Mining Sequential Pattern Changes*

I-HUI LI1, JYUN-YAO HUANG2 AND I-EN LIAO2

1Department of Information Networking and System Administration
Ling Tung University
Taichung, 408 Taiwan

2Department of Computer Science and Engineering
National Chung Hsing University
Taichung, 402 Taiwan

Many techniques have been proposed for mining sequential patterns in data streams. Nevertheless, the characteristics of these sequential patterns may change over time. For example, the sequential patterns may appear frequently in one time period, but rarely in others. However, most existing mining techniques ignore the changes which take place in sequential patterns over time, or use only a simple static decay function to assign a greater importance to the more recent data in streams. Accordingly, this study proposes an adaptive model for mining the changes in sequential patterns of streams. In this model, the current and cumulative mining results for sequential patterns within streams are found, and the significant change patterns and corresponding degree of change are identified. The degree of change between the current sequential patterns and those in the next mining round is then predicted, and the decay rate modified accordingly. The experimental results confirm the ability of the proposed model to automatically tune the decay rate in accordance with the present state of data stream and the predicted degree of change of sequential patterns in the following mining round.

Keywords: data streams, sequential pattern mining, prediction, change handling, change mining

1. INTRODUCTION

Real-time data processing is an essential requirement in many applications, such as network monitoring analysis, web log mining, wireless sensor network (WSN) analysis, stock trading application [1], and so on. The data generated over time in such applications form so-called data streams. Data streams are typically large, fast and unbounded, and are composed of continuous data elements which may only be read at most a few times. Consequently, developing data mining systems with a real-time capability for practical data streams is a highly challenging problem.

Change is an important characteristic in real-time applications, and has therefore attracted considerable attention in data mining field. According to Liu et al. [2], by knowing what is changing and how it is changing, a business can consistently provide the products and services required to satisfy changing market needs. A consumer’s recent purchasing behavior may differ from his or her past behavior, and is not a reliable indicator of his or her future behavior [3]. Thus, developing tools for analyzing a customer’s historic behavior and predicting their future behavior is highly challenging.

Received August 9, 2012; revised November 19, 2012; accepted February 25, 2013.
Communicated by Tyng-Luh Liu.
* This research was partially supported by the National Science Council, Taiwan, under Contract No. NSC 101-2221-E-005-087.
Sequential pattern mining provides the ability to detect frequently occurring patterns and to detect (or predict) anomalies in the data. For example, a frequent sequential pattern (abc) indicates that items a and b often occur at the same time, and are then frequently followed by item c. Sequential pattern mining algorithms are widely used in such applications as customer purchase sequence analysis, route suggestion systems [4], and so on. Therefore, change mining is essential in identifying potentially significant changes in sequential patterns of data streams.

Existing proposals for mining the changes in sequential patterns within data streams fall into two main groups. In the first group, a static decay function is used to assign a greater importance to new elements within the streams, such as IncSPAM [5], GraSeq [6] and eISeq [7]. The second group of change mining schemes presented in the literature focus on the identification of significant change patterns. For example, Li et al. [8] proposed a method for the online mining of five different types of item change patterns over continuous data streams. Tsai and Shieh [9] proposed a framework identifying three types of change pattern, namely “emerging sequential patterns”, “unexpected sequence changes” and “added/perished sequential patterns”. Liu et al. [10] proposed an event change detection (ECD) system to detect five types of environmental change pattern, namely emerging event patterns, added event patterns, perished event patterns, unexpected consequent changes of event patterns, and unexpected condition changes of event patterns.

The literature contains various proposals for mining changes in sequential patterns of data streams. However, these proposals generally suffer two major drawbacks. First, they use static decay functions and therefore fail to reflect the differences in sequential patterns detected in different time periods. Second, they provide no means of quantifying the degree of change in sequential patterns and modifying the mining process accordingly. Consequently, the present study proposes a model for mining the changes in sequential patterns of data streams and then modifying the decay rate adaptively in accordance with the degree of change of the patterns. To the best of the current authors’ knowledge, our previous research [11] is the first report to predict the change of sequential patterns in real-world data streams, and this study represents the first reported attempt to adapt the sensitivity of the mining model in response to changes in sequential patterns. Through the proposed model, the decision-maker can recognize emerging trends, detect and predict changes within real-world data streams and formulate appropriate strategies in response.

The main contributions of this paper can be summarized as follows:
(1) the concept of “change” in the context of the sequential pattern mining of data streams is discussed; (2) a model is proposed for mining the changes in sequential patterns within data streams; (3) a change predictor mechanism for estimating the future change in sequential patterns within data streams is presented; (4) a means of adjusting the change sensitivity of the mining model in accordance with the current state of the data stream and the predicted degree of change of the sequential patterns is proposed.

