Attribute-driven Hidden Markov Model Trees for Intention Prediction

LIAT ANTWARG, LIOR ROKACH, BRACHA SHAPIRA
Department of Information Systems Engineering, Ben-Gurion University of the Negev
and T-Labs @ BGU, Beer Sheva, 84105, Israel
{liatant,bshapira,liorrk}@bgu.ac.il

In this paper we introduce a novel approach for generating an intention prediction model of user interactions with systems. As part of this new approach, we include personal aspects such as user characteristics that can increase prediction accuracy. The model is automatically trained according to the user's fixed attributes (e.g., demographic data such as age and gender) and the user's sequences of actions in the system. The generated model has a tree structure. The building blocks of each node can be any probabilistic sequence model (such as hidden Markov models and conditional random fields)) and each node is split according to user attributes. Thus, we refer to this algorithm as an attribute-driven model tree. The new model was first tested on simulated data in which users with different attributes (such as age, gender) behave differently when trying to accomplish various tasks. We then validated the ability of the algorithm to discover the relevant attributes. We tested our algorithm on two real datasets: from a Web application and a mobile application dataset. The results were encouraging and indicate the capability of the proposed method for discovering the correct user intention model and increasing intention prediction accuracy compared to single HMM or CRF models.

Key Words and Phrases: Intention prediction, sequence learning, Hidden Markov model

1. INTRODUCTION

Predicting user intentions when using a system can improve the services provided to users by adjusting the system services to their needs. The goal is to identify what the user wants to accomplish when performing a certain sequence of activities. This goal is achieved by training a specific learning algorithm in order to generate models that will maximize the accuracy of the prediction. The challenge in predicting the intention of the user given a sequence of interactions can be categorized as a problem with a sequential problem space which is commonly solved using sequence learning algorithms [Sun & Giles, 2001].

The motivation of this paper is to achieve a higher level of compatibility between user intentions and the system she is using. This motivation derives from the fact that in recent years applications have continuously been gaining functionality and emerging features are complicating user interactions with applications. Using intention prediction methods will enable a company to improve the services provided to users by adjusting the system services to their needs (e.g., the interface). Predicting user intentions can be also used for assisting users to perform or complete tasks, to alert them regarding the availability of features, and to increase their general knowledge. Effective assistance will allow users to acquire the new skills and knowledge they need to easily operate an unfamiliar device or function.

Different users tend to select different sequences for accomplishing the same goal. Specifically, our hypothesis is that the user's attributes and context (such as age or the operating system) indicate which sequence the user will eventually perform. This hypothesis is based on studies that examined the interaction of users with systems (such as Web browsers) and found that user attributes affect the way of interaction. [For example, Hu et al., 2007, Weber and Castillo 2010; Thakor et al., 2004]. If a young male uses a system, it is possible to use the model to predict the goal that he intends to accomplish and provide him with a user interface (UI) more appropriate to his usage and intentions.

Although methods exist for predicting user intentions, they do not take into account user attributes that can increase the accuracy of prediction. In this work we present a new method that we developed that examines how employing user attributes can improve the accuracy of intention prediction. The new method is termed an attribute-driven hidden Markov model tree (ADHHMT) and it builds a separate tree of hidden Markov models (HMMs) [Rabiner, 1989] for each goal in the system. The tree is branched according to the user's attributes and each node in the tree consists of a single HMM. When the user performs a new task, we first employ her attributes to find the suitable node in each of the goal trees. Then, based on the actions the user has already performed in implementing this task, we anticipate the goal she is trying to accomplish and the action she intends to perform next.

The reason for using hidden Markov models is that the sequence of actions that users perform is not always observable but can only be observed through another set of stochastic processes that produce the sequence of observations [Rabiner, 1989]. For example, in a mobile device the sequence of buttons that the user pressed is unknown (hidden) but the sequence of screens that was created from using these buttons is known (observable). Thus, we can learn the sequence of buttons from the sequence of screens using HMM.

The use of user attributes stems from personalization-related studies. It was shown that utilizing user demographic attributes for personalization and recommendation of relevant items to users increases the accuracy of recommendations since these attributes influence users preferences, and interaction with systems [Zanker et al., 2007; Weber and Castillo, 2010; Vozalis et al., 2006]. One study [Hu, et al., 2007] was able to predict users attribute based on their browsing behavior. Thus, we utilize user attributes to predict user intentions assuming that since user attributes relate to user behavior, they naturally relate to the intentions preceding that behavior.

The contributions of this paper are threefold:

a. Developing a new model for intention prediction that encapsulates both the user's fixed attributes (e.g. demographic data) and her sequence of actions. Each of the system's goals is represented by a tree model, whose branching is performed according to the user's fixed attributes. Moreover, each node in the tree contains a HMM that can be used to predict user intentions. These comprehensive tree models can be used for segmenting users according to their usage behavior. This segmentation can be also used for other purposes, such as marketing.

b. It presents an algorithm for inducing the models from given training data. Inspired by previous works on decision trees, we developed our own top-down algorithm for creating tree models using a new splitting criterion and pruning procedure.

c. Evaluating the proposed method on real-world case-studies.

The rest of this paper is organized as follows: In section 2 we review related works on intention prediction and HMM. Section 3 formulates the problem. Section 4 presents the new algorithm for improving the accuracy of intention prediction. Section 5 reports the experiments carried out to examine the new algorithm and discusses their results. A comparison between performance of a single HMM and the proposed attribute-driven HMM is illustrated by two real-world datasets. The first involves 165 PC-users of a Web application and the second test involves 48 mobile users. Section 6 is a discussion about the proposed algorithm and section 7 concludes and presents suggestions for further research.

2. RELATED WORK

2.1. Intention prediction

Intention prediction has become a very active research area lately. In this section, we summarize real world applications that use intention prediction. Some applications treat the data as a sequence. One such example is the Lumiere project [Horvitz et al., 1998] that leverages reasoning methods using Bayesian models for capturing uncertainty about the goals of software users. The models can be employed to infer user needs by considering the user's background, actions and queries. Horvitz et al. [1998] used Markov representation of the temporal Bayesian user-modeling problem by considering dependencies among variables at adjacent time periods. The Lumiere project was first used in Microsoft's Office '97 product suite, containing the Office Assistant. Chen et al [2002] argue that although Office Assistance is probably the only one method, which has extensively studied user intention, its predictions can be further improved if semantic contexts are also used in addition to action sequences for mining user intentions.

There are few works about predicting user intentions as they surf the Web. Chen et al. [2002] used a modified naïve Bayes classifier algorithm to support incremental learning for modeling the user’s intended action on a computer. They focused on predicting actions based on the features extracted from the user's interactions such as the user’s typed sentences and viewed content. Their goal was to predict the series of basic actions that the user will be performing in a system to accomplish his intention. Sun et al. [2002] presented a method for predicting the user’s browsing intention based on the Web page sequences she had previously visited. Their method employs a multi-step dynamic n-gram model and predicts the next action that lies on the optimal path that leads to the ultimate goal.

TaskPredictor [Shen et al., 2006] is a machine learning system that attempts to predict the user's current activity. It operates in the Microsoft Windows environment and collects a wide range of events describing the user's computer-visible behavior. As a first step, feature selection, a threshold for making classification decisions and naive Bayes are applied to decide whether to make a prediction. Then, a discriminative, model linear support vector machine (SVM) is applied to make the prediction itself. In their work, they treated task prediction as a traditional supervised learning problem, ignoring the sequential aspects.

Few works attempt to identify the goal of user queries while searching the Web. Rose and Levinson (2004) identified the goal of queries through manual, query-log investigation and concluded that the goals can be divided into categories in a hierarchical structure. Lee et al. (2005) suggested automating the identification of user goals in a Web query since in their study of human subjects, the majority of queries had predictable goals. Baeza-Yates et al. (2006) try to infer the user's search intention by using supervised and unsupervised learning. Jansen et al. (2007) presented an algorithm that aims to automatically understand a user’s intention by classifying queries as navigational, transactional or informational.

Several research projects have focused on intelligent wheelchairs. Taha et al. [2008] present a technique for predicting the wheelchair user’s destination to locations much further away than the immediate surroundings. The system relies on minimal user input obtained from a standard wheelchair joystick and a learned, partially observable Markov decision process (POMDP).

Jung et al. [2003] designed a user intention recognition module for an intelligent bed robot system. Since feature values are in sequence form, they used HMM. Feng et al. [2004] developed a plan recognition-based method to predict the anomalous events and intentions of potential intruders using system call sequences as observation data. An algorithm based on a dynamic Bayesian network with parameter compensation progressively accomplishes the prediction while a recursive process identifies the intruder's hostile intentions. Yudhistira et al., [2006] presented an intelligent television set which can adaptively propose certain shows to the viewer based on previous shows that the user watched. A hidden Markov model (HMM) models the user's preferences. All of the above studies focus on usage behavior and neglected user attributes in predicting intentions.

2.2. HMM (Hidden Markov Models)

There are various methods of sequence learning including neural networks, dynamic Bayesian networks, different Markov models and others. The sequence learning algorithm we selected for our problem is a hidden Markov model (HMM) [Rabiner, 1989] due to its ability to very efficiently calculate the probability of sequences in the given model and the existence of an efficient training algorithm (the Baum-Welch algorithm) [Baum & Egon, 1967]. As far as we know, no research until now has tried to increase prediction accuracy by utilizing user attributes.

HMM is a type of dynamic Bayesian network (DBN) [Friedman et al., 1998, Chen et al., 2009]. It is a stochastic process with an underlying unobservable (hidden) stochastic process that can only be observed through another set of stochastic processes that produce the sequence of observed symbols [Rabiner, 1989]. HMM can be viewed as a specific instance of a state-space model in which the latent variables are discrete. In HMM, the probability distribution of z_n depends on the state of the previous latent variable z_{n-1} through a conditional distribution $p(z_n \mid z_{n-1})$ [Bishop, 2006].

The following characterize a HMM:

N, the number of states in the model. The individual states are denoted as $Z = \{z_1, z_2 ... z_N\}$ and the state at time t as q_t .

M, the number of distinct observation symbols per state. The observation symbols correspond to the physical output of the system being modeled. The symbols are represented as $X = \{x_1, x_2 ... x_M\}$.

The state transition probability distribution $A = \{a_{i,j}\}$ where

$$a_{i,j} = P[q_{t+1} = z_j \mid q_t = z_i] \qquad 1 \le i, j \le N$$

The observation symbol probability distribution in state j, $B = \{b_i(k)\}$ where

$$b_i(k) = P[x_k \; at \quad t \quad \mid q_t = z_j], \begin{matrix} 1 \le j \le N \\ 1 \le k \le M \end{matrix}$$

The initial state distribution $\pi = \{\pi_i\}$ where $\pi = P[q_1 = z_i], \qquad 1 \le i \le N$

2.3. How the attribute-driven HMM tree differs from HHMM and profile HMM.

Since the method we propose may be confused with the well-known, hierarchical hidden Markov model (HHMM) [Fine et al., 1998], we discuss in this section in what sense the proposed HMM tree is different. HHMM utilizes the hierarchical structure of natural sequences and generalizes the standard HMMs by making each of the hidden states a model. In other words, each hidden state in the HHMM can also be an HHMM. Although the attribute-driven HMM tree presented in this paper might be inferred as similar to HHMM, the hierarchical concept that we use is different from HHMM. The hierarchical concept that we propose is the tree itself. Each node in the tree is a flat HMM which uses the same states from the higher level or only some of them but with different transition probabilities. Because the number of available training instances drops as we move down the tree, the number of hidden states should generally drop accordingly. In particular, if the number of observations is very large, then we can choose a large number of hidden states to capture more features (Samaria and Harter, 1994). In this paper the hierarchical structure that forms a tree is used to differentiate between different user behaviors (according to user attributes). Although HMM and HHMM have been used in previous studies for modeling process behavior among users [Galassi et al., 2005, Lane, 1999], the hierarchical concept, as noted above, is very different as Fig. 1 illustrates.

