Interface Agents

Paul Schmieder
705 Advanced HCI
Computer Science, University of Auckland
Paul.Schmieder@gmail.com

ABSTRACT
Being part of everyone's life, mundane routines are a familiar, time consuming part people have to deal with. In a computer environment such tasks could be checking emails, maintaining schedules or simply finding a good movie to watch.
A possibility to reduce the time involved is the use of software agents, programs acting in behalf of the users to support them. Applied to the problems mentioned before, agents could check emails, extract important information and present it. Other tasks are to update the user's schedules or to search for movies which are likely to be enjoyed by the user.
This report introduces and discusses the problems an agent has to overcome in order to establish and maintain an effective cooperation with its human operator. In order to guarantee the efficient teamwork the agent has to know the users, their goals and their working environment. To explain current approaches six existing interface agents are introduced including their methodologies to solve the challenges of human agent cooperation. These interface agents are VIO, LookOut, MailCat, Re:Agent, Magi and Meeting Schedule agent. While all agents work within the same field, each looks at it from a slightly different angle, such as efficiency (VIO), agent user interaction fundamentals (Meeting Schedule Agent, LookOut), interface design (MailCat) and algorithms (Re:Agent, Magi).
Furthermore, an outlook is given describing possible future interface agents.

INTRODUCTION
The rise of software agents started in the 1990's and is nowadays a vibrant combination of Artificial Intelligence (AI) and Human Computer Interaction (HCI).
Its roots however, go back to the 1950's when AI as a field of computer science was born. It was the work of scientists such as Negroponte (Negroponte, 1973) and Lay (Kay, 1990) in the mid 1990's on which the new computer science field of agent-based computing was built. The first person who described the interaction between a human operator and a software based agent was Maes (Maes, 1994). According to their pioneering work the software agent is supposed to work as an assistant, like a co-worker, which acts in a partly autonomous and independent manner. It is delegated via commands and supports the user with accomplishing mundane and time consuming tasks by directly manipulating the working environment.
Being inspired by Maes' work many cooperative human computer agent systems have been developed in several different domains including maintaining schedules(Kozierok & Maes, 1993; Mitchell, Caruana, Freitag, McDermott, & Zabowski, 1994), checking emails (Boone, 1998; Payne & Edwards, 1997) and recommender systems (Dabbish, Kraut, Fussell, & Kiesler, 2005; Terveen, Hill, Amento, McDonald, & Creter, 1997). To employ a successful human computer interaction every agent has to address three fundamental questions:

1. Who is my user?
2. What should the interaction look like?
3. How can I competently help my user?

To answer the first question the agent has to learn the user's goals and habits. This can be done by observing the user or by directly asking for feedback. Once the agent knows the user's goals it has to understand the environment in which the human operator tries to accomplish the goals. Due to the fact that goals and/or environments could change over time, the agent has to keep itself updated. The learning process of the initial goals and environments as well as the updating process have to be accomplished by disturbing the user as few times as possible. Another issue is the time these procedures need which ideally should be zero.
The second question can only be answered after the user trusts the digital helper's ability to effectively predict results. Without trust the user would most likely never delegate any tasks to the software based companion. This makes winning the users trust and maintaining it one of the most important objectives of such a system. Having earned the user's trust the human operator should be able to easily understand how the agent could autonomously work and how the interaction could work. Therefore simple models which define these kinds of interactions have to be introduced whereby the focus always has to be on the model's simplicity. Once the user trusts and understands the agent's capabilities, they will transfer tasks to the agent and will accept it as an esteemed assistant. Additionally, the willingness to invest time to train the agent to increase the interaction efficiency will rise.
The answer to the third question is probably the most difficult and important one to find. Even if the agent knows the users, their goals and how to accomplish these, it has to present its results in an appropriate manner. Therefore the digital assistant has to choose the way of interaction such as directly interrupting the user to present the results or process the results automatically as if the user would have done it. After the style of cooperation is established, the agent's tasks can be automated so that the user has to invest as little time as possible and takes advantage of the digital helpers

