Recognition of User Intentions for Interface Agents with Variable Order Markov Models

Marcelo G. Armentano[1] and Analía A. Amandi[2]

[1] ISISTAN Research Institute, Fac. Cs. Exactas, UNCPBA
Campus Universitario, Paraje Arroyo Seco, Tandil, 7000, Argentina
[2] CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina

Abstract. A key aspect to study in the field of interface agents is the need to detect as soon as possible the user intentions. User intentions have an important role for an interface agent because they serve as a context to define the way in which the agents can collaborate with users. Intention recognition can be used to infer the user's intentions based on the observation of the tasks the user performs in a software application. In this work, we propose an approach to model the intentions the user can pursue in an application in a semi-automatic way, based on Variable-Order Markov models. We claim that with appropriate training from the user, an interface agent following our approach will be able both to detect the user intention and the most probable sequence of following tasks the user will perform to achieve his/her intention.

1 Introduction

Interface Agents [Maes, 1994] are computer programs designed to assist human users in their computer-based tasks in a personalized manner. This kind of agent is able to learn interests, preferences, priorities, goals and needs of a user aiming at providing him/her proactive and reactive assistance in order to increase the user's productivity regarding to the application at issue.

With the aim of assisting a user of a software application, interface agents not only have to learn the user's preferences and habits regarding the use of the application itself, but should also consider what his/her intention is before initiating an interaction with the user. Considering the status of a user's attention (i.e. his/her intention or the goal he/she is pursuing) and the uncertainty about the user's intentions are critical factors for the effective integration of automated services with direct manipulation interfaces [Horvitz et al., 1998]. A correct detection of the user's intention will avoid the agent interrupting the user in an improper moment. Users generally don't want to be interrupted while working on a specific task, unless this interruption is strongly related to the task they are performing [Whitworth, 2005]. By considering the user's intention the agent will be able to answer to his/her requirements always in the realm of his/her current intention. As a result, we must build agents capable of detecting the

G.-J. Houben et al. (Eds.): UMAP 2009, LNCS 5535, pp. 173–184, 2009.

user's intention so that it can predict opportune moments for gaining the user's attention.

In this work we propose an approach to automatically obtain a model of the user intentions in a software application to allow a posterior detection of those intentions. This model aims at being considered by an interface agent as a context that represents the user's focus of attention in a particular moment in the use of the application. The interface agent can use this knowledge to assist the user in the context of his/her intentions and, moreover, to find a suitable moment to initiate an interaction with him/her (that is, when the agent is quite sure of his/her intention). However, how the agent uses the detected intention to assist the user is out of the scope of our work.

The rest of this work is organized as follows. Section 2 describes some related work in the area and the problems detected in existent approaches. Next, in Section 3 we describe our approach to the problem of modeling and detecting a user's intention. In Section 4 we present the experiments we performed to validate our proposal. Finally, in Section 5, we present our conclusions.

2 Problem Overview

Intention recognition in this context can be defined as the process of inferring a *user* intentions based on the observation of the actions he/she performs in a *software application*. Intention recognition is a special case of plan recognition in which only the intention of the user, but not the associated plan is predicted. A complete plan recognition process is a more complex and time requiring task. In the domain of an interface agent, such as in many other domains, it is preferable a fast detection of just the user's intention than a slower detection of the complete plan needed to fulfill his/her intention.

The basic idea beyond the intention recognition process is to narrow the number of possible goals the agent believes the user is pursuing. This task is accomplished by observing the actions the user performs. For example, when starting a scheduling application, the user can have any goal $G_1, G_2, \cdots, G_n$. Now, if the agent observes that the user performs certain task, like selecting "Add new contact" in the application menu, the set of goals is reduced to those in which the task performed is included as a particular step (for example, organizing an event with the new contact as a participant, or sending a email to him/her). If the next task observed is "Compose new email", the set of candidate intentions can be further narrowed. Note that even if there is only one candidate intention, the agent will not always be absolutely sure that the user really has that intention. This way, each time the user performs an action in the application, the set of candidate intentions might be reduced and/or some intention will be more probable than others.

