

Models and Model Biases for Automatically Learning Task Switching Behavior

Hamilton Link, Terran Lane
University of New Mexico¹

Joseph P. Magliano
Northern Illinois University²

Abstract

Machine learning techniques have been applied to several kinds of human data including speech recognition and goal or user identification. When learning on such data, it is important to use models that are not strongly biased against properties of the data, or the variable assignments learned may be largely incorrect. We are working with data sources for user interface event data and examining the applicability of dynamic Bayesian networks (DBNs) to context tracking. Specifically, we identify the value and transition points of a hidden task variable; this problem is known as segmentation. Our data is drawn from command line interaction collected in a real work setting and window event traces taken during a controlled behavioral study. We have applied discrete time hidden Markov models (HMMs) and DBNs to these data sets, but these methods are fundamentally Markovian and, as a result, cannot correctly learn the properties of hidden variables with nongeometrically distributed dwell times. We believe that using semi-Markov models may better capture some underlying structure and allow for better segmentation. In this paper, we describe the experimental protocols performed, examine the bias of typical DBNs and HMMs towards geometric variable dwell times, and assess the validity of this assumption. We discuss the issues of applying semi-Markov DBNs to the available data.

1 Introduction

One strategy to improve a person's performance on computer-aided tasks is to enable the computer to recognize and respond to patterns in the user's actions. As part of this strategy, we have been researching the automatic identification of task switching from computer user logs. By building systems that detect these switches and develop likely task sets, we hope to automatically identify contexts in user interaction. We would then be able to attach semantic information to these tasks and associate responses with them, at which point reinforcement learning could be used to map the best responses from details about the state of individual users. A learning system would be more flexible than a system tailored to a particular user, while hopefully attaining comparable performance.

One problem of interest in modeling task switching is accurately capturing the probability density function describing how long a person will work on a particular task before moving on. In our experience, hidden Markov models (HMMs) can be effectively used to learn attributes of users through interface data. For example, trained HMMs are sufficient for reasonable user identification performance (Lane, 1999). Conceivably, similar models could be used to segment a data stream accurately as well, to identify a user's work on multiple tasks, but in practice it appears that accurate segmentation is hindered by the strong Markov assumption of such models. In our experiments, automatic segmentation of user data into tasks produces poor results without semi-Markov support.

We have been working with data collected at Purdue and data from a new study conducted at Northern Illinois University (NIU). The NIU data provided an empirical behavioral study of context switching in the presence of multiple information gathering tasks. This data is an application event log and includes some information about each event—for example, web page requests contain their associated URLs. Not all information from the original sessions is available—in particular document and web site content—and not all available information was used by the model. Task labels and segment boundaries are not present in the bulk of the data, so unsupervised learning was used, with one labeled data set available for validation testing. The data gathered at Purdue University is UNIX user command line interactions, collected for previous research on user modeling applied to computer security (Lane and Brodley, 1999; Lane, 2000, etc.). The Purdue data provides more general computer usage data and captures information about unconstrained user task execution patterns, in contrast to the controlled NIU study. No file content was captured. The data is sanitized and is completely unlabeled with respect to the user tasks; shell session boundaries are present in the data but were not originally used for segmentation.

¹ Dept of Computer Science, Albuquerque, NM 87131 {hamlink,terran}@cs.unm.edu

² Dept of Psychology, DeKalb, IL 60115, tj0jpm1@wpo.cso.niu.edu

This work involved the design, collection, and preparation of user interface data from NIU, and the initial efforts to automatically segment the NIU and Purdue data with HMMs and dynamic Bayesian networks (DBNs). In our initial estimation, it was hypothesized that Markov analysis would be limited by an inability to model the structured procession of a person from one task to another, and our preliminary results point to the need to more explicitly capture the temporal properties of the data. In future work we hope to leverage the domain expertise of behavioral psychology in automated learning systems.

2 Background

We base our user modeling work on the rich history of statistical user and behavioral modeling. Simple and highly popular statistical models of user behaviors are *atemporal* models such as Latent Semantic Analysis (LSA) (Foltz et al., 1998; Landauer et al., 1997) and other models derived from the “bag of words” assumption. While LSA models have proven to be surprisingly effective in clustering and classifying text documents, they explicitly do not capture temporal interactions among behaviors—precisely the class of effect in which we are interested for the goal segmentation problem. Many authors (including ourselves) have turned to statistically more complex temporal models such as Markov chains (Davison and Hirsh, 1998), hidden Markov models (Lane, 2000), or dynamic Bayesian networks (Albrecht et al., 1998; Zukerman et al., 1999). Of the three, Markov chains make the strongest assumptions about the data, assuming that all statistical information is captured in observable data and that there are no latent or “hidden” variables in the system. Such models are not a good fit for task recognition problems (such as ours) where the existence of one or more hidden variables (such as goals, emotional state, or other mental states) is assumed. User goals are typically not represented directly in observable data, but instead must be inferred from observation and knowledge of temporal dynamics.