The remainder of this paper is organized as follows. Section 2 provides the preliminaries and the problem statement. Section 3 introduces the proposed change mining model and describes each of its major components. Section 4 describes the experimental design and presents the corresponding results. Finally, Section 5 provides some brief concluding remarks.
MINING SEQUENTIAL PATTERN CHANGES

2. PRELIMINARIES AND PROBLEM STATEMENT

This section commences with some preliminaries and definitions pertinent to the sequential pattern mining of data streams. The change and change patterns considered in this study are then defined. Finally, the problem statement used in developing this study is introduced.

2.1 Sequential Pattern Mining

The problem of mining sequential patterns was first introduced in [12]. Let \( I = \{i_1, i_2, \ldots, i_m\} \) be a set of items. Furthermore, let an itemset be a non-empty set of items. Finally, let a sequence \( s \) be a set of elements ordered according to their timestamps, where each element is an itemset containing sorted items. In other words, a sequence \( s \) can be denoted as \(<s_1, s_2, \ldots, s_n>\) is an ordered set of \( n \) elements [13], where \( s_j \) is an element, \( j \in \{1 \ldots n\} \). If every element in a sequence has only one item, the sequence is referred to as an item_sequence (such as \(<abkd>\) for ordered items a, b, k, d); otherwise, it is referred to as an itemset_sequence (such as \<ab)(dk)(ace)>).

In this study, the data streams are modeled as a series of transactions. Let each transaction \( T \) consists of a customer-ID (CID), a timestamp and an itemset (see Fig. 1). Having collected all the data stream information over a certain period, the transactions relating to the same customer are grouped, sorted in ascending timestamp order and designated as a data sequence. The system repeats this process until \( w \) data sequences have been generated, where \( w \) is an application-dependent window size and has a fixed value (see Fig. 2 for example). Windows are non-overlapped. Thus, each data sequence extracted from the input stream comprises three pieces of information, namely CID, WID (Window ID) and the sequence itself. A support count (denoted as count(s)) is generated for each sequence indicating the total number of data sequences in the window containing sequence \( s \). The support of a sequence \( s \) represents the proportion of the total number

<table>
<thead>
<tr>
<th>CID</th>
<th>Timestamp</th>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>(abd)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(b)</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>(bcd)</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>(abc)</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>(ab)</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>(bcd)</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>(bcd)</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>(bd)</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>(ac)</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>(bc)</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>(abc)</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>(cd)</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>(ad)</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>(ad)</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>(cd)</td>
</tr>
</tbody>
</table>

Fig. 1. Typical example of input stream.

<table>
<thead>
<tr>
<th>CID</th>
<th>WID</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>(abd)(bcd)(bc)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>(b)(abc)(bcd)(ac)(cd)</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>(ab)(bcd)(abc)(ad)</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>(bd)(ad)(cd)</td>
</tr>
</tbody>
</table>

Fig. 2. Example of data sequences extracted from Fig. 1.

Fig. 3. Pattern types.
of sequences in the dataset which contain \( s \). This study also utilizes the semi-buffer concept from IncSpan [14] in mining process, since such an approach reduces variation in the mining results, accelerates the mining process, and prevents the loss of information as a result of changes. As shown in the following, this paper considers two types of pattern, namely a current pattern and a cumulative pattern.

**Definition 1:** A data sequence \( s \) is defined as a current frequent sequential pattern if \( \text{count}(s) \geq S_{cmin} \times w \), where \( S_{cmin} \) is a user-defined minimum support threshold parameter and has a value in the range of \( 0 < S_{cmin} < 1 \) (see Fig. 3).

**Definition 2:** A data sequence \( s \) is defined as a current semi-frequent sequential pattern if \( S_{c sig} \times w \leq \text{count}(s) < S_{cmin} \times w \), where \( S_{c sig} \) is a user-defined significance threshold parameter with a value in the range of \( 0 < S_{c sig} < S_{cmin} < 1 \) (see Fig. 3).

Let \( DS = W_1, W_2, \ldots, W_n, \ldots \) be an infinite sequence of windows, where each window is associated with a particular timestamp \( t \). In addition, let \( W_n \) be the current window and let \( DS_n \) be a cumulative window between \( W_1 \) and \( W_n \). The cumulative number of data sequences within \( DS_n \) is equal to \( |DS_n| = |W_1| + |W_2| + \ldots + |W_n| = n \times w \). In addition, the cumulative support of data sequence \( s \) in window \( n \) is defined as the ratio of the number of data sequences containing \( s \) within \( DS_n \) to the cumulative number of data sequences within \( DS_n \).

**Definition 3:** A data sequence \( s \) is defined as a cumulative frequent sequential pattern if \( \text{cum\_count}_n(s) \geq S_{amin} \times |DS_n| \), in which \( \text{cum\_count}_n(s) \) is the cumulative support count of the data sequences containing \( s \) within \( DS_n \) and \( S_{amin} \) is a user-defined minimum support threshold parameter for cumulative data streams and has a value in the range of \( 0 < S_{amin} < 1 \).

**Definition 4:** A data sequence is defined as a cumulative semi-frequent sequential pattern if \( S_{a sig} \times |DS_n| \leq \text{cum\_count}_n(s) < S_{amin} \times |DS_n| \), in which \( S_{a sig} \) is a user-defined significance threshold parameter for cumulative data streams and has a value in the range of \( 0 < S_{a sig} < S_{amin} < 1 \).

