Available online at www.sciencedirect.com

ScienceDirect

Data & Knowledge Engineering 60 (2007) 30–50

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

Benchmarking the effectiveness of sequential pattern mining methods

Hye-Chung Kum [a,*], Joong Hyuk Chang [b], Wei Wang [a]

[a] *Department of Computer Science, University of North Carolina at Chapel Hill, NC 27599, USA*
[b] *Department of Computer Science, Yonsei University, Seoul 120-749, Korea*

Available online 3 March 2006

Abstract

Recently, there is an increasing interest in new intelligent mining methods to find more meaningful and compact results. In intelligent data mining research, accessing the quality and usefulness of the results from different mining methods is essential. However, there is no general benchmarking criteria to evaluate whether these new methods are indeed more effective compared to the traditional methods. Here we propose a novel benchmarking criteria that can systematically evaluate the effectiveness of any sequential pattern mining method under a variety of situations. The benchmark evaluates how well a mining method finds known common patterns in synthetic data. Such an evaluation provides a comprehensive understanding of the resulting patterns generated from any mining method empirically. In this paper, the criteria are applied to conduct a detailed comparison study of the support-based sequential pattern model with an approximate pattern model based on sequence alignment. The study suggests that the alignment model will give a good summary of the sequential data in the form of a set of common patterns in the data. In contrast, the support model generates massive amounts of frequent patterns with much redundancy. This suggests that the results of the support model require more post processing before it can be of actual use in real applications.

Keywords: Benchmarking effectiveness; Evaluating quality of results; Sequential pattern mining

1. Introduction

Much research has been devoted to efficient mining methods that can process huge amounts of data. Yet, the mining methods have been ineffective in many real world applications with limited use because often times the results are not of direct use to a human expert. Recently, there is an increasing interest in new intelligent mining methods to find more meaningful and compact results. However, there is no general benchmarking criteria to evaluate whether these new methods are indeed more effective compared to the traditional methods. Furthermore, benchmarking the quality of the mined results is required to compare the effectiveness of new

[*] Corresponding author.
E-mail addresses: kum@email.unc.edu (H.-C. Kum), jhchang@amadeus.yonsei.ac.kr (J.H. Chang), weiwang@email.unc.edu (W. Wang).

0169-023X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2006.01.004

intelligent mining methods among themselves. Assessing the quality and usefulness of the results from different mining methods is essential in intelligent data mining research.

We are particularly interested in the problem of mining sequential patterns. In sequential pattern mining, the goal is to detect patterns in a database comprised of sequences of sets. Conventionally, the sets are called *itemsets*. For example, supermarkets often collect customer purchase records in sequence databases in which a sequential pattern would indicate a customer's buying habit. In such a database, each purchase would be represented as a set of items purchased, and a customer sequence would be a sequence of such itemsets.

One common approach in sequential pattern mining is provided by the *support model* [2]: the task is to find the complete set of frequent subsequences in a set of sequences. Much research has been done to efficiently find the patterns defined by the support model, but to the best of our knowledge, no research has examined in detail the usefulness of the patterns actually generated.

An alternative pattern definition in sequential pattern mining is the *multiple alignment model* [6]. Its goal is to organize and summarize sequences of sets to uncover the underlying consensus patterns in the data. An approximate algorithm, ApproxMAP, uses clustering as a preprocessing step to group similar sequences, and then mines the underlying consensus patterns in each cluster directly through multiple alignment.

Our purpose here is to introduce a general evaluation method to assess the quality of the mined results from the two sequential pattern mining models, and then to compare the results. In this paper, employing our new evaluation method we examined closely the results from both the support model and the multiple alignment model. The evaluation method clearly describes the different kinds of patterns generated from each model. Understanding the characteristics of the results suggests when each model should be used to mine sequential patterns.

1.1. Our approach

In general, the most suitable evaluation criteria for a given problem tends to be domain-specific. That is, the evaluation criteria depends on what type of sequential patterns are of most interest in the domain application. Thus, the most general benchmark should provide a set of target patterns and evaluation criteria that is appropriate for each target pattern. Then, users can select the appropriate target patterns and evaluation criteria required for the domain in order to determine the most appropriate mining method for the application.

