Transition Discovery of Sequential Behaviors in Email Application Usage Using Hidden Markov Models

William N. Robinson
Computer Information Systems Dept.
Georgia State University
wrobinson@gsu.edu

Arash Akhlaghi
Computer Science Dept.
Georgia State University
aaklagh1@student.gsu.edu

Tianjie Deng
Computer Information Systems Dept.
Georgia State University
tdeng1@student.gsu.edu

Abstract
Requirements monitors provide high-level feedback on software usage in real-time. Herein, we show how low-level monitoring can identify behavioral transitions that can be interpreted as learning transitions by a post-clinical team. One monitoring technique is to apply stochastic modeling to the software’s event stream. Herein, we show how dynamically generated hidden Markov models (HMMs) characterize sequence patterns in a software’s user-interface event-stream. We show how this is used to dynamically model a user’s usage of an emailing application. By differencing the resulting sequence of generated HMMs, the technique can identify transitions in software usage. This is important for identifying usage transitions, which occur with user learning. Herein, we show how the approach applies to monitoring an email application that has been simplified for users having cognitive impairments. The identified transitions provide the post-clinical team feedback on a user’s emailing progress. The team then uses the feedback to make adjustments to the emailing environment to further aid learning.

1 Introduction
Don is learning to email his friends. His cognitive impairment impedes his progress. Some days he readily reads and replies, as part of online socialization as well as task-based requests, like asking for a ride to the store. Other days, he aimlessly composes messages that are never sent. To progress, Don needs someone to monitor his efforts and provide feedback and guidance. Don is one of approximately 54 million individuals with disabilities, making up 19 percent of the U.S. population. More than one million adults in the U.S. are diagnosed each year with cognitive impairments (CI) due to neurological disease or trauma (e.g., traumatic brain injury, stroke, tumor, epilepsy, infectious disease). Currently, there are between 13.3 to 16.1 million Americans living with chronic brain disorders and associated CI[1]. Incidence rates are expected to rise due to the development of dementias associated with an aging population and increased survival rates associated with improved medical management[2]. In addition, approximately 4 million Americans have developmental disabilities that impact cognitive functioning [3]. Cognitive impairments prevent this large and growing segment of our society from fully integrating into society; they are unable to participate in mainstream computer-based activities[4].

Clinics do provide cognitive rehabilitation. In fact, Don benefitted from clinical treatment. Now, however, there are no funds to support Don’s continued clinical treatment. Fortunately, Don does take part in a post-clinical program that monitors his emailing efforts and provide feedback and guidance. Unfortunately, there are too many clients for such post-clinical programs to monitor closely—these are largely research or volunteer programs.

To assist the post-clinical monitoring of clients like Don, we are developing monitoring software. This is design-science research, in which we are constructing software to support and extend the theory of runtime monitoring[5, 6]. High-level property (requirements) monitoring has traditionally applied logic model-based approaches to monitor runtime properties, as summarized in section 2.3. The work herein extends requirements monitoring theory to address probabilities of sequences.

The work described herein focuses on raising alerts when Don needs help. This occurs during transitions—when Don is engaged in non-routinized, exploratory behaviors that normally occur as part of learning. Rather than look for specific events, the monitor looks for significant variations from historically normal behavior. Thus, the monitor applies stochastic modeling to identify Don’s transitional behavior.

This approach to the monitoring of learning is summarized as follows:
(1) A user uses target software in the context of learning higher-level goals, such as social interaction through email.
(2) The monitor builds models of the user’s behaviors and changes in the user’s behaviors. These models are structured according to the events generated by the software, which include user-interface events, such as compose an email message.
(3) A human analyst interprets the generated models to determine the changes in user learning.
The automation of the first two steps dramatically reduces the monitoring effort on the post-clinical team. The monitor provides notification and characterization of transitions, which vastly simplifies the work of the caregivers. Nevertheless, the caregivers must determine user goal satisfaction. Our future work will attempt to automate a portion of the interpretation task. The contribution of this research is the automated characterization of behavioral sequences as potential learning transitions, which is a prerequisite to the interpretation task.

1.1 Cognitive Rehabilitation

“In the rehabilitation context, the clinician is generally viewed as an instructor and the client as the learner”[7]. The PIE framework provides an approach for the clinician to implement rehabilitation therapies:

(1) **Planning** considers the learner, environment, and treatment program, which begins with a careful needs assessment.

(2) **Implementation** maximizes the efficiency and durability of learning resulting from the session sequences.