work.
If any one of the three issues briefly discussed above is not addressed correctly, the agent will rather increase the user's workload instead of reducing it and as a result soon be rejected.
This report will address the answers in a more detailed manner. Thereby several existing agents are introduced including their approaches to solve these problems. These user interface agents work in different domains and offer solutions from slightly different perspectives. To understand every system they will be briefly described in the next chapter before their problem solutions to the specific questions is discussed in section 3. The final two sections will sum up the presented issues and furthermore look into the future. Thereby questions which still have to be answered as well as questions which are only poorly answered yet will be briefly discussed.

HUMAN AGENT INTERACTION (HAI)

In this section the fundamental questions an agent has to address in order to enable an efficient cooperation with its human operator are discussed. Additionally, different approaches of already existing agents are introduced to get a better understanding about current problems and possible solutions.

Interface agents

In the following six interface agents are briefly introduced.

VIO

The interface agent VIO (Zimmerman et al., 2007) is a domain independent system which supports the user in translating requests. VIO is not designed to be the perfect agent which makes no errors but to enable an easy error detection and correction by the user. It observes the user's behaviour and filters the necessary information it needs to make meaningful suggestions to reduce the overall task completion time. Being applied to the email domain, VIO analyses incoming emails, modifies them, presents the changed version to the user and waits for an approval. While the changes made in the original email are reduced to highlightings of important text snippets, the overall analyses of the matter is leading to task proposals presented in form of a ranked list with entries such as "Add event", "Modify Person" or "Delete File".

LookOut

LookOut (Horvitz, 1999) is an agent system which supports users with managing schedules and meetings. It analyses the content of incoming mails and composes meetings or assist the user with reviewing the calendar. Besides the obvious patterns in emails such as "meeting at Tuesday at 4 pm", LookOut also has knowledge about colloquial time expressions such as "sometime tomorrow" and "afternoon".

MailCat

MailCat (Segal & Kephart, 1999) supports the user by deciding which folder to store an incoming email in. To do so, the program uses a text classifier which analyses the email content. Based on the classification results gained from the already existing email-to-folder allocation analysis, MailCat predicts the three most likely destination folders and displays 3 buttons representing these to let the user make the final decision. To cope with things like the creation of a new folder, renaming a folder and deleting folders MailCat uses incremental learning strategies. This means that the program monitors all changes to the folder structure and reruns the email classification to stay up to date.

Re:Agent

Re:Agent (Boone, 1998) analyses incoming emails to either sort, delete or store them in folder structure. Re:Agent filters features in the form of words from the email's content to generate feature vectors. Based on these features machine learning algorithms (e.g. nearest neighbour algorithm, the neural net) are used to compute action vectors. The resulting action vectors are used to determine the final action (e.g. sort, store). During the learning period the user has to classify emails into priority groups such as high priority or social.

Magi

Magi (Payne & Edwards, 1997) is an email agent which helps the user to sort incoming emails. Payne and Edwards concentrated their resources on the comparison and evaluation of different learning algorithms, prediction features and feature extraction models. They especially concentrated on evaluating the two learning algorithms CN2 and IBPL1. Depending on the time a user is willing to invest and the number of training examples the right algorithm has to be chosen.

Maes' Meeting Schedule Agent

Maes' Meeting Schedule agent (Kozierok & Maes, 1993) interacts with a user's calendar to schedule meetings. To do so, it observes the user to study their habits and preferences. If the user wants to schedule an appointment the user can either do it alone or ask the agent. Therefore the user has to feed the agent with acceptable dates. The digital helper then proposes possible meeting times by considering the user's free time slots and habits amongst other things.