As a first step towards building such a general benchmark, we propose an evaluation method that can quantitatively assess how well the models can find known common patterns in the data. Integral to the evaluation method is a synthetic dataset with known embedded patterns against which we can compare the mined results. For this purpose we have used the well-known IBM synthetic data generator [2] built by the authors of the support model, Agrawal and Srikant. We extend the well-known IBM data generator to generate a variety of situations with varying randomness and noise levels. Then, we develop a set of evaluation criteria to use in conjunction with the extended IBM data generator to measure the accuracy of the results. By mapping the mined patterns back to the known base patterns that generated the data, we are able to measure how well the base patterns are recovered and how much confounding information (in the form of spurious patterns, redundant patterns, or extraneous items) is in the results.

In summary, our evaluation method is a matrix of four experiments—(1) random data, (2) patterned data, and patterned data with (3) varying degree of noise, and (4) varying number of outliers—assessed on five criteria: (1) recoverability, (2) precision, (3) the total number of result patterns returned, (4) the number of spurious patterns, and (5) the number of redundant patterns. *Recoverability*, defined in Section 4, provides a good estimation of how well the underlying trends in the data are detected. *Precision*, adopted from ROC analysis [8], is a good measure of how many incorrect items are mixed in with the correct items in the result patterns. Both recoverability and precision are measured at the item level. On the other hand, the numbers of spurious and redundant patterns along with the total number of patterns returned give an overview of the result at the sequence level. In short, a good model would produce (1) high recoverability and precision, with (2) small number of spurious and redundant patterns, and (3) a manageable number of result patterns.

Our workshop paper [7] reports our preliminary work. In this paper, we extend the report by

- presenting the details of the synthetic data generation process,
- revising and extending the evaluation criteria,

- illustrating the evaluation method via an example,
- supplementing the comparison experiment,
- and briefly exploring some promising directions for future research.

The remainder of this paper is organized as follows. Section 2 summarizes the related works in sequential pattern mining. Sections 3 and 4 describe the synthetic data and the criteria used in the evaluation method respectively. Section 5 is a detailed example illustrating how the evaluation is done. Section 6 presents a comparison study of the sequential pattern mining methods employing the evaluation method. Finally, Section 7 concludes with a discussion of directions for future research.

2. Background and related work

Mining sequential patterns has become an important data mining task with broad applications in business analysis, career analysis, policy analysis, and security. Many papers on sequential pattern mining focus on specific algorithms and evaluating their efficiency.

2.1. IBM synthetic data generator

The IBM synthetic data generator was first presented in [1] to evaluate the efficiency of mining association rules. It was further expanded to evaluate the efficiency of sequential pattern mining methods in [2]. Since then, the IBM synthetic data has been used extensively as a performance benchmark in association rule mining and sequential pattern mining.

The computational efficiency of most mining methods are usually well documented, however, the efficacy of various definitions in producing useful results has received less attention. Although the well used IBM synthetic data generator reports the base patterns used to generate the data, to the best of our knowledge, no previous study has measured how well the various methods recover the known base patterns in the synthetic data. In this paper, we propose to extend the performance benchmark to evaluate the quality of the mined results. By mapping the mined patterns back to the base patterns that generated the data, we are able to measure how well the mining models find the real underlying patterns. We can also determine whether or not a model generates any spurious patterns or confounding information.

2.2. Support model

Sequential pattern mining is commonly defined as finding the complete set of frequent subsequences in a set of sequences. The conventional support model finds all subsequences that meet a user specified threshold, min_sup [2]. We omit the details of the support model since it is well known. GSP [11], PrefixSpan [9], SPADE [14], and SPAM [3] are some well-known algorithms to efficiently find such patterns. The efficiency of these methods are compared in detail using the IBM data generator in [9]. However, to the best of our knowledge, our workshop paper [7] was the first to examine in detail what patterns are actually generated from such a model. In this paper, we supplement our experiments in [7] to better understand whether the support model in fact generates interesting and understandable patterns.