HMMs allow a single hidden variable, allowing us to represent a single layer of structure in a goal or task. The state of the hidden variable at a given time can be inferred through the dynamic programming based forward-backward algorithm, while the model parameters can be estimated via the Baum-Welch algorithm, a special case of the Expectation Maximization (EM) algorithm (Rabiner and Juang, 1993). Often, however, we are interested in systems with multiple hidden variables or hierarchically related hidden variables such as goal/subgoal or, in a Goals/Operators/Methods/Selection (GOMS) context, the current stage of a method or operator. HMMs have been extended to hierarchical and layered models for user models (Nguyen et al., 2003; Oliver et al., 2002), but a more general approach to multivariate statistical temporal systems is the DBN model. Bayesian networks arose in the artificial intelligence community in the late 1980’s (Pearl, 1988) and have since been widely applied to temporal-process learning and inference problems (Murphy, 2002). Such models are more computationally complex to handle, but possess inference and learning algorithms closely related to the forwardbackward and EM algorithms (Huang and Darwiche, 1994).

A more fundamental difficulty, plaguing both HMMs and DBNs, is that they have a built-in bias toward geometrically distributed state dwell times. This bias arises from discrete-time sampling and an assumption of stationarity—that is, both HMMs and DBNs assume that $\Pr[\text{state}_t | \text{state}_{t-1}]$ is independent of t (ie, $\Pr[\text{state}_t | \text{state}_{t-1}] = \Pr[\text{state}_{t'} | \text{state}_{t'-1}]$ for all t and t'). As a result, the process is memory-less and the probability of remaining in a [Fig] single state for k time steps decreases geometrically in k , in contrast to many real-world temporal processes, where the probability of remaining in a particular state is a non-geometric distribution. The probability of staying in some state for k more time steps might depend on how long the process has already been in that state. For example, if a user is doing a task that takes roughly an hour to complete, then the probability that the user completes the task at any moment depends on how long the user has been doing the task. Similarly, factors like boredom thresholds can influence how long a user stays focused on a single task.

The machine learning and human computer interaction (HCI) communities have addressed problems of this nature using a variety of machine learning (ML) techniques to identify the goal or plan of the person under observation or assess a user’s cognitive load (Albrecht et al., 1998; Nguyen et al., 2003; Müller et al., 2001, etc.). We are currently using dynamic Bayesian networks (DBNs) as our learning mechanism and ultimately intend to use HCI models to structure the learning process. Comparable work in the field (Oliver et al., 2002; Gurer et al., 1995; Müller et al., 2001, etc.) has indicated that DBNs can be used to accurately infer some attributes of the work environment and the mental state of users under observation, but this work makes limited use of cognitive structure described in HCI and behavioral psychology. Particular models of interest for segmentation of the user’s activities are GOMS (Card et al., 1983), Don Norman’s seven stage model (Norman, 1986), and Barnard’s human information processor (Barnard, 1991).

3 The NIU User Data

The goal of the NIU study was to study task switching and assess human performance in several contexts. The study was conducted to provide problem solving protocols as participants solved complex and naturalistic problems that required web searches and writing documents. These problems were designed to be open-ended. That is, there were a variety of ways a participant could solve the problem. These protocols provided a basis for analyzing their goal structure, which could in turn guide machine learning. We drew upon causal theories to develop an experimental approach to gathering computer use episodes and segmenting them according to goal structures. We created open ended problems for individuals to solve using the web and Microsoft applications (e.g., Word) that would take about an hour to complete. The problems were presented in the context of hypothetical email exchanges with a relative or friend. There was a clearly stated goal provided to the participants at the outset of a session. However, the problem could change as new emails were received. In addition to the primary problem, another task was given to a participant from a different fictive person. This task was social in nature and entirely unrelated to the primary task (e.g., choose a restaurant or a movie). The purpose of this secondary task was to assess when participants would choose to solve it and provide data for domain shifts between two completely unrelated goals.

| | |
|---|--|
| <p>First message (immediately) Hey, I'm completely swamped at work today. I don't have any time to work on this week's picks. I'm putting my trust in you. Well, sort of!!!! Could you put together a list of potential players that is within the point cap? If you can get it done before lunch, I can take a look at it over my lunch break. Thanks, Jeff.</p> | <p><i>Task:</i> Go to espn Baseball Challenge website. Create a new name and password. Using the rules of the website, select 9 offensive players and pitching staff for today's team. Each player has a point value. The basic rule is that you need to select these players so that the sum of their salaries does not exceed 50 Million. Do NOT, however, enter the names of these players into the game site!!! Instead, Put the names and values of the players in a Microsoft word document. Build a brief report in Microsoft Word that justifies your choices, writing a short blurb about each player (or pitching staff) and why you have selected him for today's game. Use any kinds of info that you want here to justify your choices. Assume that your friend wants to review your choices immediately. Send him the Word file as soon as it is complete. Assume that your friend is likely to dispute your decisions because he especially likes players from the Cubs. Write your justification after you send him the Word file.</p> |
| <p>Second message (after 20 minutes) Hey, Want to see a movie tonight. What's showing in the Naiperville, Aurora, St.</p> | <p><i>Task:</i> Perform an internet search to find movie options in these areas. Create a MS word document listing potential movies and short description if possible. Then create a list of the theaters and movie times for these areas. Send this document as an attachment. Pick a movie that you would like to see and argue for that option in the body of the email.</p> |