(3) **Evaluation** of client performance within and between sessions is critical for measuring outcomes and planning future therapy.

Improvements in assistive technology (AT) have facilitated the PIE approach. In the Think and Link (TAL) project, for example, AT provides the implementation context for email learning (via a customized email client), and provides monitoring to aid analyses of session outcomes and trends (described herein)[8-11].

The Think and Link project provides email clients that are usable by people with impairments in memory and learning as a result of brain injuries. A custom email client is continually monitored to determine how well it satisfies a client’s learning goals[10]. Improved activities of daily living (ADLs) is the overall goal, which is supported through emailing. Thus, clients learn how to email to improve their daily activities. When goal satisfaction wanes, the social-technical system of email client and human “buddies” is updated in an effort to increase goal satisfaction.

Clients typically abandon AT after a short period. In the AT context, failure is often associated with a poor fit between the client’s goals and the goals supported by the AT.

The cognitive rehabilitation field uses a goal attainment scale to specify an individual’s goals. Each goal is broken into five attainment levels, ranging from not-attained to fully-attained. For example, one TAL client, Don, had goals to be socially involved through online communications. One of his goals was to learn to email with no help. He divided this goal into: (1) not learned, (2) email with lots of help, (3) email with some help, (4) email, and (5) teaches others to email[9].

AT abandonment can be mitigated by monitoring the client’s usage of the AT. In TAL, this means monitoring how the client uses the email system, which can be done by monitoring the stream of user interface (UI) events generated by the client.

As Don uses the TAL emailing system, our monitoring software evaluates his behavior. Consequently, Don’s post-clinical team of caregivers can determine Don’s variance from his normal behavior. When Don goes through periods of low or error prone activity, the caregivers can respond quickly.

The TAL team reacts to changes in user satisfaction. The TAL team provides three kinds of adaptations: (a) adaptations to the social space, i.e., changes to the (closed) list of email buddies, (b) adaptations to the UI, and (c) adaptations to the process rules that govern emailing activities[10]. The TAL team must make the appropriate adaptations at the appropriate time to retain their clients. In TAL, if a patient does not like the system, he will stop using it, which is all too common with such assistive technologies. Thus, it’s critical to recognize transitions in user behavior.

1.2 Transition Identification using HMM

Recognizing change is a core function of monitoring. Consider a stream of UI events divided into data windows, \((w_1, w_2, ..., w_n)\). Transition identification marks each data window as either being normal or transitional; for example, \((\text{normal, normal, transitional, normal, normal} \ldots)\). Transitional behavior is historically unusual behavior, according to some measure such as statistical variance. We use the term *transitional* because the behavior is unusual and transient, and thus interesting from a cognitive or learning perspective. (See §2.1.)

In our approach, the stream of TAL email events is divided into data windows. Each window is characterized by a model, \((w \Rightarrow \lambda_i)\). Given a transition point, \(p\), that occurs between two models, \(\lambda_1\) and \(\lambda_2\), we find the difference of the models to characterize the change: \(\lambda_1 \rightarrow \lambda_2\), \(\lambda_1 - \lambda_2\) / \(t_2 - t_1\), where \(t_1 < p < t_2\). In this work, the models \(\lambda_i\) are hidden Markov models (HMMs) that characterize a user’s behavior as user-interface event sequences. Now, because of this automated differencing technique, the monitoring system can quickly identify changes in the models. Thus, caregivers can review a patient’s dashboard to see when a client’s behavior changes significantly.

Event sequencing is an important characteristic of user behavior. It’s also a challenge for automated monitoring systems to recognize event sequences and notice their changes or trends. Prior monitoring approaches do not address sequences. An approach that relies on decision tree differences cannot directly recognize changes in sequence distributions. A plan recognition approach does address the sequences found
in its pre-specified plans. It’s good for strict compliance checking; however, it ignores unmatched sequences, and thus is of limited use where there is a great variety of sequential patterns. Thus, we apply an HMM approach, which builds a stochastic model of all sequential patterns. Thus, our approach can characterize sequences in user behavior efficiently. This is a unique contribution of this research into real-time user monitoring.

1.3 Article Overview

Next, this article introduces reference theories that have guided our monitor design—these include learning, cognitive rehabilitation, and sequence mining. Section 3 presents the design of our HMM-based monitor. Section 4 presents a case-study of the monitor’s application to Don’s TAL UI event stream. The final two sections present a discussion and conclusions from this study.