Some attempts have been made to improve the results from the support model. In order to reduce redundancy, [12] extends the support model to find only closed sequential patterns. Ref. [13] extends the support criteria to account for partial matches to find approximate sequential patterns.

Two papers viewed the output from the support model as intermediate results. Ref. [10] presents a system for managing interesting sequential rules using the output from the support model. Ref. [4] uses the information on frequent subsequences to cluster the sequential data.

2.3. Multiple alignment model

Recently, [6] presented an entirely different model for sequential pattern mining based on sequence alignment. Extending the rich body of literature on string analysis in computer science and computational biology,

Table 1
Multiple alignment ($\theta = 75\%$)

$seq_1 = \langle$(BC)(DE)$\rangle$	$\langle$()	()	(BC)	(DE)$\rangle$	
$seq_2 = \langle$(A)(BCX)(D)$\rangle$	$\langle$(A)	()	(BCX)	(D)$\rangle$	
$seq_3 = \langle$(AE)(B)(BC)(D)$\rangle$	$\langle$(AE)	(B)	(BC)	(D)$\rangle$	
$seq_4 = \langle$(A)(B)(DE)$\rangle$	$\langle$(A)	()	(B)	(DE)$\rangle$	
Weighted sequence wseq	(A:3, E:1):3	(B:1):1	(B:4, C:3, X:1):4	(D:4, E:2):4	4
Consensus pattern ($w \geqslant 3$)	$\langle$(A)		(BC)	(D)$\rangle$	

Kum generalized string multiple alignment to find patterns in ordered lists of sets. The power of the multiple alignment model hinges on the following insight: the probability that any two long data sequences are the same purely by chance is very low. Thus, if a number of long sequences can be aligned with respect to particular items that occur frequently in certain positions, we will have implicitly found sequential patterns that are statistically significant.

The exact solution to multiple alignment pattern mining is too expensive to be practical. Consequently the results of the alignment model are method dependent. An efficient approximation algorithm, ApproxMAP (for APPROXimate Multiple Alignment Pattern mining), was introduced in [6]. In this paper, we use the results of ApproxMAP to represent the alignment model.

ApproxMAP has three steps. First, k nearest neighbor clustering is used to partition the database. Second, for each partition, the optimal multiple alignment is approximated by the following greedy approach: in each partition, two sequences are aligned first, and then a sequence is added incrementally to the current alignment of $p-1$ sequences until all sequences have been aligned. At each step, the goal is to find the best alignment of the added sequence to the existing alignment of $p-1$ sequences. A novel structure, *weighted sequence*, is used to summarize the alignment information in each cluster. In short, a weighted sequence is a sequence of itemsets with a weight associated with each item. The item weight represents the strength of the item where *strength* is defined as the percentage of sequences in the alignment that have the item present in the aligned position. Clearly, larger strength value indicates that more sequences share the item in the same aligned position. Third, based on the user-defined threshold, θ, for item strength the weighted sequence of each partition is used to generate the consensus pattern for the partition.

Table 1 illustrates the alignment model. Given a set of sequences, ApproxMAP will first cluster the sequences into similar sequences. Then each group will be aligned, via dynamic programming, to generate a consensus pattern. For example, given a group of similar sequences, ApproxMAP will align them as in Table 1. The alignment information is summarized in the weighted sequence wseq. There are three elements in the weighted sequence. First, there is a weight associated with the full sequence. The 4 in the last column is associated with the sequence and indicates that there are four sequences in the alignment. Second, there are weights associated with each item. 3 and 1 associated with the items A and E, respectively, in the first itemset (A:3, E:1) indicate that there are three A's and one E in the first column of the alignment. Third, there is a weight associated with each itemset. The weight 3 associated with the first itemset (A:3, E:1):3 indicates that one sequence out of four sequences ($4-3=1$) has a null itemset aligned to this position. seq_1 has a null itemset in the first position of the alignment. The details of ApproxMAP can be found in [6].