2.2 Significant Change Pattern

In a practical data stream mining system, it is impossible to store all of the streaming data. Thus, as new data arrives, old data must be discarded. The underlying concepts describing the stored streaming data may become out-of-date as new data arrives. In this study, the change in sequential patterns mined from data streams is defined as the difference in two sets of cumulative sequential patterns extracted from two datasets acquired from the same data streams over two different time periods. This study uses three general types of significant change pattern, namely frequency increasing patterns (Type I), frequency decreasing patterns (Type II) and structural change patterns (Type III), for finding the change information (i.e., the variation of the pattern types) between the previous and current windows. The detailed definitions of the significant change patterns are provided in Table 1.
Given certain user-defined thresholds for sequential pattern mining and significant change pattern discovery, respectively, report the frequent sequential patterns in the current and cumulative windows and identify any significant change patterns. In the event
that a significant change pattern is detected, predict the degree of change in the following mining round and modify the decay rate accordingly.

3. MINING CHANGES IN SEQUENTIAL PATTERNS OF DATA STREAMS

As shown by the three bold rectangles in Fig. 4, the proposed model comprises three modules. The Traditional Data Stream Miner module comprises three sub-modules, namely the Stream Collector, the Current Window Miner and the Results Merger. The Stream Collector collects input streams for a specified period of time and converts these streams into a set of data sequences. The Current Window Miner mines the sequential patterns from these sequences and stores the results as the Current Window Results. Finally, the Results Merger merges the Current Window Results and the Previous Cumulative Results to form the Recent Cumulative Results.

Although the Stream Collector, Current Window Miner and Results Merger enable the mining of sequential patterns within data streams, they do not provide the decision-maker with any information regarding the change in these patterns over time. Accordingly, this study incorporates a Change Detector and a Change Handler to quantify changes in sequential patterns and to modify the decay rate accordingly. The Change Detector processes the Recent Cumulative Results and Previous Cumulative Results to detect any significant change patterns, to compute the corresponding degree of change, and to predict the degree of change in the following mining round. If the state of data stream is unchanged between the previous and current mining round, the system reports only the significant change patterns. Otherwise, the system triggers the Change Handler to modify the decay rate for the following mining round. Finally, the Recent Cumulative Results are copied to the Previous Cumulative Results, and the system enters a new mining round. The details of the three modules are presented in the following subsections.

3.1 Traditional Data Stream Miner Module

The Stream Collector collects the input streams and converts them into corresponding data sequences, until the number of customers within the current window is equal to \( w \). The Current Window Miner mines the current frequent and current semi-frequent sequential patterns from the data sequences generated by the Stream Collector in the current mining round, and reports these patterns as the Current Window Results. Within each window, the mining process is performed in batch mode using the PrefixSpan algorithm [15]. The sequential patterns detected in the window are stored using a semi-frequent buffer [14].

The Results Merger merges the Current Window Results and Previous Cumulative Results to form the Recent Cumulative Results, and mines the cumulative semi-frequent and cumulative frequent sequential patterns in the resulting cumulative data streams. The Recent Cumulative Results are updated and reported in each mining round, and are then copied to the Previous Cumulative Results prior to the start of the following mining round. In mining data streams, some infrequent data sequences not stored in the Previous Cumulative Results may appear many times in the current window. In such a case, the cumulative support count of a sequence within \( DS_{n-1} \) is estimated as \( S_{\text{sig}} \times |DS_{n-1}| - 1 \).
In most real-time mining applications, newly-arriving data are more important than older data in terms of predicting future trends in data streams. Accordingly, the data collected in different time periods are weighted using a decay mechanism. Specifically, the Results Merger first estimates the cumulative supports of the sequences which are contained in the Current Window Results but not in the Previous Cumulative Results, and then decays the Previous Cumulative Results in accordance with the current value of the decay rate. The Results Merger then merges the Current Window Results with the decayed Previous Cumulative Results. Finally, the Results Merger searches for the cumulative semi-frequent and cumulative frequent sequential patterns in the present mining round in accordance with the values of $S_{min}$ and $S_{sig}$. Let the count of a pattern $S$ in window $W_k$ be denoted as $count_k(S)$ and let the decay rate be denoted as $d$. The cumulative count of pattern $S$ within $D_{S_k}$ (denoted as $cum_{-}count_k(S)$) is therefore obtained as

$$cum_{-}count_k(S) = \begin{cases} count_k(S), & \text{if } k = 1 \\ cum_{-}count_{k-1}(S) \times (1 - d) + count_k(S), & \text{if } k \geq 2 \end{cases}$$ (1)

Similarly, the number of recent cumulative data sequences within $D_{S_k}$ is obtained as

$$|D_{S_k}| = \begin{cases} |W_k|, & \text{if } k = 1 \\ |D_{S_{k-1}}| \times (1 - d) + |W_k|, & \text{if } k \geq 2 \end{cases}$$ (2)

where $|D_{S_{k-1}}|$ is the number of previous cumulative data sequences and $|W_k|$ is the number of data sequences within the current window.
3.2 Change Detector Module

The Change Detector plays a key role in the proposed model. The functions of the Change Detector can be summarized as follows: (1) to detect the significant change patterns in the current mining window; (2) to measure the degree of change in sequential patterns from the previous mining window to the current mining window; (3) to evaluate the present state of the data stream; (4) to predict the degree of change in the following mining round; and (5) to trigger the Change Handler if required. 

