International Journal of Innovative Computing, Information and Control
Volume **8**, Number **1(A)**, January **2012**

ICIC International ©2012 ISSN 1349-4198
pp. **285-302**

PREDICTING SEQUENTIAL PATTERN CHANGES IN DATA STREAMS

I-Hui Li[1,2], Jyun-Yao Huang[1] and I-En Liao[1]

[1]Department of Computer Science and Engineering
National Chung Hsing University
No. 250, Kuo Kuang Rd., Taichung 402, Taiwan
phd9301@cs.nchu.edu.tw; allen501pc@gmail.com; ieliao@nchu.edu.tw

[2]Department of Information Networking and System Administration
Ling Tung University
No. 1, Ling Tung Rd., Taichung 408, Taiwan

Received August 2010; revised February 2011

Abstract. *Data streams are utilized in an increasing number of real-time information technology applications. Unlike traditional datasets, data streams are temporally ordered, fast changing and massive. Due to their tremendous volume, performing multiple scans of the entire data stream is impractical. Thus, traditional sequential pattern mining algorithms cannot be applied. Accordingly, the present study proposes a new sequential pattern mining model for predicting sequential pattern changes in data streams. The experimental results show that the prediction performance of the proposed model is better than that of two linear regression-based models.*
Keywords: Data streams, Sequential pattern mining, Pattern changes, Prediction

1. Introduction.

Real-time data processing is an important requirement in many fields nowadays, including network monitoring and traffic analysis, web log and click-stream mining, wireless sensor network (WSN) data analysis, dynamic stock fluctuation tracking, manufacturing process analysis, weather data and power consumption mining, and so on. The data streams in such applications are typically large, fast, unbounded and composed of continuous data elements. As a result, achieving a real-time mining of the data set is extremely challenging.

Traditional data mining techniques such as classification, clustering and association rule finding are well-developed and have spurred extensive research into the development of more advanced data mining techniques. Among the various advanced mining methods which have been proposed, sequential pattern mining, a method based on association rule mining, provides the means to identify frequently occurring orders or trends within a data stream and detect (or make predictions about) anomalies in the data. Therefore, sequential pattern mining algorithms have found use in a wide variety of applications, including DNA sequence analysis [1], website navigation patterns finding [2,3], customer purchase sequence analysis, hyponymy relation between Chinese terms extracting system [4] and route suggestion systems [5].

Data streams are utilized in an increasing number of applications nowadays. Unlike traditional data sets, data streams are temporally ordered, fast changing and massive. Due to their tremendous volume, it is impractical to perform multiple scans of the entire data stream. As a result, traditional sequential pattern mining algorithms cannot be easily applied to the analysis of most real-world data streams. Consequently, many advanced sequential pattern mining algorithms have been developed for data stream analysis. However, the problem of detecting (and predicting) changes in the sequential patterns within a dataset has been largely ignored, despite the fact that such changes may lead to an overdue mining model. Besides, the change status varies over time, and the degree of change in the sequential patterns within a dataset is important for decision-makers to give appropriate response to emerging trends.

Accordingly, this paper proposes a new sequential pattern mining model and a predictor for analyzing sequential pattern changes in large-scale data streams and predicting changes in these pattern types in accordance with their degree of change. The experimental results show that the proposed approach enables a more accurate prediction of pattern changes than that obtained using conventional linear regression-based models.

The major contributions of this study can be summarized as follows:

1. A sequential pattern mining model is proposed for identifying the sequential patterns within a data stream.
2. A method is proposed for evaluating the degree of change of the sequential patterns within a data stream based upon a root mean square (RMS) metric.
3. A mechanism is proposed for predicting future changes in the sequential patterns within a data set in accordance with their degree of change properties.

The remainder of this paper is organized as follows: Section 2 describes previous work in the sequential pattern mining field; Section 3 introduces the model proposed in this study for predicting changes in the sequential patterns within a data stream; Section 4 describes the experimental design and presents the experimental results; finally, Section 5 provides some brief concluding remarks.

2. **Related Work.**

2.1. **Sequential pattern mining over data streams.**

Raïssi et al. [6] proposed a method, designated as SPEED, for mining maximal sequential patterns over data streams. SPEED uses a titled window and finds the maximal sequential patterns by means of a region lattice. SPEED has a simple input format, and is, therefore, straightforward in use. However, it assumes the existence of only one item in each element, whereas in most practical applications, each element includes many items. IncSPAM [7] performs the mining of sequential patterns over a sliding window using a lexicographical tree and reflects the greater importance of more recent information by means of a static decay function. The mining performance is enhanced by storing the recent data stream information in the form of a bitmap within a vertical database. However, the information associated with each different customer is stored within a different window, and thus, a large memory space is required. Furthermore, if a large number of items exist, the lexicographical tree becomes large and unwieldy, and thus, the mining efficiency is reduced. DSM-PLW [8] utilizes a forest structure to accomplish the single-pass mining of path traversal patterns over streaming web click-sequences. However, for most real-world web click-sequences, the forest structure is highly complex, and thus, mining involves a cumbersome and time-consuming computational process. MILE [9,10] accomplishes the simultaneous mining of multiple streams using a pattern-growth method known as PrefixSpan [11]. However, PrefixSpan is a static mining approach, and therefore, requires the data to be scanned multiple times. Consequently, MILE is inappropriate for the real-time mining of practical data streams.

2.2. **Change mining.**

Change is an important characteristic in real-time applications and has therefore attracted considerable attention in the data mining field. Liu et al. [12] argued that by knowing *what* is changing and *how* it is changing, a business can provide the products and services required to continuously satisfy changing market needs. Chen et al. [13] identified changes in customer behavior by comparing the association rules generated from two datasets acquired over different time periods. According to the authors, the detection and prediction of changes in customer behavior is beneficial to a business in developing long-term and pleasant customer relationships. However, the sequential patterns which occur frequently in one time-period may not do so in another [14]. In other words, a user's recent behavior is not necessarily the same as his or her previous behavior, and a pattern which occurred frequently in the past may seldom (or indeed never) occur again in the present or future [15].

Despite the importance of change in many practical applications, the literature contains no sophisticated methods for the rigorous change mining of the sequential patterns within a data stream. Existing proposals for mining changes in the sequential patterns within a data stream can be broadly categorized into two groups. In the first group, a static decay function is used to assign greater importance to new elements within the data stream, while a pruning mechanism is used to remove old elements. For example, IncSPAM [7] uses a static decay function with a user-defined decay value of 0.999 and prunes any nodes which occur only infrequently in the lexicographical tree. RSP-DS [16] and SS-MB [17] perform a pruning operation at fixed time intervals. By contrast, GraSeq [18] uses the static decay function of estDec [19], and therefore, considers only the timestamp when performing the pruning process. The estDec, estWin [20] and eISeq [21] use the same decay function which is based on a *decay-base*, a *decay-base-life* and a *safety factor*. The *safety factor* represents the maximum number of the most recent consecutive transactions containing a new itemset, where the itemset is infrequent. As a result, pruning can be performed either periodically or whenever the current size of the monitoring lattice reaches a pre-defined threshold value. In estDec, estWin and eISeq, a *Coverage Rate CR(X)* metric was used to illustrate the speed of these methods in responding to information changes in a data stream. In addition, the performance of the eISeq decay function was evaluated using the *T5.I4.D1000K-AB* data set generated by Agarwal and Srikant [22]. The results showed that the speed of eISeq in adapting to the transition of information between the two subparts of the data set (i.e., subpart TA and subpart TB) increased as the value of the *decay-base-life* parameter was reduced.