3. Synthetic data generation

Integral to evaluating the effectiveness of mining methods is synthetic data with known embedded patterns against which we can compare the mined results. For this purpose we have used the well-known IBM synthetic data generator [1,2]. In this section, we detail the data generation process after a through review of the public code. We further use an example to illustrate the data generation process. In addition, in Sections 3.2–3.4 we extend the IBM data generator to generate a variety of situations varying randomness and noise level.

3.1. IBM synthetic data generator: Patterned data

As discussed in [1,2] the IBM synthetic data generator models a customer transaction database. The general idea is that people buy sequences of sets of items. Each such sequence could be similar to a frequent buying pattern. For instance, a common buying pattern might be sheets and pillows followed by a comforter. Some people will buy only some items in the potentially frequent buying pattern. In the previous example, some will only buy sheets followed by comforters. Other customer sequences will contain more than one such potentially frequent buying pattern. For example, some customers might order a dress and jacket along with the sheets. This customer might follow this initial order with a comforter and shoes. In this case, the dress and jacket followed by shoes might form another frequent buying pattern. In the original paper, these frequent patterns in the synthetic data were called *maximal potentially large sequences*. In this paper, we call them *base patterns*. These are the patterns used to generate the sequence database. In short, a customer sequence is generated by deleting items from base patterns, and then combining the different perturbed base patterns. Now, let us examine this process technically.

The input parameters for the IBM data generator is given in Table 2. We have changed the notations from the original paper for clarity. The first four parameters determine the characteristics of the sequence database, and the last four parameters determine the type of base patterns used to generate the database. Given these parameters, the IBM data generator produces a patterned database and reports the base patterns used to generate it.

The data is generated in two phases. First, it generates N_{pat} potentially frequent sequential patterns, called *base patterns*, according to user parameters L_{pat} and I_{pat}. Secondly, each sequence in the database is built by combining these base patterns until the size specified by user parameters L_{seq} and I_{seq} are met. There are N_{seq} such sequences in the database. Along with each base pattern, the data generator reports the expected frequency, $E(F_B)$, and the expected length (total number of items), $E(L_B)$, in the database for each base pattern. The $E(F_B)$ is given as a percentage of the size of the database and the $E(L_B)$ is given as a percentage of the number of items in the base pattern.

There are two steps involved in building the base patterns. First, the set of potentially frequent itemsets, Λ, is built by randomly selecting items from the distinct set of items in I. Under the assumption that some items occur often while others occur rarely, the probability of an item occurring is exponentially distributed. The size of each itemset is randomly determined using a Poisson distribution with mean I_{pat}. The assumption is that the transaction sizes are usually clustered around a mean with few transactions that are larger. The details of building the potentially frequent itemsets can be found in [1]. The number of distinct items and the number of potentially frequent itemsets are determined by user set parameters $\|I\|$ and $\|\Lambda\|$, respectively.

Second, the base patterns are then built by selecting, corrupting, and concatenating itemsets selected from the set of potentially frequent itemsets in Λ. The selection and corruption is based on the probability of select, $P(\text{select})$, and probability of corrupt, $P(\text{corrupt})$, randomly assign to each potentially frequent itemset. The selection probability is exponentially distributed then normalized to sum to 1. The corruption probability is normally distributed. Corrupting means randomly deleting items from the selected potentially frequent item-

Table 2
Parameters for the original IBM synthetic data generator (patterned data)

Type	Notation	Original notation	Meaning
Sequence database parameters	$\|I\|$	N	# of items
	N_{seq}	$\|D\|$	# of data sequences
	L_{seq}	$\|C\|$	Avg. # of itemsets per data sequence
	I_{seq}	$\|T\|$	Avg. # of items per itemset in the database
Base pattern parameters	$\|\Lambda\|$	N_I	# of potentially frequent itemsets
	N_{pat}	N_S	# of base patterns (potentially freq. seq. patterns)
	L_{pat}	$\|S\|$	Avg. # of itemsets per base pattern
	I_{pat}	$\|I\|$	Avg. # of items per itemset in the base patterns

set. N_{pat} determines how many base patterns to construct, and L_{pat} determines the average number of itemsets in the base patterns. More precisely, the number of itemsets in a base pattern is randomly assigned from a Poisson distribution with mean L_{pat}. Again, the assumption is that the sequence lengths are usually clustered around a mean, with few long sequences [2]. Base patterns built in this manner then become the potentially frequent sequential patterns.

