Pattern Discovery of User Interface Sequencing by Rehabilitation Clients with Cognitive Impairments

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

Ali Raza Syed  
Computer Information Systems Department  
Georgia State University  
asyed1@student.gsu.edu

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

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

Abstract

We demonstrate the use of sequence pattern mining as applied to monitoring the usage of emailing software by clients with cognitive impairments. We show how Max Motif, a sequence-mining algorithm, can be applied using a stream-mining approach. Consequently, clinicians can now consider sequential patterns in their analysis of client emailing behaviors. Such analysis is part of the Think and Link project, which provides personalized email clients, a kind of assistive technology, to aid client communications that facilitate activities of daily living. By using the simplified, customized system, clients can now email, whereas they could not previously. By continuously monitoring usage, the project is able continually adapt the software to a user’s changing needs. Thus, monitoring software usage, particularly email event sequences, is important. This paper introduces theory and design of stream sequence-mining for UI event streams.

1 Introduction

Between 13.3 to 16.1 million Americans are living with chronic brain disorders and associated cognitive impairments (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) can help. Many studies have found, however, that AT systems are abandoned by client at high rates[5-7]. 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. The work described herein extends our prior work on monitoring AT usage, which provides feedback for AT realignment. Herein, we demonstrate stream sequence-mining for user-interface (UI) event streams.

Monitoring client usage of a UI is difficult. Our prior work has demonstrated how changes in usage can be identified and diagnosed using a stream mining approach[8, 9]. We are not aware of an extant research for sequence pattern discovery in UI event streams. This article demonstrates one such approach.

In the case-study project, clinicians do use graphs of aggregate UI usage to monitor client progress[10]. Such analysis has been supplemented recently with data mined models of usage. Models, such as decision trees, characterize the usage context for a data window. The models have not considered event sequences. Consequently, our clinicians have had no means to consider sequential patterns, despite their value to cognitive analysis (§1.3).

Because of UI monitoring, our team has applied timely software adaptations, and thus the clients are better able to engage in social communications as part of their rehabilitation therapy.

1.1 Cognitive Rehabilitation

“In the rehabilitation context, the clinician is generally viewed as an instructor and the client as the learner”[11]. 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 AT have facilitated the PIE approach by providing automated assistance in planning, implementing, and evaluating therapies. In the *Think and Link* project, for example, AT provides the implementation context for learning emailing, assistance to monitoring the client’s learning sessions, and provides monitoring to aid analyses of session outcomes and trends.

The *Think and Link* (TAL) project provides email clients that are usable by people with impairments in memory and learning as a result of brain injuries [10, 12, 13]. A custom email client is continually monitored to determine how well it satisfies a client’s learning goals [13]. Improved 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 system is updated in an effort to reestablish goal satisfaction.

Clients typically abandon AT after a short period. In the AT context, goal failure is often associated with a poor fit between the client’s goals and the goals supported by the AT. By monitoring user goals TAL can responsively react to changes in goal satisfaction. Such changes are attributed to TAL’s success. TAL team members provided 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 [13]. The TAL team must make the appropriate software 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.

### 1.2 Goal Attainment Scales

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. As Don used the personalized emailing system, our software evaluated the fit between his specified goals and his observed behaviors. Consequently, Don’s careproviders and clinicians could easily determine Don’s variance from his idealized behavior. When Don went through periods of low or error prone activity, the careproviders could quickly respond. The diagnosed variance was precise. For example, Don reduced his overall emailing and increased his late night activity (mostly Monday – Wednesday). Differencing data-mining models from the data stream provided the necessary precise diagnosis at the end of each collected data window [8, 9, 14].

Clinicians want to see goal satisfaction, and in particular: (1) a good success-to-failure ratio over sessions, and (2) an improving trend of this ratio. In the case of Don, who is acquiring simple email skills, clinicians want to see Don succeed with: (1) read email and (2) compose and send email. In support of clinicians, the monitoring system’s goals are to [12]:

- Recognize changes in client behavior as observed in usage of the emailing AT
- Diagnosis each significant change by characterizing influential attributes of AT usage

Event sequencing is an important characteristic of user behavior. A challenge for automated monitoring systems is to recognize action sequences and diagnose changes or trends in such sequences.