3.2.1 Significant change pattern detection

The Change Detector searches for ten significant frequency change patterns and one structural change pattern. The frequency change patterns are detected in accordance with Definitions 5-15 in Section 2.2. Meanwhile, the structural change pattern is found by checking the dissimilarity between the vanishing patterns in the previous mining round and the adding patterns in the current mining round using the method proposed in [9].

3.2.2 Change predictor

The Change Detector compares the Recent Cumulative Results with the Base Cumulative Results, and uses the difference between them to quantify the extent of the current change. Note that the Base Cumulative Results storing is the starting point of the observation for pattern changes and are set by default to the Previous Cumulative Results, i.e., the target of the comparison for calculating Change Degree. Having measured the degree of change, the Change Detector evaluates the current state of the data stream and then predicts the degree of change in the following mining round. Depending on the state of the data stream, the Change Detector triggers the Change Handler if required in order to modify the decay rate used in the Results Merger in the following mining round. The following paragraphs describe the Change Degree indicator used in this study to quantify the change, and then introduce the Change Predictor mechanism used to determine whether or not the Change Handler should be triggered. Finally, the four possible states of the data stream are introduced and described.

Definition 16: Change Degree (CD). In this study, the extent of the change between the Base Cumulative Results and the Recent Cumulative Results is quantified using the Change Degree (CD) indicator.

Let $\text{diff}$ be the total number of Type I & II significant change patterns (see Definitions 5-14) in the current mining round. Furthermore, let $|BR|$ be the number of cumulative semi-frequent and frequent sequential patterns in the Base Cumulative Results, and let $|RR|$ be the number of cumulative semi-frequent and frequent sequential patterns in the Recent Cumulative Results. The Change Degree, denoted as $CD_{b,r}$, is then expressed as

$$CD_{b,r} = \frac{\text{diff}}{|BR| + |RR|}$$

(3)
and has a value in the range of $0 \leq CD_{b,r} \leq 1$.

Having calculated the $CD$ in the current mining round, the Change Detector uses a Change Predictor mechanism to predict the $CD$ between the Recent Cumulative Results and the new cumulative results in the following round (see Fig. 5, in which $R$ denotes the cumulative results obtained in each mining round). The related definitions and calculations are as follows.

**Definition 17:** Predicted Change Degree ($PCD$). The Predicted Change Degree represents the predicted difference between the Recent Cumulative Results obtained in the current mining round and the new cumulative results obtained in the following mining round. $PCD$ has a value in the range of $0 \leq PCD \leq 1$ and is obtained by multiplying the estimated $CD$ by a correction factor (see Definition 20).

![Fig. 5. Prediction of change degree in following mining round.](image)

**Definition 18:** The prediction error ($\varepsilon$) is the absolute value of the difference between the predicted change degree and the actual change degree in the following mining round, i.e.,

$$\alpha_{k-1,i} = PCD_{k-1,i} - CD_{k-1,i}$$  \hspace{0.5cm} (4)

$$\varepsilon_{k-1,i} = |\alpha_{k-1,i}|$$  \hspace{0.5cm} (5)

where $CD_{k-1,i}$ is the actual Change Degree between windows $k-1$ and $k$, $PCD_{k-1,i}$ is the Predicted Change Degree between windows $k-1$ and $k$, and $\alpha$ is the result obtained when subtracting $CD$ from $PCD$. Since $PCD$ is bounded by the interval $0 \leq PCD_{k-1,i} \leq 1$ and $CD$ is bounded by the interval $0 \leq CD_{k-1,i} \leq 1$, it follows that $\alpha_{k-1,i}$ lies within the range of $-1 \leq \alpha_{k-1,i} \leq 1$ and $\varepsilon$ therefore has a value of $0 \leq \varepsilon \leq 1$.

**Definition 19:** Estimated Change Degree ($EstCD$). The Estimated Change Degree is the default value of $PCD$ and lies in the range of $0 \leq EstCD \leq 1$. $EstCD$ is calculated from the historical value(s) of $CD$, which is determind in turn by the state of the data stream.

![Fig. 6. Four possible states of data stream.](image)
**Definition 20: Correction Factor (CF).** The Correction Factor is used to adjust the accuracy of the Change Predictor such that it approaches 100%. The appropriate value of CF is determined by the value of α obtained from Eq. (4).