Agent Issues

The efficiency of HAI is significantly determined by the agent's ability to decrease the time the user has to invest to accomplish mundane and time consuming tasks. Therefore the digital helper has to understand the user and the working environment. It has to adopt an appropriate interaction style and furthermore to solve the delegated tasks and process them on behalf of its human operator. This section discusses these agent issues including the problems and sub problems they cause. Additionally, approaches to overcome these problems from previous studies and their findings are used to explain the agent issues. The single problems and their solutions are grouped into the three categories/questions introduced before [1].

[1] "Who is my user?",
"What should the interaction look like?",
" How can I competently help my user?"

Who is my user?

Domains exits in the digital world as they do in the real one. Therefore, the environment the agent should work in has to be defined in the first place. Two basic design approaches are common in today's agent developments; generic and domain specific designs. While the specific agents are designed to operate in one particular domain only, the generic digital assistants could work in different areas. Nevertheless, every domain has to be implemented upon the generic core of a generic agent. The advantage of the generic approach is the applicability to every domain with only minor time expense [2]. However, due to the generic core it is not possible to design the agent in such a way that it can directly target special features of a certain domain. This is why specialized agents usually are more precise but still limited to one domain. Two generic agents are VIO (Zimmerman et al., 2007) and Re:Agent(Tomasic, Simmons, & Zimmerman, 2007) and two specific ones are LookOut (Horvitz, 1999) and MailCat (Segal & Kephart, 1999). Possible domains are email filtering (Boone, 1998; Payne & Edwards, 1997), recommender (Dabbish et al., 2005; Terveen et al., 1997) and schedule maintainer (Kozierok & Maes, 1993; Mitchell et al., 1994). Since within a particular domain only specific domain related tasks are possible, a human operator as well as an agent only has to accomplish these possible ones . For example, an email agent's scope of services could be to delete unimportant emails (Zimmerman et al., 2007), store them in specific folders (Payne & Edwards, 1997) or extract data to update contact information (Zimmerman et al., 2007) but not to recommend a movie because that is not possible in an email domain.

After the possible goals are predetermined due to the domain choice the particular goal has to be determined by the agent as well as the user habits. The latter can be done by observation (VIO, MailCat) or direct feedback (Re:Agent). An agent can also employ both methods (Magi, Meeting Schedule agent (Kozierok & Maes, 1993)). The first published interface agent was Maes' Meeting Schedule agent (Kozierok & Maes, 1993). It observes all user actions and stores them in so called "situation action pairs". Based on this "memory" it runs memory based learning algorithms (Stanfill & Waltz, 1986) to determine the user action which is most likely. This is done by comparing the current situation with all stored ones and looking for the closest match by calculating a distance vector. Additionally, it uses reinforcement learning algorithms to calculate weightings for meeting topic keywords which act as priorities. The weightings are calculated upon positive and negative predictions the agent did in the past and direct feedback from the user to clarify the agent's false assumptions. Another interface agent is Magi. As well as Maes' Meeting Schedule agent, it observes the user and asks for direct feedback. Being implemented as a testbed for different algorithms, Magi uses different algorithms to learn the user's habit to achieve the most effective cooperation. Payne and Edwards concentrated on the evaluation of two learning algorithms; CN2 (Clark & Niblett, 1989) and ILBP1. The latter is producing exemplars of training examples to generate weights and distance metrics. CN2's advantage over ILBP1 is the comprehensibility of the produced rules for people. Furthermore, it needs less training examples to achieve good prediction results.

This raises another critical issue. What are "good" results and how do I get results in the first place? To calculate any result the source has to be analysed. In case of an email it is the email's content. This can be done in several ways. Magi (Payne & Edwards, 1997) calculates features from the email's "From", "Subject" and the "Message" fields. Depending on the algorithm applied, it extracts one feature per field (CN2 algorithm) or sets of features (ILBP1). For example, ILBP1 compares pre-calculated feature sets which have been derived from training examples with the feature sets calculated for the new email based on a k-nearest neighbour algorithm (Dasarathy, 1991). The resulting k-closest feature sets are used to determine the feature distance to the stored exemplars. Thereby biases in the form of neural net weights are used to find the closest feature set.

