Monitoring Behavioral Transitions in Cognitive Rehabilitation with Multi-Model, Multi-Window Stream Mining

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

Arash Akhlaghi  
Computer Science Department  
Georgia State University  
aakhlaghi1@student.gsu.edu

Abstract

This paper describes how quality metrics over stream-mined models can identify potential changes in user goal attainment, as a user learns a personalized emailing system. A sequence of mined models is generated from sequential segments of logged user email commands. When the quality of some models varies significantly from nearby models—as defined by quality metrics—then the user’s behavior is flagged as a potentially significant change. This paper describes how this technique works in its application on a case study of cognitive rehabilitation via emailing.

1 Introduction

Approximately 53 million individuals have disabilities in the U.S., according to the Census Bureau. 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]. In the coming years, incidence rates are expected to rise due to the development of dementias associated with a rapidly aging population and increased survival rates associated with improved medical management of neurological impairments[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].

Assistive Technology (AT) should offer great promise for supporting people with CI in their use of computers. However, research on effective technology design that facilitates long-term adoption by individuals with acquired and developmental CI is sorely lacking. The vast majority of AT research and development activities have focused on people with physical or sensory deficits[5-8]. The literature that does exist lacks systematic examination of issues of long-term adoption. If individuals with CI are to fully participate in society, technology development must take into account the complex factors related to the design of assistive technology for long-term use.

Studies have found that AT systems are abandoned by CI users at shockingly high rates[9-11]. One of the major causes of abandonment is an eventual misalignment with: (1) user goals and abilities and (2) the functionality delivered by the system. This work, in part, supports the monitoring of this relationship between user goals and their satisfaction by the system. In our case study, CI users are given in email system to aid their cognitive rehabilitation. At appropriate times, the system is adapted to meet the changing needs of the user. By monitoring a user’s event stream, we can notice changes in user behavior that indicate that the system should be adapted.

1.1 Goal Attainment Scales

The cognitive rehabilitation field uses a goal attainment scale to specify the individual goals and desires of a person. Each goal is broken into a set of attainment levels to provide a measure of attainment. For instance, a goal might be to ‘be able to do shopping for meals’, with this broken into degrees of attainment, e.g., can shop for all meals, can shop for special meals, etc. Using this style, individual CI user goals are specified. Each user is asked to first list a goal and then five levels of attainment, ranging from not-attained to fully-attained. For example, one of Don’s goals is to be socially involved through online communications. One of his goals was to learn to email with no help. He divided this goal as follows:

- Level 1 (not attained): will not be able to learn how to use email.
- Level 2: can email, but only with lots of prompting and help.
- Level 3: can email, with some prompting and help.
- Level 4: can email with no prompting and help.
- Level 5 (fully attained): can teach others how to email.

It is important to note the temporal nature of these goals, and in particular, the division of attainment levels into
milestones. This is the norm in clinical fields: individuals state long-term goals and then work towards them.

1.2 Monitoring and Adaptation

Clinicians monitor patient attainment levels. When a patient attains mastery and at one level, they are encouraged and supported to attain the next level. When goal achievement involves software, then software monitoring can aid the analysis that determines when to transition a patient from one level to the next. Adaptive AT software applications are moving from research to practice. For example, the emailing system from our case study supports a variety of multi-level clinical goals. When a patient masters one level of emailing, the emailing system can be configured to provide greater capability and choices to the user, thereby supporting their path towards ultimate goal achievement.

A monitor is a software system that observes and analyzes the behavior of another (target) system, determining qualities of interest, such as the satisfaction of the target system’s requirements[12]. A monitor can determine the status of user goals from a stream of inputs (INmon). A monitor can be characterized as a function that processes its input data stream to determine the status of goals: MON(INmon) → Sat(GOAL).