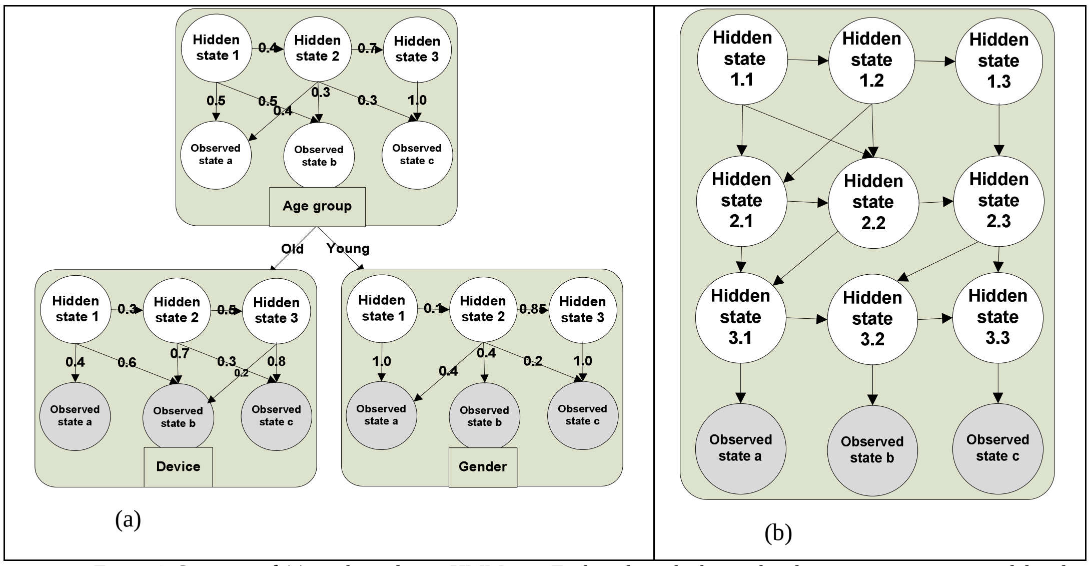


Figure 1. Structure of (a) attribute-driven HMM tree. Each node in the hierarchical structure contains a model and an attribute. All nodes contain the same model but with different transition probabilities. (b) HHMM. The hierarchical structure is inside the states. Each hidden state can be HHMM as well.

A profile HMM is a method for searching a database for other sequences from the same family [Krogh, 1998]. Profile HMMs are applied to such common tasks as simultaneously aligning multiple, related sequences to each other; aligning a new sequence to an already-aligned family of sequences; and evaluating a new sequence for membership in a family of sequences [Bhargava & Kondrak, 2009]. Profile HMM is mainly used for biological sequence analysis. The division of training sequences to goals (profile) that the profile HMM reveals is already known in our proposed method.

3. PROBLEM FORMULATION

In this section several basic definitions are introduced followed by the problem formulation. In a typical sequence learning problem, a training set of sequences $S = \{s_1, s_2, ..., s_m\}$ is given. Each sequence $s_j \in S$ is an ordered set of n_j elements (actions) $\{e_1, e_2, .., e_{n_j}\}$. *G* denotes the set of all possible goals (for example, in an email application G={'Add recipient', 'Send an email', 'Read email'}). Each training sequence is associated with one goal and several characterizing attributes. The notation *U* denotes the set of input attributes containing η attributes: $U = \{v_1, .., v_\eta\}$. The domain (possible values for each attribute) of an attribute is denoted by $dom(v_i)$. User space (the set of all possible users) is defined as a Cartesian product of all the input attribute domains. The input in the problem consists of a set of m records and is denoted as $Train = (< s_1, g_1, u_1 >, ..., < s_m, g_m, u_m >)$ where $s_q \in S, g_q \in G, u_q \in U$.

The notation L represents a probabilistic sequence learning algorithm such as HMM. In the rest of the paper we mainly focus on HMM. But we show in Section 5.6 that the same algorithm can be used with other base models, for example, CRF (conditional random field) as a base model. These algorithms generate models that can estimate the conditional probability of a goal g given a sequence and the input user attributes. Let λ represent a probabilistic model which was induced by activating L onto dataset $Train$. In this case we would like to estimate the conditional probability $p_\lambda(G = g_j \mid s_q, u_q)$ of a sequence s_q that was generated by user u_q. Thus, the aim of intention prediction is to find the most accurate probability estimations.

As indicated in Figure 1a, we are using a tree structure to improve the accuracy of training models. For each goal in the system, a different tree is generated. This structure is built using user attributes in an attempt to differentiate between various usage behaviors in the system. For example, in Figure 1a we are employing the age, gender and device for differentiating between usage behaviors. In this structure, each node contains a model and an attribute v_i that splits the node. Assuming this tree structure, the problem can be formally phrased as follows:

Given a sequence learning algorithm L and a training set $Train$ with input sessions set S, users U and goals G, the aim is to find an optimal set of trees (a tree for each goal). Optimality is defined in terms of minimizing prediction errors.

In this paper we assume that L is an algorithm for training HMMs. Particularly, the model λ in each of the tree's nodes is trained using the Baum-Welch algorithm [Baum & Egon, 1967] and the probability $p_{\lambda}(G = g_j \mid s_q, u_q)$ is estimated using the forward-backward algorithm [Baum & Egon, 1967, Baum & Sell, 1968] as we show below.

4. ATTRIBUTE-DRIVEN HMM TREE - OVERVIEW OF METHODS

In order to solve the problem defined in section 3, we suggest an attribute-driven HMM tree method. Fig. 2 presents the proposed process schematically. The left side in Fig. 2 specifies the generation of the models used for intention prediction based on an attribute-driven HMM tree for each possible goal. The input for this process is a set of user action sequences divided into goals and user attributes. The output of the process is a tree for each goal based on different usage behavior of users in attaining this goal, and user attributes. The right side of the figure presents the prediction process. The inputs for the process are: the goal trees generated during the model generation phase, a specific user session (i.e., a sequence of actions), and the attributes of a user. The output is the predicted goal that the specific user was trying to accomplish in this sequence (i.e., the goal with the highest probability estimation).

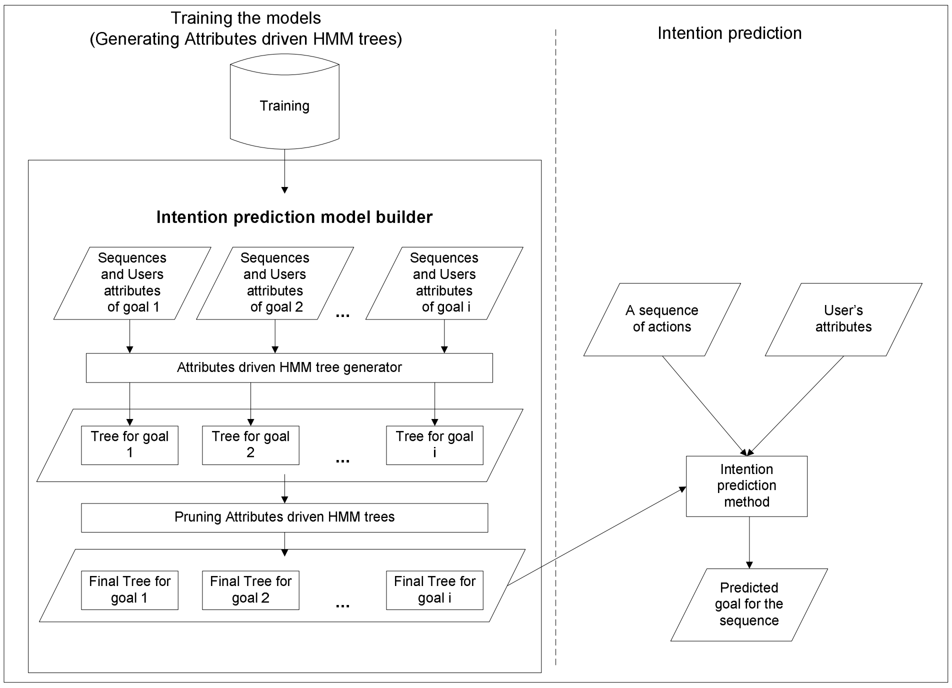

Training the models
(Generating Attributes driven HMM trees)
Training
Intention prediction model builder
Sequences and Users attributes of goal 1
Sequences and Users attributes of goal 2
Sequences and Users attributes of goal i
...
Attributes driven HMM tree generator
Tree for goal 1
Tree for goal 2
Tree for goal i
Pruning Attributes driven HMM trees
Final Tree for goal 1
Final Tree for goal 2
Final Tree for goal i
Intention prediction
A sequence of actions
User's attributes
Intention prediction method
Predicted goal for the sequence

Figure 2. Overall diagram of the proposed attribute-driven HMM tree based method for intention prediction

In the following subsections we present in detail the proposed method followed by a discussion and an example of the intuitions for some of the approaches that were applied.

4.1. Attribute-driven HMM tree

In this section we describe our new intention prediction method for predicting with a high degree of accuracy the user's goal based on her activities. Recall that for predicting the user's intentions we assume that a sequence learning algorithm is given. Specifically, in this paper our tree model wraps the HMM [Rabiner, 1989]. Our algorithm is trained according to the user's fixed attributes (e.g., demographic data) and the user’s sequences of actions in the system in order to increase the prediction accuracy of HMM. The intuition behind using fixed attributes is that user behavior patterns can be differentiated by their attributes, as shown in personalization related studies [Weber and Castillo, 2010], and thus training different models for different attributes may increase prediction accuracy.

Our algorithm is based on the following assumptions:

a. For each application, the administrator pre-defines a set of goals. Each goal can be achieved by one or more sequences of actions. For example, in an email application; the goals might include ‘Send e-mail’ or ‘Add contact’.

b. For each user, the administrator provides a set of attributes and their domain values.

Our method consists of three main phases: (1) training the tree models; (2) pruning the trees and (3) using the trees for intention prediction.

4.1.1. *Training the tree models.* In this phase, the method generates a trained tree for each goal in the system. The following is a description of the process for generating a tree for one goal (the same process is applied for each of the goals). The input to this phase includes:

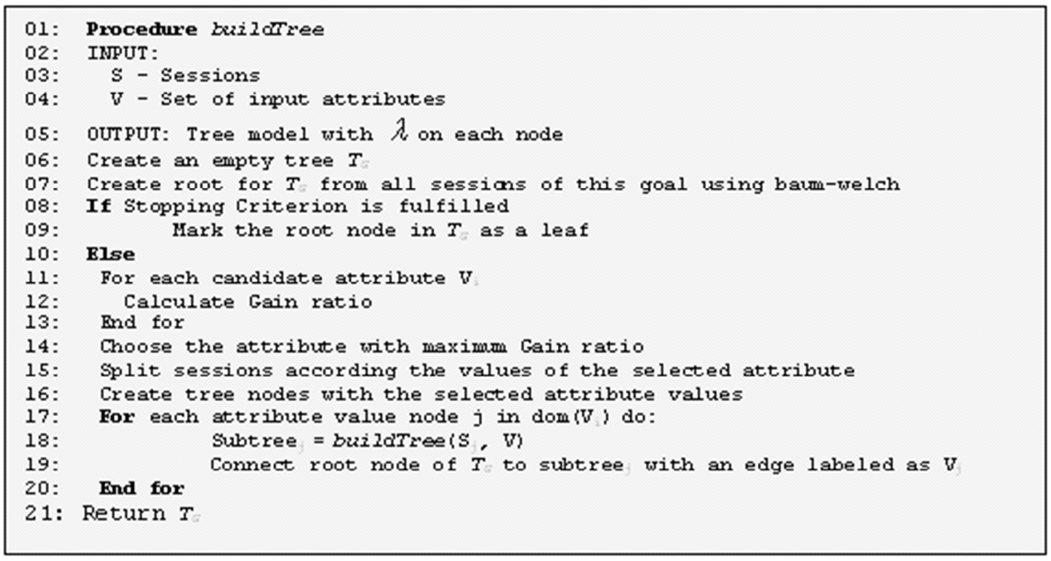
- Sequences of actions from all the users in the system, divided into goals.
- User attributes and their values.