VIO (Tomasic et al., 2007) uses a similar approach to find rules. It predefines the possible tasks which can be done according to the email's content. For the prediction the interface agent extracts text snippets and computes labels for them. Thereby every label represents a hypothesis describing the likeliness to be a certain list entry. These weak labels or hypothesises are basically simple rules which have differing ranges of accuracy. The accuracy is determined by comparing the label from the new unknown email with the labels calculated from the training set. Afterwards these weak labels are fed into a boosted decision tree model to rank the hypothesises according their likeliness (Schapire & Singer, 1999).

Both approaches, VIO and Magi's, need initial training on existing examples to train their algorithms whereby the training time depends on the quantity of training examples. This poses another important issue for interface agents; how long do they need to be trained and how much training examples do they need to achieve a satisfying accuracy. This mainly depends on the machine learning algorithm used. For example, the CN2 algorithm used in Magi needs significantly less training time and data than the ILBP1 to compute accurate results. However, if ILBP1 is trained with sufficient training examples, it achieves a higher accuracy than CN2 (Payne & Edwards, 1997). In contrast to the Magi's rules based learners, Re:Agent employs a neural net. Rule based learners as well as neural nets need sufficient training examples to generate accurate rules and to determine efficient weights respectively. The LookOut agent is another inductive learner which means that it extracts rules out of given training examples.

VIO and MailCat, in contrast, only need very little training examples. The reason is that both employ a more deductive way of reasoning. This means that they use the provided examples and try to use them as premises to conclude a prediction. Such an approach makes a confidence rating necessary because otherwise both agents would make proposals based on little evidence. To ensure highly accurate predictions both agents have to additionally train themselves by observing the human operator.

To determine the quality of the results the agent has to rank them in form of a confidence rating. The more sure it is of the result, the more confidently it presents the results to

[2]The domain still has to be implemented upon the generic core

the user. For example, VIO which presents a list of possible actions based on its email analysis designs a list based on its confidence. If the agent is absolutely sure, the list only consists of one entry representing an action. The more uncertain the digital assistant is the more items it adds to the list. If it has no confidence in its results it makes no suggestions (Zimmerman et al., 2007).
There are several ways to calculate the degree of confidence. One is to compute a predictive and confidence threshold (Payne & Edwards, 1997). Thereby a predefined threshold (predictive and confidence) is compared with the currently calculated one and if the current one is higher the agent will proceed with the task. Additionally, the agent presents its confidence ratings to the user to receive feedback. The human operator can then either confirm or reject the proposal. The agent uses the feedback to adjust its threshold for the current task. Another approach of the confidence calculation is implemented in LookOut (Horvitz, 1999). It uses a Support Vector Machine (SVM) to classify the extracted text via approximation (Platt, 1999). Additionally, the LookOut agent uses custom features for task-specific sets to run the confidence calculation not only on particular pieces of text but rather to also include the context (task-specific sets). An important task for the confidence calculation is the SVM text classifier. It is trained on runtime as well as at initialisation time.
In conclusion, to get to know the user including the goals and the environment all agents draw on different kinds of machine learning algorithms. Due to this approach a varying range of training times is necessary as well as sufficient training examples. Furthermore, to autonomously evaluate the gained predictions the agent has to employ a confidence rating to ensure an effective cooperation.

What should the interaction look like?