The second group of methods proposed in the literature for mining changes in the sequential patterns within a data stream focus on the definition and identification of *significant* change patterns. Li et al. [23] proposed a method for the online mining of item changes over continuous append-only and dynamic data streams. Five different types of item were considered, namely "*frequent frequency changed items*", "*sub-frequent frequency changed items*", "*infrequent frequency changed items*", "*vibrated frequency changed items*" and "*stable frequency changed items*". In [24], the authors extended the proposed method to enable the detection of changes in user-centered musical query streams (MQSs). In the proposed approach, designated as MQS-change, a landmark window was used to store two musical melody structures, and four melody structure changes (*positive burst*, *negative burst*, *increasing change* and *decreasing change*) were monitored using a new data structure, MSC-list. Tsai and Shieh [14] proposed a method for detecting three different types of change pattern, namely "*emerging sequential patterns*", "*unexpected sequence changes*" and "*added/perished sequential patterns*". The method provides a straightforward means of identifying changes in sequential patterns, but is not specific to the change mining of data streams.

Liu et al. [25] proposed an event change detection (ECD) method based on a change mining approach and a concept hierarchy for mining the change of environmental event trends. In the proposed approach, association rule mining was performed initially to discover the event trend, and the ECD method was then applied to detect changes in this trend in order to enable managers to respond rapidly to changes in the external environment. In detecting event changes, five types of change pattern were considered, namely *emerging event patterns*, *added event patterns*, *perished event patterns*, *unexpected consequent changes of event patterns* and *unexpected condition changes of event patterns*.

The methods described above all concern the change mining of the sequential patterns within a dataset. However, none of the methods consider the actual degree of change of these patterns, or the detection and prediction thereof. Nonetheless, the degree of change in the sequential patterns within a dataset is important since the change status varies over time. Accordingly, this paper develops a model for mining sequential patterns in data streams and for predicting changes in these patterns based upon their degree of change.

3. Model for Predicting Sequential Pattern Changes in Data Streams.

Figure 1 illustrates the proposed model for predicting sequential pattern changes in data streams. As shown, the model commences with a data preprocessing step, in which the data stream information is collected in batches of a fixed size. The sequential patterns within the current batch of data streams are mined and are then merged with the patterns detected in the previous mining rounds. A search is then made for 6 different types of change pattern. Having established the change type of each pattern, the degree of change of the corresponding pattern is computed and used to predict the pattern type in the following batch of data stream information. The entire process is then repeated as required. The details of each step are described in the following sections.

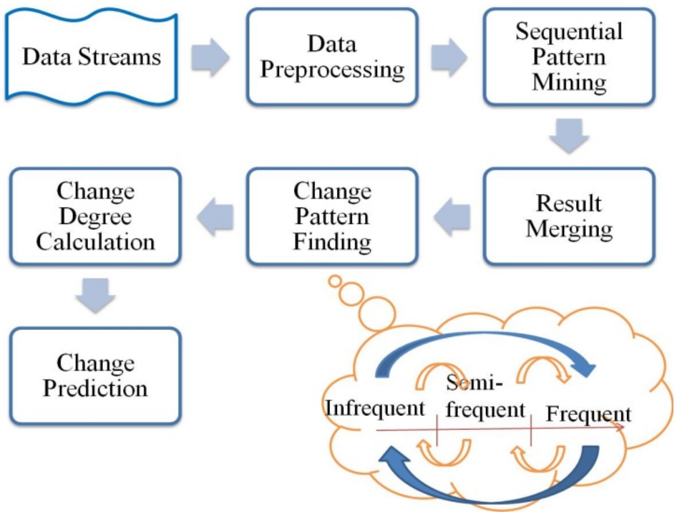

Figure 1. Proposed model for predicting sequential pattern changes in data streams

3.1. Data preprocessing and sequential pattern mining.

The problem of mining sequential patterns was introduced in [26] and then extended in [27]. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals 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 ordered items. In other words, sequence s can be denoted as $\langle s_1\ s_2 \ldots s_n \rangle$, where s_j is an element, $j \in 1 \ldots n$. A sequence $S_1 = \langle a_1\ a_2\ \ldots\ a_n \rangle$ is a subsequence of another sequence $S2 = \langle b_1\ b_2\ \ldots\ b_m \rangle$ if there exists a set of integers, $1 \le i_1 < i_2 < \ldots i_j \ldots < i_n \le i_m$, such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}, \ldots, a_n \subseteq b_{in}$. For example, sequence $\langle$A,C,D$\rangle$ is a subsequence of $\langle$A,(BC),(AD)$\rangle$. If every element has only one item

CID	Timestamp	Itemset
1	1	(abd)
2	2	(b)
1	3	(bcd)
2	4	(abc)
3	5	(ab)
2	6	(bcd)
3	7	(bcd)
4	8	(bd)
2	9	(ac)
1	10	(bc)
3	11	(abc)
2	12	(cd)
4	13	(ad)
3	14	(ad)
4	15	(cd)

Figure 2. Typical example of input streams

in a sequence, the sequence is referred to hereafter as an item_sequence; otherwise, the sequence is referred to as an itemset_sequence.

In this study, the streams are modeled as a series of transactions. Let each transaction T consist of a customer-ID (CID), a timestamp and an itemset (see Figure 2). In the data preprocessing step, having collected all the 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 continues to collect stream information until *batch_size* data sequences have been generated, where *batch_size* is a fixed value representing the size of the batch. Note that since each data sequence is associated with a particular CID, the number of customers within a batch is equal to the number of sequences (see Figure 3 for example, in which *batch_size* = 4). Thus, each data sequence extracted from the input stream includes three pieces of information, namely CID, Batch-ID and the sequence itself. For each data sequence s, a support count (denoted as count(s)) is generated to indicate the number of data sequences in the batch within which s is contained. The support of a sequence s is the proportion of total sequences in the dataset which contain s. Such as count ($\langle$(ab)(c)$\rangle$) = 3 and the support of $\langle$(ab)(c)$\rangle$ is 0.75 (i.e., 3/4) in Figure 3.

CID	Batch-ID	Sequence
1	1	(abd)(bcd)(bc)
2	1	(b)(abc)(bcd)(ac)(cd)
3	1	(ab)(bcd)(abc)(ad)
4	1	(bd)(ad)(cd)

Figure 3. Example of data sequences extracted from Figure 2

A data stream, D, can therefore be modeled as a series of batches, i.e., $D = D_1 + D_2 + \ldots + D_i + \ldots$, where the related notations are as follows:

- D_i: batch i of the data stream.
- $|D_i|$: number of sequences in batch i.
- $D_{i,j}$: sequences accumulated from batch i to batch j.

- $|D_{i,j}|$: number of sequences accumulated from batch i to batch j.
- $\text{cur\_cnt}_t(p)$: current count of sequence p in batch t.
- $\text{cur\_sup}_t(p) = \text{cur\_cnt}_t(\text{p})/|D_t|$: current support of sequence p in batch t.
- $\text{cum\_cnt}_t(p)$: cumulative count of sequence p in D_{1t}.
- $\text{cum\_sup}_t(p) = \text{cum\_cnt}_t(\text{p})/|D_{1,t}|$: cumulative support of sequence p in $D_{1,t}$.