From the definitions above, the objective function of the PCD indicator has the form

\[
P_{CD, k+1} = \begin{cases} 
    \text{EstCD}_{k, k+1} \times CF_{k, k+1}, & \text{if } k > 2 \\
    \text{EstCD}_{k, k+1}, & \text{if } k = 2 
\end{cases}
\] (6)

and

\[
CF_{k, k+1} = \begin{cases} 
    1 - \alpha_{k, k+1}, & \text{if } -1 < \alpha_{k, k+1} < 1 \\
    1, & \text{if } \alpha_{k, k+1} = 1 \text{ or } -1
\end{cases}
\] (7)

subject to the constraint that \(0 < CF_{k, k+1} < 2\) and \(0 \leq P_{CD, k+1} \leq 1\) are both satisfied.

Note that in Eqs. (6) and (7), \(k\) is the current window, \(k+1\) is the following window; \(k-1\) is the previous window; \(\text{EstCD}_{k, k+1}\) is the Estimated Change Degree between windows \(k\) and \(k+1\); \(CF_{k, k+1}\) is the Correction Factor between windows \(k\) and \(k+1\); and \(P_{CD, k+1}\) is the Predicted Change Degree between windows \(k\) and \(k+1\). And since there is no historical information available to compute the Predicted Change Degree between Windows 1 and 2, i.e., \(P_{CD, 1, 2}\), the PCD predictor is used only after the second window (see Fig. 5).

### 3.2.3 Evaluation of data stream state

As shown in Fig. 6, this study defines four basic data stream states in accordance with the degree of change detected between the Base Cumulative Results and the Recent Cumulative Results (i.e., \(CD_{b,r}\)). The four states are defined as follows:

1. **Steady State:** If \(CD_{b,r}\) is less than an application-dependent fluctuation threshold \(T_f\), the system is said to be in the Steady State. Thus, \(\text{EstCD}_{now, now+1} = CD_{b,r}\).

2. **Fluctuation State:** If \(CD_{b,r}\) is larger than \(T_f\) but less than an application-dependent change threshold \(T_c\), the system is said to be in the Fluctuation State. The variation range of \(CD\) is still acceptable, and thus \(\text{EstCD}_{now, now+1} = CD_{b,r}\).

3. **Warning State:** If \(CD_{b,r}\) is larger than \(T_c\), the system is said to be in the Warning State and \(\text{EstCD}_{now, now+1} = CD_{b,r}\). When the data streams begin changing, this study will monitor the phenomenon for a time period to prevent noise, i.e., in order to confirm the change happens. In the Warning State, the Previous Cumulative Results in the present mining round are taken as the Base Cumulative Results and are stored for comparison with the new cumulative results. The current window ID is also stored (\(w_{begin}\)). To prevent noise, the system monitors the value of \(CD_{b,r}\) over a predefined time period (\(obs\_windows\)) and replaces the Base Cumulative Results with the current Previous Cumulative Results if the condition \(CD_{b,r} > T_c\) is not continuously satisfied over the observation period. Depending on the value of \(CD_{b,r}\), the system either remains in the Warning State or transits to the Steady State or Fluctuation State.
(4) Change State: If the system is in the Warning State but $CD_{h,r}$ is consistently greater than $T_c$ over the observation period, the system switches to the Change State. The $EstCD$ indicator in the following mining round is then taken as the average value of all the $CD_{h,r}$ values obtained since the system switched to the Warning State, i.e.,

$$EstCD_{\text{est,now} + 1} = \frac{\sum_{i \in \text{windows}} CD_{h,i}}{\text{obs}}.$$  \hspace{1cm} (8)

3.3 Change Handler Module

In this study, the Change Handler tunes the decay rate dynamically in each re-mining round in accordance with the real-time state of the data stream. Applying the dynamic decay rate can make the system more sensitive to changes than the static decay rate. The Change Handler is triggered whenever one of the four following state transitions occurs:

(1) Transition to Steady State: If the data stream transits to the Steady State, the decay rate ($d$) is set as $d = \text{steady}_d$, where the value of $\text{steady}_d$ is application-dependent and satisfies the condition $\text{steady}_d < \text{fluct}_d$.

(2) Transition to Fluctuation State: If the data stream transits to the Fluctuation State, the decay rate is set to $d = \text{fluct}_d$, where $\text{fluct}_d$ satisfies the condition $\text{fluct}_d < T_c$. Note that the Change Handler considers the range of the average $CD$ when the system is in the Fluctuation State, and thus $\text{fluct}_d = T_f + (T_c - T_f)/2$.

(3) Transition to Change State: When the system is in the Change State, the sequential patterns change more significantly from one mining window to the next and thus a higher value of the decay rate should be applied. In practice, the larger the value of $PCD$, the more the decay rate should be increased. Thus, the value of the decay rate is set as $d = PCD$. The system then replaces the Base Cumulative Results with the Previous Cumulative Results, and sets $EstCD_{\text{now,now} + 1} = CD_{h,r}$.

(4) Transition to Warning State: Before switching to the Change State, the system resides in the Warning State and is in an observation mode only. Thus, if the data stream transits to the Warning State from either the Steady State or the Fluctuation State, the decay rate is not changed. However, if the system changes from the Change State to the Warning State, the decay rate is set to $d = \text{fluct}_d$ to prevent an overly rapid decay.