To earn the users trust the agent has to deliver accurate predictions. Therefore it has to successfully employ the features described in the previous section. After it has earned the user's trust the digital assistant has to maintain and extend it. From the beginning both partners have to communicate with each other. Besides the classical ways of interaction via mouse and keyboard which are employed by all agents, LookOut (Horvitz, 1999) also uses verbal communication. Thereby the agent can talk to its human operator by using a text-to-speech system. Additionally, the user can talk to LookOut which has established an audio channel to do so.
For an effective interaction the user has to understand how the agent works so that the user can delegate tasks and receive results efficiently. To achieve this understanding, simple interaction models are used. Agents which use these models are Maes' Meeting Schedule agent (Kozierok & Maes, 1993) and VIO (Zimmerman et al., 2007). The latter employs an interaction model set by Letizia (Lieberman, 1997). According to this model the assistance is not forced on the user but rather offered in an appropriate way. If the user's attention is attracted by the digital assistant the human operator can interact with it otherwise the human operator simply ignores the agent.
LookOut has a similar approach where the user is informed based on an attention model. This model defines times when the user will be statistically less distracted if interrupted by LookOut (Horvitz, 1999). These times were gained through experiments where the times between the message review and the resulting invocations by the human operator were measured. The resulted temporal-centric model of attention takes relationships between message length and the preferred interruption time into account which are approximated by a sigmoid function. Additionally, if necessary the agent can replace the predefined time model by one which it calculates while observing the user. Thereby the times are calculated in the same way they were during the experiments. Another option is the explicit time delay adjustment by the user.
A more feedback intensive approach is implemented in Maes' Meeting Schedule agent (Kozierok & Maes, 1993) where the interaction is modelled like a self-learning process. Thereby the agent has an initial knowledge about scheduling which is mostly manually determined by the user. Additional knowledge is steadily added by observing the user and studying the user's habits.
When the user trusts and understands the agent the user will most likely transfer more tasks.

How can I competently help my user?

Once the interaction style is established the agent has to formulate a plan to help the user and not to hinder the user. This includes the way the tasks are processed according to the user's preferences and the degree of automation of the tasks.
To work according to the user's preferences is a difficult task for the agent. A study by Dabbish, Kraut, Fussell, and Kiesler showed strong individual differences between people when working with their emails. The intention of the study was to model people's email behaviour which can be later used to design email user interfaces. When sorting emails users had a 48% variance in selecting a destination folder for incoming emails. Furthermore, the likelihood of replying an email was bigger for social messages than any other although they had nothing to do with work. Dabbish, Kraut, Fussell, and Kiesler reasoned that the typical user does not exist. This fact makes the successful observation of the user by the agent more important because that is currently the only way to personalize the agent's interaction style.
Currently most agents process the predicted actions if they are confident enough. It is up to the user to change the results if they are not satisfied. With increasing observation time the automation increases automatically because the agent learns more and more about the user and has therefore higher confidence. This leads to more accurate predictions and to a higher degree of the user's satisfaction. Basically the degree of automation is determined by the autonomous work accomplished by the agent without being interrupted by the user. For example, VIO fills in data extracted from emails autonomously in the corresponding forms. In case the extracted information is wrong the user has to correct them manually. In case VIO makes no mistakes the user only has to click the first item in the proposed list and the following form is correctly filled out by VIO so that the user only has to save it. In this optimal scenario the time benefit is sig-

nificant. Instead of manually filtering the email's content, choosing the correct form and inserting the new data, the user only has to choose one list entry and confirm the update with one button click.

SUMMARY

The report described the steps necessary to establish an effective human agent interaction. The agent is acting on behalf of the user to process mundane and time consuming tasks which significantly reduce the time the user has to invest to complete these tasks. After the agent has gained sufficient information about the user's habits and preferences as well as about the environment, it has to earn and maintain the users trust by successfully making predictions. Six different existing interface agents were used to explain the problems connected with the agent's work and to show methodologies to solve them; VIO (Zimmerman et al., 2007), LookOut (Horvitz, 1999), MailCat (Segal & Kephart, 1999), Re:Agent (Tomasic et al., 2007), Magi (Payne & Edwards, 1997) and Meeting Schedule agent (Kozierok & Maes, 1993). According to all of these authors, no agent is designed to be perfect but rather to allow easy interaction, error detection and error correction. Down to the present day the prediction rate of existing agents is not even close to perfect but the development is steadily continuing.