The basic algorithm to accomplish plan or intention recognition seems to be straightforward; however, most of previous approaches to the problem fail in three main aspects. The first problem that makes many previous approaches

to the problem of plan recognition unsuitable for interface agents is that plan libraries are usually hand-coded by a domain expert [Kautz, 1991] [Charniak and Goldman, 1993] [Horvitz et al., 1998] [Lesh et al., 1999]. Building a plan library is a tedious and error prone task and the success of a plan recognizer firstly relies on the correctness and completeness of the plan library itself. For this reason, in the recent years researchers have put special attention in the acquisition of plan libraries by constructing models that capture regularities in the user behavior. Nevertheless, most of this research was conducted to learn the parameters of the model, such as probabilities, while the structure of the model itself remained fixed [Nguyen et al., 2005] [Duong et al., 2006] [Liao et al., 2007]. On the other hand, few efforts were put on the task of learning plan libraries from the interaction history of a user with a software application and the proposed approaches are limited in the kind of plan structures that they are able to model [Bauer, 1999] [Gorniak and Poole, 2000] [Garland; and Lesh, 2002]. Behavior usually differs from one user to another and a predefined structure of plans may not fit a specific user behavior. For these reasons, the automatic acquisition of plan libraries is desirable.

Second, one of the most important problems that an interface agent faces when inferring the user's intention is the uncertainty related to the moment in which the user starts a new plan to achieve a new goal, that is how does the agent become aware that the user has already achieved one goal and started pursuing a new one? This issue is not usually addressed by many approaches to the problem of plan recognition, and they consider only one "session", which starts with the first observed action and ends when the algorithm recognizes the user's intention. In an interface agent environment, the user will repeatedly start pursuing new goals in the application, with no preplanned behavior. Moreover, the user can even change his/her intention without completing his/her previous goal. This problem is usually tackled by restricting the memory of the plan recognizer so that it only considers the most recent tasks performed by the user, or it considers each task for only a fixed interval of time and then they are completely disregarded [Brown, 1998] [Waern, 1996].

Another issue to take into account is that users usually follow several intentions at a time. Consequently, a plan recognizer used by an interface agent should be able to manage the realization of multiple user intentions simultaneously. Plan recognizers that limit themselves to one-at-a-time intentions are not suitable in the interface agents domain. Related to this issue is the execution of noisy tasks. Noisy tasks are tasks that the user performs but that do not belong to his/her main goal, such as checking the current time while answering an email. Predictions of a plan recognizer should not be highly affected by the presence of such kind of tasks.

Although there are many approaches to the problem of plan recognition, no previous approach is able to manage all the issues stated in this section. In the next section we present our approach to deal with all these aspects of an intention recognition system for interface agents.

3 Proposed Approach

3.1 Learning an Intention Model from Examples

Markov models are a natural way of modeling sequences of actions observed along time. In its simplest form, a Markov chain is a stochastic process with the Markov property. Having the Markov property means that, given the present state, future states are independent of the past states. In other words, the description of the present state fully captures all the information that could influence the future evolution of the process. Future states will be reached through a probabilistic process instead of a deterministic one. At each step the system may change its state from the current state to another state, or remain in the same state, according to a certain probability distribution. The changes of state are called transitions, and the probabilities associated with various state-changes are called transition probabilities.

Markov chains of fixed order are a natural extension in which the future state is dependent on the previous m states. Although this extension is beneficial for many domains, there are some main drawbacks in the use of these models. First, only models with very small order are of practical value since there is an exponential grow in the number of states of Markov chains as their order is increased. Second, for sequences of tasks performed by a user to achieve an intention, the probability of the next performed task is not always determined by the same fixed number of previous tasks. There is usually a variable length previous "context" that determines the probability distribution of what the user may perform next.

