# Goal Recognition with Variable-Order Markov Models

**Marcelo G. Armentano and Analía Amandi**
ISISTAN Research Institute, Fac. Cs. Exactas, UNCPBA
Campus Universitario, Paraje Arroyo Seco, Tandil, 7000, Argentina
CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina
{marmenta, amandi}@exa.unicen.edu.ar

## Abstract

The recognition of the goal a user is pursing when interacting with a software application is a crucial task for an interface agent as it serves as a context for making opportune interventions to provide assistance to the user. The prediction of the user goal must be fast and a goal recognizer must be able to make early predictions with few observations of the user actions. In this work we propose an approach to automatically build an intention model from a plan corpus using Variable Order Markov models. We claim that following our approach, an interface agent will be capable of accurately ranking the most probable user goals in a time linear to the number of goals modeled.

## 1 Introduction

Interface Agents [Maes, 1994] are an assistive technology that emerged to provide proactive and reactive assistance to human users in their computer-based tasks in a personalized manner. To accomplish their task of assisting users, interface agents are designed to learn the interests, preferences, priorities and needs of the following the metaphor of a personal assistant. However, 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 the user current goal is before initiating an interaction with him or her. Considering the status of the user's attention (i.e. the goal he 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]. For these reasons, it is desirable to build agents capable of detecting as soon as possible the user's goal so that it can predict opportune moments for gaining the user's attention.

A correct and early detection of the user's goal 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 requirements always in the realm of his current intention. For example, if the agent observes that the user is scheduling a work meeting for the following day, the agent can offer to automatically complete the information required and to send an email to each participant of the meeting, providing that it knows the user's preferences about the kind of meeting he is scheduling.

With this purpose, plan recognition aims at identifying the goal of a subject based on the actions he or she performs in an environment. The goal usually has one or more associated plans that can predict the user's subsequent behavior. Inputs to a plan recognizer are generally a set of goals the agent expects the user to carry out in the domain, a set of plans describing the way in which the user can reach each goal, and an action observed by the agent. The plan recognition process itself, consists in foretelling the user's goal, and determining how the observed action contributes to reach it.

Goal recognition is a special case of plan recognition in which only the goal is recognized. Goal recognition is used in domains in which it is better a fast detection of just the user goal than a more precise but time consuming detection of the complete user plan.

In this work we tackle the problem of automatically acquiring a model of a user's goals and the posterior detection of the current user's goal by means of variable order Markov models. The algorithm for learning such models is based on the observation of the actions a user performs on a software application and the goal that motivates the execution of those actions. The learning algorithm will build a model of the actions the user performs to achieve each goal. This model will then enable an interface agent both to detect the intention the user has in any given moment, and the actions or sequence of actions that he will probably perform next to achieve his goal. The agent can use this information as a context for future assistance, not to bother the user with interventions that are not related to his main goal[1].

## 2 Background and Related Work

Plan and goal recognition systems can be roughly grouped in two main categories: Consistency and Probabilistic approaches.

[1] How the information about current user goal is used by the interface agent is out of the scope of this work.

Consistency approaches face the problem by determining which of an input set of goals is consistent with the observed tasks. A goal *G* is consistent with an observed task sequence *A* if *A* might have been executed in service of *G*. Kautz [Kautz, 1991] provided the first formal theory of plan recognition in which plans and goals are represented as an event hierarchy which describes all the behavior that the user might exhibit in some particular domain. Every observed action is part of one or more top-level plans, and the plan recognition task is to minimize the set of top-level plans sufficient to explain the observed actions. The plan recognizer presented in [Lesh, 1998] uses a consistency graph that represents the relations between the actions and possible goals of the domain. The system iteratively applies pruning rules which remove from the graph the goals that are not in any consistent plan, given the observed actions. COLLAGEN [Rich et al., 2001], on the other hand, uses a set of tasks distinguishing between primitive and high level tasks. Then, it uses recipes to break down non primitive tasks in sub-goals. Recipes are presented as functions that map a task to a plan to perform that task.