Figure 1: Fantasy Baseball Protocol

Complex goal directed activities involve a hierarchy of goals, subgoals, and actions (Newtson, 1973; Trabasso et al., 1989; Zacks et al., 2001, etc.). Furthermore, the order in which an individual addresses the goals and subgoals of a problem is often highly systematic (Newell and Simon, 1972). Those subgoals that provide the biggest obstacle in meeting the highest order goal are typically accomplished first. To complicate matters, especially in terms of machine based recognition of goal driven activities, many goals are nested or are accomplished concurrently (Trabasso et al., 1989; Zacks et al., 2001), and the extent to which a person is able to construct a well sequenced set of subgoals varies with domain expertise. Complex goal directed behaviors are sequences of hierarchically related goal episodes (Newtson, 1973; Trabasso et al., 1989; Zacks et al., 2001). The causal network model (Trabasso et al., 1989) of story structure provided the basis for understanding the goal episodes that occurred in the sessions. According to the causal network model, story events can be classified by how they fit into an episodic structure, which consist of a set of story unit categories (Stein and Glenn, 1979). These categories include settings, events, goals, attempts, outcomes, and reactions, but for the present study we disregarded settings and reactions. Events are experienced by a character, but are not the direct result of their action (since they received unsolicited email on a fixed schedule).

We developed two scenarios that would vary in familiarity to a participant: constructing a fantasy baseball lineup (Figure 1) and finding medical information for a sick relative. The email exchanges for both problems are presented to the participant along with information regarding the timing of the reception of the fictive emails throughout the session. All of the participants were experts in fantasy sports, which involve creating hypothetical

sports teams based on real players. A team's performance is based on the performance of their players in any given week. There are a high number of websites that provide support for fantasy sports. For this problem, participants were tasked to construct a team for the current week's matchup, which required going to websites that provide information on individual players.

- | | |
|---|---|
| <ol style="list-style-type: none"> 1. Find a diagnosis <ul style="list-style-type: none"> Search symptoms Find a symptom search engine Find a new search site—nonspecific (1) Find a new search site (2) Find a new search site (3) Find a diagnosis website—nonspecific Search new site for symptoms (4) <ul style="list-style-type: none"> Register for symptom website 2. Find out about MDS <ul style="list-style-type: none"> Prognosis info Info about types Find doctors Find hospitals <ul style="list-style-type: none"> Search for hospital site, nonspecific Search specific site for MDS Search new site for MDS | <ol style="list-style-type: none"> 3. Find doctors and hospitals outside Chicago <ul style="list-style-type: none"> Find doctors Find hospitals 4. Find food 5. Sending document <ul style="list-style-type: none"> Locate saved file 6. Close down formerly relevant windows and irrelevant pop-up windows 7. Prepare document <ul style="list-style-type: none"> Extract info from internet: cut and paste to Word file Edit document: typing in info and/or editing pasted text 8. Troubleshoot computer problems 9. Open PDF file <ul style="list-style-type: none"> Put PDF link in word file 10. Killing time |
|---|---|

Figure 2: Goal structure of a participant engaged in the Health Scenario, as described by the analyst.

The medical information problem involved searching for a diagnosis for a sick relative, and is structurally similar to the fantasy baseball protocol, but involves more interruptions and a more complex search task. The disease chosen, myelodysplastic syndromes, was relatively obscure and would take several attempts to find relevant information. This problem started with an email from a uncle whose wife is exhibiting several unusual symptoms. He is waiting for a diagnosis from his doctor, but wants the participant to find one on the web. After approximately 10 minutes, an email is received with the diagnosis. The task then shifted to finding information regarding the prognosis and treatment options. With respect to treatment options, a restriction is initially placed for local treatment centers, but is eventually opened up to national treatment centers.

We collected three sets of data from each participant for each of these problems. The first was an event log of the computer activities, captured as XML objects. Participants were also instructed to think aloud as they solved the problems and to report whatever thoughts came to mind as they worked through the problems. There is a considerable amount of evidence that suggests that thinking aloud does not change the nature of complex problem solving behavior (Ericsson and Simon, 1994). The third set of data was a continuous record of the information presented on the computer monitor throughout the session. Both the think-aloud protocols and the visual record of the monitor were recorded on DVD. A running clock was used to synchronize all three sets of data. For the user-modeling work in this paper, we have focused on the most easily accessible data: the user interface event log.