Figure 3 presents the pseudo code for building a single goal tree. In line 07, the root node HMM is trained using all sequences that belong to the current goal. The HMM is trained using the Baum-Welch algorithm [Baum & Egon, 1967]. The output of this step is a model represented by a matrix with transition probabilities between all the elements of this goal for all the users.

The aim of the second step is to branch out the root node into its descendants. The following processes are applied to generate the level of the immediate descendant nodes. They are repeated for each level until a stopping criterion is reached (we explain the stopping criteria in section 4.3). The user attributes are employed to find the most informative child nodes, namely the nodes that provide the most accurate intention prediction (in particular those that are more accurate than the parent node). The branching out step contains two sub-steps:

1. Examine all user attributes - For each possible user attribute, a measure termed gain ratio is calculated in line 12. This measure is used as the criterion for splitting a node in the tree. We discuss the gain ratio below in section 4.2.

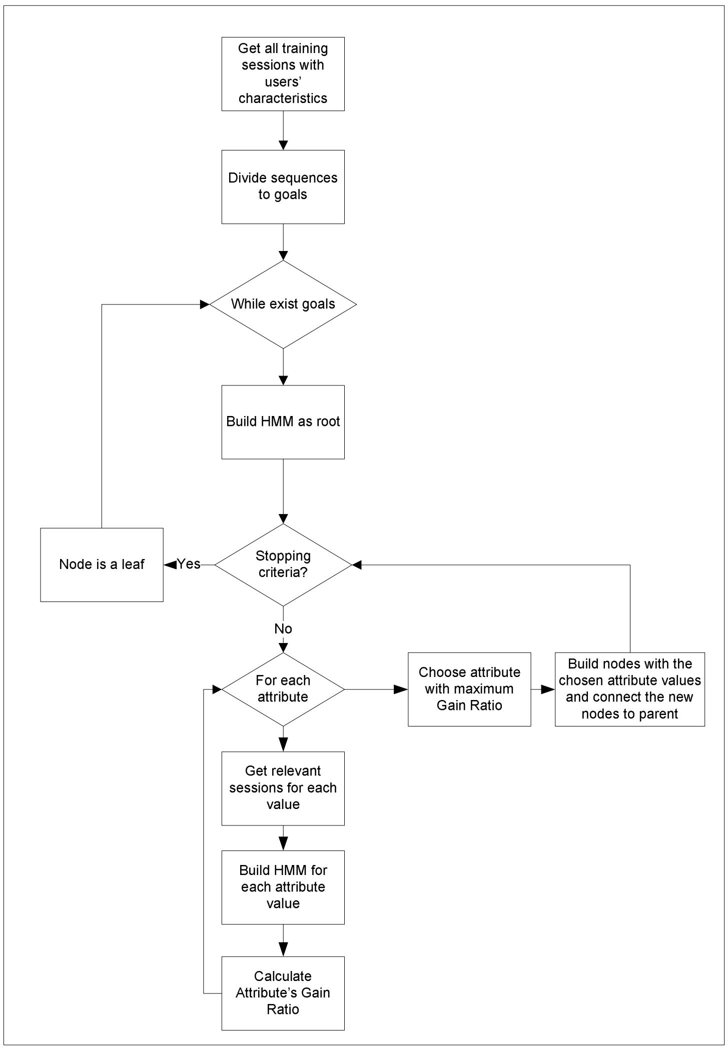
2. Choose attributes for the level and update tree accordingly - The attribute with maximal gain ratio is chosen to split the parent node in line 14. The chosen attribute name is stored in the parent node together with the parent's HMM that was generated earlier. In line 17, a loop is started over each value of the chosen attribute. For each such value, the method 'Build model' (line 18) is called up again to generate HMM models for the child nodes to be trained on the sequences of users with that node's attribute value. The subsequent levels of the tree are generated similarly, until a stopping criterion is reached. For each level, the user's attributes list becomes shorter since attributes that were used in the upper levels cannot be used again. In line 19, the arcs between the parent and the child nodes are updated to contain the attribute value. The output of this step is an attribute-driven HMM tree for each goal.



```
Procedure buildTree
INPUT:
  S - Sessions
  V - Set of input attributes
OUTPUT: Tree model with $\lambda$ on each node
Create an empty tree $T_g$
Create root for $T_g$ from all sessions of this goal using baum-welch
If Stopping Criterion is fulfilled
        Mark the root node in $T_g$ as a leaf
Else
 For each candidate attribute $V_i$
   Calculate Gain ratio
 End for
 Choose the attribute with maximum Gain ratio
 Split sessions according the values of the selected attribute
 Create tree nodes with the selected attribute values
 For each attribute value node j in dom($V_i$) do:
           $Subtree_j$ = buildTree($S_j$, V)
           Connect root node of $T_g$ to $subtree_j$ with an edge labeled as $V_j$
 End for
Return $T_g$
```

Figure 3. Building one goal tree with user attributes

Figure 4 presents the flow of the training phase:



Get all training sessions with users' characteristics
Divide sequences to goals
While exist goals
Build HMM as root
Stopping criteria?
Yes
Node is a leaf
No
For each attribute
Choose attribute with maximum Gain Ratio
Build nodes with the chosen attribute values and connect the new nodes to parent
Get relevant sessions for each value
Build HMM for each attribute value
Calculate Attribute's Gain Ratio

Figure 4. Flow of training an attribute-driven HMM tree

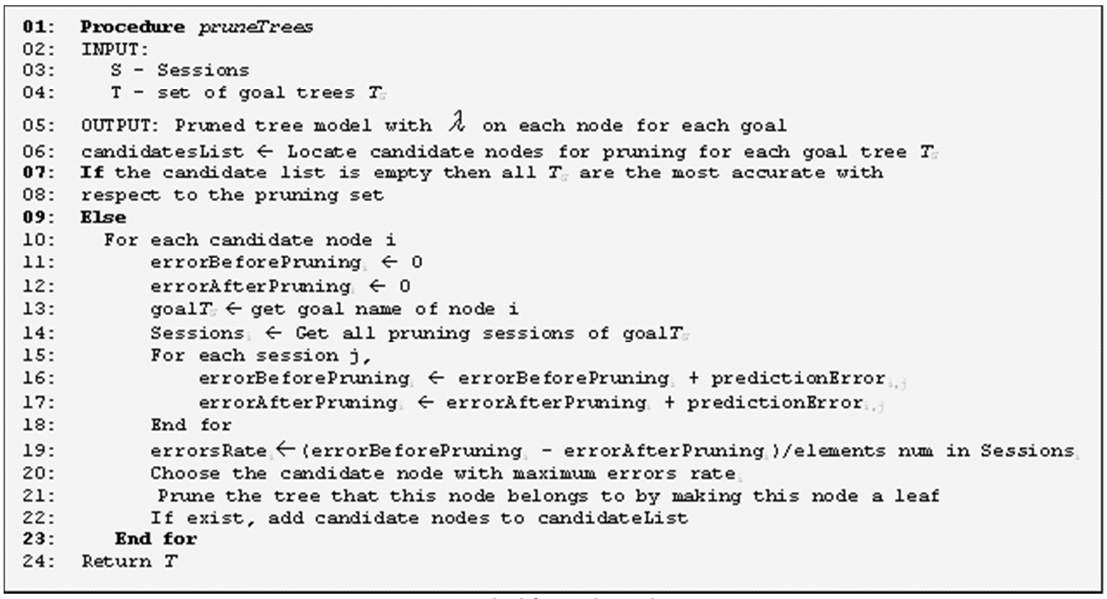
4.1.2. *Pruning the tree.* As in classification trees, counting solely on a stopping criterion for determining the tree size may cause either overfitting or underfitting that may lead to low prediction accuracy. For example if the stopping criterion is too loose, then the tree might overgrow and the HMMs on the lower nodes would be induced using only a few training sequences. This would make the model unreliable.

In classification trees, pruning methods are commonly used to avoid these problems. The idea is to let the tree over-grow and then prune it to the correct size. Inspired by this idea, we developed our own pruning procedure for attribute-driven HMM trees. Specifically we adapted the reduced error pruning method [Quinlan, 1987] to our needs. The intuition behind this method is to find the smallest version of the most accurate subtree with respect to the pruning set. A subtree is pruned if it does not contain any subtree that results in a lower error rate than the tree itself. The process is a bottom-up pruning method that compares the prediction error rate on the pruning tree when a subtree is kept and when the subtree is turned into a leaf and associated with the best leaf. Since pruning methods are usually used in decision trees algorithms, the implementation of a pruning method in an attribute-driven HMM tree requires few adjustments. In most cases, a pruning method is implemented on a single tree. In our algorithm we have a tree for each goal. The prediction accuracy of each goal tree is dependent on the others. Since the pruning process must be implemented for all goal trees simultaneously, we defined an adjusted method for reduced error pruning.

The pseudo code in Fig. 5 presents the pruning method we used. The following is a description of the pruning process for the generated attribute-driven HMM trees.

The aim of the first step is to identify candidate nodes for pruning. A candidate node is one that is not a leaf and all its children are leaves. Candidate nodes are identified from all goal trees. The candidate nodes are located in line 06. Then, we prune the trees according to prediction errors. In line 10, a loop starts over all candidate nodes for pruning. For pruning each candidate node, errors before (line 16) and after pruning (line 17) of this node are counted for this goal tree only. Thus, the test sequences for counting errors for a candidate node for pruning are only sequences of the same goal. In line 19, the difference between the number of errors before and after pruning is calculated for each candidate node. The number of errors is counted by using subsequences of each sequence and comparing the output (i.e., the most probable goal) to the real goal the user was aiming at. In line 20, the candidate node that causes maximal positive difference is pruned and this node becomes a leaf. Then, if there are new candidate nodes for pruning, they are added to the candidate list in line 22. Pruning iterations continue until there are no more nodes that can improve the current state of the trees. After pruning all nodes that do not contribute to prediction accuracy, the output of this step is an attribute-driven tree for each goal.

Unlike the original implementation of reduced error pruning, a subtree is pruned without replacing the root of the subtree with its best leaf. This is done to avoid contradictions with the idea behind our algorithm of creating a model that represents usage behavior of all the users and not only the users with the best attribute value.



```
Procedure pruneTrees
INPUT:
  S - Sessions
  T - set of goal trees $T_G$
OUTPUT: Pruned tree model with $\lambda$ on each node for each goal
candidatesList ← Locate candidate nodes for pruning for each goal tree $T_G$
If the candidate list is empty then all $T_G$ are the most accurate with
respect to the pruning set
Else
  For each candidate node i
      errorBeforePruning$_i$ ← 0
      errorAfterPruning$_i$ ← 0
      goal$T_G$ ← get goal name of node i
      Sessions$_i$ ← Get all pruning sessions of goal$T_G$
      For each session j,
          errorBeforePruning$_i$ ← errorBeforePruning$_i$ + predictionError$_{i,j}$
          errorAfterPruning$_i$ ← errorAfterPruning$_i$ + predictionError$_{i,j}$
      End for
      errorsRate$_i$ ←(errorBeforePruning$_i$ - errorAfterPruning$_i$)/elements num in Sessions$_i$
      Choose the candidate node with maximum errors rate$_i$
      Prune the tree that this node belongs to by making this node a leaf
      If exist, add candidate nodes to candidateList
    End for
Return T
```

Figure 5. Pruning method for attributes driven HMM trees

At the end of the training and pruning processes, attribute-driven HMM trees are generated for each goal in the system. As mentioned above, each node in the tree contains a HMM with transition probabilities between actions and an attribute for splitting. Each arc contains an attribute value of its parent's attribute. Each leaf in a tree holds a set of attribute-values. We term this set as user type. Thus, the leaves in each tree represent user types for each goal. These user types can be later used for other purposes such as personalization or marketing.