Probabilistic approaches to plan recognition, on the other hand, mainly make use of Bayesian networks [Charniak and Goldman, 1993], [Horvitz et al., 1998], [Huber and Simpson, 2004] and Markov models [Davison and Hirsh, 1998], [Gorniak and Poole, 2000], [Bui, 2003], [Blaylock and Allen, 2005].

Bayesian networks are a popular representation for reasoning under uncertainty because they combine a graphical representation of causal relationships with a sound Bayesian foundation. The directed acyclic graph structure of the network contains representations of both the conditional dependencies and independences between elements of the problem domain. The knowledge is represented by nodes called random variables and arcs representing the causal relationships between variables. The strengths of the relationships are described using parameters encoded in conditional probability tables (CPTs).

The structure of Markov models, on the other hand, is in the form of a state transition graph. This simple structure is due to its dependence in the Markov assumption to represent sequences of events, which states that the occurrence of the next event depends only on a fixed number of previous events. Given some previously observed events, the next event is predicted from the probability distribution of events that followed those observed in the past.

Both kinds of approaches (consistency and probabilistic) can lead to accurate predictions providing that the plan library is complete and correct. However, probabilistic approaches can find the most probable intention when the observations so far enable more than one possible intention; consistency approaches cannot select between more than one possible intention and have to wait for a single consistent explanation, since they do not consider a priori likelihood of different plans. There are some problems with Bayesian networks, however, that complicate the task of plan recognition. These problems include that the result of the inference process is not sensitive to the order in which evidence is entered in the network, that cycles are not allowed by definition and that learning the structure and parameters of the network from data is a very complex and time consuming task.

Regarding the learning of the plan libraries, most of the previous approaches to the problem centered they efforts in the recognition process itself based on a predefined hand-coded plan library [Kautz, 1991] [Charniak and Goldman, 1993] [Horvitz et al., 1998] [Lesh, 1998] [Goldman et al., 1999]. However, the success of a plan recognizer firstly relies on the correctness and completeness of the plan library. For this reason, in the recent years researchers have put special attention in the acquisition of plan libraries by learning regularities in the user behavior. Nevertheless, most of this research was conducted to learning the parameters of the model, such as probabilities, while the structure of the model itself remained fixed [Oliver et al., 2002] [Bui, 2003] [Liao et al., 2007] [Philipose, et al. 2004] [Nguyen et al., 2005] [Duong, et al. 2006]. 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 [Davison and Hirsh, 1998] [Bauer, 1999] [Gorniak and Poole 2000] [Garland et al., 2001].

## 2 Variable Order Markov Intention Model

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. 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 a fixed number of previous staes, *m*. 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 grown in the number of states of Markov chains as their order is increased. Second, for sequences of actions performed by a user to achieve a given goal, the probability of the next performed action is not always determined by the same fixed number of previous actions. There is usually a variable length previous context that determines the probability distribution of what the user may perform next.

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 leading to a great reduction in the number of model parameters [Rissanen, 1983].

Algorithms for learning VOM models over a finite alphabet Σ attempt to learn a probabilistic finite state automaton (PFA) which can model sequential data of considerable complexity. In contrast to M-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, VOM models provide the means for capturing both large and small order Markov dependencies based on the observed data.

Ron et al. introduced an algorithm for learning VOM models from data [Ron et al., 1996] and Armentano [Armentano, 2008] extended this algorithm to work incrementally as new data is available. This model is described using a subclass of PFA, which they called Probabilistic Suffix Automata (PSA). For the construction of the PSA, a construction called Prediction Suffix Tree (PST) is used. PST preserves the minimal sub-sequences of variable length that are necessary for precise modeling of the given statistical source.