Given a patient using the emailing system, we wanted to determine if it was possible to identify potential changes in user behavior that could aid the analysis about when to transition a patient. Previously, aggregations of functional goals have been monitored—for example, ‘the average number of emails per day shall be greater than 10’. However, clinicians are also concerned about unspecified behaviors that can have an impact than on a transition decision. For example, one user composed an unusual number of email messages on Wednesdays. It turned out that the patient was periodically forgoing their medication. Thus, we turned to data mining to augment the transition decision information. We wanted to know if data mining could uncover new behavioral patterns against the background of typical behaviors.

This paper describes how quality metrics over stream-mined models can identify potential changes in user goal attainment for the Think-and-Link (TAL) emailing system. A sequence of mined models is generated from sequential segments of logged user email commands. When the quality of some models varies significantly from nearby models—as defined by metrics—then the user’s behavior is flagged as a potentially significant change. This paper describes how this technique works and its application on a TAL case study.

1.3 A Monitoring Case Study

Our home health-care emailing case study provides an illustration of monitoring software quality requirements. The Think and Link (TAL) project provides personalized email clients for people with impairments in memory and learning as a result of brain injuries[13-15]. These email systems are continually monitored to determine how well they satisfy clinical and personal user goals[15]. When satisfaction wanes, the system is updated in an effort to provide higher-quality software.

Assistive technology (AT), like TAL’s emailing system, is almost always unsuccessful. Patients typically abandon AT after a short period. In the AT context, goal failure is often associated with a poor fit between the patient’s goals and the goals supported by the AT. Remarkably, the TAL patients continue to expand their email system usage. By monitoring user goals, which serve as a proxy for user needs, TAL developers can responsively react to changes in goal satisfaction.

Computer-supported monitoring provides a cost-effective means to assist those requiring personalized healthcare. Cognitive impairments, for example, are expected to grow substantially over the next decades. These include autism and various forms of dementia, such as Alzheimer’s and trauma-induced brain injury. Many of the cognitively impaired (CI) can only function well with assistance. For electronic communications, this means vastly simplified and personalized email, PDA, and telephone systems.

In a recent TAL study, software monitoring was used to assess the satisfaction of clinical goals for a small group of CI patients[15]. As part of cognitive rehabilitation, the patients were given goals of communicating through a limited and personalized emailing application[16]. Software monitoring was used, in part, to track the clinical goals. Automated user-goal monitoring techniques were found to be more efficient than a manual approach[15].

1.4 Email Goals

Figure 1 illustrates a user-goal model for emailing. The figure presents hierarchical relationships among a particular user’s emailing goals. The user, Don, wants to use email to ‘engage in online social communication’. This need is shown as the top-most, root goal in Figure 1. Supporting emailing subgoals, such as ‘composing and sending consistently’, are shown below the root. Through standard goal-oriented requirements engineering (GORE) methodology[17], an operational model, including objects and their operations, can be derived.

The goal graph is typical in that the leaves specify specific monitored events, while the high-level goals
aggregate the lower goals, using logical relationships, sum, average, etc. In the case of TAL, user (patient) goals are towards top of the graph, while clinical goals are towards the bottom—many clinical goals are aimed at achieving the patient’s goals. The last three goals of the preceding section are shown toward the bottom of Figure 1.

1.5 Identifying Goal Transitions

Prior research has shown how goals, expressed as logical properties, can be monitored. However, determining when users are ready to transition between the goals of a learning goal-tree is largely unexplored. We are looking for a change in user behavior—more specifically, a significant change in behavior that leads to new goal attainment or the loss of goal satisfaction (i.e., forgotten behaviors).

Consider a user goal set, \( G_i \), which specifies the currently attained goals. We want to know when a user transitions to new goal set, \( G_j \), where the difference between the user goal attainment is one goal, i.e., \( G_j - G_i = g \).