4.1.3. *Intention prediction.* In this phase we aim to predict the user's goal in using a system. To realize the goal, the user implements a sequence of actions, in a user session, where she intends to complete some goal. The input to this phase includes:

- Sequence of user actions
- User attributes and their values
- Goal trees (generated in the previous phase (section 4.1.2))

The result of the learning stage is a tree for each goal. Each node in the tree contains an HMM model λ with transition probabilities between actions and an attribute for splitting. Each arc contains an attribute value of its parent's attribute. When a user starts interacting with the system, she forms a session. In the beginning, the session contains only one element (one user's action), then two elements etc. The pseudo code in Fig. 6 presents the process for predicting the user's intentions. This process consists of two steps:

Locating relevant HMM for the user - In order to predict which goal the user intends to achieve, we inspect each goal tree T_G. The loop over the goal trees starts at line 07. In line 08, the hidden Markov model λ (transition matrix) in the relevant leaf (relevant user type) that fits the user's attributes is located in each goal tree.

Calculating the likelihood of user's sessions using all relevant HMMs - In line 09, given the relevant model in each goal tree, the likelihood of this session is calculated with the forward-backward algorithm [Baum & Egon, 1967, Baum & Sell, 1968]:

$$P(s \mid u, g_j) = P_{\lambda_{g_j,u}}(s) \qquad (1)$$

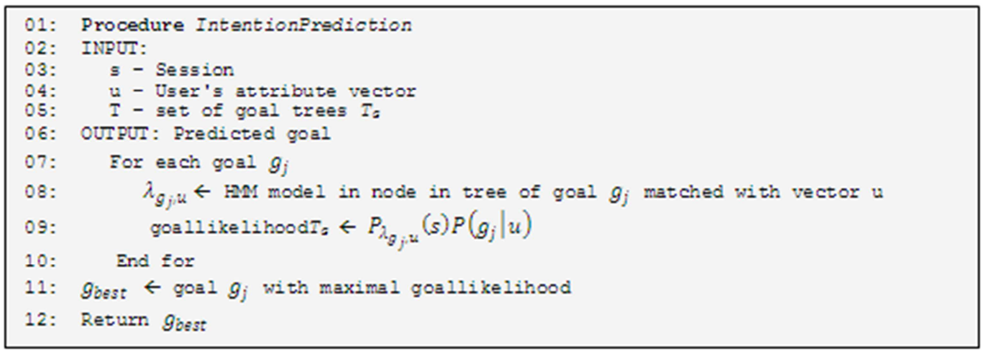
where $\lambda_{g_j,u}$ represents the HMM model located in the tree of goal g_j and in the node matched with the user-fixed attribute vector u. Using Bayes' rule twice:

$$P(g_j \mid u,s) = \frac{P(g_j,u,s)}{P(u,s)} = \frac{P(s \mid u,g_j)P(u,g_j)}{P(u,s)} = \frac{P_{\lambda_{g_j,u}}(s)P(u,g_j)}{P(u,s)} \quad (2)$$

The goal with maximal probability is selected in line 11 as the "intended" goal ' i.e., the goal that the user is most probably trying to accomplish:

$$g_{best} = \arg\max_{\forall g_j \in G}\left(\frac{P_{\lambda_{g_j,u}}(s)P(g_j \mid u)P(u)}{P(u,s)}\right) = \arg\max_{\forall g_j \in G}\left(P_{\lambda_{g_j,u}}(s)P(g_j \mid u)\right) \qquad (3)$$

Note that the term $\frac{P(u)}{P(u,s)}$ could be eliminated because it is fixed for all goals. The probability $P(g_j \mid u)$ can be estimated by using a probabilistic classifier. This classifier can be trained using various methods such as decision trees or Bayesian methods. The value of $P(g_j \mid u)$ can be approximated by the a-priori probability of each goal that can be estimated by simply counting the frequency that each goal occurs in the training set.



```
01: Procedure IntentionPrediction
02: INPUT:
03:    s - Session
04:    u - User's attribute vector
05:    T - set of goal trees $T_s$
06: OUTPUT: Predicted goal
07:    For each goal $g_j$
08:       $\lambda_{g_j,u}$ ← HMM model in node in tree of goal $g_j$ matched with vector u
09:        goallikelihood$T_s$ ← $P_{\lambda_{g_j,u}}(s)P(g_j|u)$
10:     End for
11: $g_{best}$ ← goal $g_j$ with maximal goallikelihood
12: Return $g_{best}$
```

Figure 6. Goal prediction given a sequence of actions (session)

The next stage is to predict for this goal the action the user intends to do next. In the selected leaf, the user's next intended action can be extracted from A, the transition probability matrix of $\lambda_{gbest,u}$. The output of this phase is the predicted goal and the next action.

4.2. Splitting criterion

An internal node in a tree is usually split according to the value of a single attribute. Consequently, the algorithm searches for the best attribute upon which to perform the split [Rokach & Maimon, 2008]. A common splitting criterion is *Information gain*, proposed by Quinlan [1986] and used in known decision trees algorithms (ID3,C4.5). This criterion estimates the attribute's quality by the difference between the prior and current entropy, given the attribute values. In this work instead of entropy, we use the log-likelihood function. Specifically, given a node *k* with a train set $Train_k$, we first induce a HMM model λ_k. The log-likelihood of node k is defined as:

$$Loglikelihood(Train_k \mid \lambda_k) = \sum_{\forall (s,u,g) \in Train_k} \log P_{\lambda_k}(s) \qquad (4)$$

where $P_{\lambda_k}(s)$ is estimated using the forward-backward algorithm.

The gain in the log-likelihood due to splitting the current node using attribute v_i is defined as:

$$\begin{aligned} &LoglikelihoodGain(V_i \mid Train_k, \lambda_k) = \\ &\sum_{\forall val \in dom(v_i)} Loglikelihood(Train_{k,i,val} \mid \lambda_{k,i,val}) - Loglikelihood(Train_k \mid \lambda_k) \end{aligned} \qquad (5)$$

where $Train_{k,i,val} = \{\forall(\langle s,u,g \rangle \in Train_k : u(v_i) = val\}$ and $\lambda_{k,i,val}$ is the HMM model trained on $Train_{k,i,val}$. Specifically, according to Equation 5, we go over all possible values in the domain of attribute V_i. For each value we first identify the relevant training instances $Train_{k,i,val}$ and then train the HMM model $\lambda_{k,i,val}$. Finally we calculate the log likelihood as described in Equation 2.

Information gain tends to favor attributes with many values [Quinlan, 1986]. If an attribute with many values is chosen as a root in a tree, each child node is trained with a small number of training examples. This attribute has a high information gain because the training of each attribute value is apparently very accurate. But choosing this attribute leads to poor results since it does not contain comprehensive information about the usage behavior of users with each attribute value. Our preliminary experiments show that the same phenomenon exists when we use log-likelihood. Quinlan defined the term "gain ratio" to overcome this deficiency in information gain [Quinlan, 1993]. Similarly we define the likelihood of gain ratio as:

$$gainRatio(v_i) = \frac{Loglikelihood(V_i \mid Train_k, \lambda_k)}{splitInfo(V_i \mid Train_k)} \qquad (6)$$

where the nominator is the *log likelihood gain* from Equation 5 and the denominator is calculated as follows:

$$splitInfo(V_i \mid Train_k) = -\sum_{\forall val \in dom(v_i)} \frac{|Train_{k,i,val}|}{|Train_k|} \log \frac{|Train_{k,i,val}|}{|Train_k|} \qquad (7)$$

Using gain ratio, the chosen attribute for splitting is the one that best improves the previous level in the tree, i.e., increases the prediction accuracy of the training set.

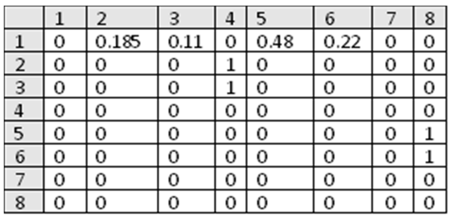
4.3. Stopping criteria

To end the process of training the models, several rules of thumb were defined as stopping criteria: (1) the improvement rate of information gain between parent and child is smaller than a pre-defined threshold; (2) a parameter is set for a minimal number of sequences for building a node (i.e., when less than X users with the same attribute value accomplish a specific goal, the node becomes a leaf); and (3) all sequences in a node belong to the same attribute value.

4.4. Illustrative example

The following section illustrates with a synthetic example our method of building an attribute-driven HMM tree for intention prediction. A simulated data set contains 1000 users with one sequence of actions per user. The users accomplished five different goals (tasks) - 'a','b','c','d','e' where 200 of the 1000 users accomplished goal 'a'. For the sake of simplicity, we illustrate the generation of an attribute-driven HMM tree only for goal 'a'. The demographic data about the users contains seven attributes (*Age group, Gender, Device, Experience, Education, Customer type, Status*). Each attribute has more than one value. In order to accomplish goal 'a', there are several possible paths each containing a sequence of action elements: [1,2,4], [1,3,4], [1,5,8], [1,6,8]. The training set for this goal contains 70% of the sequences of this goal, meaning 140 sequences. Table 1 describes the generated transition matrix after applying the HMM training method on the sequences of goal 'a' in the simulated dataset. This transition matrix is saved in the root of the tree. The cells in the matrix describe the probability of the next element occurring given the current element. For example, the probability that the user pressed button '2' just after button '1' is P(2|1)=0.185.

Table 1: Transition matrix of HMM for all training sequences for goal 'a'



	1	2	3	4	5	6	7	8
1	0	0.185	0.11	0	0.48	0.22	0	0
2	0	0	0	1	0	0	0	0
3	0	0	0	1	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

After the model is generated, the algorithm calculates the likelihood of the root using the forward-backward method of HMM. In this example the likelihood of the model in the root is --165.61. The higher the likelihood (closer to zero), the better the model represents the sequences.

The next step is to build the children level, as explained in subsection 4.1.1. The algorithm calculates for each attribute its gain ratio and chooses the attribute that produces the highest gain ratio. For example, attribute 'Gender' has two values - Male (38 sequences) and Female (102 sequences). The algorithm built one HMM using the 'Male' sequences (Table 2) and another using the 'Female' sequences (Table 3). The likelihood of 'Gender' is the sum of the likelihoods of 'Male and 'Female' (-22.86+-60.89) values: -83.75. The attribute gain of 'Gender' is 0.843. Therefore, the gain ratio of 'Gender' is (-165.61- (-83.75))/0.843=97.04. The gain ratio of all the other attributes is calculated similarly. 'Gender' is the chosen attribute to split the root because it has the highest gain ratio. The generated attribute-driven HMM tree is illustrated in Fig. 7 where the matrix in Table 1 is the root of the tree and the matrixes in Tables 2 and 3 are the first levels in the tree. The arcs from the first level represent the attribute values that would be chosen if we were to continue splitting the tree. The tree presented in Fig. 7 represents goal 'a' before pruning. Due to space limitations, in Fig. 7 we present the hidden states and the transition matrix among these states only (Table 2 and Table 3). However, as indicated in Fig. 1, like in any HMM, we use two layers of nodes: one layer holds the hidden states and the second holds the observable states (which we do not show).

Table 2. Transition matrix for Gender=Male.

	1	2	3	4
1	0	0.635	0.365	0
2	0	0	0	1
3	0	0	0	1
4	0	0	0	0

Table 3. Transition matrix for Gender=Female.

	1	2	3	4	5	6	7	8
1	0	0	0	0	0.69	0.31	0	0
2	0	0	0	1	0	0	0	0
3	0	0	0	1	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

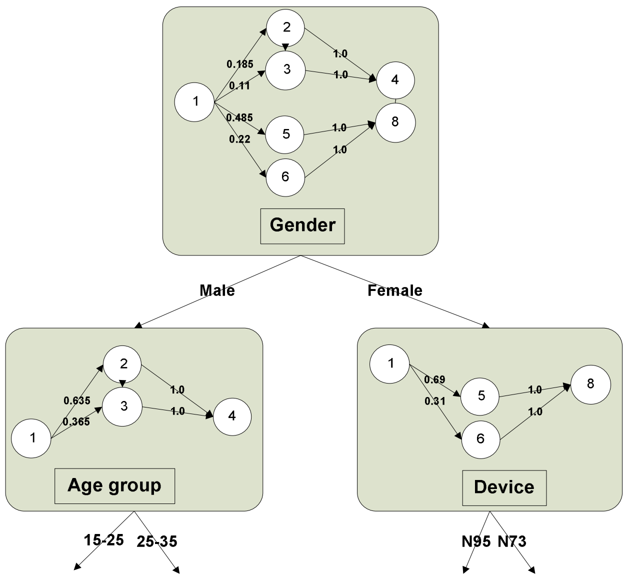


Figure 7. An attribute-driven HMM tree for goal 'a'.