For each batch, sequential pattern mining is performed using the PrefixSpan algorithm [11]. The mining results for the current batch are then merged with those for previous batches in order to obtain a cumulative mining result. The sequential patterns identified during the mining process are stored with a semi-buffer δ $(0 < \delta < 1)$, of which the support is less than but near *min_sup*. Figure 4 shows the generic pattern types considered in the mining process. If the support of pattern p is less than $\delta * min\_sup$, p is considered to be *infrequent*. Meanwhile, if the support of pattern p is less than *min_sup* but equal to or larger than $\delta * min\_sup$, p is considered to be *semi-frequent*. Finally, if the support of pattern p is equal to or larger than *min_sup*, p is considered to be *frequent*. In other words, the proposed model involves a total of six different pattern types, namely *current infrequent patterns*, *current semi-frequent patterns*, *current frequent patterns* (identified in the Sequential Pattern Mining step) and *cumulative infrequent patterns*, *cumulative semi-frequent patterns* and *cumulative frequent patterns* (constructed in the Result Merging step, see the following section).

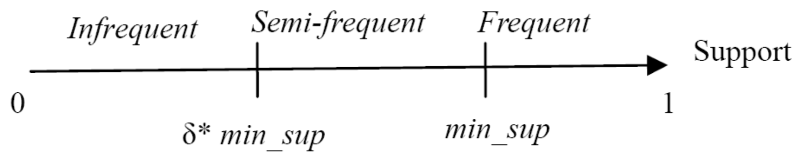

FIGURE 4. Pattern types

3.2. **Result merging.** One of the most important steps in the mining of data streams is that of calculating the support of a pattern based on both the current mining results and the previous results. In the present study, a decay mechanism similar to that used in [7,18-21] is applied to assign new data in the data stream a greater importance than old data within the stream. Thus, the cumulative count of sequence p in $D_{1,t}$ and the total number of cumulative sequences from the first batch to batch t are given respectively as follows:

$$\begin{aligned}&\text{cum\_cnt}_t(p) = \text{cur\_cnt}_1(p), \text{ for } t = 1\\&\text{cum\_cnt}_t(p) = \text{cur\_cnt}_{t-1}(p) * d + \text{cur\_cnt}_t(p), \text{ for } t > 1\end{aligned} \tag{1}$$

$$\begin{cases}|D_{1,t}| = |D_1|, \text{ for } t = 1\\|D_{1,t}| = |D_{1,t-1}| * d + |D_t|, \text{ for } t > 1\end{cases} \tag{2}$$

where d denotes the decay rate.

Moreover, as there is no information stored about an infrequent sequential pattern of previous results in data streams mining, in accordance with the method proposed in [20,21], the previous cumulative count of the infrequent sequential pattern p from the first batch to batch $t-1$ is estimated as:

$$\text{cum\_cnt}_{t-1}(p) = \delta * min\_sup * |D_{1,t-1}| - 1 \tag{3}$$

If $\text{cum\_sup}_t(p) \geq min\_sup$, then p is a *cumulative frequent sequential pattern*. Meanwhile, if $min\_sup > \text{cum\_sup}_t(p) \geq \delta * min\_sup$, p is a *cumulative semi-frequent sequential pattern*. Finally, if $\text{cum\_sup}_t(p) < \delta * min\_sup$, p is a *cumulative infrequent sequential pattern*.

3.3. **Change pattern finding.** Due to the huge size of real-world data streams, it is impossible to store all of the data in the stream for mining purposes. Thus, in practice, only the most recent data is stored, and as new data sequences are added, old sequences are discarded. The concepts describing the stored streaming data may change as new data sequences are added. Thus, this paper proposes the following definition for the change in the sequential patterns within a data stream.

Definition 3.1. *The change in the sequential pattern mined from a data stream is defined as the difference between two sets of cumulative sequential patterns generated from two datasets obtained over different time periods.*

The problem of detecting sequential pattern changes in data streams has received extensive attention in the literature. However, previous studies have not provided a detailed description of the types of changes which may occur. Therefore, as shown in the following, the present study defines six specific types of change pattern which may occur within a data stream over time.

Definition 3.2. *A cumulative semi-frequent sequential pattern p is called an ISCP change pattern, if:*
(1) $min\_sup \times |D_{1,t}| > cum\_cnt_t(p) \geq \delta * min\_sup \times |D_{1,t}|$ *and*
(2) $cum\_cnt_{t-1}(p) < \delta * min\_sup \times |D_{1,t-1}|$,
where t is the index of the current batch.

In other words, an *ISCP* change pattern is said to occur when pattern p transits from an *in-frequent* pattern to a *semi-frequent* pattern.

Definition 3.3. *A cumulative frequent sequential pattern p is called an IFCP change pattern, if:*
(1) $cum\_cnt_t(p) \geq min\_sup \times |D_{1,t}|$ *and*
(2) $cum\_cnt_{t-1}(p) < \delta * min\_sup \times |D_{1,t-1}|$.

In other words, an *IFCP* change pattern is said to occur when pattern p transits from an *in-frequent* pattern to a *frequent* pattern. The change pattern is dubbed an *adding pattern.*

Definition 3.4. *A cumulative frequent sequential pattern p is called an SFCP change pattern, if:*
(1) $cum\_cnt_t(p) \geq min\_sup \times |D_{1,t}|$ *and*
(2) $min\_sup \times |D_{1,t-1}| > cum\_cnt_{t-1}(p) \geq \delta * min\_sup \times |D_{1,t-1}|$.

In other words, an *SFCP* change pattern is said to occur when pattern p transits from a *semi-frequent* pattern to a *frequent* pattern. The change pattern is dubbed an *adding pattern.*

Definition 3.5. *A cumulative semi-frequent sequential pattern p is called an FSCP change pattern, if:*
(1) $min\_sup \times |D_{1,t}| > cum\_cnt_t(p) \geq \delta * min\_sup \times |D_{1,t}|$ *and*
(2) $cum\_cnt_{t-1}(p) \geq min\_sup \times |D_{1,t-1}|$.

In other words, an *FSCP* change pattern is said to occur when pattern p transits from a *frequent* pattern to a *semi-frequent* pattern.

Definition 3.6. *A cumulative in-frequent sequential pattern p is called an FICP change pattern, if:*
(1) $cum\_cnt_t(p) < \delta * min\_sup \times |D_{1,t}|$ *and*

(2) $cum\_cnt_{t-1}(p) \geq min\_sup \times |D_{1,t-1}|$.

In other words, an *FICP* change pattern is said to occur when pattern p transits from a *frequent* pattern to an *in-frequent* pattern. The change pattern is dubbed a *vanishing pattern.*

Definition 3.7. *A cumulative in-frequent sequential pattern p is called an SICP change pattern, if:*

(1) $cum\_cnt_t(p) < \delta * min\_sup \times |D_{1,t}|$ *and*
(2) $min\_sup \times |D_{1,t-1}| > cum\_cnt_{t-1}(p) \geq \delta * min\_sup \times |D_{1,t-1}|$.