Transition probabilities are computed as follows. Let $e^1, e^2, \cdots, e^m$ be the set of $m$ training examples over the alphabet Σ. The length of the i-th training example is given by $l_i$, that is $e^i = e_1^i.e_2^i, \cdots, e_{l_i}^i \; \forall e_j^i \in \Sigma$. The empirical probability of a sequence $s$ of length $l$ is computed as shown in Equation 1.

$$\widetilde{P}(s) = \frac{\sum_{i,j} \chi_s^{i,j}}{\sum_{i \text{ s.t. } l_i \geq |s|} (l_i - (|s|-1))} \tag{1}$$

$$where\ \chi_s^{i,j} = \begin{cases} 1 & if\ s_1, s_2, \cdots, s_l = e_j^i, e_{j+1}^i, \cdots, e_{j+(l_i-1)}^i \\ 0 & otherwise \end{cases}$$

The numerator is the number of times the sequence $s$ was observed in the sample and the denominator is an estimation of the maximal number of possible overlapping occurrences a pattern of the same length could have had.

The conditional empirical probability of observing an action σ right after a given sequence $s$ is given by Equation 2.

$$\tilde{P}(\sigma|s) = \frac{\tilde{P}(s \cdot \sigma)}{\tilde{P}(s)} \tag{2}$$

The training algorithm [Armentano, 2008] involves building a suffix tree where each node is labeled with a string up to a predetermined length $L$. It keeps track of the number of times each symbol σ is observed in each context $s$. Transition probabilities are then computed using Equation 2 and a pruning procedure is applied to build a prediction suffix tree. The pruning procedure eliminates those nodes with similar prediction capabilities to other nodes corresponding to shorter contexts. It also eliminates nodes corresponding to rarely observed sub-sequences and nodes that do not predict any action with a significant probability value. All these pruning schemes are controlled by a set of parameters of the learning algorithm. Finally, a smoothing technique is applied to face the fact that an action that was not observed in a given context in the training examples might be observed when the PST is used for prediction. If $\gamma_{min}$ is the minimum probability we allow the PST to assign to any action in a given context (which corresponds to the execution of an unobserved task in the context), the algorithm collects a probability mass of $|\Sigma|\gamma_{min}$ from the observed tasks in the given context and then redistributes it among all the tasks in that context. To prevent negative numbers we must assure that $\gamma_{min} < 1/|\Sigma|$

The PST structure is then converted to an equivalent Probabilistic Suffix Automata that is able to assign a probabilistic value to a sequence of observed actions in a time linear to the length of the sequence.

## 3 Goal Recognition with a Variable Order Markov Intention Model

Once the agent owns a model of a user's intentions, it should be able to make use of it to recognize the user's intention while he/she interacts with the application. When checking a given sequence $r$ against a PSA, we initialize the automata in its unique initial state (that corresponding to the empty context) and for each task in the sequence we simply follow the states transitions and compute the probability of the sequence by multiplying the probability of the corresponding task at each state. This process is linear to the length of the sequence.

To perform goal recognition, the agent will have a PSA model for each goal. 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. The goal recognition process itself will consist in classifying any given sequence of tasks performed by the user into one of the possible user goals that is into one of the PSA models. For doing so, as the user performs tasks in the application the conventional use of PSAs (or PSTs) for classification will make the corresponding state transitions and compute the probability that each PSA $k$ generated the given sequence of tasks as shown in Equation 3, where $\gamma(s_{i-1}, \sigma_i)$ indicates the transition probability in state $s_{i-1}$ for the observation $\sigma_i$ ($s_0$ is the state corresponding to the empty sequence)

$$PSA_k(r = \sigma_1, \cdots, \sigma_{|r|}) = \prod_{i=1}^{|r|} \gamma(s_{i-1}, \sigma_i) \tag{3}$$

Then, the goal recognizer would 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 are interested in giving more importance to recent observations than to older observations in a way that we can better detect the underlying trend in the process.

Furthermore, the problem we are facing is not a classical problem of classification as we do not predict a "class" (goal) after observing a complete sequence of performed actions. In our domain, the interface agent should be able to predict the most probable goal after each performed action, and the limit between sequences of actions corresponding to different goals is often fuzzy.

To tackle this problem we choose to 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. This is done by applying weighting factors which decrease 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. λ may be expressed as a percentage, so a smoothing factor of 10% is equivalent to λ=0.1. Alternatively, λ may be expressed in terms of N time periods, where $\lambda = 2/N + 1$.