We assume that around the time that goal transitions occur, a user’s behavior becomes less consistent. Initially, a user engages in a stable set of behaviors. Next, as a user tries to attain a new goal, he or she will engage in new behaviors. Thus, transitions may be associated with new behaviors. By monitoring the stream of user events, we can identify potential transitions by identifying unusual patterns of behavior against a common background. We apply a data-stream mining approach to identify unusual behavioral patterns and specific metrics to select those most likely to be associated with goal transitions.

1.6 Related Research

Runtime monitoring of software for specified behaviors is increasingly of importance to software engineering. Computer published a toolkit for monitoring user-interface activities[18]. More generally, runtime analysis of monitors is a growing trend in software development[19, 20]. Requirements monitoring integrates and generalizes prior monitoring technologies in support of high-level user or system requirements[21]. Here is a partial list demonstrating a variety of concerns and techniques that influence the interpretation of a running system.

- Functional models describe the states and transitions allowed by system[22, 23].
- Goal models typically augment a functional model to describe user behaviors that can be refined to operational specifications of system behaviors[24].
- User models describe user goals and capabilities, including cognitive impairments or software skills[15].
- Anti-models describe undesirable behaviors, such as attackers, to support analysis of anticipated problems[25].
- QoS models describe qualitative measures of systems, which may be included in a goal model[26].
- Diagnostic models infer causes of property violations[27].
- Architectural and related models describe software architecture, design, and code models, often in support of QoS analysis[28].
- Evolution models describe potential changes to the

![Figure 1](image.png)  
Figure 1 Some of Don's emailing goals, as depicted by an Eclipse goal modeling plugin.
system, typically focusing on software changes; however, user changes, such as skill learning, are possible. Evolution models can be associated with QoS and or cost models[29].

- Discovery models, such as data mining or learning, enables dynamic discovery of behaviors, such as hacking attacks, as well as predictions of eventual property violations and successes [30].
- Decision models, such as utilities, use the other models to select an appropriate evolution path[31].

Although these techniques do address monitoring, they do not recognize deviations from a stream of relatively stable behaviors. The novelty in our approach to monitoring derives from the use of data stream mining. Gaber presents a recent review of the literature [32]. A variety of techniques can be applied to stream data [33-36]–much of the work is focused on the efficiency of incrementally updating the model [37]. Phua [38] addresses the related issue of recognizing 'spikes' in the data stream. Our work is unique in using analysis of a continuous, model-quality function to predict significant behavioral changes.

2 An Approach to Recognizing Behavioral Transitions

Our general approach to discovery in user-goal analysis is that of a classic data mining approach. We include two slight variations. First, the logged events are processed with stream-mining methods. Second, changes in model quality are used to suggest changes and user behavior. Overall, the approach automatically models user behavior based on events and recognizes significant changes in user behavior.

The data miner component is part of a larger monitoring framework. As an illustration, consider monitoring a cognitively impaired (CI) patient in his use of a simplified email client. (All of the examples herein are drawn from real, clinical case studies.) The patient should email more than one person each day according to the clinicians’ treatment plan.

1. Goals are specified formally, by a software analyst, to allow for automated monitoring. Goals are formalized in OCL, a temporal-variant of the Object Constraint Language. Goals may be marked for data mining (in additional to propositional analysis).
2. The analyst uses a development environment (IDE) to generate automated monitors from the formalized goals and configures the email client to provide notifications of significant events, such as an outgoing email.
3. As the patient uses the email client, significant events travel from the client to the analyzer, which calculates goal satisfaction. Propositional goals may be updated with each event, while data mined analysis is updated according to specified windows.

To discover patterns in data, Data Mining and classically applies the following steps[39].
1. Data cleaning, to remove noise and inconsistent data
2. Data integration, to combine multiple data sources
3. Data selection, to retrieve relevant from the database
4. Data transformation, to provide consolidate data into forms appropriate for mining
5. Data mining, the central process methods extract data patterns
6. Pattern evaluation, to identify the interesting patterns
7. Presentation, to present the mined information to the user

Our approach applies these classic steps; however, step five includes stream-mining methods. In step six, we use model quality to recognize significant behavioral changes, and thus infer goal transitions.

