Learning and Predicting User Behavior for Particular Resource Use

Jung-Jin Lee
Dept. of Computer Science and Engineering, The Catholic University of Korea, BuCheon, Korea
JungJin@songsim.cuk.ac.kr

Robert McCartney and Eugene Santos, Jr.
Dept. of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269-3155 U.S.A.
{robert,eugene}@engr.uconn.edu

Abstract

To successfully interact with users in providing useful information, intelligent user interfaces need a mechanism for recognizing, characterizing, and predicting user actions. In particular, it is our interest to develop the mechanism for recognizing and predicting simple user intentions, i.e., an activity involves in using particular resources. Much work to date in adaptive user interfaces has resulted in ad-hoc approaches such as simply capturing user preferences at a shallow level ignoring the more difficult problem of capturing the user intention. We frame the modeling task of user interface systems in terms of learning user patterns of using particular resources by understanding temporal information of activity, user intentions, and abstraction of user behavior. Our approach learns the individual user models through time-series action analysis and abstraction. After capturing the dynamics of user behavior into regularities of user behavior(patterns), probabilistic user models are constructed to facilitate the predictions of resource usage with a sequence of currently observed actions in the Unix domain.

Introduction

When humans predict the future for what's coming next, one tries to discover existing facts at the present time, employs past findings and experiences and then utilizes them to envision the future. These are the core concepts we want to transform from humans to our predictive agents. In particular for this paper, we are focused on managing resources such as printer or file system in the UNIX domain; it is for, specifically, assessing the likelihood of upcoming demands by users on limited resources and detecting potential problems by observing human-computer interactions.

Systems which reason about real-world problems can represent only a portion of reality. Moreover, the observation of user behavior itself is not easy. If we want to predict the performance of multiuser computing systems, while having the uncertainty of what the users are going to do, and how that affects system performance, the agent's ability to determine what to observe will contribute to identifying patterns of user behavior by dealing with uncertainty. Therefore, the user modeling task in our work involves understanding temporal user behavior and learning predictive patterns from user's past actions regarding how a user uses particular resources in order to predict the user's resource demands.

Specifically, we are looking at two problems associated with predictive pattern learning: (1) decoding the intention of user action or plan recognition, and (2) learning regularities of user actions as patterns of user plans. In the first problem, we want to identify user intent hidden in his/her actions. The goal of intent recognition is to recognize the category of an action regarding resource use, which can be viewed as a process of classification of action data. In the second problem, we want to learn predictive patterns or skills of user plans of using a particular resource corresponding to user action data.

Recognition/Prediction Problem

Suppose an interface agent's task is to manage given resources, such as a printer and/or a file server, and part of this task involves transferring work elsewhere to avoid overloaded conditions. One way of predicting such use is to recognize when users are likely to use that resource in the near future; that is, to recognize the user's intentions and plans relative to the resource in question. Many systems would benefit from the ability to infer user's knowledge and intention throughout interaction in order to arrive at a particular solutions. The benefits of predicting the future behavior of a user, the usage of resources, or a short-term goal conflict are that the interface agent can use the prediction to take control actions both to help users and to better achieve its own task. For users, the agent can suggest, based upon the predictions of their behaviors, that users send a file to printer–2, since printer–1 is jammed and printer–3 has many jobs in the queue. For overall system performance, based on the measure of predicted use of printers, the interface agent can take the action of changing cartridges or warming up the printers or gather some information from other agents. When the interface agent has multiple resources to manage and its intermediate or procedural actions are common to the plans of using those resources, recognizing which plan is more plausible will be beneficial for managing the right resources. We look at two prediction problems: predicting the possibility of using resources with the partial sequence of actions observed and predicting which resource is more likely to be

used among competing ones within bounded lengths of next actions.

Domain Characteristics

Our domain of interest is human-computer interaction in a large, ongoing, and dynamic environment such as *Unix* and *WWW*. Some difficult features in these domains [1] include the nonstrict temporal orderings of actions, the interleaving of multiple tasks, the large space of possible plans, some sequence of actions are shared, suspended, and resumed to lead to multiple goals, and conditional plans where the condition is neither explicit nor directly observable. The *Unix* domain is used as a testbed for this work.

Prediction problems in *Unix* domain have been explored in other work. Davison and Hirsh [2] builds an adaptive system to predict the very next command prompt and Korvemaker and Greiner [3] extended [2]'s work by providing top-rated commands to shortcut keys on the user's keyboard. Recently, a similar problem to ours was investigated by Lau and Horvitz [4] in a WWW domain, where they use Bayesian networks to infer the probability of a user's next action, the user's informational goal, based on a consideration of partial evidence about the status of a search.