### 1.3 Cognition and Sequencing

Goals are 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 [15]. In each view, sequencing of tasks increases complexity, especially if it includes alternative tasks and subsequences [16]. The ability to think about task sequences that are lengthy and variable is an aspect of good cognitive processing [17]. Unvarying short task sequences are considered less indicative of good cognition than moderate to long, varied and repeated sequences.

Recognition and differential diagnosis of action sequences is the primary contribution of this study. Our monitoring software records and analyzes task sequences initiated by the client. In particular, the software finds repeated patterns, or motifs, in the UI event stream. Motif analysis provides caregivers insight into cognitive effort that a client devotes to emailing. A smaller effort suggests a lack of interest, software obstacles, or troubles in cognitive processing. Trend analysis is most important. Clinicians consider waning motifs as a trouble sign,
whereas motifs that are more complex suggest improvement in interest, UI configuration, or client cognition. Interesting UI event motifs suggests that the client has mastered emailing, and thus their ADLs will improve through improved communications.

### 1.4 Stream Mining Sequences

To recognize and diagnose sequences, we apply sequence mining logged UI actions. Our software identifies usage sequences and supports trend analysis of usage sequences.

Events occur in sequence. A sequence pattern specifies a sequence of events that frequently occurs within a period. Finding patterns of reoccurring sequences is the goal of sequence mining[18].

Sequence mining can be applied to a series of data windows using an approach called stream mining. Stream mining systems analyze voluminous, continuous data streams where it is not practical to store all the data. Instead, sequential data subsets, called windows, are analyzed as they arrive. This is the approach we apply—sequence mining of data windows and trend analysis across the windows.

Such analysis is useful to clinicians, as they monitor their clients. There had been no knowledge of UI sequences in the TAL project. With the stream sequence-mining described herein, clinicians now gain insight into client UI trends, which they may associate with changes in client interest, software obstacles, or troubles in cognitive processing.

### 1.5 Article Outline

This article describes a case study in applying stream sequence-mining to a client’s actions as logged by the TAL emailing software. Next, we summarize related research. In section 3, we describe the software design task of integrating the Max Motif sequence miner into our stream-mining system. The case study (§4), discussion (§0), and conclusions (§6) follow. We conclude that stream sequence-mining provides an effective, automated means to recognize and diagnosis TAL UI sequences. Such analysis aids clinicians in their care of clients with CI.

### 2 Related Research

#### 2.1 Characteristics of Cognitive Rehabilitation Data Streams

Individuals with cognitive impairments vary greatly in their needs and skills. Moreover, each individual may display significant variation within a session and between sessions. Consider a client enrolled in the TAL project[10]:

Matt is a 43-year-old male with severe cognitive communicative and physical disabilities due to a traumatic brain injury from a motor vehicle accident 25 years prior to the study. He demonstrated severely impaired anterograde memory, executive functions, and information processing speed as well as severely dysarthria speech and impaired gross and fine motor ability. He was single and living in a supported living residential community. Staff assisted Matt in remembering to take his medications and in managing his finances.

A client like Matt can perform increasingly well and then suddenly perform very poorly. Such was the case of one client who periodically forgot his medication on Wednesdays, and then engaged in the unusual behavior of “drawing”, via placement of characters, within the email compose form. This was noticed as a periodic increase in the metric of edits per email. This sort of randomness can confound traditional data mining methods, which assume more normal distributions. Thus, mining methods must be robust in the face of randomly occurring events that can arise from individuals with cognitive impairments.

#### 2.2 Plan Recognition

Planning and event abstraction may aid analysis. Schmidt et. al. developed a theory of plan recognition for a single-actor action sequences[19]. A plan recognizer observes event sequences, S, and characterizes them as a plan, P, that is a sequence of abstract actions satisfying a goal. There is a great deal of plan recognition research, which addresses issues such as probabilistic reasoning, temporal relationships, event hierarchies, and concurrent goals[20]. However, most plan recognizers rely on plan libraries, which require substantial human effort. Learning techniques can increase automation by building a predictive model from past behaviors[21].