In the behavioral analysis, each event recorded was categorized as an event, goal, attempt, or outcome. The goal hierarchy used is shown in Figure 2. We designed the protocols used to allow identification of the major components of the goal episodes a priori. Specifically, the email messages from the relative and friends were considered events that would initiate the goals and sub goals built into the problem and we knew exactly when these would occur in the event log. Furthermore, participants were instructed to send emails to the relative/friends when the explicitly stated goals were completed, which enabled us to identify outcome events in the log. The monitor recordings and think aloud protocols were used to make these classifications as well. For example, when a participant received the initial message in the health scenario, he might say "I need to now find a diagnosis for my relative." As such, we classified the next action as an attempt to achieve this goal. Participants often explicitly stated subgoals, such as finding a symptoms search engine. As such, we would code the next action in the event log as a subgoal attempt. We kept track of and recorded the hierarchical relationships between goals and subgoals. If participants encountered a failed attempt to achieve an explicit goal, they often articulated that failure. As such, these events in the log were coded as a failed outcome. We considered an event a successful outcome of a subgoal if the think-aloud protocols or monitor recordings indicated that it was completed.

4 The Purdue User Data

The Purdue user data was drawn from studies of user modeling applied to computer security—specifically, intrusion and anomaly detection. These studies aimed to differentiate computer users purely by their command line behavior. The data set is UNIX command line session traces from eight graduate students in the Purdue ECE department gathered in the late 1990s. The amount of data available varies among the users from just over 15,000 tokens to well over 100,000 tokens, depending on their work rates and when each user entered and left the study. Because of computational constraints and for testing uniformity, we employed a subset of 10,000 tokens from each user, representing approximately four months of computer usage. All users employed the UNIX shell `tcsh` and worked in the Xwindows environment. During the original studies, we sanitized the data for privacy by removing file and directory names, user names, email addresses, and web addresses. Command names, switches, and shell metacharacters were preserved, as were the count of arguments to each command. No attempt was made to correct typos in command names or switches, as we hypothesized that patterns of typos might, themselves, be revealing about user identity. Altogether, after sanitization, we identified 2360 unique tokens. An example of raw and sanitized data is given in Figure 3.

| | | |
|---------------------------------------|------------------------|---|
| | <code>**SOF**</code> | <code># start of session</code> |
| | <code>cd</code> | <code># command name</code> |
| <code># Start session</code> | <code><1></code> | <code># one "file name" argument</code> |
| <code>cd ~/private/docs</code> | <code>ls</code> | <code># next command</code> |
| <code>ls -laF more</code> | <code>-laF</code> | <code># command switches</code> |
| <code>cat foo.txt bar.txt \</code> | <code> </code> | <code># shell metacharacter</code> |
| <code>zorch.txt > somewhere</code> | <code>more</code> | |
| <code>exit</code> | <code>cat</code> | |
| <code># End session</code> | <code><3></code> | <code># three "file" arguments</code> |
| | <code><1></code> | |
| | <code>exit</code> | |
| | <code>**EOF**</code> | <code># end of session</code> |

(a) (b)

Figure 3 Example of Purdue data, pre- and post-sanitization. Example data is synthetic to preserve privacy, but is close in spirit to the original data. (a) Pre-sanitized data. (b) Post-sanitized data. The comment strings (`# ...`) are purely for clarification and are not present in the actual data streams see by the modeling software.

These data were gathered under uncontrolled circumstances as the users worked normally, and thus represent a spectrum of tasks and goals. All users were informed and consented to the data collection, but we found no evidence that the presence of data gathering affected user behaviors. The command monitoring and recording utility was a built-in capacity of the `tcsh` shell and imposed no noticeable latency or other overhead that would interfere with user work. We did not attempt to label any data manually, as the original study employed unsupervised learning.

In our prior work with the Purdue data, we showed that users could be distinguished purely through their behavioral patterns with reasonably high reliability (60 - 80% accuracy). Both the instance-based learner and hidden Markov model employed automatically detected natural and semantically meaningful “clusters” of behaviors in the data.

For example, we identified clusters corresponding to text editing, email, and program development. However, segmenting the data into “tasks” was a side effect and not the primary intent. To the degree that the Markov assumption did not interfere with user identification, it was not inappropriate, so we did not examine the dwelltime distribution in detail in that work.

5 Properties of UI Data

It has been noted in domains as varied as economic modeling, semiconductor process design (Ge and Smyth, 2000), control theory (Puterman, 1994), and speech recognition (Russell and Moore, 1985; Ratnayake et al., 1992; Rabiner and Juang, 1993) that many interesting time series processes are significantly nonMarkovian. Specifically, it is often the case that (hidden) state dwell times violate the geometric distribution that naturally arises from a discrete time Markov chain. We are not, however, aware of an examination of this issue for task dwell times in user modeling contexts.