2.1 Stream Mining

Recent emerging applications, such as sensor networks, web click-streams, and power consumption measurement, generate voluminous data in real-time. For these kinds of applications, the data is too voluminous to be stored and then scanned several times for data mining. Instead, a subset of the incoming data sequence, called a window, is analyzed. By processing the sequence of data windows, the data stream can be processed as data arrives. Each window is data mined. Useful data and deductions are persisted and the windowed data is discarded in preparation for the next window. This process is called stream mining.

The data window plays an important role in data stream mining. Its size, ws, is fixed, typically. When the buffered steam data has more than ws elements, data is transferred to the window and processed. This process is repeated indefinitely. This emulates an analysis window sliding over a static data. A small window size results in models that change quickly with small changes in the data, whereas a large window size results in models that change more slowly, in effect averaging data variations.

Concept drift is an important consideration for stream mining. The distribution of instances associated with a classification may change with new observations from the data stream. In such cases, the classification scheme should change with the distribution. Thus, as the underlying concept drifts, so too should its model drift.
2.2 Recognizing Behavioral Changes

Stream mining methods can be used to discover significant changes in behavior. Stream mining produces a sequence of models, \( m_1, m_2, \ldots, m_n \), that predict the behavior observed in windows \( w_2, w_3, \ldots, w_{n+1} \). After a predicted window is observed, \( w_i \), the prediction quality of its model, \( q(m_i, w_i) = [0,1] \), can be calculated. There are a number of ways in which quality can be defined. For example, the classic metrics of accuracy and precision can be used individually or in combination.

Accuracy measures how error-free the model’s predictions are, according to this equation:

\[
\text{accuracy} = \frac{\text{true negative cases} + \text{true positive cases}}{\text{all cases}}
\]

where all cases = true negative + true positive + false negative + false positive cases

Precision measures fidelity, according to this equation:

\[
\text{precision} = \frac{\text{true positive cases}}{\text{(true positive + false positive cases)}}
\]

A model can have high accuracy but low precision, which indicates the model is not a good model. As an example, consider predicting 100 patients for cancer. In the data, 95% of patients are cancer free, but our model predicts 100% cancer. The model is 100% accurate, but has 5% precision—a very poor model indeed.

Predictive quality can be automatically evaluated as windows update during stream mining. Given that model \( m_{i-1} \) is trained over window \( w_i \), one can evaluate the predicted values of \( m_{i-1} \) against the known data in \( w_i \).

Consider the case where the predictive quality is nearly a constant 0.9 on a scale from 0 to 1, where 1.0 is prefect quality. This suggests that the models are good and that the behavior from one window to the next is nearly constant. Now, consider a sequence of some \( n \) predictions, with \( q_1, q_2, \ldots, q_n \), where each \( q_i \) is 0.9 with the exception of \( q_k (1 < k < n) \) which is 0.1. Again, this suggests that the models are good with the exception of \( m_{k,j} \), which was trained on window \( w_{k-1} \) to predict window \( w_k \). From this, we infer that something interesting happened during window \( w_k \). That is, the events in window \( w_k \) is so different from the events of window \( w_{k-1} \) that the model trained on \( w_{k-1} \) cannot reasonably classify the new window \( w_k \) behavior. This is the technique we use to recognize significant behavioral changes.

The predictive quality change is \( dq/dt \). Thus, \(|q'| > \varepsilon \) implies behavioral changes from window \( w_{k-1} \) to window \( w_k \). We can also consider how quickly the predictive quality changes, which is \( q'' \). An analysis of typical domain values for \( q' \) and \( q'' \) can provide guidelines that distinguish normal behavioral variations from significant behavioral changes.

Our particular use of stream mining and predictive quality to discover behavioral changes will be elaborated in the context of a case study in user-goal analysis (§3). There, we show how a negative value of \( q' \) for multiple window models suggests a behavior change.