$EMA_t$ expresses the value of the EMA at any time period $t$. $EMA_1$ is set to the a priori probability of the first observed action $\sigma_1$. Then, the computation of the EMA at time periods $t \geq 2$ is done according to Equation 4.

$$EMA_t = \lambda\gamma(s_{t-1}, \sigma_t) + (1 - \lambda)EMA_{t-1} \quad (4)$$

The parameter λ determines the rate at which older probabilities enter into the calculation of the EMA statistic. A value of λ=1.0 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 its value should be determined empirically.

After each observation, the goal recognizer computes the EMA value for each PSA model and builds a ranking corresponding to the most probable goals the user may be pursuing at each moment. Then, if the most probable goal has an EMA value over a given confidence threshold τ it makes a prediction. The goal recognizer is also able to make predictions of the N-best goals in the ranking, instead of only the most probable goal. This is useful to make further analysis on this reduced set of the N most probable goals.

# 4 Experimental evaluation

## 4.1 The plan corpus

Plan corpus is the term used to describe a set of plan sessions and consists of a list of goals and the actions a user executed to achieve them. Although many corpora is available for testing Machine Learning algorithms in other domains, corpora for training plan recognizers are hard to obtain. There are a few plan recognizers [Lesh, 1998; Bauer, 1999; Garland and Lesh, 2002; Stumpf *et al.*, 2005; Blaylock and Allen, 2005] that make use of corpora to build the plan library and a few others for which the sequences of actions collected are not labeled with the user goal [Davison and Hirsh, 1998; Gorniak and Poole, 2000] which are not used for plan recognition but for next action/command prediction. Moreover, each of the plan recognizers using plan corpora to learn the plan library uses its own data making comparison between them difficult.

For our experiments we selected to use the Linux Plan Corpus kindy provided by Nate Blaylock. This plan corpus [Blaylock and Allen, 2005] is modeled after Lesh's Unix Plan Corpus [Lesh, 1998] and was collected from 56 human Linux users at the University of Rochester's Department of Computer Science. Users involved volunteer students, faculty and staff with different levels of expertise in the use of Linux, categorized from 1 (lowest) to 5 (highest).

Each user was given English descriptions of a set of 19 goals and was instructed to solve them using any Linux commands with some restrictions, such as not using pipes or some special commands which simplifies achieving the desired goal. Users were given the possibility of performing more than one session for each goal.

All sessions, consisting in the sequence of commands performed by a user to achieve a given goal, were automatically recorded. At the end of the session the user was asked to indicate whether he/she succeeded or not in achieving the goal. Other information was also recorded in the session, such as the time the session was initiated, the directory structure and the system response after each command, among others. For more details about how the data was collected, refer to [Blaylock and Allen, 2005].

The first step in our experiments was to pre-process the raw user sessions. From the data recorded for each user session, we were only interested in the user goal and in the sequence of commands he/she performed. We automatically removed commands with typos from each session, and sessions consisting in only 1 command. Attributes, flags and parameters for the rest of the commands were removed so that we only kept the name of the command as action schemas. There is an exception for two commands, *ls* and *find*, for which some flags change the command functionality. For these commands we create more than one action schemas as detailed below.

The command *find*, which searches the directory tree rooted at each given file name by evaluating a given expression, was split in four action schemas: *find-ctime*, representing the command *``find [path...] -ctime n''* which searches for files whose status was last changed *n\*24* hours ago; *find-name*, representing the command *``find [path...] -name pattern''* which searches for files whose name matches the given pattern; *find-size*, representing the use of the command *``find [path...] -size s''* which searches for files using *s* units of storage space; and the action schema *find* grouping all other uses of the command.

In a similar fashion, command *ls*, which lists information about files in the current directory by default, was split in

two action schemas: *ls-R*, representing the command ``*ls -R*'', which lists information about files in the base directory and recursively in its subdirectories; and *ls* grouping all other uses of the command.

After preprocessing, we got 19 goal schemas and 48 action schemas resulting from the plan corpus.