In other words, an *SICP* change pattern is said to occur when pattern p transits from a *semi-frequent* pattern to an *in-frequent* pattern. The change pattern is dubbed a *vanishing pattern.*

3.4. **Change degree calculation.** Since the *ISCP* and *SICP* change patterns do not belong to *frequent* patterns, the degree of change computation is based only on the *FICP*, *IFCP*, *SFCP* and *FSCP* patterns. In the model proposed in this study, the degree of change is based on the root mean square (RMS) metric (discrete distribution mode). In general, the RMS of a variable x, denoted as $R(x)$, is calculated as:

$$R(x) = \begin{cases} \sqrt{\dfrac{\sum_{i=1}^{n} x_i^2}{n}}, & \text{for a discrete distribution} \\ \sqrt{\dfrac{\int P(x)x^2dx}{\int P(x)dx}}, & \text{for a continuous distribution} \end{cases} \tag{4}$$

In calculating the degree of change of the sequential patterns in a data stream, let A be a change pattern set from the set *FICP*, *IFCP*, *SFCP* and *FSCP*, t be the current batch index, and a_i be a change pattern i from the set *FICP*, *IFCP*, *SFCP* and *FSCP*. Furthermore, let d_{a_i} represent the difference ratio of the cumulative support of a_i in two continuous batches. Here, d_{a_i} corresponds to variable x in Equation (4). Let the numerator of the radical in Equation (4) be denoted as Diff_A. d_{a_i} and Diff_A are given respectively as:

$$\text{Diff}_A = \sum_{i=1}^{|A|} d_{a_i}^2 \quad a_i \in A \tag{5}$$

$$d_{a_i} = \frac{|\text{cum\_sup}_{t-1}(a_i) - \text{cum\_sup}_t(a_i)|}{\max\left\{\text{cum\_sup}_{t-1}(a_i), \text{cum\_sup}_t(a_i)\right\}} \tag{6}$$

Note that d_{a_i} in Equation (6) has a value in the range $0 \leq d_{a_i} \leq 1$. From Equations (5) and (6), the Change Degree (*CD*) can be calculated as:

$$CD = \max\left\{\sqrt{\frac{\text{Diff}_{SFCP}}{|SFCP|}}, \sqrt{\frac{\text{Diff}_{FSCP}}{|FSCP|}}, \sqrt{\frac{\text{Diff}_{IFCP}}{|IFCP|}}, \sqrt{\frac{\text{Diff}_{FICP}}{|FICP|}}\right\} \tag{7}$$

As shown in Equation (7), the *CD* is based on the RMS values of the number of *FICP*, *IFCP*, *SFCP* and *FSCP* change patterns. Specifically, the *CD* is taken as the maximum RMS value among the four counts. A large *CD* indicates a larger variation in support of the corresponding change pattern between the previous batch and the current batch.

3.5. **Change prediction.** Prediction methods are generally based on stored historical data. Moreover, the prediction accuracy increases as the amount of historical data available increases. However, due to the sheer volume of real-world data streams, most data stream mining models store only the current results, and thus, the prediction accuracy is inevitably degraded. Accordingly, in the present study, the number of historical batches considered in the Change Prediction step is specified by a parameter α, where the value of α represents a compromise between the prediction accuracy and the corresponding storage space requirement.

In the proposed model, the types of the patterns in the next batch are predicted on the basis of the CD metric. Through the user-defined steady threshold γ and change threshold τ $(0 < \gamma < \tau < 1)$, the value of CD can be divided into three regions according to the degree of change, i.e., (1) $CD \geq \tau$, (2) $\tau > CD \geq \gamma$ and (3) $CD < \gamma$. In the model proposed in this study, a different change prediction method is used in each region, as discussed in the following sections. These methods generate a predicted cumulative support for each pattern, and the pattern type of each pattern can be found. The details are as follows.

(a) Direct Prediction Method

If $CD \geq \tau$, a significant difference exists between the patterns in the current batch and those in the previous batch, and thus, the past results are not relevant in predicting the pattern types in the following batch. Accordingly, the past results for the support information are discarded, and the prediction process is performed on the basis of the current support information alone. Specifically, the model predicts the type of each pattern in the following batch using a direct prediction model. That is, the predicted cumulative support of each pattern in the following batch is simply set equal to the current cumulative support of the corresponding pattern (i.e., the number of historical batches used equals one). Other history of all patterns about support information will be deleted from memory.

(b) Linear Regression Prediction Method

If $\tau > CD \geq \gamma$, a regular difference exists between the patterns in the current batch and those in the previous batch. In this case, the cumulative support of each pattern in the following batch is computed using the following linear regression model:

$$y = w_0 + w_1 x \tag{8}$$

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2}, \tag{9}$$

$$w_0 = \bar{y} - w_1 \bar{x} \tag{10}$$

In Equation (8), x represents the Batch ID for predicting, and y is the predicted cumulative support of the patterns in batch x. Moreover, in Equations (9) and (10), x_i is Batch ID i, $\bar{x}$ is the average of all x_i, y_i is the cumulative support of the patterns in batch i, $\bar{y}$ is the average of all y_i, and D is the total number of referred historical batches (i.e., $D = \alpha$).

(c) Average Prediction Method

If $CD < \gamma$, a steady difference exists between the patterns in the current batch and those in the previous batch. In this case, the proposed model uses the average prediction method to predict the cumulative support of each pattern in the following batch. That is, the support information of each pattern in the α historical batches is averaged, and the resulting average cumulative support is taken as the predicted cumulative support of the corresponding pattern in the following batch.

Table 1. Support of pattern ⟨(A)(BC)⟩

Batch ID	1	2	3
Cumulative Support	0.3	0.4	0.5

Example 3.1. *To illustrate how the three different change prediction methods works, we assume that the cumulative support values used to predict the type of pattern ⟨(A)(BC)⟩ in batch 4 are shown in Table 1. Assuming that min_sup =* 55%*,* $\delta = 72\%$ *and* $\alpha = 3$*, three possible scenarios arise:*

(*a*) $\boldsymbol{CD \geq \tau}$: *the direct prediction method is used, the predicted cumulative support of* $\langle(A)(BC)\rangle$ *in batch 4 is set equal to its cumulative support in batch 3. In other words, the predicted cumulative support of pattern* $\langle(A)(BC)\rangle$ *in batch 4 is set to 0.5. And all patterns' support information of Batch IDs 1 and 2 will be deleted from memory. Referring to the pattern types shown in Figure 4, pattern* $\langle(A)(BC)\rangle$ *is predicted to be semi-frequent in batch 4 (i.e.,* $0.72 * 0.55 < 0.5 < 0.55$*).*

(*b*) $\boldsymbol{\tau > CD \geq \gamma}$: *the predicted cumulative support of* $\langle(A)(BC)\rangle$ *in batch 4 is calculated using the linear regression method in Equations (8)-(10). The derivation processes are shown in Equations (11)-(15). In this case, the predicted cumulative support is equal to 0.6, and thus, pattern* $\langle(A)(BC)\rangle$ *is predicted to be frequent in batch 4 (i.e.,* $0.55 < 0.6$*).*

$$\bar{x} = \frac{1+2+3}{3} = 2 \tag{11}$$

$$\bar{y} = \frac{0.3+0.4+0.5}{3} = 0.4 \tag{12}$$

$$w_1 = \frac{(1-2)(0.3-0.4)+(2-2)(0.4-0.4)+(3-2)(0.5-0.4)}{(1-2)^2+(2-2)^2+(3-2)^2} = 0.1 \tag{13}$$

$$w_0 = 0.4 - 0.1 \times 2 = 0.2 \tag{14}$$

$$y = 0.2 + 0.1 \times 4 = 0.6 \tag{15}$$

(*c*) $\boldsymbol{CD < \gamma}$: *the average prediction method is used, the predicted cumulative support of* $\langle(A)(BC)\rangle$ *in batch 4 is set equal to the average cumulative support of the pattern over batches 1 to 3. In other words, the predicted cumulative support is equal to 0.4 (i.e.,* $(0.3+0.4+0.5)/3$*), and thus pattern* $\langle(A)(BC)\rangle$ *is predicted to be semi-frequent in batch 4 (i.e.,* $0.72 * 0.55 < 0.4 < 0.55$*).*