2 Reference Theories

2.1 Learning & Cognitive Rehabilitation

Transtheoretical theory (TTM) is a popular stage model to explain behavioral changes. Developed by Prochaska in 1977 by analyzing theories in psychotherapy and behavior change[12], it was first used by Prochaska and his colleagues to model self-change and therapy change of individuals of their smoking behavior[13]. In this model, individual’s change is defined as a “process involving progress through a series of stages”, from precontemplation to action and maintenance[14]. This theory specifies learning states and their transitions, including relapse and recycling back to earlier stages. This state-transition view of learning is still common in more recent theories (e.g., [15]).

As introduced in Section 1.1, cognitive rehabilitation theory also considers a staged approach, but with explicit goals satisfied by task sequences. Planning and executing complex task sequences entails greater cognitive loads than simple sequences. Task complexity can be viewed as: (1) primarily a psychological experience, (2) an interaction between task and person characteristics, and (3) a function of objective task characteristics[16]. In each view, sequencing of tasks increases complexity, especially if it includes alternative tasks and subsequences[17]. The ability to think about task sequences that are lengthy and variable is an aspect of good cognitive processing[18]. Random and short task sequences are considered less indicative of good cognition than moderate to long, moderately varied (i.e., parameterized) and repeated sequences.

With regard to our emailing domain, a smaller cognitive effort by a client suggests a lack of interest, software obstacles, or troubles in cognitive processing. Trend analysis is most important. Significant changes in behavior suggest that the client is either transitioning, to a more knowledgeable (learned) state or relapsing to a prior state.

Our monitoring software records and analyzes task sequences initiated by the client. In particular, the software models sequential patterns and the probability that they fit into the current model. Additionally, the sequentially obtained models are compared to determine changes over time.

This approach recognizes:

1. Normal behavior as common, routinized, sequential patterns within a data window.
2. Transitional behavior as less common behaviors within a data window.
3. Transitions from one model of normal behavior to a new model of normal behavior (i.e., the new normal). This may be transitioning to a more knowledgeable state or relapsing to a prior state.

To automatically recognize these behavioral forms, we apply sequence modeling within a data stream mining approach.

2.2 Sequence Stream-Mining

Sequence data mining concerns analysis of events in sequence. The event data are often nominal-valued or symbolic and the goal is to discover variables and their correlations[19, 20]. This contrasts to the well-studied domain of time series analysis, which considers real or complex-valued time series of known parameters using methods such as autoregressive integrated moving average (ARIMA) modeling. Sequence mining techniques address:

(1) prediction, (2) classification, (3) clustering, (4) search and retrieval, and (5) pattern discovery. We apply hidden Markov models (see §2.1) to discover sequences.

We apply sequence mining in the context of stream mining. Stream mining aims to find interesting relationships over a sequence of data segments [21-23]. Stream mining algorithms can vary substantially from their more traditional forms: data may be analyzed incrementally rather than as a batch, old data may be discounted or removed in favor of newer data, the created model may be an approximation when compared to its traditional form[24]. A variety of techniques can be applied to stream data [25-28]—much of the work is focused on the efficiency of incrementally updating the model [29].

Detecting changes in data-streams is important for monitoring. Two types of algorithms are common: (1) distribution detection, which watches for changes in the data distributions, and (2) burst detection, which watches for sudden large and unusual changes in a data-stream. Distribution detection algorithms have two common forms: (a) data from two windows (current and reference) are compared using some distance measure, (b) a predictive model is created from a prior window and then its prediction is compared with the current
window—high prediction error indicates a significant change. We demonstrate both technique using HMMs.

2.3 Requirements Monitoring Tool Support

Runtime monitoring of software is increasingly importance to software engineering[30]. Computer published a toolkit for monitoring user-interface activities[31]. More generally, runtime analysis of properties is a growing trend in software development[32]. Goal monitoring integrates and generalizes prior monitoring technologies in support of high-level requirements. A recent article summarizes this research and presents systems demonstrating a variety of concerns and techniques that influence the interpretation of a running system[30].

Table I summarizes closely related tool support for behavioral monitoring, mostly from the RE and associated literature. The first column presents the concepts, which are defined in the second column, while the third column presents representative references.