Hidden Markov Models are an alternative way of modeling natural sequences. Although these models are a powerful and popular representation, there are theoretical results concerning the difficulty of their learning [Ron et al., 1996].

Variable Order Markov (VOM) models arose as a solution to capture longer regularities while avoiding the size explosion caused by increasing the order of the model. In contrast to the Markov chain models, where each random variable in a sequence with a Markov property depends on a fixed number of random variables, in VOM models this number of conditioning random variables may vary based on the specific observed realization, known as *context*. These models consider that in realistic settings, there are certain realizations of states (represented by contexts) in which some past states are independent from the future states conducting to a great reduction in the number of model parameters.

Algorithms for learning VOM models over a finite alphabet Σ attempt to learn a subclass of Probabilistic Finite-state Automata (PFA) called Probabilistic Suffix Automata (PSA) which can model sequential data of considerable complexity. Formally, a PSA can be described as a 5-tuple$(Q, \Sigma, \tau, \gamma, \pi)$, where Q is a finite set of states, Σ is the task universe, $\tau : Q \times \Sigma \rightarrow Q$ is the transition function, $\gamma : Q \times \Sigma \rightarrow [0, 1]$ is the next task probability function, where for each $q \in Q$, $\sum_{\sigma \in \Sigma} \gamma(q, \sigma) = 1$, $\pi : Q \rightarrow [0, 1]$ is the initial probability distribution over the starting states, with $\sum_{\sigma \in \Sigma} \pi(q) = 1$.

A PFA is a PSA if the following property holds. Each state in a PSA M is labeled by a sequence of tasks with finite length in Σ^* and the set of sequences S labeling the states is suffix free. Σ is the domain task universe, that is the finite set of tasks that the user can perform in the domain. A set of sequences S is said to be suffix free if $\forall s \in S$, $Suffix^*(s) \cap S = \{s\}$, where $Suffix^*(s) = \{s_i, \cdots, s_l | 1 \leq i \leq l\}$ is the set of all possible suffixes of s, including the empty sequence e. For every two states q_1 and $q_2 \in Q$ and for every task $\sigma \in \Sigma$, if $\tau(q_1, \sigma) = q_2$ and q_1 is labeled by a sequence s_1, then q_2 is labeled by a sequence s_2 that is a suffix of $s_1 \cdot \sigma$.

In contrast to N-order Markov models, which attempt to estimate conditional distributions of the form $Pr(\sigma|s)$, with $s \in \Sigma^N$ and $\sigma \in \Sigma$, VOM algorithms learn such conditional distributions where context lengths $|s|$ vary in response to the available statistics in the training data. Thus, PSA models provide the means for capturing both large and small order Markov dependencies based on the observed data. In [Armentano, 2008] we proposed an algorithm for learning such models in an incremental way.

For learning a user's intention model we follow a Programming By Example (PBE) [Lieberman, 2001] approach in the sense that the user will teach the agent what sequence or sequences of tasks he/she usually performs when he/she has a given intention. However, unlike the classic programming by demonstration approach, our aim is not to create a program that allows the agent to perform repetitive tasks on behalf of the user, but to detect the user's intention that lead him/her to perform a set of tasks.

Learning the user's intention model by example has the main advantage that we do not need any additional information of the domain being modeled more than the tasks that can be performed in the domain. The agent will be able to learn regularities in the user's behavior just by analyzing the examples given by the user. This way the agent will be able to learn any intention the user may have in the domain, just by giving an example of how to fulfill this intention.

By using the examples provided by the user, the agent will build a PSA model for each goal the user can pursue in the domain. When a new example for an existent model is provided, it may correspond to an alternative way of reaching the same goal, so the corresponding model is updated to reflect this fact.

3.2 Recognizing a User's Intentions

To perform plan recognition, the agent will have a PSA model for each goal for which it was trained by means of examples provided by the user. By having a separate model for each goal, the agent will be able to track several goals that are being pursued simultaneously by the user.