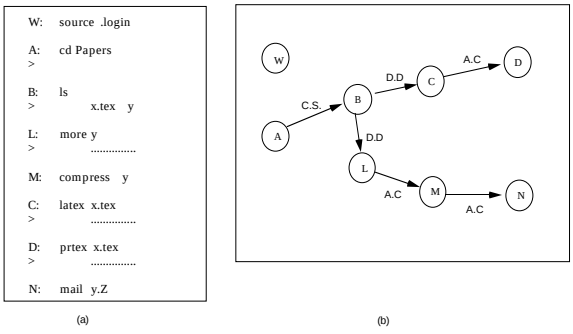

Figure 1: The Correlations of an Example

Less attention has given to issues of *ambiguity, distraction* and *interleaved execution* for uncertainty of user behavior. The issues are investigated in our work by finding correlations of actions using time-series action analysis. Suppose the current observation is (a) in Figure1. Through correlations of actions, the sequence of the observation can be separated into three coherent subsequences: *(W)*, *(A-B-C-D)*, and *(A-B-L-M-N)* as (b) in Figure 1. Each *coherent* sequence means here a partial sequence of the whole observation sequence, which is grouped by correlations of its actions involved. *Ambiguity* of actions to identify user intent in his/her action is dealt with by recognizing shared actions, such as action *B* in Figure 1 where *'ls'* is considered to be an information gathering action as in [5]. *Distraction* is examined by excluding extraneous actions related to resources in concern, for example, action *W* in Figure 1 can be considered irrelevant action with no relation to other actions and no relevance in using resources. And *multiple interleaved plans* are investigated by recognizing shared actions and finding correlations among actions. As in Figure 1, users in *Unix* domain could and tend to execute actions in multiple plans interleaved. Finding correlations through action analysis segments the sequence into the subsequences: *(W)*, *(A-B-C-D)*, and *(A-B-L-M-N)*.

Hidden Markov Model Approach

General-purpose machine learning algorithms have been used to deal with uncertainty to learn user behavior and can efficiently search large and flexible spaces of classifiers for a good fit to training data. Although these algorithms are general, they have a major drawback. In a practical problem where there is extensive prior knowledge, it can be difficult to incorporate this prior knowledge into these general algorithms. A secondary problem is that the classifiers constructed by these general learning algorithms are often difficult to interpret, that is, their internal structure might not have any correspondence to the real–world process that is generating the training data. A time-series analysis (segmentation and labeling) of action sequences is done to deal with uncertainty and ambiguity of actions. Once the sequences of actions are segmented and labeled, the coherent subsequences, which are called predictive patterns learned, are represented in each HMM in terms of input parameters. A hidden Markov model (HMM) in our work is used only to represent and to compute the likelihood of behaviors and the probability of multiple resource usage.

Learning Predictive Patterns

Inducing regularities/preferences in each user's behavior to carry out his/her intended task helps to guide users the daily work. Each user has a different way of doing the same thing called *preferences* and identifying the information that can characterize the user and be automatically collected is crucial.

Learning predictive patterns involves observing repetitive or regular behavior. We learn user regularities into probabilistic user models incrementally. There are two ways of learning predictive patterns: (1) assigning a label to an entire sequence of actions and (2) segmenting the sequence of actions into subsequences and labeling them individually. The latter one is more difficult because there are temporal information to consider within each subsequence and among the subsequences. In our work, segmentation and labeling method is used to learn the patterns taking observations of task executions as a time-series analysis. Segmentation of this problem is extracting coherent partial sequences through correlation of actions. Once the partial sequences are extracted the labeling of the subsequences are determining states relating to using particular resources.

Time-Series Action Analysis: Segmentation

Most work for user-adaptive interface system has analyzed only the sequence of user behavior. In our work, both se-

quential and relational information are used to extract correlations among actions. Sequential information is a command sequence and relational information are arguments of the commands and *Unix* system's responses to the commands. In Figure 1, partial sequences: *(W)*, *(A-B-C-D)*, and *(A-B-L-M-N)* are extracted by finding correlations among actions as contextual temporal information. Correlations among actions are determined by coherence rules and action knowledge such as command *C.C*and argument coherence *A.C*, data dependency *D.D*, anytime action *A.A*, redundant action *R.A*, and conditional sequence *C.S*. A current working directory of an action issued is captured and compared to make sure the actions compared for correlations are in the same directory. Any argument each action might have, results of some actions from the *Unix* system, and time-stamps of actions which describe the sequence of actions are gathered for the action reasoning of finding correlations. For instance, if the current action takes an argument from the result of its previous action then the *data dependency* rule is attached as a link and the link represents the two actions are correlated with the data dependency relation.