We observe statistical properties in the available data inconsistent with the model bias of Markov HMMs and DBNs. By their iterative nature, hidden variables in such models can be expected to have a geometric distribution of dwell times with a shape parameter corresponding to the fixed likelihood of staying in the same state from one step

to the next. Graphing the distribution of session and command lengths for a user against MLE-fit geometric distributions, we see session length likelihoods climb to a sharp peak of a few tokens or commands before trailing off (Figures 4a and 4b). This peak is consistent throughout the Purdue users.

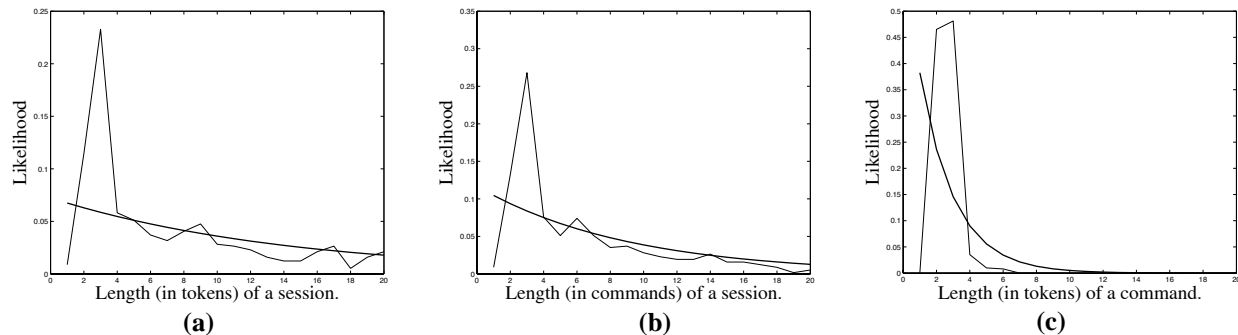


Figure 4: Comparison of geometric distributions fit to Purdue user 1’s behavior. The natural behaviors in (a) and (b) climb to an early peak before falling. The shell-constrained behavior in (c) shows that short commands with one or two arguments are the norm, with next to no tail on the distribution. (axes differ)

In contrast to the task directed length of sessions, the distribution of *command* lengths in tokens is a function of the UNIX environment. In the original study command boundaries were not used, however the distribution of command lengths does not substantially change the shapes of the curves in Figures 4a and 4b. Bearing in mind that tuples of sanitized tokens were replaced by $\langle 1 \rangle$, $\langle 2 \rangle$, etc. we see in Figure 4c that short commands of 13 tokens are all fairly common, and that extremely long commands are almost never seen.

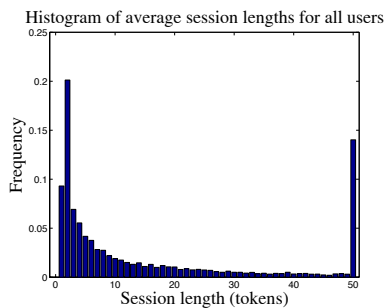


Figure 5: Mean session length distributions across users. The peak seen in User 1 is well represented in the averaged distributions. Zero-length sessions are not included in the graph. The right-most bar is all sessions >50 .

Figure 5 shows the average distribution of session lengths (in tokens) across all users in the Purdue study. Zero length sessions have been omitted from the data for this analysis, because they do not represent any particular task information because the windows environment is typically configured to create some number of terminals when a user logs in. In fact the prevalence of zero length sessions had the most variance between users, with the nine users having between five and almost one thousand empty sessions. The graph of “used” session lengths and their average frequency across users shows a robust peak in length and a smooth decline in longer sessions. Looking at the sessions in more detail, we observed a common practice of spawning off one or two independent tasks and exiting; the alternative seems to show users performing more lengthy tasks entirely within the command line. These two different behaviors and the frequency with which an individual user generates zero length sessions could be captured with a mixture model. Such a probability distribution would represent the three individual distributions (zero length sessions, single task sessions, and indefinite sessions) and their relative likelihood. Unfortunately, although this may accurately model terminal window sessions, this mixture model can not be captured by any geometric distribution.

Session boundaries in the Purdue data were available and the quantity of data lends itself to such analysis, but we would expect similar results to hold in the NIU data. Even the simplest task in a graphical user interface will require window selection followed by a spike of mouse or keyboard manipulation within a window, analogous to the creation and brief use of short terminal sessions.

6 ML Methods and Results

In the NIU data, window events are logged as XML objects. We extracted feature values from these sequences, and trained HMMs and DBNs on the resulting sequences of variable values. For the HMMs, we had an unlabeled hidden variable with two to five possible values and used event type as the observable. The DBN used a hidden task variable with eleven possible states and two observables corresponding to the event’s application and the event type (Figure 6a). We intended to have a fourth observable mapped to an LSA-chosen cluster for document related event types, but the technical details have not yet been addressed.