4. **Experimental Design and Results.**

The performance of the proposed model was evaluated by performing a series of simulations. In each simulation, the prediction accuracy was defined as the number of correct pattern predictions divided by the total number of pattern predictions. The prediction accuracy (from the third batch onwards) of the proposed method was compared with that of two linear regression methods based on a limited number of recent batches and an unlimited number of historical batches, respectively. Most experimental mining studies only consider synthetic datasets containing item_sequences. However, in the present study, the experiments also considered synthetic datasets comprising itemset_sequences (mentioned in Subsection 3.1). The simulations were conducted using two itemset_sequences datasets, designated as C200T2.5S10I1.25 and C200T2S8I1.25, respectively, which generated using the IBM Quest Synthetic Data Generator [28]. Note that: C denotes how many thousands of customers exist within the database; T denotes the average transaction length; S denotes the average sequence length; and I denotes the average number of itemsets in the maximal frequent sequential pattern. Note that both datasets comprised a total of 1000 items.

An appropriate setting of parameter τ, was obtained by performing a pretest on C200T2.5S10I1.25 and C200T2S8I1.25 both datasets. In performing each test, the value of τ was set initially to 0.32, and was then reduced by 1% in each iteration. The results showed that a value of $\tau = 0.2$ was appropriate for both datasets. Specifically, the prediction accuracy obtained when using a value of $\tau = 0.2$ was $5 \sim 10\%$ higher than that obtained when using $\tau = 0.32$.

In the first simulation, the prediction accuracy was calculated for two different values of *min_sup*, i.e., 0.75% and 1.0%. The corresponding results are presented in Figure 5. Note that the results relate to the C200T2.5S10I1.25 dataset and are based on the following parameter settings: $\delta = 70\%$, $d = 0.995$, $\tau = 0.2$, $\gamma = 0.05$ and $\alpha = 3$. Note also that each of the 20 batches comprised 1000 sequences. From inspection, the average prediction accuracy for *min_sup* = 1% is found to be 0.93, while that for *min_sup* = 0.75% is equal to 0.88. Since more patterns are generated, the smaller value of *min_sup* results in a lower accuracy. However, the prediction accuracy (i.e., 0.88) is still acceptable.

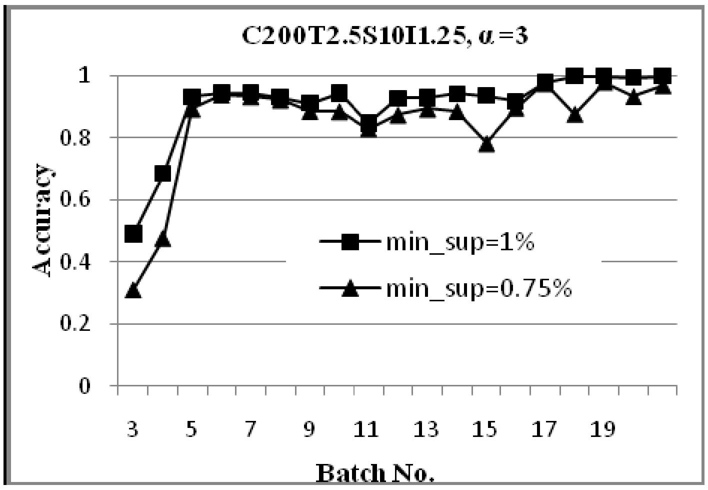

FIGURE 5. Prediction performance of proposed method for different *min_sup* values

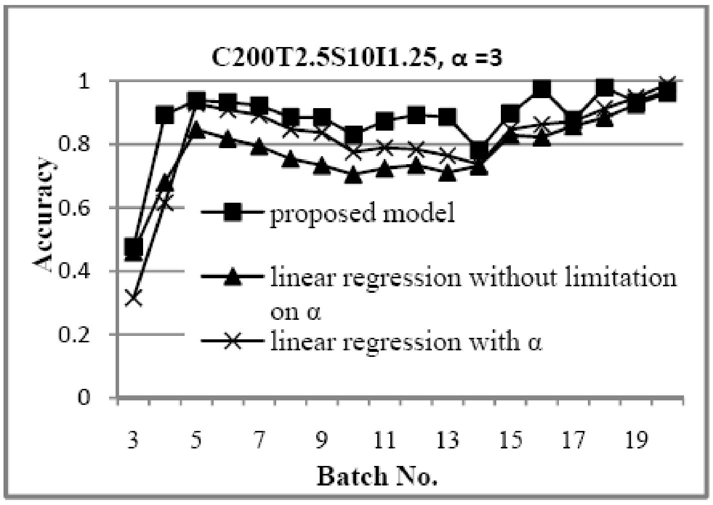

FIGURE 6. Prediction performance of different methods given a batch size of 20

The second set of simulations evaluated the effect of the batch size on the prediction accuracy. The simulations were performed using the C200T2.5S10I1.25 dataset with parameter settings of $\delta = 70\%$, $min\_sup = 0.75\%$, $d = 0.995$, $\tau = 0.2$, $\gamma = 0.05$ and $\alpha = 3$. Two different batch sizes were considered, namely 20 batches with 1000 sequences per batch, and 10 batches with 2000 sequences per batch. Moreover, the prediction performance of the proposed approach was compared with that obtained using the two linear regression methods described above. The results presented in Figure 6, corresponding to a batch size of 20, show that the average prediction accuracies of the proposed model, the linear regression model with no storage limitation, and the linear regression model with a finite storage are 0.88, 0.77 and 0.81, respectively. In other words, the prediction accuracy of the proposed model is around $7 \sim 11\%$ higher than that of the two linear regression models. In Figure 7, corresponding to a batch size of 10, the average prediction accuracies of the three models are 0.88, 0.75 and 0.75, respectively. In other words, the prediction accuracy of the proposed model is 13% higher than that of the linear regression models. Overall, the two figures show that the proposed method outperforms the linear regression models irrespective of the batch size. The superior prediction performance of the proposed method is attributed to the use of different prediction methods in accordance with the degree of change of the sequential patterns (i.e., the value of CD).