2.3 Multiple Windows

Selecting the appropriate window size, \( ws \), is important. If \( ws \) is small, then insufficient data will be available when the mining model (e.g., decision tree) is derived. Conversely, if \( ws \) is large, then the analyst must wait, perhaps a long time, before the model is derived—moreover, model construction itself can take a long time for large data sets. Thus, there is a tradeoff between accuracy and availability of the mined model.

Some data stream miners incrementally update a single model as the data becomes available. Thus, the model grows with the window size. Such an approach is good for incrementally changing the model; however, for us, we are looking for transitions between models, where each model represents the learned behavior associated with a user goal. Therefore, we aim to acquire a goal model, and then notice when it changes significantly. Such significant changes may become hidden among many small, incremental model changes, but become apparent when a static model is compared with new behaviors.

A series of static models in stream mining begs the question of window size. Our approach is to run multiple concurrent windows; for example, two-week, four-week, and eight-week. Note that averaging four two-week models is not the same as one eight-week model because model construction is data dependent.

2.4 Multiple Models

Ensemble models are popular in data stream mining because model accuracy is data sensitive. Ensemble is the data mining term for concurrently using multiple models on the data window. Model predictions can be combined (e.g., averaged) or weighted according historical evaluation of prior predictions. We use multiple models to increase the accuracy of our predictions, as described in the case study presented next.

3 Emailing Case Study

The Think and Link (TAL) project developed a specialized (closed) email system as part of a clinical treatment package[13-15, 40, 41]. In TAL, the email software is personalized uniquely to each individual, as required by his or her treatment plan. Personalization is accomplished through evolution. User behaviors are continuously monitored to ensure that proper software adaptations are made to accommodate changing user needs. In contrast to most assistive technologies, TAL
patients do not abandon the TAL email system. In fact, their emailing skills grow with usage, and in response, their unique email interfaces require continuous updates.

In the commercial version of TAL, much of user monitoring is a manual process. This approach is not feasible for a large patient population. To address the scaling issue, TAL aims to automate much of monitoring in the research version of TAL. In particular, TAL’s monitor must notify clinicians when unusual behavioral patterns against the background of typical behaviors. To address this, we applied the data mining steps outlined in section 2.

The two goals that we will consider here are simply ‘read email’ and ‘compose and send email’. These two goals are directly supported by two observed events: ReadEmailEvent and ComposeMailEvent.

3.1 Data Logging

The TAL project provides an automated custom logger, which appends to a text file. To obtain real-time data access, the log file can be monitored with changes sent to a server. Here is a simplified entry from the log file. 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 e-mail message, such as its length, are also included. The two events of this case study concern reading and composing email.

3.2 Data Transformation

To simplify the processing of the data, the event log is transformed into a standard format, Common Base Event (CBE). We use a standard tool, the Eclipse Generic Log Adapter (GLA), to simplify this transformation. Finally, the CBE data is persisted to an SQL database. To support data mining we do provide a variety of computed fields, such as date views for hour of day, day of week, day of year, etc. The entire process can execute in real-time as each event is generated from the TAL system.

3.3 Data Cleaning and Integration

Classically, data mining includes cleaning and integration. Our data was largely correct; however, we did eliminate some TALMailEvent events because there were only a few, which occurred early in the dataset. Similarly, integration was unnecessary because the log data represents entire dataset.

3.4 Data Selection

With the goal to monitor a patient’s hourly behavior, we specify a scheme to predict a patient’s events at each hour. We partition the data based on a patient ID and event type; each patient has four views (by event type). Additionally, we select the fields for data mining. These include hour, day of week, and word count. The word count field (for compose events) is an integer; to increase predictability, it is mapped to enumerated sets: \{0 – 49\}, \{50 – 99\}, \{100, 499\}, \{500, 999\}, \{1000, 1999\}, \{2000, 2999\}, \{3000, 4999\}, \{5000, 6999\}, \{7000, 9000\}.