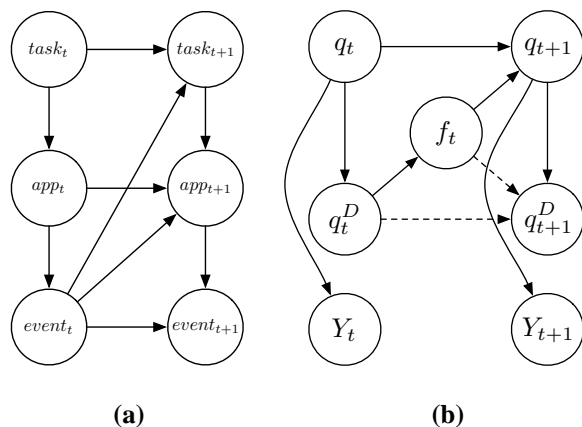


Figure 6: (a) The DBN structure trained on the NIU data set. App and event were observed for most time steps, task was an unsupervised hidden variable. (b) The DBN structure of a semi-Markov from HMM (Murphy, 2002) we plan to apply to the Purdue data set. Based on those results, a hierarchical semi-Markov HMM may be used on the NIU data.

high mutual transition probabilities, randomly dividing up the entire session (Figure 7c). Relatively common short bursts of activity, such as we see in the Purdue data, would be underpredicted and could lead to a similar result, with transitions being missed and a single state being ascribed to multiple phenomena.

To weakly corroborate this hypothesis, we returned to earlier results from a separate experiment with an HMM—distinct from the DBN that was used to process the NIU data—that had a comparable problem, with the system becoming unable to learn states that had particularly distinct behaviors. An HMM trained to predict event types in the NIU data performed no better than uniform prediction of the most prevalent type identifier. Upon investigation, we found the system bouncing back and forth erratically between the two highest potential states, which were associated with nearly identical probability distributions. This was in an early attempt to predict event classes, but upon reflection the model bias was one reason that the patterns in the data were not learned.

Labelling biases also present a difficulty when applying automated learning methods to partially annotated data. In sparse data, automated learning systems will tend to drift towards whatever patterns are in the data. In unsupervised experiments in particular, these patterns may be somewhat orthogonal to the semantically laden labels an analyst may give to data points in a validation set. Ultimately, an analyst has a lot of prior knowledge about the world, the manner in which people typically solve problems, how information that may be available in the world relates to problems being solved, etc. Take, for example, the descriptive name “register for symptom web site”, which semantically captures features about the world and how they relate to the data stream. Even in a large DBN, it would be difficult to represent that the Internet is filled with web sites that are typically comprised of many pages of interrelated material, that some of this material is medical in nature and describes the features an individual may be likely to exhibit when they have succumbed to particular pathogens, and that such information may be accessible only to seekers who have provided identifying information.

7 Conclusions

We have data gathered for two human usage studies, and there are many useful methods in machine learning that we would like to apply to this data. The methods that we originally considered using have a strong bias to interpret data as Markovian and learn properties consistent with that world view. Because the data appears to have several temporal properties that are inconsistent with the model used, we believe that segmentation is not possible with any degree of accuracy using these models unaltered. The direct approach to relaxing the Markov requirement on the data is to use semi-Markov variable structures (Figure 6b), but naïvely running semi-Markov models with a generic DBN is highly inefficient. With linear programming and other methods, it is possible to algorithmically exploit the

The HMMs were trained using the Bayes Net Toolbox (Murphy, 2004), and the DBNs using our own junction tree and EM software (Huang and Darwiche, 1994). As mentioned, this was unsupervised learning using a single validation set. The trained model was used to segment the labeled data, and we chose the heuristically best permutation to map between the abstract task variable values and the semantically laden analyst’s labels (Figure 2). The resulting labels were added as annotations to a validation data set (Figure 7a).

We applied the DBN in Figure 6a to the NIU data, but the system proved unable to segment the data into task episodes analogous to those identified by the analyst (Figure 7a). Given the scale and distribution of the specified episodes, this is consistent with our operational hypothesis. If, due to a model bias, the system imposed unnecessary task switches, the system would be forced to conclude that states joined by a highly likely transition exhibited a common, and likely incorrect, pattern of behavior. No clear distinction between those values of the hidden variable would be learned, and the system might segment a new session as we see in our results: a few states, capturing very

deterministic nature of the added variables in a semi-Markov DBN. It may also be appropriate to use approximate learning and inference methods. At the moment we are considering these options to making learning and inference tractable while preserving the ability to learn nonMarkov properties and segment the data more accurately.