### 4.2 Evaluation metrics

In the experiments shown in this section we evaluate three different metrics. The *error* for a model $Q$ given an observed task sequence $Seq=a_1, \ldots, a_n$ is computed as the sum of the absolute differences between the value assigned by the model after observing each task, $Q(a)$, and the highest value assigned by any of the PSAs after observing that task, $Q_{best}(a)$, as shown in Equation 5.

$$error_Q(Seq = a_1, a_2, \ldots, a_n) = \frac{\sum_{i=1}^{n} |Q(a_i) - Q_{best}(a_i)|}{\sum_{i=1}^{n} Q_{best}(a_i)} \quad (5)$$

On the other hand, *precision* measures the number of times a model $Q$ was in the N-best predicted models, $best_Q(a_i)$, divided the number of predictions made, $m$, as shown in Equation 6. We consider that the goal recognizer makes a prediction every time the higher value assigned by any PSA after some observations is over a threshold τ

$$precision_Q(Seq = a_1, a_2, \ldots, a_n) = \frac{\sum_{i=1}^{n} best_Q(a_i)}{m} \quad (6)$$

On the other hand, *convergence* is a metric that indicates how much time the recognizer took to converge in what the current user goal was. If from the time step $t$ to the time step corresponding to the last performed action for the current goal the algorithm predicted correctly the actual user goal, the convergence is computed as shown in Equation 7. The time step $t$ is called convergence point.

$$convergence_Q(Seq = a_1, a_2, \ldots, a_n) = \frac{m - t + 1}{m}, \quad (7)$$

$$with\ \ not_best_Q(a_{t-1})\ and\ best_Q(a_j)\ \forall j\ \ t \le j \le n$$

$$where\ best_Q(a_i) = \begin{cases} 1 & if\ Q(a_i) = q_{best}(a_i) \\ 0 & otherwise \end{cases}$$

### 4.3 Goal schema recognition

For our experiments, we trained 19 different PSA models, one for each goal schema in the plan corpus, using the sequences of action schemas performed by the different users that took part in the experiments. Next, we performed leave-one-out cross validation to evaluate the performance of our goal recognizer.

We evaluate different values of the smoothing factor λ varying from 0.1 to 1.0 with intervals of 0.1. We also evaluate different thresholds τ to make predictions.

For all the experiments, a value of λ=0.3 and τ=0.2 led to better results. Table 1 shows the recognition results for the Linux corpus.

We obtained an error of 1.15%. Notice that the error metric is independent of the number of models we consider for prediction because this metric measures the *distance* in the prediction of the current user goal from the model that is predicted with *highest* probability by the plan recognizer.

| N-best | 1-best | 2-best | 3-best |
|---|---|---|---|
| **Error** | 0.115 | 0.115 | 0.115 |
| **Precision** | 0.646 | 0.765 | 0.808 |
| **Convergence** | 0.589 | 0.706 | 0.745 |

**Table 1**: Goal schema recognition results

On the other hand, we obtained a precision of 64.6% that is increased to 80.8% for the case of 3-best prediction, with a convergence of 58.9% for 1-best prediction. Convergence can be increased to 74.5% by considering 3-best prediction.

These values improve results presented in Blaylock research [Blaylock and Allen, 2005] for a bigram model of the user goals. By using variable order Markov models with exponential moving average, we got an increment of 21.5% better convergence for 1-best prediction, 14.1% for 2-best prediction and 14.8% for 3-best prediction. Precision on the other hand was improved by 26.8% for 1-best prediction, 12.4% for 2-best prediction and 7.1% for 3-best prediction. Since our goal recognizer has the same complexity of O(|G|) than Blaylock's goal recognizer, where G is the set of goal schemas in the domain, we believe that our improvements are significant.

## 6 Conclusions and future work

We have presented a goal schema recognition approach based on variable order Markov models to extend existent approaches that use fixed order Markov models (especially unigrams and bigrams models). We also make use of a smoothing technique, namely an exponential moving average, to better detect the underlying trend in the predictions.

Our plan recognizer is able to make partial predictions after each observed task and takes a linear time in the number of goals modeled.