3.5 Data Mining

Our data mining procedure applies classification to sliding windows. In pre-testing, we considered a variety of classification models, including neural networks, Bayesian networks, decision trees, and association rules. We settled on three decision tree algorithms.

- M1: Gain Ratio (C4.5) is a successor of ID3.
- M2: Information Gain (ID3) minimizes the information needed to classify the data represented as tuples and the resulting partitions reflect the least randomness or impurity in these partitions.
- M3: Gini Index (CART) uses a formula based on probability to branch on nodes.

In pre-testing, we also considered a variety of window sizes. We settled on running three window sizes simultaneously: two-, four-, and eight-week windows.

Our analysis of one patient included 3,695,086 records occupying 737 MB in Microsoft SQL 2005. The data was automatically processed by Rapid Miner using our custom stream miner plug-in. Our stream-miner plug-in differs from the standard plug-in in that it: (1) can divide data by date or time (e.g., 2-week windows) in addition to a count of datum (e.g., 100); and (2) generates a log of information about the models it creates, including precision, accuracy, and other metrics.

The classification problem can be summarized as follows.

**Input:** event type, (email) word count, hour, day of week, week of year, month of year

**Predict:** event type, (email) word count, hour, day of week

On average, the models have reasonable quality.

3.6 Pattern Evaluation

Our application of data mining seeks to find interesting changes in behavior rather than interesting rules of behavior. Thus, we look for inflection points in predictive quality, as described in section 2.2.
The models predict for a time-slot within a day-of-week. Table 1 summarizes accuracy and precision for the read and compose event types. The values for the different window sizes are very similar. Notice that the read models are very good—meaning that the models predict well the number of emails that will be read during a time-segment on a day-of-week. The compose model is useful but not as good. It weakly predicts the number of emails that will be composed during a time-segment on a day-of-week. Because read models have the best quality and least variability, we rely on them to predict behavioral changes. In particular, the read-model is reliable, and thus can predict changes. Whereas, the compose-model is less reliable and thus its compose predications can be useful, they can also be wrong.

### Table 1 Qualities of model M1 for read and compose email.

<table>
<thead>
<tr>
<th>Window Size</th>
<th>Read Average</th>
<th>Read Precision</th>
<th>Compose Average</th>
<th>Compose Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-week</td>
<td>0.967</td>
<td>0.970</td>
<td>0.631</td>
<td>0.484</td>
</tr>
<tr>
<td>StdD.</td>
<td>0.049</td>
<td>0.018</td>
<td>0.241</td>
<td>0.144</td>
</tr>
<tr>
<td>4-week</td>
<td>0.984</td>
<td>0.971</td>
<td>0.657</td>
<td>0.478</td>
</tr>
<tr>
<td>StdD.</td>
<td>0.027</td>
<td>0.011</td>
<td>0.202</td>
<td>0.119</td>
</tr>
<tr>
<td>8-week</td>
<td>0.983</td>
<td>0.971</td>
<td>0.657</td>
<td>0.478</td>
</tr>
<tr>
<td>StdD.</td>
<td>0.027</td>
<td>0.011</td>
<td>0.202</td>
<td>0.119</td>
</tr>
</tbody>
</table>

The predictive qualities of the three models for three windows are graphed in Figure 2. Of course, it’s difficult to see the details in the graph, but the overview is useful. The graph shows that the models are windows roughly track the same events. Windows of four and eight weeks are nearly the same. They both miss or diminish events considered interesting by the two-week window (e.g., weeks 66 – 68). In general, the two-week window analysis has greater variability and foreshadows the larger window analysis, as expected. Notice that week 34 has poor predictive quality (q) as identified in both the two- and four-week window analysis. Moreover, the quality in the immediately surrounding weeks is good; thus, the rate in change of the quality (q') around week 34 is also high. Together the dramatic decrease in q and zeroing of q' suggest that week 34 may be a behavioral inflection point worthy of further analysis. (There are other inflection points in the graphs also worthy of further analysis (e.g., week 82); however, due to space limitations, we present only one detailed analysis.)