<table>
<thead>
<tr>
<th>#</th>
<th>Concept</th>
<th>Definition</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Compliance checking</td>
<td>Identify absent events compared to specified event sequence.</td>
<td>Deviation from sequences implied by plan—see survey: [30]; Deviation from event-stream properties (e.g., sequences, data predicates, etc.)(6, 33-36)</td>
</tr>
<tr>
<td>2</td>
<td>Aggregate behavior identification</td>
<td>Identify a change in event distribution compared to a model of past events.</td>
<td>Decision tree differencing[37]</td>
</tr>
<tr>
<td>3</td>
<td>Aggregate behavior diagnosis</td>
<td>Identify the specific events (types and quantities) that have changed from one observation window to the next.</td>
<td>Decision tree differences and enumerate of associated difference metrics[11, 38]</td>
</tr>
<tr>
<td>4</td>
<td>Aggregate sequential behavior transition identification</td>
<td>Identify a change in event-sequence distributions compared to a model of past event sequences.</td>
<td>Simple difference metrics [39]; Markov probability distribution differences (this article)</td>
</tr>
<tr>
<td>5</td>
<td>Aggregate sequential behavior diagnosis</td>
<td>Identify the specific event-sequence (types and quantities) that have changed from one observation window to the next.</td>
<td>Markov probability differences enumerate as event-sequence metrics (this article)</td>
</tr>
</tbody>
</table>

Most work in requirements monitoring checks for runtime compliance with design-time properties. We call this compliance checking because the properties imply or explicitly specify a sequence of actions that shall be executed by the software.

Another approach to monitoring analyzes trends of aggregated software events. Rather than enumerating and checking all properties, this approach focuses on finding (row #2) and diagnosing (row #3) unusual events, assuming that most software events are normal. Where there are many potential acceptable event sequences, this approach can be more efficient in specification and monitoring effort. Conversely, where there a few potential acceptable event sequences, the specify-and-check approach of row #1 is more efficient.

The last two rows of Table I address the discovery of event sequences. Rows 2 – 5 are only necessary when one wants to monitor and analyze implicitly specified properties (i.e., data mining properties). If (1) all properties of interest can be specified, and (2) the closed world assumption applies, then the compliance checking approach of row #1 is appropriate. If either of these preceding assumptions are inappropriate, then the stochastic approaches of rows 2 – 5 are appropriate. The best approach may be to apply compliance checking for important properties along with stochastic anomaly analysis; together, they increase the likelihood that the monitor will identify interesting software activities.

It is important to consider the unusual characteristics of the events in our domain. Users with CI can engage is some unusual behaviors, which result data with large variance. One client periodically forgot his medication on Wednesdays, and then engaged in the unusual behavior of “drawing”, via placement of characters, within the email compose form. The sequence of edits without email sent, as well as the randomness of this episode, calls for monitoring methods that address sequences and probabilities.

Once a change and patterns is identified, the next question that arises is “what exactly did change?” This leads to the research in a row #5. Here, we show how Motif differencing can be applied to the windows associated with the transition. This approach originated in [39] (row #4). That approach only provided simple counts of the sequential patterns discovered. Herein, we show how the probabilities of such patterns can also be obtained.

From the domain of credit card fraud detection, Srivastava et al. uses HMMs in a manner similar to that presented herein[40]. To detect fraud within a client’s transaction stream, a single HMM characterizing past transaction is created. Now, as the client uses her credit card, a window on the growing transaction sequence is incrementally compared with the HMM—a low acceptance probability suggests the presence of fraud. (See equation (8 ) in the following Section 3.2.) Our work differs in that: (1) we dynamically create HMMs with each data window, and (2) we apply two different techniques to detect distribution changes.

3 Monitor Design

3.1 Markov Modeling

A hidden Markov model (HMM) is a stochastic signal model[41]. In our application, the signals are sequences of
Compute the probability that an observed sequence, $O$, is represented by a HMM, $\lambda$ (using the Forward-Backward Procedure[42]).

Adjust the parameters of a HMM, $\lambda$, to maximize the fit to an observed sequence, $O$ (using the Baum-Welch Algorithm[43]).

Compute the optimal HMM state sequence that best explains an observed sequence, $O$ (using the Viterbi Algorithm[44]).

In this presentation, we show how applying the first two algorithms help to identify unusual (transitional) behaviors from an event stream.

As an introduction to the following formalism, keep in mind that an HMM specifies $N$ probability models (one for each state $S_i$) for event sequences (where each event is an observation $O_t$). A HMM can be characterized as follows:

- $N$ is the number of states in the model, $S = \{S_1, S_2, \ldots, S_N\}$, where $S_i$ is a state. Time $t$ (1) denoted by $q_t$.

- $M$ is the number of distinct observation symbols per state; the set is denoted as $V = \{V_1, V_2, \ldots, V_M\}$.