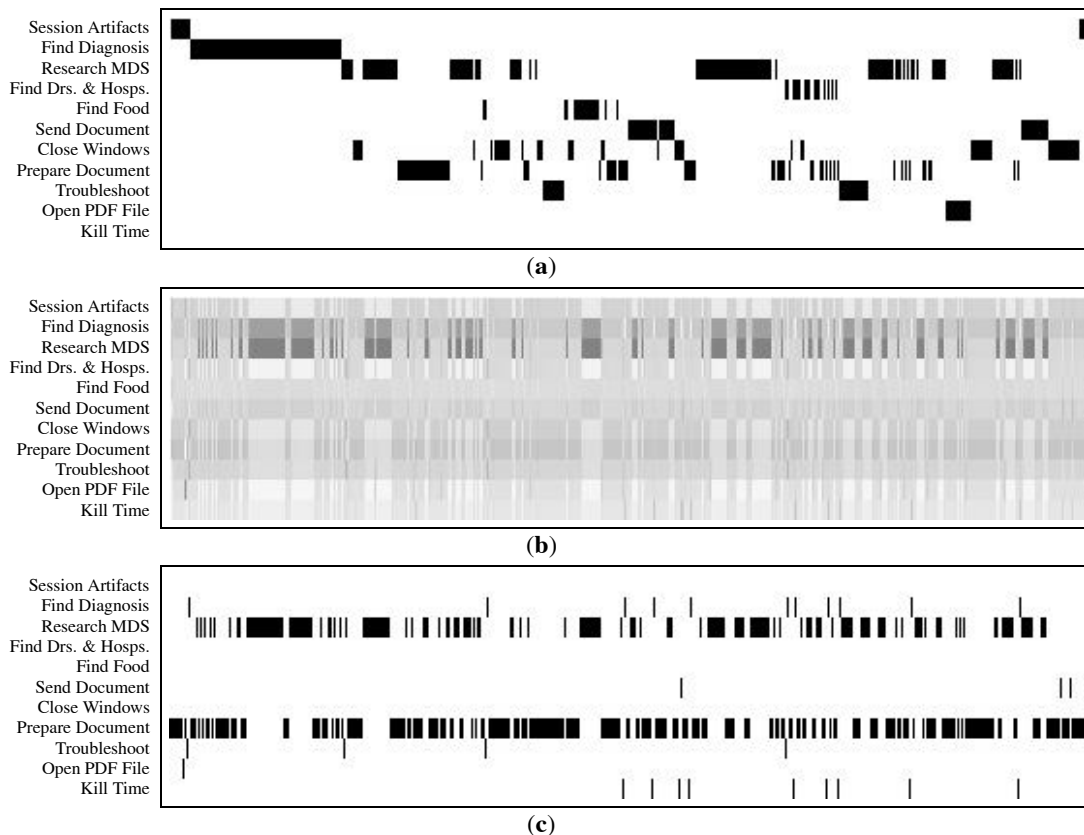


Figure 7: Task variable values over time, depicting analyst labels, inferred likelihoods, and maximum likelihood labels. **(a)** Labels assigned by an analyst to events taken for one user in the Health Scenario. **(b)** Relative likelihood of each task label, after training on unlabeled data. Relative darkness vertically means a label was deemed more likely at that time. **(c)** Labels given to events based on maximum likelihood at each time step.

8 Future Work

Adding semi-Markov support to a DBN implementation naïvely can be computationally intensive. Without algorithmically exploiting the counter’s deterministic nature, the standard junction tree for a semi-Markov model will have relatively large forward interfaces. For example, the semi-Markov HMM depicted in Figure 6b has a three-dimensional joint probability table as its forward interface, when a Markovian HMM with the same hidden variable has only a single-dimensional potential. In a semi-Markov hierarchical HMM with two levels as might be used to replace the DBN in Figure 6a, the three forward-pointing variables would be replaced by seven, all in the forward interface that determines the structure of the junction tree.

In addition to the computational burden, the addition of hidden variables with potentially high cardinality requires that more information be extracted from the available data, making sparse samples problematic. In both data sets, Purdue and NIU, task labels were not present in the data—forcing us to do unsupervised learning, and further exacerbating the need for data. In future experiments fully annotated data would be desirable, but a balance between the quantity of unsupervised data and annotated data will have to be found. One likely solution is to use richer raw data for events. Document content for the NIU protocols was originally collected for many event types, but was unavailable for this project. We are considering the feasibility of follow-on experiments that preserve such material, allowing us to use LSA or similar techniques for DBN feature extraction.

Although validating our hypotheses on user data is essential, it is difficult to control the necessary variables for an experiment and time consuming to collect enough data for machine learning work. Fortunately, supplementing

experimental data with synthetic data is possible and would allow us to study the properties of ML techniques applied to data from a known model. Using generative Markov and semi-Markov variables with the same fundamental variables and similar statistical properties, we can generate representative data sets corresponding to our beliefs about how people work with computers. If the results from small studies on real data and simulated data were analogous, it would justify studies to validate the models experimentally.