Trees for all the goals are generated in a similar way. Fig. 8 illustrates trees for goals 'a','b','c'. The next step is to prune the trees in order to avoid overfitting. In this example of trees with one level, the root of each tree is a candidate for pruning.

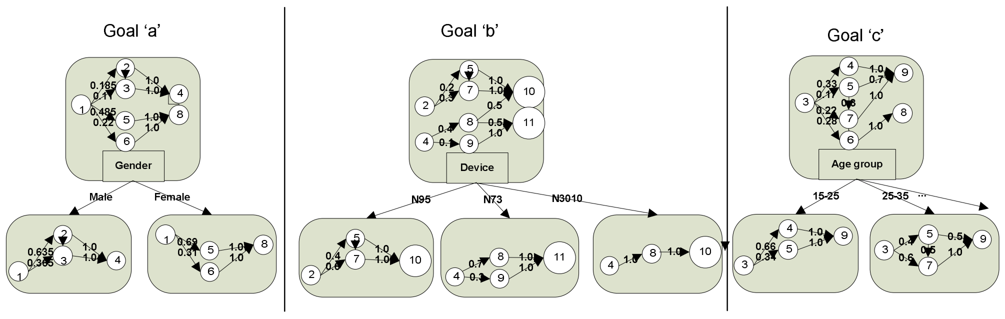
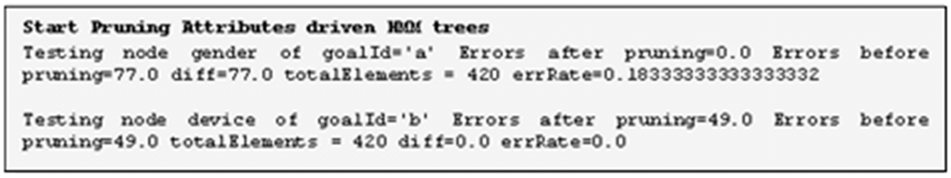


Figure 8. Goal trees before pruning.

The pruning method is executed in iterations. During each iteration, each candidate node for pruning is examined and the node that produces the maximum error rate is pruned. All candidate nodes are examined and the difference between errors before and after pruning each node is calculated and then divided by the number of sub-sequences in the training set. The node that produces maximum error rate is pruned. In this example, it is the root node in goal 'a'. (In Figure 9 we can see that its error rate is 0.1833, while the error rate for root node in goal 'b' was 0. Goal 'a' was the highest error rate). At the end of each iteration, the method searches for new candidate nodes for pruning and adds them to the existing ones. The iterations continue until no node produces a positive error rate. Fig. 10 illustrates the goal trees after the pruning methods.



```
Start Pruning Attributes driven HMM trees
Testing node gender of goalId='a' Errors after pruning=0.0 Errors before
pruning=77.0 diff=77.0 totalElements = 420 errRate=0.18333333333333332

Testing node device of goalId='b' Errors after pruning=49.0 Errors before
pruning=49.0 totalElements = 420 diff=0.0 errRate=0.0
```

Figure 9. Example of Pruning process for candidate nodes for pruning.

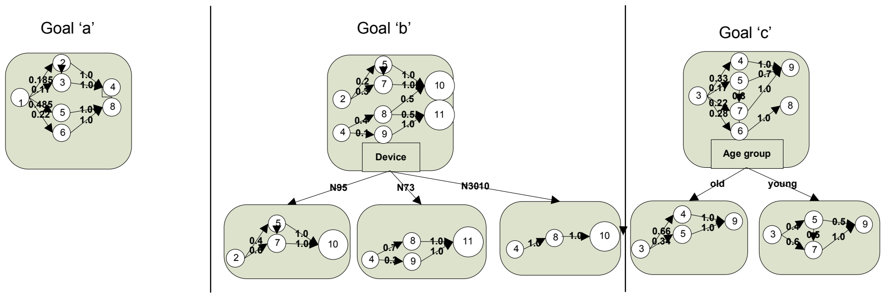


Figure 7. Goal trees after pruning

After applying the pruning method, the training process ends and we get the final tree for each goal. The leaves in the trees represent segmentations of users, i.e., user types. For example in goal 'b', the user types are characterized by their different devices and by age groups in goal 'c' .

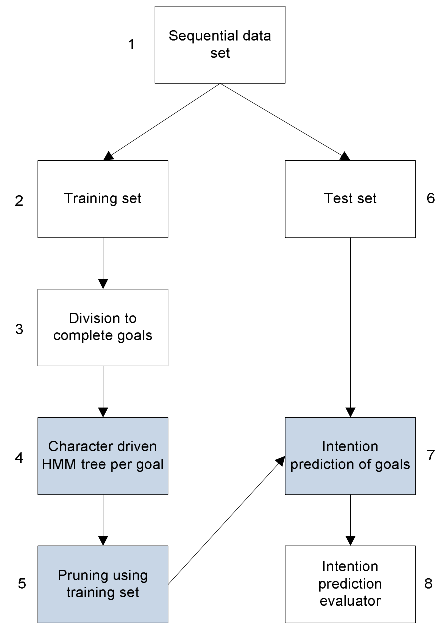
Intentions are predicted using the generated goal trees. When a user is interacting with the system, he generates a sequence of actions. This sequence, together with the user's set of attribute values, is the input to the intention prediction method. For example, a user with the following attributes -- 'Gender=Male', 'AgeGroup=young', 'Device=N95', 'Status=Single', 'Education=Graduate' -- generated the sequence [1,2]. As Fig. 10 illustrates, in goal 'a' the model that fits this user is the root; in goal 'b' the model that fits this user exists in the left child node (with device 'N95'); and in goal 'c' it is the right child node that fits the user (Age group='young'). We calculate the probability of the sequence [1,2] for each fitting leaf in each goal tree, and select the node with maximal probability as the intended goal.

5. EXPERIMENTAL STUDY

In order to evaluate its performance and examine the potential of the attribute-driven HMM tree algorithm to improve the HMM algorithm, we conducted an experiment on three datasets. The following subsections describe the experimental setup and the results obtained.

5.1. Experimental process

The main aim of the experimental study is to compare the generalized accuracy (i.e., the probability that the prediction of a goal given a sequence and a user is correct) of the abovementioned methods. Figure 11 presents the various steps in the experimental process that we conducted. The shaded boxes represent algorithms. First, the sequential dataset (box 1) was divided 10 times randomly into a training (box 2) and test sets (box 6). We then divide the training sequences into complete sessions (i.e., a sequence that starts when the goal begns and ends with completion of the goal). We apply (box 4), the attribute-driven HMM on the training set, and obtain a tree for each goal. A pruning method is then applied to the generated trees (box 5). Finally, the intention prediction model (box 7) estimates the predicted goal of sequences from the test set (box 6). Then, performance of the algorithm over the test set is estimated (box 8).



1
Sequential data set
2
Training set
Test set
6
3
Division to complete goals
4
Character driven HMM tree per goal
Intention prediction of goals
7
5
Pruning using training set
Intention prediction evaluator
8

Figure 8. The Experimental Process

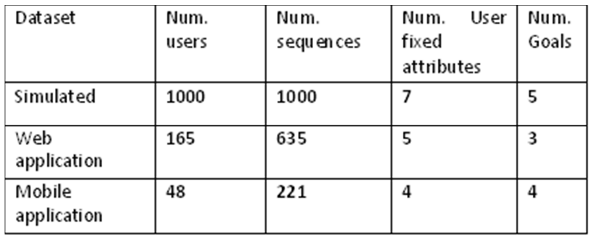
5.2. Algorithms used

This study examined the implementation of HMM in a tree structure using the suggested attribute-driven HMM algorithm. The attribute-driven HMM tree aims at improving the accuracy that HMM provides by utilizing user attributes. Therefore, the attribute-driven HMM tree algorithm is compared to HMM. Recall that in HMM, there is a hidden layer and an observed layer. Each layer can have a different number of states. In our study, however, these layers are identical in terms of the number of states (by default). Namely, as in the case of Gunterand and Bunke (2004), we choose the number of hidden states to be equal to the number of observed states in the processed node. It should be noted that one can set the number of hidden states using cross-validation procedure and use the number of hidden states which provide the highest cross-validated likelihood (please refer to Smyth (2000) and Celeux and Durand, 2008 for additional information). We leave this potential improvement for future work.

5.3. Datasets

The attribute-driven HMM algorithm and the HMM algorithm were evaluated on three datasets: (1) a simulated data set; (2) a real dataset obtained from a Web application used by faculty at Ben-Gurion University and (3) real data obtained from a mobile application. Prior to the process of training the attribute-driven HMM trees, the data was "cleaned" so all the sequences begin a goal and end with completion of that goal. A summary of statistics about the datasets is presented in Table 4. For each dataset, the table contains how many users interacted with the system; the total number of sequences from all the goals; the number of user attributes that are relevant for splitting the tree (have more than one value but less than a pre-defined number); and the number of goals (tasks) the users can accomplish in the system.

Table 4. Statistics of datasets used for test



Dataset	Num. users	Num. sequences	Num. User fixed attributes	Num. Goals
Simulated	1000	1000	7	5
Web application	165	635	5	3
Mobile application	48	221	4	4

5.4. Measures

In this experiment, metric accuracy was used. This represents the probability that the prediction of a goal, given a sequence and user, is correct. In order to estimate the accuracy, each dataset was randomly divided 10 times into a training set and test set. The training set and the test set comprises 70% and 30% of the dataset, respectively. Using the training set, the models for the attribute-driven HMM trees were built. Following this stage, the trees were tested using the test set. Each sequence on the test set was divided into subsequences by action steps (only first action, then two first actions in the sequence, etc.). The input for the test was these subsequences of the sequence together with the user's attributes. The output of the test is a goal that the algorithm predicts as the goal that the user will do with the highest probability. After the test, a pruning is carried out by using the same training set and then the 30% test set.

We ran the test once again to measure the performance of the algorithm after pruning in order to compare between the prediction accuracy of HMM and the attribute-driven HMM before and after pruning the trees. Since in the test we knew what the user actually did, we estimated the prediction accuracy of each goal in the system. Wilcoxon's test, with a confidence level of 95% verified whether the differences in accuracy between the HMM and the attribute-driven HMM with and without pruning were statistically significant. Wilcoxon's test is a non-parametric test for measuring a difference in central location (median) between two paired samples without assuming a normal distribution of the population.