Temporal abstraction facilitates characterizing event sequences. The knowledge-based temporal abstraction (KBTA) method applies domain-specific knowledge types to characterize event sequences as temporal abstractions[22]. KBTA is a data pre-processor that simplifies subsequent analysis, such as pivot table analysis, data mining, or plan recognition (potentially). The approach has been generalized to apply incrementally, so it can be applied to large volumes of continuously arriving events[23]. This approach notifies users when given a query’s results change. Thus, one can monitor concepts that can be represented as a query in its TAQILA language.
2.3 Model-Based Monitoring

Runtime monitoring of software for specified behaviors is increasingly of importance to software engineering[24]. Computer published a toolkit for monitoring user-interface activities[25]. More generally, runtime analysis of properties is a growing trend in software development[26, 27]. Goal monitoring integrates and generalizes prior monitoring technologies in support of high-level user or system requirements[28]. A recent article summarizes this research and presents systems demonstrating a variety of concerns and techniques that influence the interpretation of a running system[24].

Related work in the context of assisted living includes (1) the use biometric and sensor data from home activity to identify trends and drifts from those trends[29], and (2) the use of positional data to detect behavioral deviations from routine patterns[30]. Our work differs in that we are concerned with mining sequences.

2.4 Process Mining

Process mining extracts information from event logs to derive business-process descriptions[31]. Constructing a business process model from a transaction log is a typical application. Inferring casual relationships, including concurrency, loops, conditionals, task hierarchies, etc., of the task network from time-stamped logs in the face of noisy and incomplete (erroneous) logs of process events is the focus. Nascent research areas address noisy and incomplete logs, visualizing results, and delta analysis, which explain differences between two derived models.

In many respects, process mining is related to AT mining. A significant difference may be that concurrency is more common in business processing than the pursuit of concurrent goals by a human. Moreover, a human is more likely to be inconsistent when pursuing goals, as well as opportunistic in pursuing new goals. Thus, human event logs are less uniform and therefore more weakly structured than business process logs.

2.5 Sequence 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[18, 32]. 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. Our task of sequential pattern discovery naturally leads to the consideration of motif analysis[33]. Work in this area discovers patterns from a database of protein sequences[34], for example. Similar work finds frequently reoccurring patterns[35], and some address explicit time constraints[36].

We apply the Max Motif algorithm, which efficiently enumerates all maximal motifs in an input string[37]. 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.

2.6 Data Stream Mining

We apply sequence mining in the context of stream mining. Stream mining aims to find interesting relationships over a sequence of data segments [38-40]. 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[41]. A variety of techniques can be applied to stream data [42-45]—much of the work is focused on the efficiency of incrementally updating the model [46].

Detecting changes in data-streams is important for monitoring, in particular for AT monitoring systems. 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 adopt this last approach.

To discover sequence patterns, we apply Max Motif to each window. To detect pattern changes, we difference the patterns sets discovered in consecutive windows. We describe our software design for stream sequence-mining next.
3 Max Motif Sequence Mining

We use KNIME for data mining (www.knime.org). Figure 1 shows a KNIME workflow for applying Max Motif in batch mode. (The same operators apply in real-time.) The Database Reader, at the left, identifies the data windows that are retrieved by the Database Reader at the lower left in Figure 1. Each window is processed within the loop by our Max Motif operator.

Time is an important concern for the TAL event sequences. Consider the following TAL email-client events:

1. NewMailEvent
2. ReadEmailEvent
3. ComposeEmailEvent

If the events all occur within a few minutes of each other, then they may be interpreted as a user creating (and sending) an email in response to a newly arrived and read email. Such a sequence is considered an important pattern. On the other hand, if there is a 6-hour gap between the Read and Compose events, then it seems less likely that the user is responding to the newly arrived email. Consequently, a time-gap splits a sequence into two conceptually significant sequences.

To address time gaps, we pre-process the input events. The leftmost JPython Script in Figure 1 segments sequences, ensuring that there is no more than a 2-hour gap between events. This approach allows us to address time constraints prior to applying Max Motif, which efficiently discovers sequence patterns without regard to time.

The remainder operators in the KNIME workflow of Figure 1 address data representation: (1) encoding and decoding the TAL events names for Max Motif, and (2) adding window information to the data record.