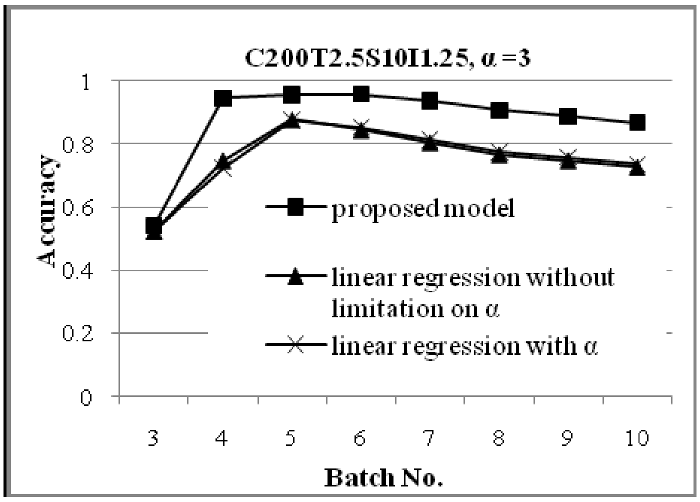

FIGURE 7. Prediction performance of different methods given a batch size of 10

The prediction performance of the three methods was also examined when applied to the C200T2S8I1.25 dataset. The content of the C200T2S8I1.25 dataset with smaller "T" and "S" is simpler than that of the C200T2.5S10I1.25 dataset, and thus, all three methods were found to achieve a prediction accuracy of more than 96% for values of min_sup in the range $0.5 \sim 1\%$. Figure 8 shows the prediction performance of the three methods for $min\_sup = 0.25\%$. The proposed model achieves an average prediction accuracy of 0.88, while both linear regression models achieve an average prediction accuracy of 0.75. These results are identical to those obtained for the C200T2.5S10I1.25 dataset (see Figure 7). Thus, the general applicability of the proposed method is confirmed.

A third set of simulations was performed using the C200T2.5S10I1.25 dataset in order to investigate the effect of α (i.e., the number of batches considered in the prediction step) on the prediction performance. In every case, the model settings were specified as follows: $\delta = 70\%$, $d = 0.995$, $min\_sup = 0.75\%$, $\tau = 0.2$, $\gamma = 0.05$ and batch size = 10. The results presented in Figure 9, corresponding to $\alpha = 2$, show that the average prediction accuracies of the proposed model, the linear regression model with no storage limitation and the

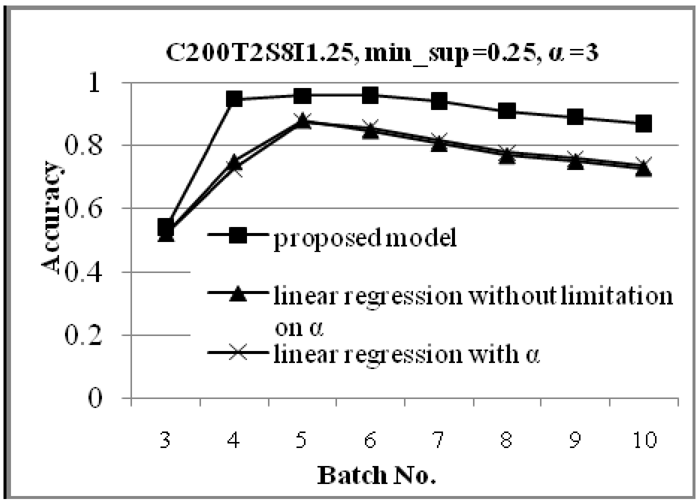

FIGURE 8. Prediction performance of different methods for C200T2S8I1.25 dataset

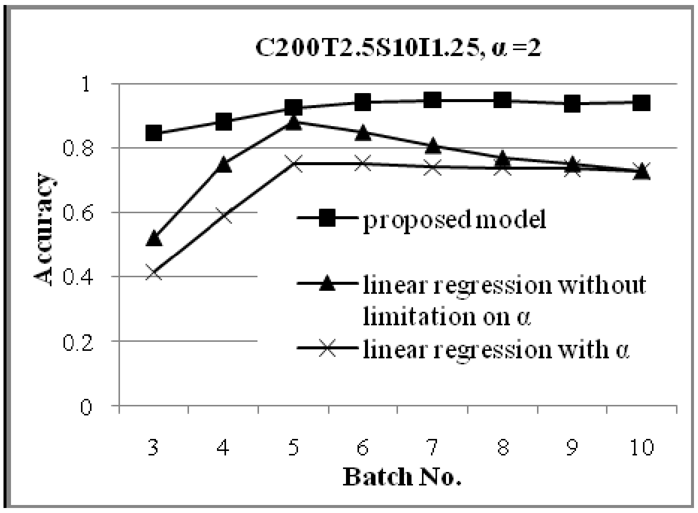

FIGURE 9. Prediction performance of different methods given $\alpha = 2$

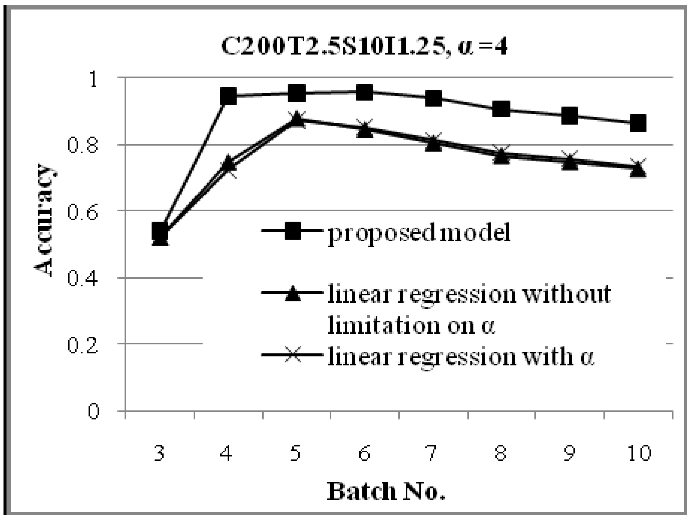

FIGURE 10. Prediction performance of different methods given $\alpha = 4$

linear regression model with a finite storage are equal to 0.92, 0.75 and 0.68, respectively. Since just two batches are considered in the prediction process, the linear regression model with a finite storage inevitably yields low prediction accuracy. By contrast, the proposed method yields high average prediction accuracy (0.92) since the proposed method using different prediction methods in accordance with the degree of change of the sequential patterns.

Figure 10 shows that when the number of batches considered in the prediction process is increased to $\alpha = 4$, the average prediction accuracies of the three methods are 0.88, 0.75 and 0.75, respectively. Theoretically, the value of α increases, i.e., a greater volume of historical information is used in the prediction process, the prediction performance of the proposed method converges toward that of the linear regression method with no storage limitation.

Finally, the three methods were applied to the FoodMart2000 database (extracted from SQL Server 2000). The database consists of two datasets, namely Sales_Fact_1997 and Sales_Fact_1998. Sales_Fact_1997 contains 20,522 transactions collected from 5581 customers in 1997, while Sales_Fact_1998 contains 34,015 transactions collected from 7824 customers in 1998. Overall, the transactions involve a total of 1559 product items. Figure 11 shows the prediction performance of the three methods. Note that the results presented for the proposed model relate to parameter settings of $\delta = 70\%$, $d = 0.995$, $\tau = 0.2$, $\gamma = 0.05$ and $\alpha = 3$. Moreover, the datasets were divided into 24 batches by per month and *min_sup* was assigned a value of 0.5%. Thus, each of the 24 batches comprised a different number of sequences. (The average number of sequences per batch per month was found to be 2826.5.) The results presented in Figure 11 show that the average prediction accuracies of the three methods are 0.96, 0.94 and 0.95, respectively. The simulations were repeated using a *min_sup* value of 1.0%; resulting in an average number of sequences per batch per quarter of 16666.75. The corresponding results are presented in Figure 12. From inspection, the prediction accuracies of the three methods are found to be 0.90, 0.85 and 0.86, respectively.