Since the decay rate varies dynamically in accordance with the degree of change and the state of the data stream, the equations used to compute the cumulative count of the recent data sequences and the number of recent cumulative data sequences (i.e., Eqs. (1) and (2), respectively) must be modified as follows:

$$cum_{-count}(S) = \begin{cases} count_1(S) & \text{if } k = 1 \\ cum_{-count}_k(S) \times (1 - d_k) + count_k(S) & \text{if } k \geq 2 \end{cases}$$  \hspace{1cm} (9)
\[ |DS_k| = \begin{cases} |W_k| & \text{if } k = 1 \\ |DS_{k-1}| \times (1 - d_k) + |W_k| & \text{if } k \geq 2 \end{cases} \]  \tag{10}

where \(d_k\) is the decay rate used in \(W_k\).

### 4. EXPERIMENTAL RESULTS

The performance of the proposed model was evaluated and compared with that of alternative mining schemes by means of a series of simulations. The Current Window Miner was implemented using the PrefixSpan algorithm [15] while the Results Merger was implemented using the method described in Section 3.1. The proposed model was coded in C++, and was applied to both a real dataset and various synthetic datasets.

<table>
<thead>
<tr>
<th>Table 2. Notations used to describe synthetic datasets.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>T</strong></td>
</tr>
<tr>
<td><strong>I</strong></td>
</tr>
<tr>
<td><strong>D</strong></td>
</tr>
<tr>
<td><strong>L</strong></td>
</tr>
<tr>
<td><strong>C</strong></td>
</tr>
<tr>
<td><strong>S</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3. Parameter settings used in mining datasets.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>T10I4L50K</strong></td>
</tr>
<tr>
<td>(w=50000) [16]</td>
</tr>
<tr>
<td>(s_{\text{min}}=0.01) [16]</td>
</tr>
<tr>
<td>(s_{\text{avg}}=0.00999) [16]</td>
</tr>
<tr>
<td>(s_{\text{min}}=0.00995) [16]</td>
</tr>
<tr>
<td>(d=0), without decay</td>
</tr>
<tr>
<td>(\text{obs windows}=3)</td>
</tr>
</tbody>
</table>

Four synthetic datasets were generated using the IBM data generator [17], namely **T10I4L50K**, **C200T2S8I1.25**, **T5I4D1000K-AB, C200T2S8I1.25-AB**, and **FoodMart2000**. (Note that the notations used in describing each dataset are shown in Table 2.) **T5I4D1000K-AB** contains item sequences and comprises two consecutive subparts, namely **TA** and **TB**. Subpart **TA** contains a set of transactions relating to a set of items **A**, while subpart **TB** contains a set of transactions relating to a set of items **B**. Note that subparts **TA** and **TB** have no items in common. Both subparts contain 500,000 transactions relating to 1000 items. **C200T2S8I1.25-AB** has the same structure as **T5I4D1000K-AB**. However, the two subparts contain a total of 1,000,000 item sequence.

The threshold values of the four data stream states (see Fig. 6) were assigned in accordance with a normal distribution. Specifically, the statistical confidence level for the Change State was specified as 68%, i.e., \(T_c=32\%\) (significance level), while the statistical confidence level for the Steady State was set as 95%, i.e., \(T_f=5\%\) (significance level). Thus, the \(CD\) of the system when in the Fluctuation State varied from 5% to 32%. Consequently, \(\text{fluct}_d\) was assigned the median value of \(\text{fluct}_d=18.5\%\). The default decay
rate was specified as 0.001, i.e., the same value as in [5]. Finally, the window size, minimum support and semi-buffer parameter values were assigned the same values as those used in the particular method chosen for comparison purposes. Table 3 summarizes the parameter settings used in the simulation experiments. Note that the citation numbers in Table 3 indicate the studies from which the related parameter settings are taken.

The first set of simulations evaluated the precision of the Traditional Data Stream Miner in mining sequential patterns within data streams. The precision of the mining results was quantified using the parameter Precision(R1/R2) = |R1 ∩ R2|/|R2| [18], where R1 denotes the mining results obtained using this study and R2 denotes the results obtained using PrefixSpan. The simulations were performed using T10I4L50K dataset used in SS-MB [16] and the C200T2.5S10I1.25 dataset used in PrefixSpan.

The average precision value for the T10I4L50K dataset was found to be 0.89. This result is similar to that obtained when evaluating the same dataset using the SS-MB with the same parameter settings [16]. Meanwhile, the average precision value for the C200T2.5S10I1.25 dataset was found to be 0.91 when using the same parameter settings as those used in IncSpan. The mining performance of the Traditional Data Stream Miner was also quantified using the ASE measure defined in eISeq given the same datasets and parameter settings. The performance was found to be similar to that obtained using eISeq. (Note that the corresponding results are not present here due to length constraints.) Overall, the results show that the Traditional Data Stream Miner achieves a good precision when applied to datasets comprising item_sequences or itemset_sequences.