In support of trend analysis, the software determines differences in pattern sets, as discovered in consecutive windows. Consider the patterns discovered in window $w_i$ as model $m_i$. We apply model differencing to discover pattern changes from window $w_i$ to $w_{i+1}$: $(m_{i+1} - m_i)$.

Figure 2 shows how pattern differencing is applied within KNIME. The JPython Script in Figure 2 calculates differences for the consecutive models produced by the Max Motif workflow of Figure 1. The Java Snippet and Value Counter operators calculate aggregate pattern information, including the
number of wild cards, the total number of patterns identified within a window and within all windows, etc. The results of these analyses are presented as pivot tables and graphs.

4 Emailing Case Study

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 e-mail message, such as its length, are also included. The significant event types are: read email, compose email, delete email, and new (arriving) email. A database view provides a continuously updated stream. 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 Preprocessing and Mining

Rather than data mine the raw TAL event stream, we preprocess the data using our property-based monitoring system, EEAT. This system allows us to encode temporal relationships as properties. For example, a response property can be defined as A followed by B within 6 hours.

The Event Engineering and Analysis Toolkit (EEAT) supports specifying, instrumenting, data collecting, and property analysis[8, 9, 24, 47-52]. It is a requirements monitoring system in that it supports (1) analysis of abstract, requirements-level properties, and (2) automation of runtime requirements evaluation, which interprets low-level software events as contributors to the eventual satisfaction or violation of requirements[24].

The EEAT goal specification language is a variant of the OCL 2.0, which we call OCL_TM—meaning OCL with Temporal Message logic[53]. It is similar to other languages that support some form of predicate calculus and temporal logic over an object model (e.g., KAOS [54]), and thus supports Goal Oriented Requirements Engineering (GORE)[55, 56]. OCL_TM also includes real-time operators.

A typical EEAT scenario, as applied to TAL, is as follows:
1. A software analyst specifies TAL goals as formal properties to allow for automated monitoring.
2. The analyst uses an integrated development environment (IDE) to generate automated monitors from the formalized properties.
3. As the user exercises the TAL client, events travel from TAL to EEAT, which calculates property satisfaction.

Figure 3 shows three kinds of TAL properties:
1. Email to buddy. Each of the 31 buddies is enumerated by their ID. The property is true each time the corresponding buddy is emailed.
2. Read from buddy. The same as email to buddy, but for reading emails.
3. Reply to buddy. A read from and then reply to a buddy that occurs within a specific period (6 hours).

```
package talx::model
context TransportToolkit
def: timeout1: LTL::Timeout = timeout('0d:6h:0m:0s')
def: rReadMail: LTL::OclMessage = receivedMessage('talx.events.ReadMailEvent')
def: rComposeMail: LTL::OclMessage = receivedMessage('talx.events.ComposeMailEvent')
def: rDeleteMail: LTL::OclMessage = receivedMessage('talx.events.DeletMailEvent')
-- argument(1) is the buddy associated with the email
inv emailToBuddy_1: eventually(rComposeMail.argument(1)='-1266262578')
//...
inv emailToBuddy_31: eventually(rComposeMail.argument(1)='81287508')
inv readFromBuddy_1: eventually(rReadMail.argument(1)='-1266262578')
//...
inv readFromBuddy_31: eventually(rReadMail.argument(1)='81287508')
inv replyToEmail_1: after(eventually(rReadMail.argument(1)='-1266262578'),
  eventually(rComposeMail.argument(7)=rReadMail.argument(6)),timeout1)
//...
inv replyToEmail_31: after(eventually(rReadMail.argument(1)='81287508'),
  eventually(rComposeMail.argument(7)=rReadMail.argument(6)),timeout1)
endpackage
```

Figure 3 EEAT OCL expression of TAL properties.
Properties like reply to buddy, are true when the time (and data) constraints are met. Thus, they encode temporal properties. These EEAT properties are then mined for sequences.

Using workflows like that shown in Figure 1 and Figure 2, we applied sequence discovery to Don’s usage of the TAL email client. The resulting tables about the discovered patterns describe:

- Sequence property patterns (including wildcards) e.g., read_16,*read_16.
- Property counts for windows, e.g., current = 60, next = 119
- Window count, indicating the number of windows that contained the pattern

Analysis of the resulting patterns is presented next.