The state transition probability matrix $A = \{a_{i,j}\}$, where $a_{i,j} = P(q_{t+1} = S_j | q_t = S_i)$, $1 \leq i \leq N$, $1 \leq j \leq N$; $t = 1, 2, \ldots$ Where any state can reach another state in a single step, we have $a_{i,j} > 0$ for all $i,j$. Additionally, $\sum_{j=1}^{N} a_{i,j} = 1, 1 \leq i \leq N$.

The emission (aka observation symbol) probabilities matrix $B = \{b_j(k)\}$, where $b_j(k) = P(V_k | S_j)$, $1 \leq j \leq N$, $1 \leq k \leq M$ and $\sum_{k=1}^{M} b_j(k) = 1, 1 \leq j \leq N$.

The initial state probability vector $\pi = \{\pi_i\}$, where $\pi_i = P(q_1 = S_i)$, $1 \leq i \leq N$, such that $\sum_{i=1}^{N} \pi_i = 1$.

The observation sequence $O = O_1, O_2, O_3, \ldots O_R$ where each observation $O_t$ is from $V$, and $R$ is the number of observations

We use the common notation $\lambda = (A, B, \pi)$ to indicate the complete set of parameters of the model, where $A$ and $B$ implicitly include $N$ and $M$.

In our HMM application, the symbols of the observation sequence contain two email event types, $V = \{write, read\}$, where write is an email composition. (An HMM can efficiently handle hundreds of observation types.) The states are the two learning states of the client, $S = \{normal, transitional\}$. HMMs typically have a few states, with about 10 being the maximum to handle efficiently. The states represent contingencies under which different probabilities are observed. Here, we assume two sets of probabilities, based on the user’s learning state—either normal learning, or transitional learning.) We use HMM algorithms to construct and compare HMMs representing the stream of observed email events.

### 3.2 Identifying Transitions

Transition identification detects significant changes in modeled events between consecutive windows of event data. HMMs can be used to identify transitions by: (1) comparing consecutive HMMs generated from the observation sequences, or (2) comparing consecutive acceptance probabilities. We elaborate these two techniques through a series of equations, presented next.

Consider how an HMM may be applied to an observation sequence to compute an acceptance probability. The acceptance probability $\alpha_1$ is defined as follows: $\alpha_1 = P(O_1, O_2, O_3, \ldots O_R | \lambda)$. The acceptance probability of the next overlapping sequence of length $R$ is $\alpha_2 = P(O_2, O_3, O_4, \ldots O_{R+1} | \lambda)$. Now, consider comparing the two acceptance probabilities:

$$\Delta\alpha = \alpha_2 - \alpha_1$$

$$\%\Delta\alpha \equiv \frac{\Delta\alpha}{\alpha_1}$$

We can also consider non-overlapping sequences (aka data windows). Let the probability $\beta_2 = P(O_{R+1}, O_{R+2}, \ldots O_{R+2}) | \lambda)$. The preceding equation (10) is technique 2, of comparing acceptance probabilities of consecutive data windows. The technique applies as follows:

1. Observation sequence $O_1$ is observed, $\alpha_1 = P(O_1 | \lambda_1)$
2. Observation sequence $O_2$ is observed, $\beta_2 = P(O_2 | \lambda_1)$
3. $\%\Delta\beta$ is computed
4. If $\%\Delta\beta \geq \text{Threshold}_\beta$, then a transition is recognized

Technique 1, of comparing consecutive HMMs, is computed as follows:

$$\Delta\lambda = \lambda_2 - \lambda_1$$

To find the distance between two HMMs, we apply the widely applied Kullback-Leibler algorithm[45]. What’s noteworthy here is that there are two different HMMs generated from two consecutive observation sequences. To see why, consider how each HMM is generated from an observation sequence.

Given an observation sequence, $O$, the emission probabilities matrix, $B$, is generated from the observed distribution of the symbols from $V$ found in $O$. In our application, this is simply done by counting the email event, $V$, to determine their distribution in $O$ and then adjusting the probabilities according to the state transition probability matrix $A$. Thus, each observation
sequence, $O^i$, is modeled by an HMM $\lambda_i = (A, B, \pi)$, where $A$ and $\pi$ are given, and $B$ models the probabilities of $O^i$. Each HMM is improved through optimization, which updates $A$, $B$, and $\pi$ for a better fit with $O^i$.