FUTURE WORK

Hong and Landay envision a domain-spanning agent which provides the right information at the right time. The agent constantly searches for information in the background. In contrast to the other introduced agents Hong and Landay's one will not automatically present its findings but will rather wait until the user needs them and then call attention to itself. To implement an agent as sophisticated as the one proposed by Hong and Landay the human prediction models have to be further developed. Therefore a closer investigation of human habits is necessary. Additionally, due to human individuality the agent has to adopt too its user's habits to a higher degree. LookOut's (Horvitz, 1999) temporal-centric model of attention is a beginning but has to be further developed.

References

Boone, G. (1998). Concept features in re:agent, an intelligent email agent. In *Agents '98: Proceedings of the second international conference on autonomous agents* (pp. 141–148). New York, NY, USA: ACM.

Clark, P., & Niblett, T. (1989). The cn2 induction algorithm. *Mach. Learn.*, *3*(4), 261–283.

Dabbish, L. A., Kraut, R. E., Fussell, S., & Kiesler, S. (2005). Understanding email use: predicting action on a message. In *Chi '05: Proceedings of the sigchi conference on human factors in computing systems* (pp. 691–700). New York, NY, USA: ACM.

Dasarathy, B. (1991). *Nearest neighbor (NN) norms: nn pattern classification techniques*. Los Alamitos, Calif.: IEEE Computer Society Press; Washington: IEEE Computer Society Press Tutorial.

Hong, J. I., & Landay, J. A. (2001). A context/communication information agent. *Personal Ubiquitous Comput.*, *5*(1), 78–81.

Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In *Chi '99: Proceedings of the sigchi conference on human factors in computing systems* (pp. 159–166). New York, NY, USA: ACM.

Kay, A. (1990). User Interface: A Personal View. *The Art of Human-Computer Interface Design*, 191–207.

Kozierok, R., & Maes, P. (1993). A learning interface agent for scheduling meetings. In *Iui '93: Proceedings of the 1st international conference on intelligent user interfaces* (pp. 81–88). New York, NY, USA: ACM.

Lieberman, H. (1997). Autonomous interface agents. In *Chi '97: Proceedings of the sigchi conference on human factors in computing systems* (pp. 67–74). New York, NY, USA: ACM.

Maes, P. (1994). Agents that reduce work and information overload. *Commun. ACM*, *37*(7), 30–40.

Mitchell, T. M., Caruana, R., Freitag, D., McDermott, J., & Zabowski, D. (1994). Experience with a learning personal assistant. *Communications of the ACM*, *37*(7), 80–91.

Negroponte, N. (1973). *(The) Architecture machine*. MIT Press.

Payne, T., & Edwards, P. (1997, January). Interface agents that learn: An investigation of learning issues in a mail agent interface. *Applied Artificial Intelligence*, *11*(1), 1-32.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. , 185–208.

Schapire, R., & Singer, Y. (1999). Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, *37*(3), 297–336.

Segal, R. B., & Kephart, J. O. (1999). Mailcat: an intelligent assistant for organizing e-mail. In *Agents '99: Proceedings of the third annual conference on autonomous agents* (pp. 276–282). New York, NY, USA: ACM.

Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. *Commun. ACM*, *29*(12), 1213–1228.

Terveen, L., Hill, W., Amento, B., McDonald, D., & Creter, J. (1997). Phoaks: a system for sharing recommendations. *Commun. ACM*, *40*(3), 59–62.

Tomasic, A., Simmons, I., & Zimmerman, J. (2007). Learning information intent via observation. In *Www '07: Proceedings of the 16th international conference on world wide web* (pp. 51–60). New York, NY, USA: ACM.

Zimmerman, J., Tomasic, A., Simmons, I., Hargraves, I., Mohnkern, K., Cornwell, J., et al. (2007). Vio: a mixed-initiative approach to learning and automating procedural update tasks. In *Chi '07: Proceedings of the sigchi conference on human factors in computing systems* (pp. 1445–1454). New York, NY, USA: ACM.