4.3 Sequence Analysis

Don’s 10 most frequently occurring motifs are:

1. read_10
2. read_16
3. read_16,*read_16
4. read_16,read_16
5. read_9
6. reply_10
7. reply_16
8. reply_16,*reply_16
9. reply_16,reply_16
10. reply_9

The number indicates the buddy involved, while the ‘*’ indicates a wildcard in which any single event may be substituted.

Table 1 shows characteristics of the discovered sequence patterns.

<table>
<thead>
<tr>
<th>Sequence pattern</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
<th>StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-variable Length</td>
<td>1</td>
<td>3.75</td>
<td>15</td>
<td>3.32</td>
</tr>
<tr>
<td>Variable length</td>
<td>1</td>
<td>11.32</td>
<td>20</td>
<td>3.64</td>
</tr>
<tr>
<td>Wildcards</td>
<td>0</td>
<td>3.68</td>
<td>9</td>
<td>1.84</td>
</tr>
</tbody>
</table>

Figure 4 graphs the average pattern length over 2 years of 2-week data windows, revealing the slightly decreasing pattern length. Figure 5 graphs non-variable pattern lengths, revealing a slightly increasing length. Figure 7 reveals the patterns by the initiating event, showing three properties from Figure 3. The topmost dashed line shows great variability in the increasing length of patterns that begin with read. The middle dotted line shows patterns beginning with compose are more consistent in length, but are slightly decreasing over time. Finally, the bottom-most solid line shows the eventual increase in length of patterns beginning with a reply.

Lastly, Figure 6 shows the average occurrence of patterns beginning with one of the three properties from Figure 3. The topmost dashed line shows patterns beginning with read, the middle dotted line shows patterns beginning with compose, while the bottom-most solid line shows reply. Again, reply sequences are most increasing, read least increasing, and compose is decreasing.

Overall, this may be interpreted as Don becoming (1) more consistent over time because of the increasing length of non-variable sequences, and (2) becoming more conversational with email because of the increased usage of reply. This latter point is significant, because it suggests that Don may be transitioning from using email for simple notifications (e.g., “I’m low on meds”) or requests...
(e.g., “I need a ride to the library.”) to dialogs (e.g., “I do remember that about our father. That reminds me of ...”). Prior analysis of simple metrics (e.g., message length, composition speed, etc.) supports this common trend of TAL users becoming increasingly skilled emailers[10].

The dramatic drops in pattern length in the figures are noteworthy. They are consistent with transitions in Don’s emailing behavior, previously identified and diagnosed[8, 9].

5 Discussion

Analysis of sequence patterns supports clinicians in their efforts to link AT usage with cognitive theories. A domain like emailing allows for planned behavior. A client can read an email and respond. In the context of a discussion, a client may re-read a few prior emails from a buddy and then compose a response. The emailing usage log provides a direct means for observing such planned behavior. Pattern analysis can reveal user plans as variable sequences, even when a user makes occasional mistakes or pursues concurrent goals.

The approach presented herein does not consider a statistical analysis of the discovered patterns. It would be helpful to know not just the occurrence counts for the patterns, but the statistical significance of the patterns within the data. It is possible to consider the pattern likelihood against the background of random sequences. This line of reasoning has been explored considering Bernoulli[57] and Markov[58] model assumptions. Our future work will consider these concepts in building more complete UI usage models.

6 Conclusions

Recognition and differential diagnosis of action sequences is the primary contribution of this study. Our monitoring software records and analyzes task sequences initiated by the client. In particular, the software finds repeated patterns, or motifs, in the UI event stream. The software has been applied to monitoring AT usage by clients with cognitive impairments.

The approach appears appropriate for analysis of human event sequences. In particular, it addresses the randomly occurring events that can arise from individuals with cognitive impairments. Parameters of Max Motif can limit pattern detection by their occurrence and randomness (i.e., number of variables). The data windowing of stream mining limits analysis to data segments rather than the whole dataset. These scope limitations focus analysis on localized patterns and their incremental changes. This is the kind of trend analysis needed for monitoring TAL clients.

Clinicians can now consider sequential patterns when assessing their TAL clients. It is likely that the approach applies more broadly to human planned action domains requiring monitoring.

7 References