Abstraction of Patterns Learned – Labeling

In a command-driven system like *Unix*, a plan of using a particular resource is identified by the presence of distinguished actions. For instance 'lpr' in a plan indicates that the plan uses a printer. In many cases, the accuracy of predictions can be improved by inventing a more appropriate set of features to describe the available data. For abstraction, distinguished actions in each plan of using a resource are used as a fixed feature to determine a underlying state (resources used) of each subsequence. Distinguished actions used for each plan are: `'lpr, prtex'` for "PrinterUse" plan, `'ftp, telnet, mail, ping, netscape, gopher'` for "RouterUse" plan, and `'uncompress, cc, gcc, latex, tar, dvips'` for "MemoryUse" plan. We define an event for each resource of interest. Suppose a coherent partial sequence which is extracted from an observation sequence in a training phase is *'latex-compress-prtex-ftp'*. Since `prtex` is present, this sequence is an element of event E_1, the set of "PrinterUse" plans. Since `ftp` is present, it is an element of event E_2, the "RouterUse" plans. Since `latex` is present, it is an element of event E_3, the "MemoryUse" plans. Therefore, the state of this sequence is viewed to be the set of events, that it is in $\{E_1, E_2, E_3\}$ of using multiple resources and the sequence is represented as a state of S_{123} in this case. This definition of state has the useful feature that each element of the sample space is in exactly one state, so the states are disjoint. The event probabilities relate to the state probabilities in the obvious way: the probability of an event is the sum of the probabilities of the states that include that event. The parameters generated from these events are inputs to the prediction system of each HMM. This work puts an emphasis on producing *better* parameters for the HMMs, namely, initial state distribution, state transition probability and output probability. Unlike problems such as patterns or voice recognition which use a *general* HMM to deal with uncertainty stochastically or mathematically, our prediction problem uses a *specialized* HMM to represent what we learn as predictive patterns and to facilitate the prediction computation of a sequence of actions. Therefore, the input parameters for our HMM are produced (rather than guessed) through data analysis by dealing with the uncertainty of filtering and extracting relevant information only.

Consider our domain of porbability theory as related to coherent partial sequences of actions namely, pattern-of-actions identification. We are interested in estimating the probability of each coherent partial sequence being in each resource plan (an event) as observing part of that sequence. Let the current observation of a sequence of actions in a test phase be (a) in Figure 1. For example, the likelihood of using a resource `'Printer'` $L_{printer}$ given a sequence w= {`cd-ls-latex`} observed at time t_c, can be computed.

$$\begin{aligned} L_{printer} &= P(`PrinterUse'|w) = P(`PrinterUse'|\text{cd-ls-latex}) \\ &= \sum_{i=1,12,13,123} P(S_i\&\text{cd-ls-latex})/P(\text{cd-ls-latex}) \end{aligned}$$

Using the definition of conditional probability and Bayes rule, we compute the likelihood of the coherent partial sequence to be an exact pattern or a part of a pattern of using each resource which is learned from previous observations. That is, the problem of interest here is calculating the probabilities of using the various resources given a subsequence of commands w. In other words, the probability of any state containing the appropriate event given the subsequence, that is,

$$P(E_j|w) = \Sigma_{i:E_j \in S_i} P(S_i|w) \text{ for resource } j \quad (1)$$

and the probability of each state $P(S_i|w)$ is

$$P(S_i|w) = P(S_i\&w)/P(w) = P(S_i) * P(w|S_i)/P(w) \quad (2)$$

In order to make such a prediction, the appropriate model needs to be built: that is, we need to be able to estimate the above probabilities for any state and subsequence.

Representations of Models

We consider two possible models given these states: one where the set of observation symbols are all possible coherent sequences, and another where the observation symbols correspond to individual commands.

The first approach, which we term the *ideal model*, is to have all possible coherent sequences as observation symbols and a state for each combination of resources as in `(a)` of Figure 2. The size of the set Σ of observation symbols for each state in the model is then equal to the possible number of coherent sequences, bounded roughly by the number of possible actions raised to the size of an observation sequence. Each state of a combination of multiple resources can be represented as a subset of the m resources. In practice, the probability may be nonzero for a fairly small subset of Σ, but that may still be prohibitively large. This means that is may be quite unrealistic to try to obtain reliable probability values from real observations.