The database is built in a similar manner by selecting, corrupting, and combining the base patterns. As with potentially frequent itemsets, each base pattern is also assigned a separate $P(\text{select})$ and $P(\text{corrupt})$. The $P(\text{select})$ is exponentially distributed then normalized to sum to 1 and $P(\text{corrupt})$ is normally distributed. The $P(\text{select})$ is the likelihood a base pattern will appear in the database. Thus, it is equal to the expected frequency of a base pattern, $E(F_{\mathrm{B}})$, in the database. The $P(\text{corrupt})$ is the likelihood of a selected base pattern to be corrupted before it is used to construct a database sequence. Corrupting base patterns is defined as randomly deleting items from the selected base pattern. Hence, $1 - P(\text{corrupt})$ is roughly the expected length (total number of items), $E(L_{\mathrm{B}})$, of the base pattern in a sequence in the database.

$$\begin{aligned} &E(F_{\mathrm{B}}) = P(\text{select}) \\ &E(L_{\mathrm{B}}) \simeq 1 - P(\text{corrupt}) \end{aligned} \tag{1}$$

Each sequence is built by combining enough base patterns until the size required, determined by L_{seq} and I_{seq}, is met. Hence, many sequences are generated using more than one base pattern. Base patterns are combined by interleaving them so that the order of the itemsets are maintained. Table 3 demonstrates how three base patterns are combined to build a database sequence. The items that are crossed out were deleted in the corruption process.

In essence, noise is introduced into each data sequence in the form of tiny bits of another base pattern. Hence, sequential pattern mining is difficult because the data has confounding noise rather than random noise. In Section 3.3, we discuss how to add controlled level of random noise in addition to the confounding noise in the patterned data in order to test for the effects of random noise.

Similarly, outliers may exist in the data in the form of very weak base patterns. The expected frequency of the base patterns has exponential distribution. Thus, the weakest base patterns can have expected frequency be so small that the base pattern occurs in only a handful of the database sequences. These are in practice outliers that occur rarely in the database. For example, when there are 100 base patterns, 11 base patterns have expected frequency less than 0.1%, of them 1 base pattern has expected frequency less than 0.01%. Thus, even when $N_{\mathrm{seq}} = 10{,}000$, the weakest base pattern would occur in less than one sequence ($10{,}000 * 0.01\% = 1$ seq). In Section 3.4, we discuss how to add controlled level of strictly random sequences in addition to outliers in the form of very weak base patterns in the patterned data to test for effects of outliers.

3.1.1. Example

To better understand the properties of the synthetic data generated, let us look closely at a particular IBM synthetic database. A common configuration of the synthetic database is given in Table 4. The configuration can be understood as follows:

Table 3
A database sequence built from three base patterns

		(A, J)	**(B)**		**(A)**	**(E, H)**	**(C)**	**(L)**	
Base patterns	**(D)**							**(F)**	~~**(M)**~~
	(G)	~~**(L)**~~		**(K)**	**(F, I)**			**(D)**	
DB sequence	**(D, G)**	**(A, J)**	**(B)**	**(K)**	**(A, F, I)**	**(E, H)**	**(C)**	**(D, F, L)**	

Table 4
A common configuration of the IBM synthetic data

Sequence database	$\|\|I\|\| = 1000$	$N_{\mathrm{seq}} = 10{,}000$	$L_{\mathrm{seq}} = 10$	$I_{\mathrm{seq}} = 2.5$
Base pattern	$\|\|\Lambda\|\| = 5000$	$N_{\mathrm{pat}} = 100$	$L_{\mathrm{pat}} = 7$	$I_{\mathrm{pat}} = 2$