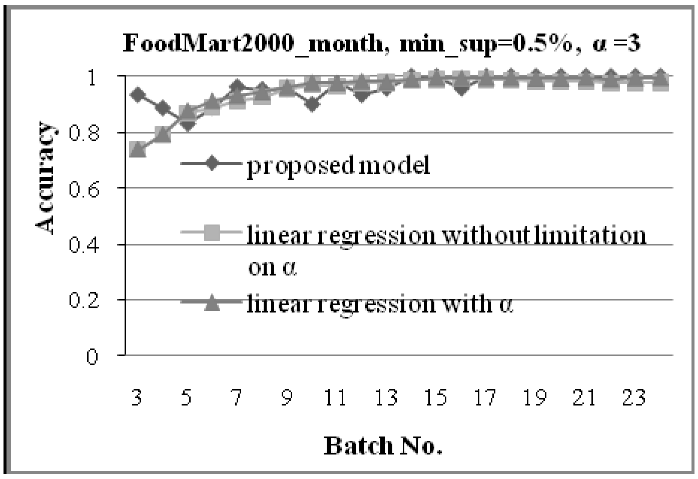

Figure 11. Prediction performance of different methods for FoodMart2000_month dataset given *min_sup* = 0.5%

A detailed observation of the FoodMart2000 database revealed that the data within the two datasets have a near uniform distribution. In other words, the database contains very few (if any) *frequent* patterns, and each element consists of only one item (i.e., item_sequence) in all *frequent* patterns. Thus, the proposed model can achieve high

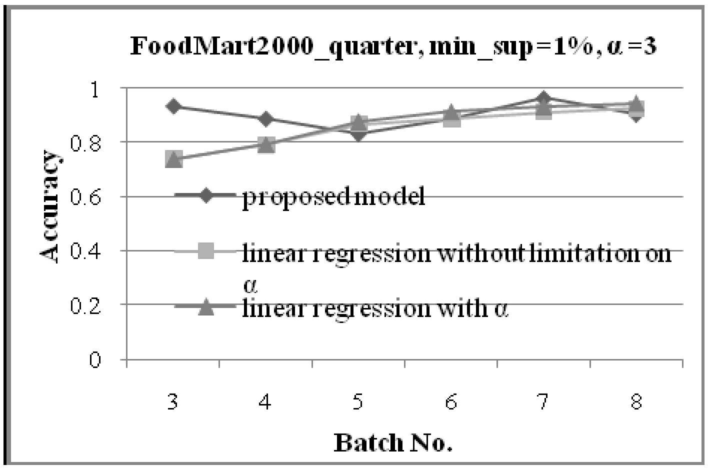

FIGURE 12. Prediction performance of different methods for FoodMart2000_quarter dataset given $min\_sup = 1.0\%$

TABLE 2. Change pattern examples from batch 2 to batch 4 of FoodMart2000_month database

Batch / Pattern	FoodMart_1997_2	FoodMart_1997_3	FoodMart_1997_4
Washington Cream Soda	*FSCP*	*SICP*	*N/A*
Tell Tale Corn on the Cob	*ISCP*	*N/A*	*N/A*
Excellent Berry Juice	*IFCP*	*N/A*	*FSCP*
High Top Lemons	*N/A*	*N/A*	*SICP*
Denny C-Size Batteries	*SFCP*	*FICP*	*N/A*
Golden Lime Popsicles	*SICP*	*N/A*	*N/A*

prediction accuracy, especially for the small average number of sequences with per month (0.96).

Table 2 presents typical examples of the change patterns within batches 2-4 of the FoodMart2000 database in 1997. Note that the notation *N/A* indicates that the batch contains no change pattern for the corresponding product. Note also that most of the mined sequential patterns include only one item (i.e., equates in product here) since the items in the FoodMart2000 database are sparsely distributed. As shown, the pattern "Washington Cream Soda" exhibits an *FSCP* change pattern in February 1997, an *SICP* pattern in March 1997 and no change pattern in April 1997. The proposed model allows the pattern type of each product to be predicted in advance, and therefore enables decision-makers to formulate a timely and appropriate response to emerging trends.

Figure 13 reveals the average historical batches used (denoted as p) with different methods for all datasets given $\alpha = 3$. And the average number of patterns in each batch (denoted as q) of the four datasets are found to be 2463.88, 7379.50, 27.29 and 1344.83, respectively. Assume that the memory size for storing the support information of one pattern is 4 bytes. Therefore, the average memory size for keeping history data is $p * q * 4$ bytes. The average memory size with different methods for all datasets given $\alpha = 3$ and the corresponding average accuracy from previous simulations are shown in Table 3. Obviously, the proposed model can use little memory for storing history data to achieve higher accuracy than the other two compared methods. Moreover, the more the batches, the more memory been saved in the proposed model.

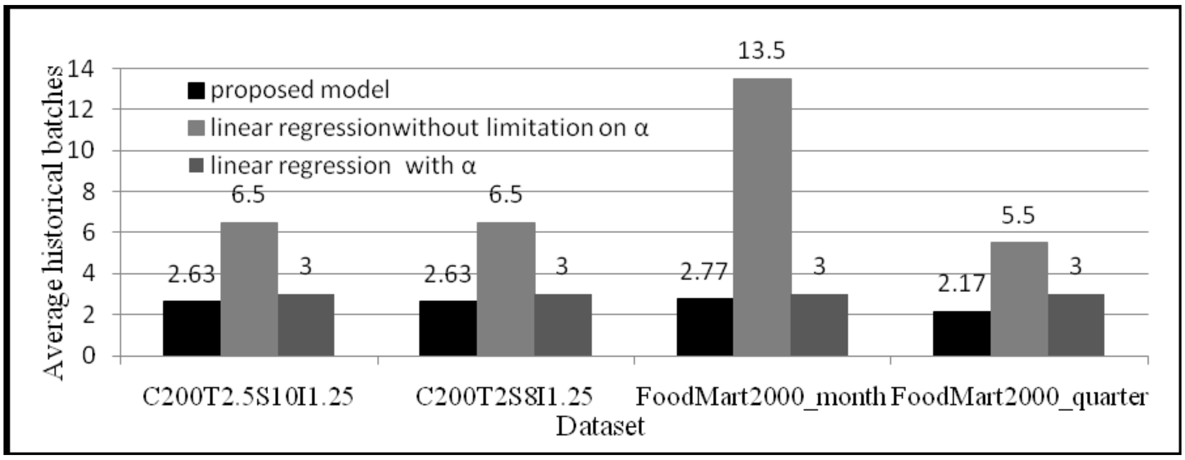

FIGURE 13. Average historical batches used for different methods with given $\alpha = 3$

In summary, the model proposed in this study provides a robust means of predicting the sequential pattern types within a data stream. The experimental results show that the proposed method consistently achieves a prediction accuracy of more than 88% and outperforms conventional linear regression-based prediction models. The average prediction accuracy of the linear regression model based on a specified number of historical batches is close to, but less than, that of the proposed model. However, even though the regression model achieves a good performance when the degree of change exceeds the change threshold τ, it requires a greater number of historical batches than the proposed model. The results have shown that the linear regression model based on all the historical batches achieves lower prediction accuracy than the proposed model or the regression model based on a limited number of recent batches. This is because the changes may lead to an overdue mining model. Overall, the experimental results demonstrate the importance of utilizing the change information within a data stream to achieve a real-time and reliable prediction capability.