As an alternative (termed the *simplified model*), we've looked at using a single command as an observation symbol as in `(b)` of Figure 2, that is, a sequence through the

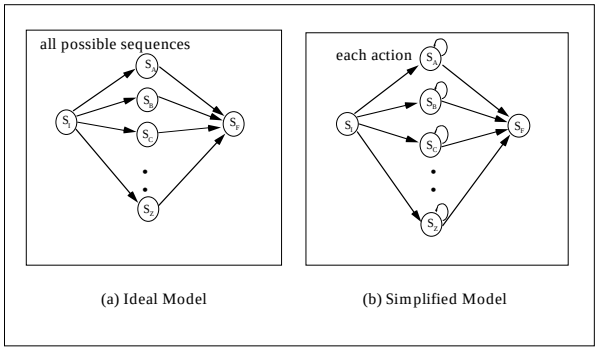

Figure 2: Possible Markov Models

states involves going from S_I to some state S_i, emitting the first command, going from S_i to S_i, emitting the second command, and so forth, until the sequence is finished. In this model, the size of observation symbol set Σ is kept to the number of possible *Unix* commands, and so the problem of obtaining (and storing) probabilities for observation symbols is mitigated, and traversing one step at a time corresponds directly with the incremental predictions.

Once we have the model and an observation w, we can calculate the probability of a state using equation (2). To do so, we need to calculate $P(S_i), P(w|S_i)$, and $P(w)$ from our model. It should be noted that in the ideal model, $a_{i,i} = 0$ for all states i, so all traversals are two transitions and one emission; the partial sequences that we observe, however, may be prefixes of the observation symbols. For both models, $P(S_i) = \alpha_i$. $P(w)$ is simply the sum over all states of $P(S_i) * P(w|S_i)$. $P(w|S_i)$, however, is calculated differently for the different models. For the ideal model, $P(w|S_i)$ is the sum of the probabilities of all strings of which w is a prefix, that is,

$$P(w|S_i) = \sum_{v:w \text{ is a prefix of } v} b_i(v).$$

For the simplified model, $P(w|S_i)$ is the product of probabilities of emitting the first symbol, going to the same state, emitting the second symbol, and so forth to the length of w. For simplicity, define $\beta_i = a_{i,i}$ (since the rest of the $a_{i,j}$'s are zero). Let $w = C_1C_2...C_n$. Then

$$\begin{aligned} P(w|S_i) &= b_i(C_1)\beta_i b_i(C_2)\beta_i \cdots b_i(C_n) \\ &= \beta_i^{n-1}\Pi_{j=1,n} b_i(C_j) \end{aligned}$$

For example, The likelihood of using a resource `'Printer'` given a sequence w= {`cd-ls-latex`}, can be computed by applying the obtained numbers from both models in the equation for $L_{Printer}$.

$$L_{printer_I} = \sum_{i=1,12,13,123} \alpha_i b_i(\text{cd-ls-latex})/P(\text{cd-ls-latex})$$

$$L_{printer_S} = \sum_{i=1,12,13,123} \alpha_i P(\text{cd-ls-latex}|S_i)/P(\text{cd-ls-latex})$$

The values of α (and β for the simplified model) are good indicators of general user behavior. High probabilities of α_i denote high use of the resources corresponding to the states and $\alpha_i = 0$ implies no use of the particular resource corresponding to the state. β (simplified model) indicates the tendency of a user's toward long or short plans: if the $\beta_i = 0$ and $\alpha_i <> 0$, then the plans corresponding to state i are always a single action; larger β values correspond to a tendency toward longer plans.

Experimental Results

Data collection for training models is done from four different users and the number of actions in each reference file varies with various periods of data collection ranging from 527 to 1,978 actions. The difficulty of obtaining subjects for experimentation needs to be stated. It might result from either gathering both commands and Unix system responses, which might expose private information or taking a willingness to install our script on their machines to collect the data. The volunteered subjects are all graduate students at two different universities.

The prediction of a user using particular resources can be made in two levels of information. One is classifying the type of a user in resource use and the other is predicting the user's future behavior to predict near–term demands on the resources. The prediction of upcoming resource uses is made with a current partial observation by answering the question of likelihood of resource usages, that is, computing the conditional probability of each resource use, given an action or a sequence of actions observed currently through the models. Once the parameters of the prediction system are obtained from the training models, the information for a class of a user, which explains how likely the user uses particular resources, can be extracted from the proportions of the user using resources. In this paper, we present only prediction accuracies of our approach and a statistical approach. The predictions of resource use given the observation of partial sequence PS are made at a certain time. The prediction accuracy is tested in different ratio of training data sets, that is, 60 to 40 and 90 to 10. Prediction hit ratio is measured by looking ahead of predicted results of testing data only knowing what likelihood of resource use has to be predicted but not knowing accurate predictions on the resources. We measured not only prediction accuracies over total number of predictions and but also prediction accuracies of resource uses only. For predicting patterns, simplified models outperform ideal models in all users and all of the 90% training models perform better than all of the 60% training models in our models.