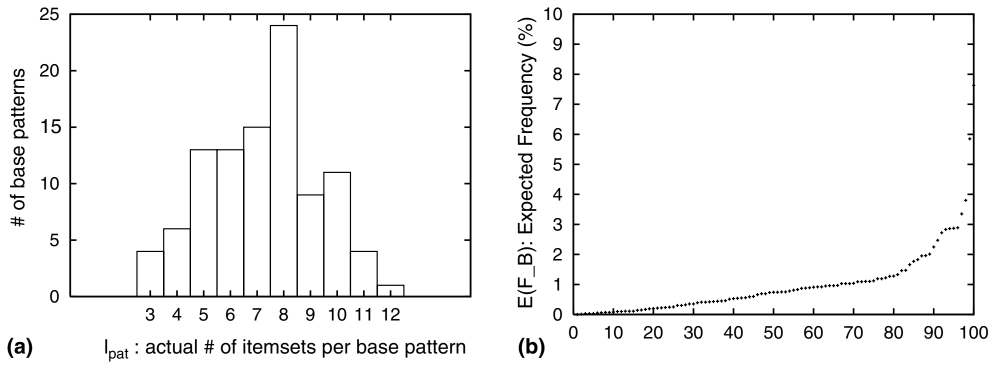


Fig. 1. Distributions from the synthetic data specified in Table 4: (a) distribution of the actual l_{pat} and (b) distribution of the $E(F_B)$.

(1) There are $\|I\| = 1000$ unique items in the synthetic database.
(2) Using these $\|I\| = 1000$ unique items $\|\Lambda\| = 5000$ itemsets were generated at random. These are the potentially frequent itemsets used to construct the base patterns.
(3) On average there are $I_{pat} = 2$ items per itemset in these 5000 potentially frequent itemsets.
(4) $N_{pat} = 100$ base patterns were randomly constructed using the 5000 potentially frequent itemsets.
(5) On average there are $L_{pat} = 7$ itemsets per base pattern. The actual distribution of the number of itemsets for each of the 100 base patterns, l_{pat}, is given in Fig. 1(a). Remember that l_{pat} has a Poisson distribution with mean at L_{pat}. When $L_{pat} = 7$, 10% of the patterns have between 3 and 4 itemsets in the base patterns. On the other hand 5% of the patterns have more than 10 itemsets per base pattern. The remaining 85% of base patterns have between 5 and 10 itemsets per pattern. Values of $L_{pat} < 7$ start to introduce base patterns of less then three itemsets per pattern. Thus, $L_{pat} = 7$ is the practical minimum value that will embed sequential patterns of more than two itemsets into the synthetic data.
(6) $N_{seq} = 10{,}000$ data sequences were constructed using the 100 base patterns.
(7) The distribution of the expected frequencies, $E(F_B)$, of the 100 base patterns is given in Fig. 1(b). Of 100 base patterns, 11 have $E(F_B) < 0.1\%$ ($0.1\% * 10{,}000 = 10$ seq). Of them, one base pattern has expected frequency less than 0.01% ($0.01\% * 10{,}000 = 1$ seq). As discussed above these are the practical outliers that occur rarely in the database. On the other hand, there are 12 base pattern with $E(F_B) > 2\%$ ($2\% * 10{,}000 = 200$ seqs). Of these the four largest $E(F_B)$ are 7.63%, 5.85%, 3.80%, and 3.35%, respectively. The other 8 are all between 2% and 3% ($2\% < E(F_B) \leqslant 3\%$). The majority, 77 base patterns, have $E(F_B)$ between 0.1% and 2% ($10 \text{ seq} = 0.1\% < E(F_B) \leqslant 2\% = 200$ seqs).
(8) The base patterns were combined so that on average there are $L_{seq} = 10$ itemsets per data sequence and $I_{seq} = 2.5$ items per itemset in a data sequence. Note that since $L_{pat} = 7$ is the practical minimum for embedding sequential patterns into the synthetic data, L_{seq} should be greater than 7. Thus, a reasonable range of L_{seq} would be from 10 to 50.