5.5. Experimental results

5.5.1.*Simulated data set.* To validate the proposed attribute-driven HMM trees for intention prediction, we simulated training data for five goals for different user types, i.e., we simulated different behaviors for some specific attributes. Then, the algorithm generated a tree for each goal from the data we simulated. We wanted to examine (1) if the algorithm identifies the same user attributes that we defined when simulating the training data and (2) to prove that prediction accuracy increases when using user attributes. The data consisted of 1000 users divided into five goals (we simulated one sequence of actions per user, i.e., 200 sequences for each goal). The number of actions for accomplishing a goal was three to four steps. There were seven user attributes (Age group, Gender, Device, Experience, Education, Customer type, Status). Each attribute had two to seven values. In order to generate a two-level tree (root and another level) we decided which attribute would differentiate usage behavior for each goal, the percentage of users with same attribute value and the transition probabilities between actions in the goal. Then, we built the sequences accordingly. The division of users into goals is detailed in Table 5

Table 5: Division of users into goals and differentiating attributes

Goal ID	Num of users	Attribute	Value	Precent	User ID	Sequence
1	200	Gender	Male	20%	1-40	1,2,4
				10%	41-60	1,3,4
			Female	50%	61-160	1,5,8
				20%	161-200	1,6,8
2	200	Device	N95	30%	201-260	2,7,14
			N73	10%	261-280	2,5,15
			Samsung	10%	281-300	3,4,15
				15%	301-330	3,5,15
			iphone	15%	331-360	2,7,15
				20%	361-400	2,5,14
3	200	AgeGroup	12-15	15%	401-430	3,4,10
			15-20	15%	431-460	3,5,10
			20-30	20%	461-500	3,8,10
			30-40	25%	501-550	3,7,9
			40-50	15%	551-580	3,4,9
			50-60	5%	581-590	3,4,5,9
			60-80	5%	591-600	3,4,5,10
4	200	Status	Married	30%	601-660	1,7,16
				25%	661-710	1,9,16
			Single	35%	711-780	1,7,12
			Unknow	10%	781-800	1,9,12
5	200	Education	Under	65%	801-930	4,9,11
			Graduate	35%	931-1000	5,9,13
Total	1000					

It was expected that the algorithm would build the trees according to the same attributes. For example, there are several different ways of implementing goal 1. The significant attribute was defined as 'Gender', indicating that males accomplish this goal using the paths illustrated in Fig. 12(a) while females utilize the paths illustrated in Fig. 12(b).

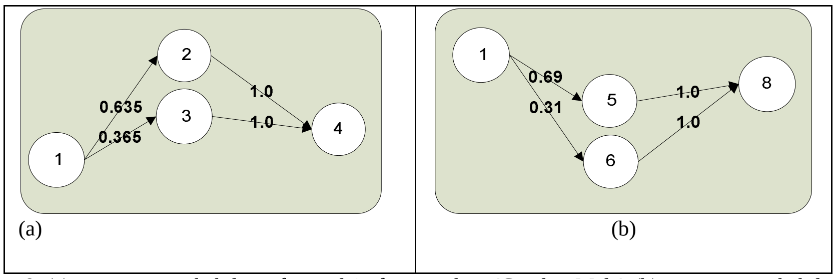


Figure 9. (a) transition probabilities for goal 1 of users where 'Gender=Male'. (b) transition probabilities for goal 1 of users where 'Gender=Female'.

The results were encouraging. We expected that all five tree goals would be generated according to the attribute we selected as the most significant for each goal. Although the data was randomly divided ten times into training and test sets, the trees were built in all runs according to expectations for all the goals. These results confirmed that using the HMM algorithm together with user attributes for differentiating usage behaviors was feasible.

Regarding the prediction accuracy of the algorithm, we see that in Table 6 the accuracy for each goal is measured in relation to a single HMM algorithm (the root in each tree) and our proposed algorithm. This indicates that prediction is based on the leaves in the tree with or without a pruning process. The results represent an average of ten runs of the algorithm and show an increase in prediction accuracy between a single HMM and the final attribute-driven HMM tree. The Wilcoxon test rejected the null hypothesis that a single HMM and our proposed algorithm perform in the same way with a confidence level of 95%. Thus we conclude that our attribute-driven hidden Markov model tree algorithm significantly outperformed the single HMM for this data set.

Table 6. Prediction accuracy for each goal in a simulated data set

Goal Id	Single HMM	Attribute driven Hidden Markov Model Tree without pruning	Attribute driven Hidden Markov Model Tree
1	0.9	0.82	0.86
2	0.83	0.95	0.89
3	0.98	0.93	0.98
4	0.73	0.83	0.80
5	0.93	1	1
Grand Total	0.87	0.90	0.90

Since the pruning algorithm is applied on all goal trees together, each tree affects the other tree's prediction accuracy. Therefore, in some cases, pruning one tree can increase or decrease another tree's accuracy. On average, the prediction accuracy of our algorithm with and without pruning was almost equal for this dataset. Fig. 13 presents the test results from a different perspective. The prediction accuracy averaged over 10 runs is presented for each step. Step 1 consists of predicting the most probable goal after only one step (action) that the user performed while step 2 means predicting the goal after two steps, etc. Prediction accuracy for all algorithms increased as the number of steps increased. We can see that for the second step, our algorithm is better than a single HMM by about 8% and achieves almost 100% accuracy.

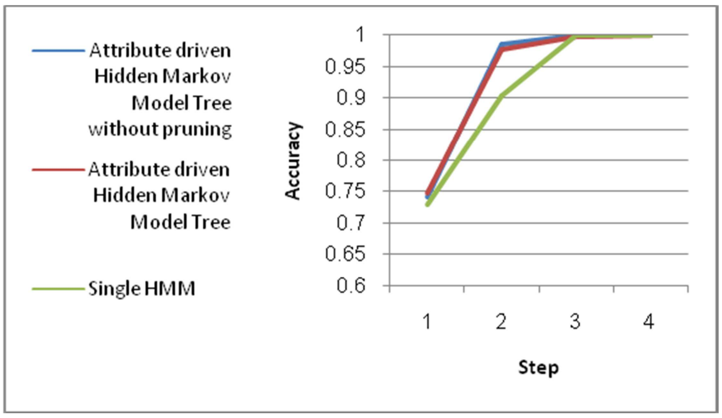


Figure 13. Increase in prediction accuracy with steps averaged over all goals in simulation data

5.5.2. *Web application dataset.* Faculty members at BGU were required to use a Web application in order to complete an annual survey of their activities during the academic year. Goals in the Web form included 'Add course', 'Delete course', 'Edit student details', 'Add research', etc. Since this application was new to the users, it made an ideal testing ground for recording user interactions and testing our algorithm. Each user had to fill out the form, but not all the goals were mandatory. A total of 165 users from the Faculty of Engineering at BGU used the application. Table 7 summarizes some statistics about the users.

Table 7. User statistics in the Web application

Attribute	Value	Percentage	Number of Sessions
Gender	Male	78%	397
	Female	13%	55
Age group	36-45	31%	186
	46-60	37%	168
	60 and up	22%	97
Title	Dr.	44%	241
	Prof.	51%	220

A session was regarded as the user interacting with the system. The dataset consisted of around 500 sessions from participants. Each session contained one to six goals. Some of the goals could only be accomplished in one possible way (as defined by the structure of the application). For such goals our algorithm is not superior to the HMM algorithm because there is no division of sequences by usage behavior (since all the users must accomplish the goal in the same way, using the same path). The number of sequences that relate to goals with more than one possible path is 635. The length of each sequence was between two to ten steps. Five attributes were candidates to differentiate between users for building each goal tree *(Rank, Department, Title, Gender and Age group)*. Comparing a single HMM and our proposed algorithm after the pruning process, the average prediction accuracy results for ten runs showed an increase of 2.5% on average for all goals (see Fig. 14). The accuracy for goals "94" and "111" is significantly higher than the accuracy of goal "105". This can be explained by the fact that goal 105 had only 113 training sequences, while the other two goals had about 300 training sequences. Goal 105 ("large course") had much less training examples than goal 94 ("general course"), because most of the courses that professors teach in BGU are not considered large courses (more than 80 students in a single classroom).

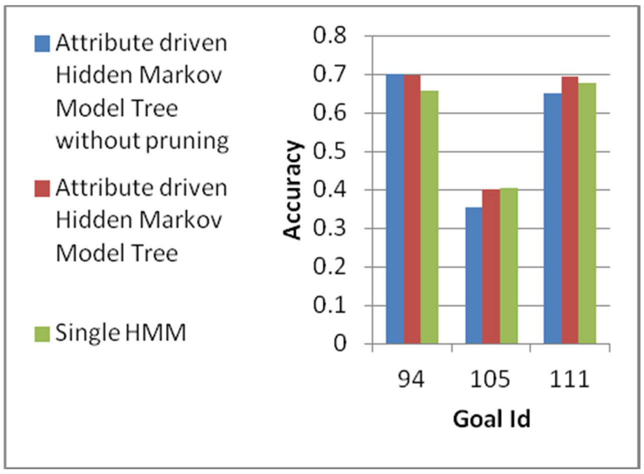


Figure 10. Prediction accuracy for each goal for the Web application dataset.

Another perspective from the results may be seen in the progression within subsequences. The main difference in the prediction accuracy between our approach and a single HMM is observed for the second step. Our algorithm outperforms a single HMM by more than 10% for the second user's step (i.e., prediction of goal after two steps is more accurate when our approach is applied). Results of an average of 10 runs (for goals number 94, 105, 111) by steps are presented in Fig. 15:

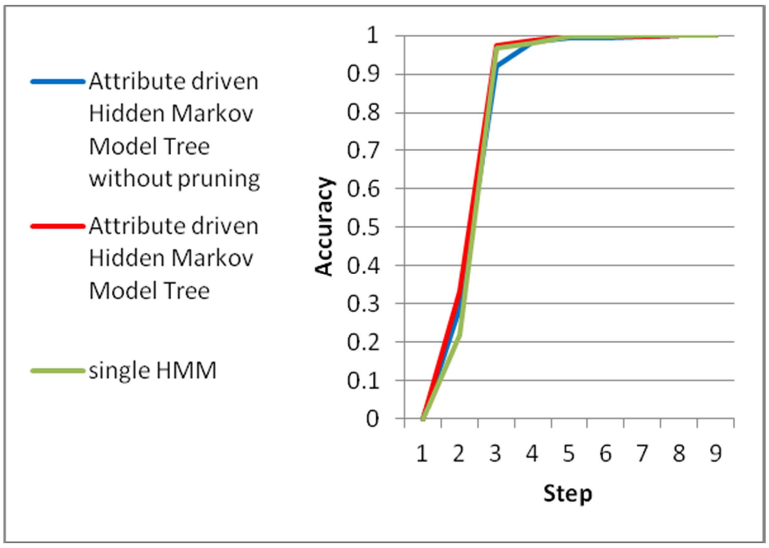


Figure 11. Increase in prediction accuracy with steps in average of all goals in Web application data

Although the improvement is moderate, it is still consistent. Moreover the Wilcoxon test rejected the null hypothesis that a single HMM and our proposed algorithm perform the same with a confidence level of 95%. Our algorithm significantly outperforms a single HMM for this dataset. The Wilcoxon test also rejected the null hypothesis that our attribute-driven HMM tree performs the same before and after pruning with a confidence level of 95%. Our algorithm after pruning significantly outperforms our algorithm before pruning for this dataset.

5.5.3. Mobile application data set. The third dataset was obtained from a test with 48 participants. The users were given tasks (goals) to perform using a mobile phone. They were aware that data was being collected about them. Goals included "Configure account settings", "Read/Send an email", "Send a picture to an email address", "Add a contact to address book", etc. We received also demographic data about the users and from this data four attributes were used to differentiate between users in order to build each goal tree (Age, Mobile provider, Customer type, Payment type). The actions users performed (usage data) with their device were recorded and transformed as input for the intention prediction algorithm.

The test was conducted on four goals with 17 to 96 sequences for each goal. Table 8 presents the prediction results.

Table 8 Prediction accuracy for each goal of the mobile application dataset

Goal ID	**Single HMM**	**Attribute Driven HMM Tree without pruning**	**Attribute Driven HMM Tree with pruning**
1000 (Configure “connection plan” and “message limit”	0.55	0.8	0.55
1004 (Add a contact to the address book)	0.44	0.54	0.44
1005 (count the number of words of an attachment)	0.57	0.34	0.58
1007 (Store an attachment; write e-mal with cc-function)	0.36	0.27	0.36
Grand Total	**0.48**	**0.36**	**0.48**

For the mobile application dataset, our algorithm does not outperform a single HMM algorithm on the average of over all the goals in the application. From Table 8, we see that the pruning process pruned all leaves from all the trees. The trees after pruning contained only roots, indicating that the model for each goal is equal to a single HMM. Although it looks as if the pruning methods decided to prune nodes that provided better predictions using its leaves, this is not correct. The method first pruned nodes that caused lower prediction accuracy (such as in goals 1005, 1007). The next iterations showed that the nodes that looked better than the root in the beginning did not provide better predictions after pruning the other nodes. From these results we see that our adjusted pruning method works properly.