Conventionally, to compute the probability assigned by a PSA k to a given sequence of observations, we should compute $P_{\mathrm{PSA_k}}(r) = \prod_{i=1}^{N} \gamma(s_{i-1}, r_i)$, where $\gamma(s_{i-1}, r_i)$ is the probability value assigned in state s_{i-1} to the observed task r_i, and will select the PSA that assigns the maximum probability as the PSA corresponding to the user's intention. However, as the user continues performing

tasks, the total cumulative probability value assigned by each PSA will become smaller and smaller as we are multiplying values in the range $(0, 1]$. Furthermore, we must consider the uncertainty related to the moment in which the user starts a new plan to achieve a new goal. The agent will be faced with a continuous stream of tasks and should be able to recognize changes in the user's current intention. Moreover, the plan recognition process should not be affected by the execution of noisy tasks. The problem we are facing is not a classical problem of classification as we do not predict a "class" (intention) after observing a complete sequence of performed tasks. In our domain, the interface agent should be able to predict the most probable intention after each performed task, and the limit between sequences of tasks corresponding to different intentions is often fuzzy.

To tackle this problem we use an *exponential moving average* on the prediction probability $\gamma(s, \sigma)$ at each step in each PSA as the predicted value for each corresponding user intention. Moving averages are one of the most popular and easy to use tools to smooth a data series and make it easier to spot trends. An exponential moving average (EMA) [Hunter, 1986] is a statistic for monitoring a process that averages the data in a way that gives less and less weight to data as time passes. The weighting for each step decreases exponentially, giving much more importance to recent observations while still not discarding older observations entirely. By the choice of a weighting factor $0 \leq \lambda \leq 1$, the EMA control procedure can be made sensitive to a small or gradual drift in the process. Alternatively, λ may be expressed in terms of N time periods, where $\lambda = \frac{2}{N+1}$.

EMA_t expresses the value of the EMA at any time period t. EMA_1 is set to the a priori probability of the first observed task σ. Then, the computation of the EMA at time periods $t \geq 2$ is done according to equation 1

$$EMA_t = \lambda \gamma_{PSA_i}(s, \sigma) + (1 - \lambda) EMA_{t-1} \tag{1}$$

The parameter λ determines the rate at which *older* probabilities enter into the calculation of the EMA statistic. A value of $\lambda = 1$ implies that only the most recent measurement influences the EMA. Thus, a large value of λ gives more weight to recent probabilities and less weight to older probabilities; a small value of λ gives more weight to older probabilities. The value of λ is usually set between 0.2 and 0.3 [Hunter, 1986] although this choice is somewhat arbitrary and should be determined experimentally.

To sum up, the plan recognition process works as follows: as the user performs tasks in the application at issue the agent will keep making the corresponding state transitions in each PSA and computing the exponential moving average of the transition probability of the performed tasks given each PSA. At each step, the agent will own a probabilistically ranked set of PSAs which correspond to the most probable intentions the user may have at each moment.

The problems we pointed out in Section 2 are then solved with our approach. The uncertainty related to the moment in which the user starts a new plan to achieve a new goal is managed by the exponential moving average by giving more importance to recent observations than to older ones. The rate at which previous intentions are forgotten is controlled by parameter λ of the EMA calculation.

The fact that users usually pursue several goals at a time is managed by keeping a set of PSA models, one for each goal the user can try to accomplish. With each task performed by the user, a transition is made in every model. Noisy tasks are also considered as the prediction of each PSA is computed as an EMA. Again, how much this task influences the prediction is controlled by parameter λ of the EMA calculation. Finally, personalization of the plan library is implicit in our approach, as it is the user who gives examples on which his/her intentions are and how to achieve each of them.