A HMM $\lambda$ can be optimized to maximize the probability of an observation sequence, $O$. Let $\lambda = \max P(O|\lambda)$, the Baum-Welch algorithm is widely applied algorithm to maximize $P(O|\lambda)$, which we use. Thus, each observed sequence results in an optimized model: $\lambda_1 = \max P(O^1|\lambda_1), \lambda_2 = \max P(O^2|\lambda_2), \ldots, \lambda_n = \max P(O^n|\lambda_n)$.

The technique 1, of comparing consecutive HMMs, applies as follows:

1. Observation sequence $O^1$ is observed, $\lambda_1 = \max P(O^1|\lambda_1)$
2. Observation sequence $O^2$ is observed, $\lambda_2 = \max P(O^2|\lambda_2)$
3. $\lambda_\lambda$ is computed
4. If $\lambda_\lambda \geq$ Threshold, then a transition is recognized

Technique 1 directly compares two HMMs, each generated from observation sequences; the consecutive distributions of $O^1$ and $O^2$ are compared via their representative HMMs, $\lambda_1$ and $\lambda_2$. Technique 2 compares the acceptance probabilities of the observation sequences using the first HMM. The two techniques produce similar results—when two observation sequences are significantly different, both $%\Delta B$ and $\Delta \lambda$ will be above their thresholds.

### 4 The Markov Stream-Mining Approach

Having described the use of HMMs in modeling event streams and identifying transitions, we next show how we apply the approach to analysis of the TAL email client’s, like Don from the Introduction section.

#### 4.1 TAL Data Stream

The TAL email client provides an automated custom logger. To obtain real-time data access, a log file can be monitored. Here is a simplified entry from the log:

```
09:48:41 NewMailEvent [id=765406159;in-reply-to=311149530;chars=770;words=179;sentences=16]
```

This logged event specifies the time, the program event, and its associated arguments. The example logs the arrival of a new email that is in reply to previous e-mail; the identity of the sender and receiver and characteristics of the email message, such as its length, are also included. The significant event types are: read email, compose email, delete email, and new (arriving) email. (For this study, we focus only on read and compose.) The logged events also include mouse movement and other events, so the dataset if very large. The dataset for one client, Don, included 3,695,086 records occupying 737 MB in Microsoft SQL Server 2005. These data are assumed for the remaining discussion.

#### 4.2 Markov Model Parameters

User behavior depends on the state they are in, according to the reference theories of cognitive rehabilitation and learning (§2). In particular, the probabilities of their actions differ with their state. Because some states of theory are contemplative, non-action-taking states, we only consider two user states: $S = \{normal, transitional\}$. These state are consistent with the reference theories (§2) and allow us to focus on the monitoring needs of the post-clinical team, i.e., identifying when learning transitions occur. The states effectively partition event sequence probabilities into two sets: normal and transitional.

We selected the observation symbols from the email event types. Although there are a variety of types, we selected $V = \{compose, read\}$. These are the focus of the email learning. For example, in the simplified system, there is no reply event. Instead, read message from buddy $X$, followed by compose message to buddy $X$ is in effect a reply. Thus, routinized sequences of read and compose indicate normal behavior, while more randomized sequences suggest transitional behavior.

For the initial state probability vector, $\pi$, we chose $P(normal) = 0.9$, $P(transitional) = 0.1$. We derive these numbers from our interpretation of the reference theories (§2), which show that learning is comprised of mostly normal behaviors interleaved with transitional learning episodes of less routinized behavior. Optimization updates $\pi$ and the emission probabilities matrix, $B$, to fit the observed sequence.

The emission probabilities matrix $B$, is the final HMM parameter. The emission probabilities are generated from the observed distribution of the symbols in the observation sequence. For example, an emission probabilities matrix $B = $ |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read</td>
<td>Compose</td>
</tr>
<tr>
<td>Normal</td>
<td>0.35</td>
<td>0.65</td>
</tr>
<tr>
<td>Transition</td>
<td>0.65</td>
<td>0.35</td>
</tr>
</tbody>
</table>

These probabilities are obtained by counting the email event types, $V$, to determine their distribution in $O$ and then adjusting the probabilities according to the state transition probability matrix $A$.

In summary, our $\lambda = (A, B, \pi)$ is parameterized as follows:

- The states in the model, $S = \{normal, transitional\}$, the two learning states of the client
- The observation symbols $V = \{write, read\}$, which are events from the email system
- The state transition probability matrix $A = $ |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
<td>Transition</td>
</tr>
<tr>
<td>Normal</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>Transition</td>
<td>0.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>

- The initial state probability vector $\pi = [P(normal) = 0.9, P(transitional) = 0.1]$
- The emission probabilities matrix $B$ is generated from each window of event data.
4.3 Streaming Email Markov Models

HMMs are created automatically for each data window, as they arrive from the data stream. After optimization, the two techniques from Section 3.2 are applied to identify transitions.