We shall discuss in section 6 several possible reasons as to why our algorithm did not outperform a single HMM for the dataset. Moreover, we see from the results that the accuracy of the prediction is low, 48% on average. According to Wickens and Dixon [2007] below a level of around 70% accuracy, providing automation, assistance or guidance is worse than not providing it at all. Probably this number is too general and depends on many issues, so it may be a little lower or higher, but 48% will apparently be lower than the threshold.

5.6. Experiments with conditional random fields (CRF)

Up to this point we have focused on HMM as our base model. However, as indicated in section 3, the same idea can be easily applied to any other probabilistic sequence learning algorithm, specifically conditional random fields (CRF). CRFs which have been developed by Lafferty et al. (2001), define a conditional probability over label sequences given a certain observation sequence. This relaxes the unwarranted independence assumptions about the sequences which HMMs make.

For examining a CRF in our framework, we have used the open-source MALLET package (McCallum, 2007) to train a single CRF model by maximizing the log-likelihood of the training data. The same log-likelihood function is then used for choosing the best attribute split in the tree as indicated in Equation 6.

Table 9 presents the obtained accuracy using CRF as the base model and compares it to the previously reported performance using the HMM as the base model. Although previous works in the text mining domain have shown that CRF significantly outperforms HMM, in this experimental study, the CRF's improvement is relatively moderate. Still, in most of the cases the attribute-driven CRF is better than a single CRF. Actually, the attribute-driven CRF obtained the highest score in 10 out of 12 cases. Specifically, for the mobile application (where the attribute- driven HMM did not succeed in improving the performance of a single HMM), we now see that the attribute-driven CRF succeeded in improving the performance in 3 out of 4 goals. Using CRF as base model, we found that the attributes Age and Customer Type affect the way users are performing the various tasks. This encouraging result emphasizes the contribution of the proposed data-driven method. Thus the inability to improve HMM, in the mobile dataset, probably stems from the fact that HMM requires more training instances in order to estimate the model parameters [Rabiner, 1989]. The mobile application dataset illustrates that it is hard to assess in advance the contribution of the user attributes. On the one hand, when going down the tree, we are focusing more and more on similar users with the same characteristics, thus increasing the probability that these users operate the device in the same manner. On the other hand, as we go down the tree, we reduce the number of instances that are available for training each of the base models, thus jeopardizing their ability to accurately classify unseen instances.

As indicated in the original paper of CRF (Lafferty et al., 2001), the authors mention two main reasons for CRF to outperform HMM:

1. CRF has the ability to add overlapping features and, as was seen in various experiments, CRF benefits significantly from the use of these features, with accuracy improvement.
2. "When the data is mostly second order the discriminatively trained CRF typically outperforms the HMM." (Lafferty et al., 2001)

In the current study we do not use overlapping features, thus the improvement of CRF cannot be attributed to the first reason. Thus, of the two known reasons for the superiority of CRF, the second mentioned reason may apply, i.e., its ability to better fit second-order data using only first order models. However, other unknown reasons may also apply.

Table 9. Prediction accuracy for each goal using CRF model – The reported results are based on the mean of ten repetitions

Dataset	Goal	Single CRF	Attribute Driven CRF	Single HMM	Attribute Driven HMM
Simulated data set	1	0.88	0.9	0.9	0.86
	2	0.92	0.95	0.83	0.89
	3	0.98	0.98	0.98	0.98
	4	0.83	0.85	0.73	0.8
	5	0.97	1	0.93	1
Web application	94	0.69	0.73	0.65	0.7
	105	0.52	0.51	0.4	0.4
	111	0.65	0.71	0.67	0.69

Mobile application	1000	0.65	0.77	0.55	0.55
	1004	0.49	0.55	0.44	0.44
	1005	0.57	0.56	0.57	0.58
	1007	0.35	0.42	0.36	0.36

5.7 Comparison with other classification methods

The aim of this section is to compare the performance of our framework against the performance of other classification methods commonly used for intention prediction tasks. Specifically, we have examined the following learning algorithms: Bayesian network (see for instance Horvitz et al., 1998;), SVM (see for instance Shen et al., 2006; Park et al., 2009) and decision trees (see for example Qu, S. and Chai, 2008). Since SVM is designed to handle binary classification tasks, we applied the "one-against-all" method to convert the multi-class classification tasks into multiple binary classification tasks.

Each algorithm has been trained in two different modes: non-sequential and sequential. In the non-sequential mode we ignored the sequential aspects of the problem and used only the user's fixed attributes. In the sequential mode, we also used the sequential navigation actions as input attributes. Following Sun et al. (2002), we used a tri-gram model for representing the sequential patterns. Each three consecutive actions that have been performed by one of the users is considered as a candidate input attribute, where the value of "1" indicates that the user performed this sequence in the current session and "0" otherwise. Then using chi-square statistic with respect to the class, we selected the top 100 tri-grams[1] and use them as the input attributes (in addition to user fixed attributes).

Table 40 presents the performance of the above algorithms in the two different modes. In addition, it presents the performance of the attribute-driven CRF for comparison purposes. It is clearly observed that using the sequential 3-gram attributes substantially improves the performance of all methods. Nevertheless, in most of the cases, the attribute-driven CRF still outperforms all other methods in terms of prediction accuracy with mean accuracy of 0.74. The second to best is attribute-driven HMM with a mean accuracy of 0.69, followed by the Bayesian network (sequential mode) with a mean accuracy of 0.67.

Table 10. Prediction accuracy for each goal using various classification algorithms

		Bayesian Network		SVM		Decision Tree			
Dataset	Goal	Non Sequential	Sequential	Non Sequential	Sequential	Non Sequential	Sequential	Attribute Driven CRF	Attribute Driven HMM
Simulated data set	1	0.51	0.85	0.46	0.83	0.48	0.72	0.9	0.86
	2	0.45	0.92	0.41	0.85	0.43	0.9	0.95	0.89
	3	0.59	0.92	0.56	0.84	0.5	0.9	0.98	0.98

[1] We have examined various settings (50,100,500 and 1000 attributes) and found that the top 100 provided the most accurate results.

	4	0.64	0.79	0.49	0.78	0.6	0.79	0.85	0.8
	5	0.72	0.92	0.72	0.97	0.63	0.91	1	1
Web application	94	0.34	0.68	0.23	0.63	0.28	0.61	0.73	0.7
	105	0.21	0.35	0.15	0.36	0.21	0.29	0.51	0.4
	111	0.36	0.67	0.27	0.63	0.35	0.67	0.71	0.69
Mobile application	1000	0.39	0.53	0.34	0.54	0.33	0.48	0.77	0.55
	1004	0.42	0.55	0.39	0.52	0.36	0.51	0.55	0.44
	1005	0.43	0.54	0.37	0.53	0.39	0.5	0.56	0.58
	1007	0.26	0.35	0.19	0.32	0.23	0.34	0.42	0.36

6. DISCUSSION

We have presented accuracy results for our data-driven algorithm compared to a single HMM and CRF. First we saw that similar to results in other domains, CRF-based models usually outperform their corresponded HMM models. Moreover, for the first two datasets, the proposed attribute-driven algorithm improved the predictive performance of both single HMM and single CRF models. However, for the last dataset there was no significant improvement in the case of HMM. We believe that this may be due to the various reasons described in the following subsections.

6.1 Training data size

As we go deeper in the tree, we split the sequences between the nodes according to the attribute values of the users who created the sequences. Thus, the number of training instances available to train an HMM in a certain tree node reduces as we move away from the root node to deeper levels. In particular, if the training sequences are uniformly distributed among the tree's branches and all demographic attributes are binary, then the number sequences that are available for a specific node is $n\frac{1}{2}^{k}$, where n is the original number of sequences and k is the depth of the node (i.e. the length of the path to the root).

It is well known that for training a HMM, a sufficient number of training events should be provided in order to achieve a good estimate of the model parameters (Rabiner, 1989). Small amounts of training data in a node can lead to an inaccurate HMM model i.e., a model that does not represent all or most of user behavior with that attribute value. Thus, such a model will provide incorrect predictions. This can be clearly observed from the results in Table 8 where the performance of a single HMM is comparable to an attribute-driven HMM tree on mobile applications (where the training set was relatively limited,namely, containing only 48 participants).

Our method addresses this issue in two ways:

1. As indicated in Section 4.3 (stopping criterion), we use a parameter for determining the minimal number of training sequences needed for building a HMM. The minimum should be set to a value which ensures that states are visited often enough to yield good estimates of transition probabilities of the HMM. In this paper, we use a default value of 20 (our tests show that this value is usually sufficient). It should be noted that the same challenge exists in any

decision tree that combines another model in the leaf. For example, LeBlanc and Crowley (1993) set the minimum allowed size to 10 training instances while inducing a survival tree (a tree in which the leaves hold a survival model). Landwehr et al. (2005) require a minimum of 15 instances in order to train a logistics regression in the tree's node.

2. The pruning procedure used in the proposed method allows us to decrease the value of the abovementioned parameter (minimal number of training sequences). This allows the tree to grow deeper than necessary so that unreliable nodes be pruned later (in particular nodes, that were trained with insufficient training instances are pruned).

In fact, the results of Table 8 indicate that in cases with small training sets (48 participants), pruning procedures almost always prune the entire tree up to the root. Thus, the single HMM and the attribute driven HMM tree with pruning obtained almost the same results. We conclude that it is futile to use the proposed attribute driven HMM tree if there are only a few training sequences. In such casesm one can benefit from using a single HMM because both single HMM and attribute-driven HMM tree provide an equal predictive performance while the computational cost of single HMM is lower.

6.2. User attributes

One of the inputs to our algorithm is user attributes. We assume that the division of users into different usage behaviors in a system can be based on the given user attributes. However it is possible that these attributes do not characterize user behavior. For example, from the structure of a system, there are two paths for accomplishing this goal. There are 100 sequences of users who accomplished this goal. The algorithm starts the training with a search for an attribute for splitting the tree. Starting with 'Gender', the algorithm finds that both males and females use both paths almost equally to accomplish the goal. This is the same for all other given attributes about the users. In such cases, the splitting of the tree according to user attributes does not contribute to prediction accuracy. Indeed, just the opposite occurs. The splitting reduces the accuracy because instead of strengthening the training with either more sequences or a clearer path to accomplish a goal, the training contains fewer sequences (less training examples) for each path.

An attractive aspect of any decision tree and, in particular the method presented in this paper, is that feature selection is actually embedded in the tree induction process. Thus, the automated method for determining the contribution of users' attributes is the splitting criteria. Hence, due to their low splitting criteria, value features that are not relevant have a low chance of being selected. Moreover, if non-relevant features were still selected, the pruning procedure should discover and dismiss them.

Although we examined the proposed attribute-driven hidden Markov model tree for intention prediction in software applications, the proposed method can be applied to applications far beyond this scope. For example, the method can be used to analyze the trajectory of tourists in a new city they are visiting (Kisilevich et al., 2010). In such cases the trajectory is represented as a sequence of point-of-interests which a tourist would explore. We can then use intention prediction to predict the next point of interest the tourist will visit and provide her relevant information. These trajectories will be affected by tourist demographic attributes, such as: age or nationality. For example, elderly people may avoid trajectories of long hikes.

Furthermore, the proposed method can be used for analyzing any sequential patterns and not solely for predicting the intention of humans. For example, HMM and CRF can be used to induce a part-of-speech tagger (Lafferty et al., 2001). Consider a case where a corpus of textual documents is available. Every document in the corpus can be associated with various meta-attributes such as domain (finance, sport, art, etc.) or target audience for the document (such as professional, news reader, and children). To maximize accuracy, it is desired to select the POS tagger for a specific document based on its meta-attributes. For example, a narrative medical report can benefit from a specific adjusted POS tagger (Gunter and Bunke, 2004). Our algorithm can identify the relevant attributes according to which the documents should be clustered and to induce a POS tagger accordingly.

6.3. Paths per goal

Our algorithm divides the training data according to different usage behaviors, i.e. different paths to accomplish a goal. If there is only one path to accomplish a goal, our algorithm is unnecessary. In such cases it is preferable to use a single HMM that will probably provide a better prediction than the attribute-driven HMM tree.

