conf. Very Large Data Bases (VLDB) – 487 – 499 – (1994)

[2] Arroyo, E., Selker, T, Stouffs, A. – Interruptions and multimodal outputs: Which are less disruptive? – In IEEE International Conference on Multimodal Interfaces ICMI 02 – 479 - 483 (2002)

[3] Bickmore, T. Unspoken rules of spoken interaction. Communications of the ACM, 47 (4): 38 – 44 – (2004)

[4] Brown, S. and Santos, E. - Using explicit requirements and metrics for interface agent user model correction. In Proc. 2nd International Conference on Autonomous Agents – (1998)

[5] Cheung, D., Lee, S., Kao, B. A general incremental technique for maintaining discovered association rules. In Proc.5th Int. Conf. on Database Systems for Advanced Applications – (1997).

[6] Czerwinski, M., Cutrell, E., Horvitz, E. Instant messaging and interruption: Influence of task type on performance. In Proc. OZCHI2000 – 2000.

[7] Czerwinski, M., Cutrell, E., Horvitz, E. Instant Messaging: Effects of Relevance and Timing. People and Computers XIV: Proceedings of HCI 2000 – 71 – 76 (2000)

[8] Gillie T., Broadbent, D. What Makes Interruptions Disruptive? A Study of Length, Similarity and Complexity - Psychological Research, Vol. 50, 243 – 250 (1989)

[9] Horvitz, E., Jacobs, A., Hovel, D. Attention-Sensitive Alerting - Proceedings of UAI 99, Conference on Uncertainty and Artificial Intelligence – 305 - 313 (1999)

[10] Klementinen, M., Mannila, H., Ronkainen, P., Toivonen, H., Verkamo, A. I. - Finding interesting rules from large sets of discovered association rules. In 3rd Int. Conf. on Information and Knowledge Management – (1994) 401 – 407

[11] Latorella, K. Effects of Modality on Interrupted Flight Deck Performance: Implications for Data Link - Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting (1998)

[12] Lee, S., Cheung, D. Maintenance of discovered association rules: When to update? In Proc. SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery – (1997)

[13] Liu, B., Hsu, W., Ma, Y. - Pruning and summarizing the discovered associations - In Proc. 5th ACM SIGKDD – (1999)

[14] Mc. Crickard, S., Chewar, C. Attuning notification design to user goals and attention costs. Communications of the ACM, 46 (3): 67 – 72 – (2003)

[15] Mc Farlane, D. Coordinating the interruption of people in human-computer interaction. INTERACT 99, 295 – 303 – (1999)

[16] Miller, C. Definitions and dimensions of etiquette. In Proc. AAAI Fall Symposium on Etiquette and Human-Computer Work – (2002)

[17] Miller, C. Human-computer etiquette: Managing expectations with intentional agents. Communications of the ACM, 47 (4): 31 – 34 – (2004)

[18] Nass, C. Etiquette equality: Exhibitions and expectations of computer politeness. Communications of the ACM, 47 (4): 35 – 37 – (2004)

[19] Schiaffino, S., Amandi, A. – User – Interface Agent Interaction: Personalization Issues – International Journal of Human – Computer Studies, 60 (1): 129 – 148 – (2004)