3.2. *Extention 1: random data*

In this section, we detail how to generate random data with the same properties as the patterned sequence database. Recall that the first four parameters, $\|I\|$, N_{seq}, L_{seq}, I_{seq}, specify the characteristics of the sequence database. We use the same four parameters to specify a random database.

Random data is generated by assuming independence between items both within and across itemsets. Following the assumption that some items occur often while others occur rarely, the probability of an item occurring is exponentially distributed. The number of distinct items and the number of sequences generated are determined by user set parameters $\|I\|$ and N_{seq}, respectively. Same as done in the IBM data generator, the number of itemsets in a sequence and the number of items in an itemset follow a Poisson distribution with mean L_{seq} and I_{seq}, respectively.

3.3. Extention 2: patterned data with varying degree of noise

Noise occurs at the item level in sequential data. Therefore, to introduce varying degree of controlled noise into the IBM patterned data, we use a corruption probability α. Items in the patterned database are randomly changed into another item or deleted with probability α. This implies that $1 - \alpha$ is the probability of any item remaining the same. Hence, when $\alpha = 0$ no items are changed, and higher values of α imply a higher level of noise [13].

3.4. Extention 3: patterned data with varying degree of outliers

Outliers are sequences that are unlike most other sequences in the data. That is there are very few sequences similar to the outlier sequence. A randomly generated sequence, such as the sequences generated for the random data, can be such an outlier sequence. Thus, we introduce controlled level of outliers into the data by adding varying number of random sequences to the IBM patterned data. The random sequences are generated using the same parameters L_{seq}, I_{seq}, and $\|I\|$ as those used to generate the patterned data. In the rest of the paper, random sequences added to patterned data are referred to as outliers.

4. Evaluation criteria

The effectiveness of a sequential pattern mining method can be evaluated in terms of how well it finds the real underlying patterns in the data, and whether or not it generates any confounding information. The extensively used IBM synthetic data generator reports the underlying base patterns in the data along with the database. Therefore, used in conjunction with the proper evaluation criteria it would be possible to benchmark not only the efficiency of the algorithms but also the efficacy under varying conditions. The number of base patterns found or missed is not alone an accurate measure of how well the base patterns were detected because it can not take into account which items in the base pattern were detected or how frequent the pattern is in the data. Instead, we report a comprehensive view by measuring this information at two different levels; (1) at the item level and (2) at the sequence level.

4.1. Evaluation at the item level

At the item level, we adapt the ROC analysis to measure recoverability and precision. ROC analysis is commonly used to evaluate classification systems with known actual values. The confusion matrix contains information about the actual and predicted patterns [8]. The confusion matrix for the evaluation is given in Table 5. The actual patterns are the base patterns that were embedded into the database. The predicted patterns are the result patterns generated from any sequential pattern mining algorithm. Then the true positive items, called *pattern items*, are those items in the result patterns that can be directly mapped back to a base pattern. The remaining items in the result patterns, the false positive items, are defined as *extraneous items*. These are items that do not come from the embedded patterns, but rather the algorithm falsely assumes to be part of the base patterns. The items from the base pattern that were missed in the result patterns, the false negative items, are the *missed items*. In this context, there are no true negative items (cell a). Thus, only the cells b–d are used in the evaluation.

Using the confusion matrix we measure two criteria at the item level. *Recoverability* measures how much of the base patterns have been found. *Precision* measures how precise are the predictions made about the base

Table 5
Confusion matrix

		Predicted (result patterns generated)	
		Negative	Positive
Actual (base patterns embedded)	Negative	a (NA)	b (extraneous items)
	Positive	c (missed items)	d (pattern items)

patterns. That is, precision measures how much confounding information (extraneous items) are included with the true pattern items.

Normally recall, $(\frac{d}{c+d})$, the true positive rate, is used to measure how much of the actual pattern has been found. However, recall is not accurate in this application because base patterns are only *potentially* frequent sequential patterns in the data. The actual occurrence of a base pattern in the data, which is controlled by $E(F_{B_i})$ and $E(L_{B_i})$, varies widely.