A large amount of user study data and exploitable behavioral models could come from HCI projects. We plan to ground our learning methods in these or similar models to add explanatory power. By modeling cognitive processes more explicitly, we hope to gain the ability to assess activities more accurately.

9 Acknowledgements

We thank Chad Lundgren whose editing and formatting assistance with this paper was invaluable. This work was performed in collaboration with Sandia National Laboratories and supported, in part, by the United States Department of Energy under Contract DEAC0494AL85000.

10 References

- Albrecht, D. W., Zukerman, I., and Nicholson, A. E. (1998). Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User Adapted Interaction*, 8(12):5–47.
- Barnard, P. (1991). Bridging between basic theories and the artifacts of human computer interaction. In Carroll, J. M., editor, *Designing interaction: psychology at the human computer interface*, pages 103–127. Cambridge University Press, Cambridge.
- Card, S., Moran, T., and Newell, A. (1983). *The Psychology of human computer interaction*. Hillsdale, NJ: Erlbaum Associates.
- Davison, B. D. and Hirsh, H. (1998). Predicting sequences of user actions. In *Proceedings of the AAAI98/ICML98 Joint Workshop on AI Approaches to Timeseries Analysis*, pages 5–12.
- Ericsson, K. A. and Simon, H. A. (1994). Verbal reports as data. *Psychological Review*, 87:215–251.
- Foltz, P. W., Kintsch, W., and Landauer, T. K. (1998). The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25:285–307.
- Ge, X. and Smyth, P. (2000). Deformable Markov model templates for timeseries pattern matching. In *Proceedings of the 2000 ACM SIGKDD*. ACM Press.
- Gurer, D., DesJardins, M., and Schlager, M. (1995). Representing a student’s learning states and transitions.
- Huang, C. and Darwiche, A. (1994). Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263.
- Landauer, T. K., Laham, D., Rehder, B., and Schreiner, M. E. (1997). How well can passage meaning be derived without using word order? A comparison of latent semantic analysis and humans. In Shafto, M. G. and Langley, P., editors, *Proceedings of the 19th annual meeting of the Cognitive Science Society*, pages 412–417, Mahwah, NJ: Erlbaum.
- Lane, T. (1999). Hidden Markov models for human/computer interface modeling. In *Proceedings of the IJCAI99 Workshop on Learning About Users*, pages 35–44.
- Lane, T. (2000). *Machine Learning Techniques for the Computer Security Domain of Anomaly Detection*. PhD thesis, Purdue University, W. Lafayette, IN.
- Lane, T. and Brodley, C. E. (1999). Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2(3):295–331.
- Müller, C., GroßmannHutter, B., Jameson, A., Rummer, R., and Wittig, F. (2001). Recognizing time pressure and cognitive load on the basis of speech: An experimental study.
- Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division.
- Murphy, K. (2004). Bayes Net Toolbox for Matlab. <http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html>.
- Newell, A. and Simon, H. A. (1972). *Human problem solving*. PrenticeHall, Englewood Cliffs, NJ.
- Newtson, D. (1973). Attribution and the unit of perception of ongoing behavior. *Journal of Personality and Social Psychology*, 28:28–38.
- Nguyen, N., Bui, H., Venkatesh, S., and West, G. (2003). Recognising and monitoring highlevel behaviours in complex spatial environments.

- Norman, D. A. (1986). Cognitive engineering. In Norman, D. A. and Draper, S. W., editors, *User centered system design: New perspectives on human computer interaction*. Hillsdale, NJ: Erlbaum Associates.
- Oliver, N., Horvitz, E., and Garg, A. (2002). Layered representations for human activity recognition. In *Fourth IEEE Int. Conf. on Multimodal Interfaces*, pages 3–8.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
- Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York.
- Rabiner, L. and Juang, B. H. (1993). *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, New Jersey.
- Ratnayake, N., Savic, M., and Sorensen, J. (1992). Use of semi-Markov models for speaker independent phoneme recognition. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP92)*, volume 1, pages 565–568. IEEE Press.
- Russell, M. and Moore, R. (1985). Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition. In *Proceedings of the 1985 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP92)*, volume 10, pages 5–8. IEEE Press.
- Stein, N. L. and Glenn, C. G. (1979). An analysis of story comprehension in elementary school children. In Freedle, R. O., editor, *New directions in discourse processing*, volume 2. Erlbaum, Hillsdale, NJ.
- Trabasso, T., van den Broek, P., and Suh, S. (1989). Logical necessity and transitivity of causal relations in the representation of stories. *Discourse Processes*, 12:1–25.
- Zacks, J. M., Tversky, B., and Iyer, G. (2001). Event structure in perception and conception. *Psychological Bulletin*, 127:3–21.
- Zukerman, I., Nicholson, A., and Albrecht, D. (1999). Evaluation methods for learning about users.