4 Experimental Evaluation

In the experiments shown in this section we evaluate two different metrics. The *Error* for a model q given an observed task sequence $Seq = \sigma_1, \cdots, \sigma_N$ is computed as the sum of the absolute differences between the value assigned for each model with respect to the higher value assigned by all the PSAs, as shown in Equation 2

$$error_q(Seq) = \frac{\sum_{i=1}^{N} |q(\sigma_i) - q_{best}(\sigma_i)|}{\sum_{i=1}^{N} q_{best}(\sigma_i)} \tag{2}$$

On the other hand, the *Convergence* is a metric that indicates how much time took the recognizer to converge in what the current user goal was. If from the time step t to the time step corresponding to the last performed task the algorithm predicted correctly the actual user goal, the convergence is computed as shown in Equation 3. The time step t is called *convergence point*.

$$convergence_q(Seq) = \frac{N - t + 1}{N}, \quad not\, best_q(\sigma_{t-1}), \; best_q(\sigma_j), \forall j \; t \leq j \leq N \tag{3}$$

$$where \quad best_q(\sigma_i) = \begin{cases} 1 \; if \; q(\sigma_i) = q_{best}(\sigma_i) \\ 0 \; otherwise \end{cases} \tag{4}$$

4.1 Recognition in the Presence of Noisy Tasks

The purpose of the experiments described in this section is to test the ability of the proposed model to perform well in the presence of noisy tasks. We considered three kinds of "noise" that can be observed while performing plan recognition: *Omitted tasks* (tasks that were observed in a training sequence and that are not executed in the recognition process), *Inserted tasks* (tasks that were not observed in a training sequence and are observed during the execution of a sequence corresponding to a given intention) and *Replaced task*s (tasks that were performed in the place of another task that was expected for a given intention).

To test the influence of the length of the task sequences corresponding to the intentions being modeled in the accuracy of the predictions, we run different experiments for different sequences length. For each experiment we used sequences

of a fixed length for training 20 PSA models. Each sequence was generated randomly from a set of 26 abstract tasks. Then we altered each training sequence by introducing different combinations of noise, both in amount and kind to build 25 testing sequences for each model and each combination of noises. The noise was introduced in amounts varying from 0 to 90 percent of the length of the sequence and in the three different kinds detailed above.

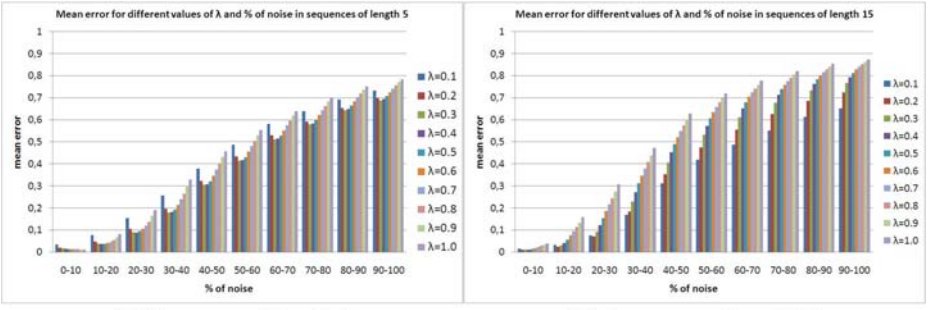
Figure 1 shows a comparison between the mean error for different values of λ for sequences of length 5, 15, 25 and 50. We did not compute convergence in this case because our goal is to evaluate individual sequences so that we can have a better perception of the influence of noise. We can observe that with few amount of noise present in the testing sequences, longer sequences lead to lower error in the prediction for any value of λ. However, as we increase the noise we introduce in the testing sequences there is a strong dependence on the value selected for λ and the length of sequences for the error of the predictions. As a general observation, we can state that higher values of λ make the system predict shorter sequences better than longer sequences, while lower values of λ make the system predict better longer sequences of tasks. For long sequences, it is suggested that the value of λ has to be reduced to 0.1 to obtain better results. However, for sequences of length 5, the value of λ that leads to lower error in the predictions is 0.3. This result is due to the fact that a lower value for the smoothing constant will take into consideration a history longer than 5 tasks, and this will include tasks not belonging to the current user intention.