There are several areas of future work that we are exploring such as hierarchical goal recognition and handling parameterized goals and actions. A simple solution might be to extend the number of actions creating one action for every pair <action schema, parameter>. However, this approach produces an explosion in the number of actions besides data sparseness, because the likelihood of observing a symbol in a given context will be very low.

Finally, the initial results presented in this work are based on a relatively small dataset for one particular domain. Exploration of other domains with different characteristics will be required to ensure that the performance of the goal recognizer is consistent with the results we obtained in this work.

## References

[Armentano, 2008] Armentano, M. G. Recognition of User Intentions with Variable-Order Markov Models. PhD

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 '99: Proceedings of the 6th International Joint Conference on Artificial Intelligence*, pages 962-967, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 1999.

[Blaylock and Allen, 2005] Blaylock, N. and Allen, J. Recognizing instantiated goals using statistical methods. In Gal Kaminka, editor, *IJCAI Workshop on Modeling Others from Observations (MOO-2005)*, pages 79-86, Edinburgh. 2005.

[Bui, 2003] Bui, H. H. A general model for online probabilistic plan recognition. In *Gottlob, G. and Walsh, T., editors, IJCAI 03, Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 1309-1318, Acapulco, Mexico. Morgan Kaufmann. 2003.

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

[Davison and Hirsh, 1998] Davison, B. D. and Hirsh, H. Predicting sequences of user actions. In *Predicting the Future: AI Approaches to Time Series*. AAAI Press. 1998.

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

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

[Goldman et al., 1999] Goldman, R., Geib, C., and Miller, C. Learning hierarchical task models by defining and redefining examples. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 245-254, San Francisco, CA. Morgan Kaufmann. 1999.

[Gorniak and Poole, 2000] Gorniak, P. and Poole, D. Building a stochastic dynamic model of application. In *Boutilier, C. and Goldszmidt, M., editors, 6th Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pages 230-237, Stanford University, Stanford, California, USA. Morgan Kaufmann. 2000.

[Horvitz et al., 1998] Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. The Lumière project: Bayesian user modeling for inferring the goals and needs of software users. In *Cooper, G. F. and Moral, S., editors, Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 256-265, San Mateo. Morgan Kaufmann. 1998.

[Huber and Simpson, 2004] Huber, M. and Simpson, R. Recognizing the plans of screen reader users. In *Modeling Other Agents from Observations (MOO2004)*. Workshop W3 at the International Joint Conference on Autonomous Agents and Multi-Agent Systems, Columbia University, NY, USA. 2004.

[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., and Tenenberg, J., editors, *Reasoning About Plans*, pages 69-125. Morgan Kaufmann Publishers, San Mateo (CA), USA. 1991.

[Lesh, 1998] Lesh, N. (1998). Scalable and Adaptive Goal Recognition. *PhD thesis, University of Washington*. 1998.

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

[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., and 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*, pages 955-960. IEEE Computer Society. 2005.

[Oliver et al., 2002] Oliver, N., Horvitz, E., and Garg, A. Layered representations for human activity recognition. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces (ICMI 2002)*, pages 3-8. IEEE Computer Society. 2002.

[Philipose et al., 2004] Philipose, M., Fishkin, K. P., Perkowitz, M., Patterson, D. J., Fox, D., Kautz, H., and Hahnel, D. Inferring activities from interactions with objects. *Pervasive Computing Magazine*, 3(4) pages 10-17. 2004.

[Rich et al., 2001] Rich, C., Sidner, C. L., and Lesh, N. (2001). COLLAGEN: Applying collaborative discourse theory to human-computer interaction. *AI Magazine*, 22(4) pages 15-26. 2001.

[Rissanen, 1983] Rissanen, J. "A Universal Data Compression System". *IEEE Transactions on Information Theory* 29 (5): 656–664. 1983.

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

[Stumpf *et al*., 2005] Stumpf, S., Bao, X., Dragunov, A., Dietterich, T. G., Herlocker, J., Johnsrude, K., Li, L., and Shen, J. Predicting user tasks: I know what you're doing! In *20th National Conference on Artificial Intelligence (AAAI-05), Workshop on Human Comprehensible Machine Learning*. 2005.

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