We also examined a pure statistical approach such as n–grams [6], which has been used in general as a method for prediction problem with same data. Although the prediction problem looked at in each approach is different, that is, the pure statistical approach as in the work [2] is to predict only the next immediate action, while our problem is to predict

resource use in the upcoming next actions, we investigate the pure statistical approach and compare results to have a base line for measuring the predictability of our approach.

Statistical approaches tested for the evaluation purpose are first order *(bigram)* and second order *(trigram)* Markov chains. Except `User 1`, the 90% training model predicts better than the 60% training model. The reason for `User 1` could be a different/peculiar set of behaviors at a particular period of the 10% test set.

It is assumed to be known that having more information means better than having less information. Since trigram models have more information of one more previous action than bigram models have as history data, so it is expected that trigram models would outperform bigram models. However, bigram models outperform trigram models in predicting the next behavior. A major problem with the assumption is that of sparse data. Observing new trigrams in current observations which were never observed in the training models. Also,taking the characteristics of `Unix` domain where both non-strict temporal orderings of actions and extraneous actions are common, the reason why bigram models outperform trigram models can be explained. `User 3` has high prediction accuracy in statistical models since the behavior observed is simple and many repetitive actions in a short pattern like `(from mail)`. The results from both the statistical approach and our approach are based on the same real data gathered from same users. The overall results show that the predictions from our approach of learning patterns using correlations have higher accuracies than pure statistical models and it is particularly distinguished in predicting resource usages.

Discussion

This work is directly related to the automated induction of user models, focusing on developing an adaptive interface and learning agent for managing resources by predicting users' tendency of using the resources. This work aims to construct the regularities (patterns) of user behaviors, including user preferences, into a probabilistic model based on observations of real data. The probabilistic model is used to predict a likelihood of resource use through the plan structure.

The prediction is based on not only just one previous command but also a sequence of commands including relational information. It is a different approach from the ones using sequential information only, in that it brings the issues of interleaved actions and relevant actions in characterizing cognitive views of user behaviors. As the parameters in a HMM are emphasized for the validation of the model, the right analysis of data in the first place is very important to get reliable predictions. The expectation of this work is that if looking at the correlations among actions helps to convey possible contextual information into the data analysis. Using local contextual information has been explored based on the supportive work in machine learning area [7], [8] and [9]. Although the numbers of prediction accuracy in our approach look promising comparing to the ones in other approaches, whether the quality of data: fitness for use is actually improved is to be examined. The work in [3] tried to match the current pattern against abstracted command lines using a parser instead of actual commands and achieved degraded performance. Our reasoning to their result is that unlike our approach, where contextual information is used *temporarily* (or locally) to learn patterns of sequences of commands in a generalized sense, abstraction in their work is done to arguments of commands and by including them into patterns, they actually specify the patterns to be matched.

Iterative update and improvement of model parameters need to be done to maximize the probability of the observation sequence given in the model as more observations are made.

Acknowledgment

The author wishes to acknowledge the financial support of the Catholic University Settlement Research Fund Granted in the Program Year of 2001.

References

[1] D. Albrecht, I. Zukerman, A. Nicholson, and A. Bud. Towards a Bayesian model for keyhole plan recognition in large domains, In *Proceedings of the Sixth International Conference on User Modeling* pp. 365–376. Sardinia, Italy 1997.

[2] B. Davison and H. Hirsh. Predicting sequences of user actions. In *AAAI 98 Workshop Notes on Predicting the Future: AI Approaches to Time-Series Problems*, 1998.

[3] B. Korvemaker and R. Greiner. Predicting Unix command lines:Adjusting to user patterns. In *Proceedings of the 17th National Conference on Artificial Intelligence* 2000.

[4] T. Lau and E. Horvitz. Patterns of search:analyzing and modeling Web query refinement. In *Proceedings of the Seventh International Conference on User Modeling* 1999.

[5] D. Weld and Oren Etzioni. The First Law of Robotics;(a call to arms). In *Proceedings of the 11th National Conference on Artificial Intelligence* 1994.

[6] E. Charniak. Statistical language learning. The MIT Press, 1996.

[7] J. Allen. Natural language understanding. The Benjamin/Cummings Publishing Company, 1995.

[8] H. Ng and H. Lee. Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Examplar-Based Approach, In *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pp 40–47

[9] Y. Choueka and S. Lusignan. Disambiguation by Short Contexts. In *Computers and the Humanities*, Vol 19, pp 147–157, 1985.