The results presented above compare the mining performance of this study with that of PrefixSpan (i.e., a static mining method in which the sequential patterns remain static over time). However, in practical mining systems, the sequential data patterns change over time, the precision values of data stream mining methods are hardly the same as those of a static mining method. Accordingly, a second series of simulations was conducted to evaluate the performance of the Change Detector/Change Handler in responding to changes within sequential patterns of the data streams in the T5I4D1000K-AB and C200T2S8I1.25-AB datasets. In performing the simulations, the accuracy of the Change Predictor (from the third window onwards) was quantified via the average prediction error (see Definition 19 and Eq. (5)). Meanwhile, the adaptability of the model in responding to changes in data streams was evaluated using the Coverage Rate CR(X) measure proposed in eISeq, i.e., CR(X) = (# of frequent patterns induced by an item set X )/ |R|)*100 (%), where |R| is the total number of frequent patterns in result set R. The corresponding results are presented in Figs. 7 and 8 for the T5I4D1000K-AB and C200T2S8I1.25-AB datasets, respectively. Note that in both figures, the annotation “with C.H.” indicates that the Change Handler was used in the corresponding experiment, while the annotation “no C.H.” indicates that the Change Handler was not used.

Figs. 7 (a)-(c) show the variations of the data stream state and the Coverage Rate with and without the Change Handler, respectively, when mining the T5I4D1000K-AB dataset using the same minimum support threshold (Smin = 0.001) as that used in eISeq. Since T5I4D1000K-AB is an item_sequence dataset, most of the CD values are less than 5%, and thus the system resides in the Steady State in the first five windows, irrespective of whether or not the Change Handler is applied (see Fig. 7 (a)). However, in Windows 6, 7, 8 and 9, the data stream resides continuously in the Fluctuation State when the Change Handler is not applied, but transits through the Fluctuation-Warning-Warning-Change
States when the Change Handler is applied. Although the data stream begins shifting from the Steady State to the Fluctuation State between Windows 5 and 6, the Change Handler does not tune the decay rate until Window 7 (i.e., the Change Handler response lags the change by one window). The average prediction error over Windows 3-10 was found to be 0.02 without the Change Handler and 0.03 with the Change Handler.

Figs. 7 (b) and (c) show the effect of the Change Handler on the Coverage Rate when applied to the T5I4D1000K-AB dataset. The results confirm that the Change Handler renders the model more sensitive to changes in the significant sequential patterns; particularly after Window 5, where the decay rate is significantly increased in order to compensate for the shift in the stream state from the Steady State to the Fluctuation State. In mining the T5I4D1000K-AB dataset using the elSeq algorithm, the best case is $CR(A) = 0$ in 900000th stream given $h = 100000$, while the worst case is $CR(A) = 0.4$ in 1000000th stream given $h = 500000$. By contrast, $CR(A) = 0$ in Windows 9 and 10 ($w = 100000$) when using this study (see Fig. 7 (c)). $CR(A) = 0$ in this study appeared earlier than in elSeq. Thus, it is inferred that the proposed Change Handler mechanism results in a more adaptive mining model than the elSeq scheme. (Note that Fig. 7 shows the results obtained for a setting of $S_{min} = 0.001$. However, similar results (not presented here) were also found for settings of $S_{min} = 0.01$.) Consequently, the Change Handler is sensitive to changes.

The performance of the Change Handler when applied to a dataset comprising two subparts with different itemset_sequence data was evaluated using the C200T2S8I1.25-AB dataset. The corresponding results were found to be very similar to those obtained for the T5I4D1000K-AB dataset (see Figs. 8 (a)-(c)). The average prediction errors with and
without the Change Handler were 0.03 and 0.02, respectively. In other words, the accuracy of the Change Predictor is independent of the type of input stream, i.e., itemset_sequence data or item_sequence data. However, comparing Fig. 8 (b) with Fig. 8 (c), it is seen that the Change Handler has a more notable effect on the adaptability of the mining model when applied to datasets containing itemset_sequence data.

Fig. 8. (b) Coverage Rate for C200T2S8I1.25-AB without the Change Handler.

Fig. 8. (c) Coverage Rate for C200T2S-8I1.25-AB with the Change Handler.

Fig. 9. (a) State of the data stream for FoodMart-2000 using $S_{\text{min}} = 0.01$.

Fig. 9. (b) Significant change pattern examples of FoodMart2000.