4.2 Prediction of Consecutive Intentions

In this section we will analyze the amount of tasks the plan recognizer needs to observe to detect a change in the user intention (convergence) and how the execution of consecutive plans affects the error of the plan recognizer. We used the same set of 20 PSA models as in the previous experiments. For testing, we generated 50 sequences by concatenating sequences belonging to a set of η models selected randomly for each case. We experimentally set $\lambda = 0.9$, since we obtained better results with this value of the smoothing constant.

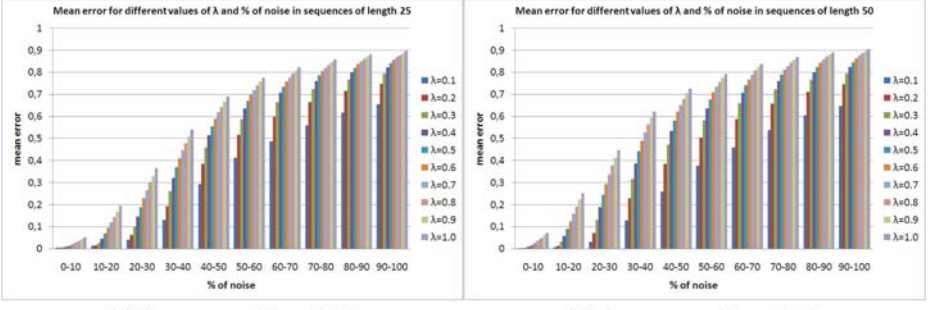
Figure 2 shows a comparative plot of the values obtained for convergence and error metrics for different sequence lengths when 3, 10 and 20 successive intentions were simulated.

We can observe that there is a fall in convergence, that is more notorious for shorter sequences, when we increase the number of successive intentions simulated. For longer sequences, however, there is almost no change in the value for this metric for 10 and 20 successive intentions. The convergence point is risen to 1.60, 2.02, 2.06 3.45 for sequences of length 5, 10, 15 and 25 respectively and for both cases of 10 and 20 successive intentions; for 10 successive intentions using sequences of length 50 the convergence point obtained is 7.55 tasks and for 20 successive intentions 7.9. For the error metric, on the other hand, the value obtained is higher as we simulate more successive intentions, but the difference tends to be smaller for longer sequences of tasks.



(a) Sequences of length 5

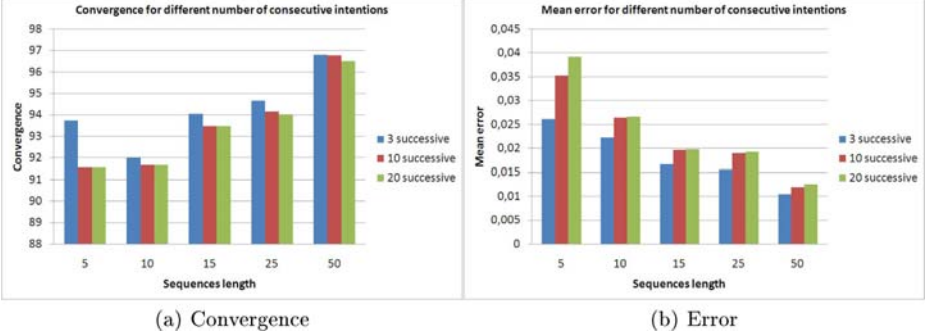
(b) Sequences of length 15



(c) Sequences of length 25

(d) Sequences of length 50

Fig. 1. Average error for different sequences length and different values of the smoothing constant