Table 2 presents q' for two- and four-week windows for the three models; it distills a significant contribution of this paper. The table shows where q' turns negative between 0 and 36 weeks. The highlighted cells indicate where the value is less than one-half of the standard deviation for values in the column. The analyst can...
combine these values—for example, using average, logical And or logical Or—to automate the recognition of “interesting” events. In our case, we used logical Or—if any \( q' \) turns negative, then we considered that week (or consecutive weeks) to be a potential behavior transition for attaining or forgetting a goal.

<table>
<thead>
<tr>
<th>Week</th>
<th>2-week window</th>
<th>4-week window</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M1</td>
<td>M2</td>
</tr>
<tr>
<td>10</td>
<td>-0.012</td>
<td>-0.012</td>
</tr>
<tr>
<td>12</td>
<td>-0.007</td>
<td>-0.007</td>
</tr>
<tr>
<td>16</td>
<td>-0.037</td>
<td>-0.037</td>
</tr>
<tr>
<td>18</td>
<td>0.021</td>
<td>-0.043</td>
</tr>
<tr>
<td>20</td>
<td>0.005</td>
<td>0.054</td>
</tr>
<tr>
<td>30</td>
<td>-0.014</td>
<td>-0.014</td>
</tr>
<tr>
<td>32</td>
<td>-0.040</td>
<td>-0.040</td>
</tr>
<tr>
<td>36</td>
<td>0.033</td>
<td>0.009</td>
</tr>
</tbody>
</table>

We applied an OLAP analysis to weeks surrounding week 34. During the weeks from 33 through 36 (Figure 2) the decrease in quality corresponds to a decrease and then increase in email counts, with week 34 being the minimal email counts. Furthermore, analysis of the preceding weeks 29 through 32 and the subsequent weeks 37 through 40 reveal that there are more emails in the afternoon and less in the morning after the inflection point. Table 3 summarizes the differences between the four weeks surrounding the four weeks in question. Together, the changes in model quality and observed events around week 34 suggest a significant behavioral change around week 34 that continued for at least one month.

<table>
<thead>
<tr>
<th>Weeks</th>
<th>Sleeping</th>
<th>Afternoon</th>
</tr>
</thead>
<tbody>
<tr>
<td>29-32</td>
<td>Average 9.00</td>
<td>8.75</td>
</tr>
<tr>
<td></td>
<td>StdD 4.24</td>
<td>3.10</td>
</tr>
<tr>
<td>37-40</td>
<td>Average 4.25</td>
<td>14.25</td>
</tr>
<tr>
<td></td>
<td>StdD 2.22</td>
<td>1.71</td>
</tr>
</tbody>
</table>

### 4 Evaluation

To evaluate this approach to identifying unusual behavioral changes against the background of typical behaviors, we consider the model quality itself and the correlation of identified inflection points with real-world changes.

### 4.1 Accuracy and Precision

Accuracy and precision are standard metrics for determining predictive model quality. As Table 1 shows, the read model is very good and while the compose model is useful not as good. This may be an anticipated because the read model depends on received emails and free time of the patient; these elements are mostly routinized for the CI user population. On the other hand, email composition depends on the content of the received email as well as external, non-modeled events, such as the correct usage of medicine or the arrival of visitors. In a prior study, we showed that patient interest and email composition increased with the addition of new ‘email buddies’, while decreasing slowly thereafter. Therefore, given the limited information in the data available and the real-world behavioral variations of the users, the availability of at least one very good model seems to provide adequate information to identify some significant behavioral changes.