The third series of simulations investigated the performance of this study when applied to the mining of a real dataset, i.e., FoodMart2000 [9]. This real dataset was extracted from the Microsoft SQL Server 2000. The simulations considered the Sales_Fact_1997 and Sales_Fact_1998 datasets in FoodMart2000. The two datasets comprised a total of 54537 transactions collected from 13405 customers for 1559 different product items. Fig. 9 (a) shows the variation of the data stream state when using $S_{\text{min}} = 0.01$ and a window size of one quarter (i.e., three months). (Note that the average number of sequences per quarter was equal to 16666.75.) As shown, the data stream state transits through the sequence Warning–Warning-Change over Windows 3, 4 and 5, respectively. The average prediction error over Windows 2-8 was found to be 0.021 and 0.025 with and without the Change Handler, respectively. Fig. 9 (b) shows some typical examples of the significant change patterns mined from the FoodMart2000 dataset. Note that the annotation “none” indicates that no significant change patterns were detected in the window for the corresponding product item. Note also that since the items in FoodMart2000 are sparsely distributed, most of the mined sequential patterns included only one item. As shown, the product “Token Diet Cola” exhibited an FFCP change pattern in...
Q3 1997, an FSCP change pattern in Q4 1997, and an SFCP change pattern in Q1 1998. In other words, through this study, the manager can find out the sales volume trend of a frequent or semi-frequent product purchase sequence (i.e., a frequent or semi-frequent sequential pattern), and the manager could adjust the inventory or give promotion of such product purchase sequence in some specific quarters. Even he could inspect the reasons for such emerging trend via questionnaires or phone interview for customers. Overall, the results presented in Fig. 9 (b) confirm the ability of this study to detect the significant change patterns within a real-world dataset, and to provide decision-makers with the information required to detect emerging trends and to formulate a timely and appropriate response. Note that the performance of the proposed model was also verified using a window size of one month. The results (omitted here due to length constraints) were found to be similar to those presented in Figs. 9 (a) and (b).

We had performed experiments about efficiency. The experimental environments are Intel Xeon 3.2GHz, 4 cores, 4GB RAM, and the OS is Linux 2.6.18. In order to simplify (due to length constraints), only the efficiency results of FoodMart2000 are shown. The time shown in Table 4 represents the total execution time (i.e., the summary time of all windows) of each module. As shown in the table, the Change Handler can modify decay rate adaptively in accordance with the degree of change of the patterns, most out-of-date streams could be dropped, therefore, the mining and detecting time is usually less than that of without the Change Handler module. Moreover, the work of the Change Handler module is only to modify decay rate, if required, most of the execution time are far less than 0.01 sec. Thus, the Change Handler Cost of with C.H. is denoted by “~0 sec”. In summary, the Change Detector and Change Handler modules can provide change information, drop out-of-date streams, and give appropriate responses for changing environments.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Traditional Data Stream Miner Cost</th>
<th>Change Detector Cost</th>
<th>Change Handler Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>FoodMart2000</td>
<td>With C.H. 0.06 sec</td>
<td>0.07 sec</td>
<td>0 sec</td>
</tr>
<tr>
<td></td>
<td>No C.H. 0.07 sec</td>
<td>0.2 sec</td>
<td>none</td>
</tr>
</tbody>
</table>

5. CONCLUSION

A model has been proposed for mining changes in sequential patterns within data streams. In the proposed approach, the degree of change between the current mining round and the following mining round is predicted and used to modify the decay rate accordingly. The model provides an effective means of mining the sequential pattern changes in real-world applications such as WSNs, web logs, consumer purchase sequence analysis systems, and so on. The experimental results have confirmed the ability of the proposed model to detect the significant change patterns in both item_sequence and itemset_sequence datasets. Moreover, it has been shown that the Change Detector and Change Handler yield a significant improvement in the adaptability of the mining model in response to changes in the data stream patterns. As a result, this study provides decision-makers with a powerful tool for detecting and formulate changes within real-
world data streams and formulate appropriate strategies in response. A future study will aim to enhance the performance of the proposed model by means of a more efficient mining algorithm and the use of parallel or cloud computing techniques.

REFERENCES

14. H. Cheng, X. Yan, and J. Han, “IncSpan: Incremental mining of sequential patterns in large database,” in *Proceedings of the 10th ACM SIGKDD International Confer-


17. Data generator (http://www.almaden.ibm.com/cs/quest) of IBM.


**I-Hui Li** (李怡慧) received her M.S. degree from National Chiao Tung University, Hsinchu, Taiwan, in 1995, and the Ph.D. degree in Computer Science and Engineering from National Chung Hsing University, Taiwan in 2011. She is currently an Assistant Professor in the Department of Information Networking and System Administration of Ling Tung University, Taiwan. Her research interests are in data mining, algorithms and wireless networks.

**Jyun-Yao Huang** (黃俊耀) received the BS degree in mathematics from National Chia Yi University, Taiwan in 2004 and MS degree in institute of networking and multimedia from National Chung Hsing University, Taiwan. He is currently a Ph.D. student in Department of Computer Science and Engineering Department at National Chung Hsing University, Taiwan. His research interests are in the areas of applied cryptography, network security and semantic web, with current focus on secure data services in cloud computing.

**I-En Liao** (廖宜恩) received the Ph.D. degree in Computer and Information Science from the Ohio State University in 1990. He is currently a Professor and the Chairman of the Department of Computer Science and Engineering of National Chung Hsing University, Taiwan. His research interests are in data mining, XML database, and wireless networks. He is a member of the ACM and the IEEE Computer Society.