TABLE 3. The average memory size and the average accuracy (in parentheses) for $\alpha = 3$

Dataset / Method	C200T2.5S10I1.25	C200T2S8I1.25	FoodMart2000_month	FoodMart2000_quarter
proposed model	25920 bytes (0.88)	77632 bytes (0.88)	302 bytes (0.96)	11673 bytes (0.90)
linear regression without limitation on α	59133 bytes (0.75)	191867 bytes (0.75)	1474 bytes (0.94)	29587 bytes (0.85)
linear regression with α	29567 bytes (0.75)	88554 bytes (0.75)	328 bytes (0.95)	16138 bytes (0.86)

5. **Conclusions.** This study has proposed a model for predicting changes in the sequential patterns within a dataset in accordance with their degree of change. The experimental results have shown that the proposed model achieves higher prediction accuracy than linear regression methods based on either a limited number of recent batches or an infinite number of historical batches. Thus, the proposed method provides an effective means of mining the sequential patterns within real-world applications such as WSNs or web logs.

The scalability problem of data stream mining for processing an enormous amount of data on a single CPU is still an issue. In the future study, the prediction speed of the proposed method will be further enhanced by means of parallel or cloud computing techniques. Moreover, the current study inspects literal items data only, complex sequences such as hybrid sequential patterns [29], or numeric data sequences like stock or sales volume will be considered.

REFERENCES

[1] Q. Zhang, R. Zhang and B. Wang, DNA sequence sets design by particle swarm optimization algorithm, *International Journal of Innovative Computing, Information and Control*, vol.5, no.8, pp.2249-2255, 2009.

[2] Y. Wang, W. Zhang, C. Zhang, H. Ling and F. Zhao, Establishing website navigation support systems by mining sequential patterns, *ICIC Express Letters*, vol.4, no.2, pp.395-400, 2010.

[3] G. T. Raju, P. S. Satyanarayana and L. M. Patnaik, Knowledge discovery from web usage data: Extraction and applications of sequential and clustering patterns – A survey, *International Journal of Innovative Computing, Information and Control*, vol.4, no.2, pp.381-389, 2008.

[4] Z. Sui, Y. Liu and Y. Hu, Extracting hyponymy relation between chinese terms based on term types' commonality and sequential patterns, *ICIC Express Letters*, vol.3, no.4(B), pp.1233-1238, 2009.

[5] C.-Y. Tsai and P.-H. Lo, A sequential pattern based route suggestion system, *International Journal of Innovative Computing, Information and Control*, vol.6, no.10, pp.4389-4408, 2010.

[6] C. Raïssi, P. Poncelet and M. Teisseire, SPEED: Mining maximal sequential patterns over data streams, *Proc. of the 3rd International Conference on Intelligent System*, pp.546-552, 2006.

[7] C.-C. Ho, H.-F. Li, F.-F. Kuo and S.-Y. Lee. Incremental mining of sequential patterns over a stream sliding window, *Proc. of IEEE International Workshop on Mining Evolving and Streaming Data*, pp.677-681, 2006.

[8] H.-F. Li, S.-Y. Lee and M.-K. Shan, DSM-PLW: Single-pass mining of path traversal patterns over streaming web click-sequences, *Computer Networks: Special Issue on Web Dynamics*, vol.50, no.10, pp.1474-1487, 2006.

[9] G. Chen, X. Wu and X. Zhu, Mining sequential patterns across data streams, *Computer Science Technical Report CS-05-04*, University of Vermont, 2005.

[10] G. Chen, X. Wu and X. Zhu, Sequential pattern mining in multiple streams, *Proc. of IEEE International Conference on Data Mining*, pp.585-588, 2005.

[11] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal and M.-C. Hsu, Mining sequential patterns by pattern-growth: The PrefixSpan approach, *IEEE Trans. Knowl. Data Eng.*, vol.16, no.11, pp.1424-1440, 2004.

[12] B. Liu, W. Hsu, H.-S. Han and Y. Xia, Mining changes for real-life applications, *Proc. of the 2nd International Conference on Data Warehousing and Knowledge Discovery*, *LNCS*, vol.1874, pp.337-346, 2000.

[13] M.-C. Chen, A.-L. Chiu and H.-H. Chang, Mining changes in customer behavior in retail marketing, *Expert Systems with Applications*, vol.28, pp.773-781, 2005.

[14] C.-Y. Tsai and Y.-C. Shieh, A change detection method for sequential patterns, *Decision Support Systems*, vol.46, no.2, pp.501-511, 2009.

[15] Y.-L. Chen and Y.-H. Hu, Constraint-based sequential pattern mining: The consideration of recency and compactness, *Decision Support Systems*, vol.42, pp.1203-1215, 2006.

[16] H. S. Kim, J. J. Shin, Y. I. Jang, G. B. Kim and H. Y. Bae, RSP-DS: Real time sequential pattern analysis over data streams, *APWeb/WAIM 2007 Ws, LNCS*, vol.4537, pp.99-110, 2007.

[17] L. F. Mendes, B. Ding and J. Han, Stream sequential pattern mining with precise error bounds, *Proc. of IEEE International Conference on Data Mining*, pp.941-946, 2008.

[18] H. Li and H. Chen, GraSeq: A novel approximate mining approach of sequential patterns over data stream, *The 3rd International Conference on Advanced Data Mining and Applications, LNAI*, vol.4632, pp.401-411, 2007.

[19] J. H. Chang and W. S. Lee, Finding recent frequent itemsets adaptively over online data streams, *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.487-492, 2003.

[20] J. H. Chang and W. S. Lee, estWin: Online data stream mining of recent frequent itemsets by sliding window method, *Journal of Information Science*, vol.31, no.2, pp.76-90, 2005.

[21] J. H. Chang and W. S. Lee, Efficient mining method for retrieving sequential patterns over online data streams, *Journal of Information Science*, vol.31, no.5, pp.31-36, 2005.

[22] R. Agarwal and R. Srikant, Fast algorithms for mining association rules, *Proc. of the 20th International Conference on Very Large Data Bases*, pp.487-499, 1994.

[23] H.-F. Li, S.-Y. Lee and M.-K. Shan, Online mining changes of items over continuous append-only and dynamic data streams, *Journal of Universal Computer Science*, vol.11, no.8, pp.1411-1425, 2005.

[24] H.-F. Li, M.-K. Shan and S.-Y. Lee, Detecting changes in user-centered music query streams, *Proc. of IEEE International Conference on Multimedia and Expo*, pp.1977-1980, 2006.

[25] D.-R. Liu, M.-J. Shih, C.-J. Liau and C.-H. Lai, Mining the change of event trends for decision support in environmental scanning, *Expert Systems with Applications*, vol.36, no.2, pp.972-984, 2009.

[26] R. Agrawal and R. Srikant, Mining sequential patterns, *Proc. of the 11th International Conference on Data Engineering*, pp.3-14, 1995.

[27] R. Agrawal and R. Srikant, Mining sequential patterns: Generalizations and performance improvements, *Proc. of the 5th International Conference on Extending Database Technology*, pp.3-17, 1996.

[28] IBM Almaden Research Center, *Quest Synthetic Data Generation Code*, http://www.almaden.ibm.com/cs/quest/syndata.html.

[29] K.-F. Jea, K.-C. Lin and I-E. Liao, Mining hybrid sequential patterns by hierarchical mining technique, *International Journal of Innovative Computing, Information and Control*, vol.5, no.8, pp.2351-2367, 2009.