(a) Convergence

(b) Error

Fig. 2. Metrics values for different number of consecutive simulated intentions

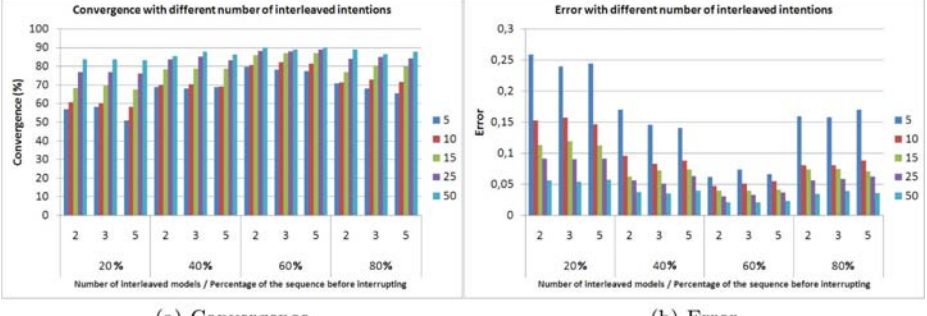
4.3 Prediction of Interleaved Intentions

In this section we will describe another series of experiments performed to analyze the behavior of the plan recognizer when the user performs tasks belonging to different intentions.

The set of models used are the same we used the previous section. We varied two variables in these experiments: the number of simultaneous models being tested η and the percentage of tasks performed before changing the current intention ζ. The smoothing constant λ was set experimentally to a value 0.9. For all the experiments η was varied to take values in the set $\{2, 3, 5\}$ and ζ to take "chunks" corresponding to the 20, 40, 60 and 80 percent of the length of the sequences.

For creating testing sequences we randomly selected a subset of the 20 models and generated a testing sequence by interleaving the tasks belonging to each model in this subset, taking chunks of a specified size each time. We also randomly selected the next model from which to take tasks, not considering the same model used immediately before. The last chunk remained usually shorter than the chunk size ζ. When this was the case, the performance of the recognizer usually decreased. For each combination of η and ζ we repeated the experiment 50 times selecting different models.

Figure 3 presents the values obtained for convergence and error metrics for sequences of length 5, 10, 15, 25 and 50.



(a) Convergence (b) Error

Fig. 3. Metrics values for interleaved intentions for different sequences length

Notice that for sequences of length 5 by using subsequences of 20% and 80% of the length of the sequence highly increases the error of the plan recognizer. This is due to the fact that a subsequence of 20% leads to isolated tasks (sequences of length 1). This represents a very uncommon situation in which the user would alternate between different intentions performing one task of each one. Sequences of length 1 are not sufficient to activate the memory of the model. Something similar happens with 80% of the sequence length; we will have a subsequence of length 4 and a sequence of length 1 would be left (the last task in the sequence). In general, we can observe that the number of simultaneous models considered in the experiments does not have a major influence in the resultant values for the error and convergence metrics. We can observe that there is a general better behavior of the recognition algorithm for both metrics for the case of chunks with size $\zeta = 60\%$. The reason for this is that this is the test case with a better equilibrium in the sizes of the chunks that lead to a better performance.

A final observation from the graphics presented in Figure 3 is that our plan recognizer predict longer sequences better. We obtained an error lower than 5% and a convergence of more than 80% for the best case of sequences of length 50 for all tested cases of interleaved intentions, and an error lower than 25%, with a convergence of more than 50% for the worst case of sort sequences of length 5.

One advantage we found for our plan recognizer is that there is no need to "remember" which was the last task performed before interrupting the current intention and start pursuing a different goal. Subsequences of the sequences used to train a model lead to correct predictions, although we do not start executing the sequence from its beginning.

5 Conclusions and Discussion

In this article, we presented an approach to model and recognize a user's intentions from the unobtrusive observation of his interaction with a software application. We propose the use of Variable Order Markov models to model each user intention and the use of an exponential moving average to tackle the evolution of the process through long user sessions. We evaluated our proposal with promising results. However, there is still a challenge that need further study that is the way the user will provide the system with training examples for building the intention models needed by the agent. Currently, we are evaluating our approach in a concrete application domain, using data collected from the observation of real users.