In stream mining, when the buffered stream data has more than window-size elements, data is transferred to the data window and processed. This process is repeated indefinitely. We selected a data window size of two-weeks, based on pre-testing and prior analysis of the TAL data[11, 37-39]. This window size provides sufficient data for accurate modeling, a reasonable period to represent client behavior with the modeled states (i.e., time enough for normal or transitional behavior), and sufficient time to provide meaning feedback that the client is transitioning.

Within the two-week window, the data is divided into 4-hour segments. A data chunk aggregates like segments by weekday. Thus, the first Sunday segment (12 AM – 4 AM) is aggregated with the second Sunday. This provides (6 inter-day segments) x (7 days) = 42 data chunks. A HMM is created from each chunk. The reasoning behind the chunking is twofold: (1) it is the same as our prior modeling using decision trees, and thereby allows for direct comparison[11, 38], and (2) it assumes that behavior is mostly consistent by segment and day-of-week (i.e., clients do similar behaviors within a 4-hour period each day of week).

As a final step of processing, each HMM is improved through optimization using the Baum-Welch algorithm.

4.4 Transition Graphs

As presented in Section 3.2, transition identification detects significant changes in modeled events between consecutive windows of event data. Recall that a transition is a significant variation from one model of normal behavior to a new model of normal behavior (i.e., the new normal). This may be transitioning to a more knowledgeable state or relapsing to a prior state. We apply two HMM-based techniques: (1) comparing consecutive HMMs, and (2) comparing consecutive acceptance probabilities. Any change above a threshold is a transition, i.e., $\Delta \lambda \geq \text{Threshold}$ or $\% \Delta \beta \geq \text{Threshold}$. Figure 1 graphs the result of using the Kullback-Leibler distance to compare HMMs (i.e., $\Delta \lambda$). The x-axis represents the comparison of the HMMs generated from the data chunks: (6 inter-day segments) x (7 days) = 42 data chunks, for 104 weeks (2 years) of two-week windows is 2,142 comparison points. Each point represents the comparison between two HMMs for the same segment from each bi-week; for example, Sunday segments 1 of weeks 1 and 2 compared with Sunday segments 1 of weeks 3 and 4.

The graph details of Figure 1 are not intended to be readable. Instead, notice the general trend. Many data points (the differences) are near 0—the average is 1.9. Nevertheless, there are 47 (of 2,142) points greater than 25. This suggests that the HMM differencing approach is good at discriminating unusual periods of sequential behaviors from the more common background.

4.5 Evaluation

Figure 2 also presents the HMM differencing approach ($\Delta \lambda$, technique 1). However, rather than showing the bi-weekly segment differences between HMMs for the 42 data chunks, the bi-weekly differences are totaled. The resulting line graph shows the aggregate bi-weekly differences. These differences allow us to consider how the two-week data windows are changing over time. Figure 2 also shows the differences of consecutive acceptance probabilities ($\% \Delta \beta$, technique 2), graphed as a dotted line. The graph scaled at 10 times (e.g., $10 \times \% \Delta \beta$) to aid visual comparison.

Notice that both $\% \Delta \beta$ (technique 2) and $\Delta \lambda$ (technique 1) have similarly shaped graphs; however, $\% \Delta \beta$ is damped compared to $\Delta \lambda$, especially for small differences. This is because of the multiplication of the small probability values ($< 10^{-2}$). In contrast, $\Delta \lambda$ uses Kullback-Leibler comparison, which is less sensitive to the small probability values. Thus, $\Delta \lambda$ appears to be more sensitive, although its computation for large HMM is slightly more complex.

The bi-weekly HMM differences can be compared to prior bi-weekly difference analysis conducted using decision trees[11, 37-39]. Figure 2 shows the bi-weekly decision tree differences as an area graph, for the same data. In comparing the HMM and decision tree differences, we see similar peaks, indicating that they both find differences in the same weeks (e.g. points 5, 13, 17, 26, 33, 37, 39, 43, 48). However, the HMMs and decision trees model the data differently.
The HMM models the distribution of event sequences of length two and greater. The decision tree models the distribution of events regardless of their sequencing. Two different distributions of event sequences may contain the same distribution of unordered events; thus, a change in the HMM model does not necessarily imply a change in the decision tree. A change in a decision tree model is likely to result in a change in the HMM. This arises simply from the fact that the decision tree model is indifferent to the sequencing that is central to the HMM.