In our experiments, we measured the accuracy of our algorithm for predicting user intentions. Predicting a user's intentions during his or her interaction with a system could serve as a useful tool for assisting users in accomplishing their tasks. This would be accomplished by providing guidance as to how to perform the intended task. However, offering users assistance may be intrusive when initiated by the system and not by the user since assistance may be proffered at an inconvenient time [Jameson, 2003]. Providing inaccurate guidance or assistance can also annoy users. Wickens and Dixon [2007] examined the implications of different levels of unreliability or imperfection in diagnostic automation. The analysis suggested that performance is quite sensitive to the level of imperfection and that below a level of around 70%, unreliable automation is worse than no automation at all.

Another issue is the added value gained by applying this algorithm - the creation of trees leaves. All leaves in a tree can be used to segment users by their attributes and attribute values for each goal. We refer to this segmentation as user types. As noted above, this segmentation can be used for different purposes such as marketing or personalization. For example, if a company wants to promote a product related to SMS use in a cell phone, it can use the user type segmentation for goals like 'send SMS'. The attributes that differentiate between users while sending an SMS could be used for segmenting promotions. If, for instance, 'age' is a differentiating attribute, then the company may market the new product to different age groups accordingly. Very possibly the types of segmentation can be used as well for personalization purposes. Using the same example, a cell phone company could provide a different SMS user interface based on the age of the user. Another tool for marketing and personalization might derive from testing the algorithm on non-fixed user attribute values. For example, proficiency may be different for different goals and might change (to more proficient) over time as the user gains experience. If a user is considered as proficient in a goal, a company may decide not to provide him assistance or guidance while he is implementing this goal. Other goals, on the other hand, where proficiency is less, would be accompanied by assistance.

From the comparison with existing machine learning algorithms we conclude that the attribute-driven method should be a method of choice when both sequential and fixed user attributes are available.

7. CONCLUSIONS AND FUTURE WORK

When predicting user intentions, the accuracy is crucial thus increasing it is an important task. In this paper we have presented a new method for intention prediction that improves the accuracy of the single HMM/CRF algorithm. The algorithm is using user attributes to build attribute-driven HMM/CRF trees for intention prediction. The results of a simulation study demonstrated the capability of the algorithm for discovering a highly accurate intention prediction model. During our research we discovered the need for a pruning phase to avoid overfitting and developed a new pruning procedure. We tested the intention prediction algorithm on three datasets and then compared the accuracy of the results obtained by the HMM algorithm introduced by Rabiner and our algorithm, both with and without pruning the attribute-driven HMM trees.

One limitation of the current version of our algorithm is that it cannot handle multitasking. In recently developed applications and devices, multitasking has become more widespread and therefore our algorithm should be extended to these applications. As discussed earlier, future research should also include testing the algorithm on non-fixed user attribute values.

To conclude, the algorithm is beneficial when data about users is available and is helpful for differentiating usage behavior of different users, particularly where there is a large number of training sequences and the goals in the system can be accomplished using more than one possible path.

REFERENCES

1. Baeza-Yates, R., Calderón-Benavides, L., & Gonzalez-Caro, C. (2006). The Intention Behind Web Queries. In Proceedings of String Processing and Information Retrieval 2006 (pp 98-109). Glasgow, Scotland.
2. Baum L.E. and Egon J.A. An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology, Bull. Amer. Meteorol. Soc., vol. 73, pp. 360-363, 1967.
3. Baum L.E. and Sell G.R., “Growth functions for transformations on manifolds,” Pac. /. Math., vol. 27, no. 2, pp. 211-227, 1968.
4. Bishop, C.M. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc. 2006.
5. Bhargava, A. and Kondrak, G. 2009. Multiple word alignment with profile hidden Markov models. In Proceedings of Human Language Technologies: the 2009 Annual Conference of the North American Chapter of the Association For Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium (Boulder, Colorado, June 01 - 01, 2009). Human Language Technology Conference. Association for Computational Linguistics, Morristown, NJ, 43-48.
6. Celeux, G. and Durand, J.B., Selecting hidden Markov model state number with cross-validated likelihood, Computational Statistics, 23(4):541-564, 2008.
7. Chen, C. and Liang, J. and Zhao, H. and Hu, H. and Tian, J., Factorial HMM and parallel HMM for gait recognition, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 39(1):114-123, 2009.
8. Chen, Z., Lin, F., Liu, H., Liu, Y., Ma, W., and Wenyin, L. User Intention Modeling in Web Applications Using Data Mining. World Wide Web 5, 3 (Nov. 2002), 181-191. 2002.
9. Dempster A.P., Laird N.M. and Rubin D.B., “Maximum likelihood from incomplete data via the EM algorithm,” 1. Roy. Stat. Soc., vol. 39, no. 1, pp. 1-38, 1977.

10. Feng L., Guan X., Guo S., Gao Y., Liu P. Predicting the intrusion intentions by observing system call sequences (2004) Computers and Security, 23 (3), pp. 241-252.
11. Fine,S, Singer, Y, and Tishby, N. The hierarchical hidden Markov model: Analysis and applications. Machine Learning, 32:41–62, 1998.
12. Friedman N., Murphy K., and Russell S., Learning the structure of dynamic probabilistic networks. 14th Conf. on Uncertainty in Artificial Intelligence, 1998.
13. Galassi, U, Giordana, A, Mendola, D., Learning User profile from Traces. In Proceedings of the 2005 Symposium on Applications and the internet Workshops (January 31 - February 04, 2005). SAINT-W. IEEE Computer Society, Washington, DC, 166-169, 2005.
14. Ghahramani, Z., and M.I. Jordan., (1997). Factorial Hidden Markov models. Machine Learning, 29, 245–273.
15. Gunter, S. and Bunke, H., HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and Gaussian components, Pattern Recognition, 37(10):2069-2079, 2004.
16. Horvitz E., Breese J., Heckerman D., Hovel D., and Rommelse K., The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pages 256--265, Madison, WI, 1998.
17. Hu, J., Zeng, H., Li,H., Niu,C., and Chen, Z., 2007. Demographic prediction based on user's browsing behavior. In Proceedings of the 16th international conference on World Wide Web (WWW '07).
18. Jameson, A. 'Adaptive Interfaces and Agents', in Jacko, J.A., Sears, A. (Eds.), Human-Computer Interface Handbook. Mahwah, NJ, Erlbaum, pp. 305-330. : 2003.
19. Jansen B., Booth D. and Spink A., Determining the User Intent of Web Search Engine Queries. In Proceedings of the International World Wide Web Conference, (pp 1149-1150). Alberta, Canada, 2007.
20. Jung J., Lee C., Lee J. and Bien Z. "User Intention Recognition for Intelligent Bed Robot System", Proceedings of the ICORR 2003 (The Eighth International Conference on Rehabilitation Robotics), 23-25 April 2003.
21. Kisilevich, S. and Keim, D. and Rokach, L., A novel approach to mining travel sequences using collections of geotagged photos, Proceedings of the Thirteenth International Conference on Geographic Information Science. Berlin, Springer-Verlag, pp. 163-82, 2010.
22. Krogh, A. An Introduction to Hidden Markov Models for Biological Sequences. In Salzberg, S., Searls, D. and Kasif, S. (eds), Computational Methods in Molecular Biology. Elsevier, New York. pp. 45-63. 1998.
23. Lafferty, J., McCallum, A., Pereira, F. (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". Proc. 18th International Conf. on Machine Learning. Morgan Kaufmann. pp. 282–289.
24. Landwehr N., Hall M., Frank E. (2005). Logistic Model Trees. Machine Learning. 95(1-2):161-205.
25. Lane, T. Hidden Markov models for human/computer interface modeling. In Proceedings of the IJCAI-99 Workshop on Learning About Users. 35–44. 1999.
26. LeBlanc, M., Crowley, J., 1993. Survival trees by goodness of split. Journal of the American Statistical Association 88, 457-467.
27. Lee U., Liu Z. and Cho J., Automatic identi?cation of user goals in web search. In Proceedings of the International World Wide Web Conference 2005, (pp. 391–400), Chiba, Japan. 2005.
28. McCallum A.K., MALLET: A Machine Learning for Language Toolkit, Home page. Retrieved 2 Febuary. 2011, http://mallet.cs.umass.edu.
29. Park, J.I. and Baek, S.H. and Jeong, M.K. and Bae, S.J., dual features functional support vector machines for fault detection of rechargeable batteries, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 39(4):480-485, 2009.
30. Qu, S. and Chai, J.Y., Beyond attention: the role of deictic gesture in intention recognition in multimodal conversational interfaces, Proceedings of the 13th international conference on Intelligent user interfaces, pp. 237-246, 2008.
31. Quinlan, J.R. "Induction of decision trees", Machine Learning 1, 81-106, 1986.
32. Quinlan, J.R. “Simplifying Decision Trees,” Int’l J. Man-Machine Studies, vol. 27, pp. 221-234, 1987.
33. Quinlan, J.R., C4.5: Programs for Machine Learning, Morgan Kaufmann, Los Altos, 1993.
34. RABINER, L.R. A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE Volume 77, Issue 2, Page(s):257 - 286, Feb 1989.

35. Rokach L., and Maimon O., Top down induction of decision trees classifiers: a survey, IEEE SMC Trans. Part C 35(4) 476–487, 2005.

36. Rokach L. and Maimon O., Data Mining with Decision Trees: Theory and Applications (World Scientific, New York, 2008).

37. Rose D. and Levinson D., Understanding User Goals in Web search. In Proceedings of the International World Wide Web Conference 2004, (pp 13–19), New York, USA. 2004

38. Samaria, FS and Harter, AC, Parameterisation of a stochastic model for human face identification Proceedings of the Second IEEE Workshop on Applications of Computer Vision, 1994., pp. 138-142, 1994.

39. Shen, J., Li, L., Dietterich, T. G., and Herlocker, J. L., A hybrid learning system for recognizing user tasks from desktop activities and email messages. In Proceedings of the 11th international Conference on intelligent User interfaces (Sydney, Australia, January 29 - February 01, 2006). IUI '06. ACM, New York, NY, 86-92.

40. Smyth P., Model selection for probabilistic clustering using cross-validated likelihood. Statistics and Computing, 10(1):63–72, 2000.

41. Sun X., Chen Z., Liu W., and. Ma W.Y. Intention modeling for web navigation. In Proceedings of the 11th World Wide Web Conference (WWW), 2002.

42. Sun, R. and Giles, C. L., Sequence learning: From recognition and prediction to sequential decision making. IEEE Intelligent Systems, 16(4), 67–70. 2001.

43. Taha T., Valls M.J., DISSANAYAKE G. POMDP-based long-term user intention prediction. Proceedings of the IEEE 2008 International Conference on Robotics and Automation (ICRA'08) - accepted for publication, May, 2008

44. Thakor, M.V.,. Borsuk, W., Kalamas, M., Hotlists and Web browsing behavior- and empirical investigation. Journal of Business Research, Volume 57, Issue 7, July 2004, Pages 776-786

45. Vozalis, M., Margarits, K.G., On the enhancement of collaborative filtering by demographic data, Web Intelligence and Agent Systems, 4(2), 117-138, 2006

46. Wickens, C. D., and S.R. Dixon., 'Is there a magic number 7 (to the minus 1)? The benefits of imperfect diagnostic automation: A synthesis of the literature'. Theoretical Issues in Ergonomics Science, 8(3), 201–212. 2007.

47. Weber, I., Castillo, C., The demographics of web search. In Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval(SIGIR '10). 523-530, 2010.

48. Yudhistira As., Kim M., Kim H. and H. Lee, An Automatic Personal TV Scheduler Based on HMM for Intelligent Broadcasting Services. PSIVT: 1123-1132, 2006.

49. Zanker, M., Jannach, D.; Gordea, S.; Jessenitschnig, M., Comparing recommendation strategies in a commercial context. Intelligent Systems, 22(3), 69-73, 2007.