### 4.2 Real-World Events

We hypothesize that model quality changes suggest behavioral changes. Our case study illustrates this with the inflection point around week 34, where is a dramatic decrease in quality. To validate the use of model quality as a behavioral indicator, we used OLAP to analyze the event data in detail. We found significant changes in email composition—changes in time of day, day of week, number of words. These changes seem to confirm the use of model quality as a behavioral change indicator, at least for this domain. Model quality suggests similar behavioral changes for week 82, which is confirmed through detailed event analysis. It may be noteworthy that week 34 of the data corresponds to 8/20/2006 - 8/26/2006, while week 82 of the data corresponds to 7/29/2007 - 8/4/2007 (week 31 of 2007). We hypothesize that something interesting happens to the patient in August, such as a family member visit. Patient anonymity prevents us from correlating such real-world events.

### 5 Discussion

While the use of predictive quality change as an indicator of behavioral change appears to be useful, our case study suggests issues that require further research.

- Window size is important. Too large and behavioral changes are effectively regressed toward a mean and remain undiscovered. Too small and every transient event appears to be a newly acquired behavior. The window size should be matched to the context in which new behaviors are commonly acquired. Not only is this a domain dependent parameter, it is likely to be user specific. Moreover, it can change
dynamically. Early in treatment, users may acquire new behaviors quickly, while later in treatment behaviors may be acquired more slowly. Our use of multiple simultaneous windows may, in part, address the issue of varied window size.

- Model quality is important. We were fortunate that the read model is very good. However, obtaining a good and consistent model may be difficult. Multiple models provide a generic method to obtain consistently good modeling.

- Real-world correlation is important. Once an inflection point is found, behavioral changes must be confirmed by analyzing the data or asking an expert. Of course, the method is intended to reduce the need for experts. Thus, the approach would be improved if an inflection point could automatically index into the underlying data where more direct analysis can be obtained. One potential method is to compare models before and after the inflection point to determine the most significant of the change features. For example, week 34 from our case study presented significant changes in the features of email composition. The automated presentation to clinicians of such differences is likely to reduce the need for further analysis.

- Quality function shape may be important. If the predictive quality is monotonically increasing (\( q' > 0 \)), it suggests that patient’s behavior is becoming more consistent. Conversely, a negative slope indicates less consistency. A domain expert may be able to infer the consequences of such changes. For example, it may be that such changes precede learning (or forgetting). We have applied both \( q' \) and \( q^n \) and find that \( q' \) with multiple models and windows appears to provide a reasonable basis for an automated transition identification function in our domain.

We have begun exploring these issues, and will report on them in the future.

6 Conclusions

This paper presents a method for monitoring behavioral changes from an event stream of user actions. This problem arises in the context of applying AT to aid cognitive treatment plans. Users work towards long-term goal achievement by applying planned behaviors. The monitored events from this domain contain complex data and temporal relationships applied to the attainment of a few goals; the behavioral patterns can be highly complex and vary over time-periods. To address this problem, we have analyzed the quality of models generated, in sequence, by mining the stream of user events. Changes in quality suggest significant changes in user behavior. These inflection points can suggest the need to change a user’s treatment plan, moving forward with plan achievement and revisiting prior treatments with user regression. Our case study of use demonstrates the feasibility of this approach. The entire process can be automated—from event acquisition in the AT software to recognizing changes in model quality.

Data stream mining is central to our approach. It depends on parameters of data window size, the model quality function (\( q \)), and the interpretation of the quality change function (\( q' \)). Although we have demonstrated the application of these parameters, important characteristics of their application require further research.

The outcome of this research is important to the field of AT-based clinical therapy. For example, as the population ages, more people will be struck by cognitive impairments, which can be ameliorated through software-supported therapy. Dynamically interpreting and adapting therapy plans for individuals currently requires substantial effort of clinicians. With AT software improving, CI user population increasing, and clinicians decreasing, the need for some automation in therapy analysis is critical. Methods, such as the one we describe herein, will be critical factor in addressing the needs of the millions of people with cognitive disabilities.

7 References