References

[Armentano, 2008]Armentano, M.G.: Recognition of User Intentions with Variable-Order Markov Models. Ph.D thesis, Universidad Nacional del Centro de la Provincia de Buenos Aires. Argentina (2008)

[Bauer, 1999]Bauer, M.: From interaction data to plan libraries: A clustering approach. In: IJCAI 1999: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pp. 962–967. Morgan Kaufmann Publishers Inc., San Francisco (1999)

[Brown, 1998]Brown, S.: A Decision Theoretic Approach for Interface Agent Development. Ph.D thesis, Faculty of the Graduate School of Engineering of the Air Force Institute of Technology Air University (1998)

[Charniak and Goldman, 1993]Charniak, E., Goldman, R.P.: A bayesian model of plan recognition. Artificial Intelligence 64(1), 53–79 (1993)

[Duong et al., 2006]Duong, T.V., Phung, D.Q., Bui, H.H., Venkatesh, S.: Human behavior recognition with generic exponential family duration modeling in the hidden semi-markov model. In: International Conference on Pattern Recognition, vol. 3, pp. 202–207 (2006)

[Garland; and Lesh, 2002]Garland, A., Lesh, N.: Learning hierarchical task models by demonstration. Technical report, Mitsubishi Electric Research Laboratories (2002)

[Gorniak and Poole, 2000]Gorniak, P., Poole, D.: Building a stochastic dynamic model of application. In: Boutilier, C., Goldszmidt, M. (eds.) Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI 2000), Stanford University, Stanford, California, USA, pp. 230–237. Morgan Kaufmann, San Francisco (2000)

[Horvitz et al., 1998]Horvitz, E., Breese, J., Heckerman, D., Hovel, D., Rommelse, K.: The Lumière project: Bayesian user modeling for inferring the goals and needs of software users. In: Cooper, G.F., Moral, S. (eds.) Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 256–265. Morgan Kaufmann, San Mateo (1998)

[Hunter, 1986]Hunter, J.S.: The exponentially weighted moving average. Journal of Quality Technology 18(4), 203–209 (1986)

[Kautz, 1991]Kautz, H.: A formal theory of plan recognition and its implementation. In: Allen, J.F., Kautz, H.A., Pelavin, R., Tenenberg, J. (eds.) Reasoning About Plans, pp. 69–125. Morgan Kaufmann Publishers, San Mateo (1991)

[Lesh et al., 1999]Lesh, N., Rich, C., Sidner, A.L.: Using plan recognition in human-computer collaboration. In: International Conference on User Modeling (UM 1999), pp. 23–32. Mitsubishi Electric Research Laboratories (1999)

[Liao et al., 2007]Liao, L., Patterson, D.J., Fox, D., Kautz, H.A.: Learning and inferring transportation routines. Artificial Intelligence 171(5-6), 311–331 (2007)

[Lieberman, 2001]Lieberman, H.: Your Wish Is My Command: Programming by Example. Morgan Kaufmann, San Francisco (2001)

[Maes, 1994]Maes, P.: Agents that reduce work and information overload. Communications of the ACM (1994)

[Nguyen et al., 2005]Nguyen, N.T., Phung, D.Q., Venkatesh, S., Bui, H.H.: Learning and detecting activities from movement trajectories using the hierarchical hidden markov model. In: IEEE Computer Vision and Pattern Recognition or CVPR, pp. 955–960. IEEE Computer Society, Los Alamitos (2005)

[Ron et al., 1996]Ron, D., Singer, Y., Tishby, N.: The power of amnesia: Learning probabilistic automata with variable memory length. Machine Learning 25(2-3), 117–149 (1996)

[Whitworth, 2005]Whitworth, B.: Polite computing. Behaviour and Information Technology 24(5), 353–363 (2005)

[Waern, 1996]Wærn, A.: Recognizing Human Plans: Issues for Plan Recognition in Human-Computer Interaction. Ph.D thesis, Royal Institute of Technology (1996)