Prior work has manually and automatically identified transitions in the client behavior for the data analyzed here[11, 37-39]. Figure 2 shows that the HMM transition analysis discovers transitions similar to the past studies, indicating that this approach is valid. (Compare the peaks of the area chart and the line graphs.) This seems reasonable, given that we rely on well-known algorithms that identify sequence distributions and calculate their differences.

To confirm that there are indeed differences, we analyzed event sequences for each month of data associated with a significant peak in the HMM graph of Figure 2. We applied Motif analysis.

The Max Motif algorithm efficiently enumerates all maximal motifs in an input string[46]. Algorithm parameters consider the minimum occurrence threshold for pattern consideration, and the maximum of variables (aka wildcards) within the pattern. For example, B*AB**B is a pattern of length 7 with max wildcard length of 2. If the minimum occurrence threshold is 3, then this pattern will only be reported if it occurs 3 or more times in the input. The algorithm only reports on maximal motifs, which is a motif that is not properly contained by other motifs.

To discover sequence patterns, we applied Max Motif to each window[39]. Motif analysis confirmed that identified transitions reveal changes in event sequences. For example, Figure 2 shows that point 26 has a transition. Point 26 compares the combined weeks 51 and 52 with the combined weeks 53 and 54. Figure 4 shows the read and write totals and the average

Figure 2 Markov model differences (dashed line, scaled left) and probabilities differences (dotted line, 10x scaled right) over decision tree differences (area chart, scaled right).
sequence length per window around transition point 26. All three factors decrease at point 26 (i.e., fewer reads, writes, and shorter emails lengths), confirming that something of interest occurred around point 26. We know that the client did less emailing. We just don’t know why.

Motif analysis combines well with HMM analysis. For example, from the Motif analysis, Figure 3 graphs non-variable pattern lengths per window, revealing a slightly increasing length[39]. Thus, it appears that the client, Don, is increasingly using sequential patterns, implying increased expertise in emailing.

5 Discussion

In this article, we showed how dynamically generated hidden Markov models (HMMs) characterize the distribution of sequence patterns in a software’s user-interface event-stream. By differencing the resulting sequence of generated HMMs, this approach can identify transitions in software usage. This has been important for identifying behavioral transitions in clients with cognitive impairments as they learn to use their customized email system. When such transitions occur, caregivers provide assistance to ensure the client is not relapsing and encouragement to aid a progressing client.

We verified our approach by comparing the transitions it identified with those identified with other, independent techniques (e.g., decision tree differencing)—see Figure 2. Additionally, we applied the approach to four other clients, for a total of about 10 years of data. After reviewing the transitions, we find that it efficiently finds significant transitions from voluminous stream data. Members of the TAL team validated that the transitions were significant periods in each client’s learning.

This stochastic approach to monitoring provides the following benefits:

- Discovery of sequence patterns and their probabilities of occurrence over streamed data windows
- Recognition of transitions to new distributions of patterns, which can be interpreted as new learning states
- Integration with diagnostic mining methods (e.g., MaxMotif), which characterize the exact nature of the transition, through presentation of the pattern differences

These monitoring features, when used in conjunction with compliance checking approaches, provide for a more comprehensive approach to runtime monitoring.

Future research will need to address the issue of learning durability. That is, does the client continue to apply the learned behavior, or is the learning lost? Additionally, the monitor should interpret its results in terms of user goal satisfaction. Addressing these issues will further simplify the monitoring activity so that caregivers can devote more time to caretaking and less to data analysis.

6 Conclusions

Software execution generates event sequences that should comply with its requirements. Hidden Markov models (HMMs) are good for characterizing the distribution of sequence patterns. Thus, HMMs are well suited to characterize software events. This is especially true for monitoring human activities where a threshold of variance can be established to distinguish acceptable from worrisome behaviors; thus, it appears well suited to health-care monitoring.

HMMs can be used to characterize software usage, look for unusual behaviors, and guide diagnosis of significant behavioral changes, as we demonstrated.
inconsequential. Conventional monitors do runtime compliance checking of specified properties. Such monitors can increase their event coverage and analysis by analyzing statistical models, such as those represented in HMMs. Statistical models can also discover unanticipated legal ways in which users employ software. Thus, HMMs are a powerful technique to consider when designing a monitor.

7 References