$E(F_{B_i})$ is exponentially distributed then normalized to sum to 1. Thus, some base patterns have tiny $E(F_{B_i})$. These base patterns do not exist in the data or occur very rarely. As discussed in Section 3.1 these weak base patterns are in practice outliers that occur rarely in the database. Thus in practice, recovering these patterns is not as crucial as recovering the more frequent base patterns.

$E(L_{B_i})$ controls how many items on average in the base patterns are injected into one occurrence of the base pattern in a sequence. This means that one sequence in the database is not expected to have all the items in the embedded base pattern. Remember that before a base pattern is embedded into a data sequence, the base pattern is corrupted by randomly deleting items from it. $E(L_{B_i})$ controls how many items on average are deleted in this process.

Therefore, taking $E(F_{B_i})$ and $E(L_{B_i})$ into account, we designed a weighted measure, *recoverability*, which can more accurately evaluate how much of the base patterns have been recovered. Specifically, given (1) a set of base patterns, $\{B_i\}$, along with $E(F_{B_i})$ and $E(L_{B_i})$ for each base pattern, and (2) a set of result patterns, $\{P_j\}$, let each result pattern map back to the most similar base pattern. That is, the result pattern, P_j, is matched with the base pattern, B_i, if the longest common subsequence between P_j and B_i, denoted as $B_i \otimes P_j$, is the maximum over all base patterns. We indicate this matching by referring to the matched result patterns with two indices. $P_j(i)$ denotes that pattern P_j has been mapped to base pattern B_i.

Now let $P_{\max}(i)$ be the max pattern for base pattern B_i. A *max pattern*, $P_{\max}(i)$, is the result pattern that shares the most items with a base pattern, B_i, over all result patterns mapped to the same base pattern. Furthermore, at least half of the items in P_j has to come from the base pattern B_i. Thus, $\max_{\text{rslt pat}\{P_j(i)\}} \|B_i \otimes P_j\|$[1] is the most number of items recovered for a base pattern B_i. In essence, max patterns recovered the most information about a particular base pattern. Note that, there is either one or no max pattern for each base pattern. There could be no max pattern for a base pattern if none of the result patterns recovered enough of the items from the base pattern.

$E(L_{B_i}) \cdot \|B_i\|$ is the expected number of items in one occurrence of B_i in a sequence. Hence, $\frac{\max \|B_i \otimes P_j\|}{E(L_{B_i}) \cdot \|B_i\|}$ is the fraction of the expected number of items found. $E(L_{B_i})$ is an expected value, and sometimes the actual observed value, $\max_{\{P_j(i)\}} \|B_i \otimes P_j\|$ is greater than $E(L_{B_i}) \cdot \|B_i\|$. In such cases, the value of $\frac{\max \|B_i \otimes P_j\|}{E(L_{B_i}) \cdot \|B_i\|}$ is truncated to one so that recoverability stays between 0 and 1.

In sum, recoverability is defined as follows:

$$\text{Recoverability } R = \sum_{\text{base pat}\{B_i\}} E(F_{B_i}) \cdot \min \left\{ \begin{array}{l} 1 \\ \left(\frac{\max_{\text{rslt pat}\{P_j(i)\}} \|B_i \otimes P_j\|}{E(L_{B_i}) \cdot \|B_i\|} \right) \end{array} \right. \tag{2}$$

Intuitively, if the recoverability of the mining is high, major portions of the base patterns have been found.

In ROC analysis, *precision*, $\frac{d}{b+d}$, is the proportion of the predicted positive items. It is a good measure of how much of the result is correct [8]. In sequential pattern mining, precision measures how much extraneous items are mixed in with pattern items in the results. Remember that when the result pattern P_j is mapped to base pattern B_i, the items in both the result pattern and the base pattern, $B_i \otimes P_j$, are defined as pattern items. The result pattern P_j is mapped to base pattern B_i, when $\|B_i \otimes P_j\|$ is maximum over all base patterns. Thus, the number of pattern items for a result pattern, P_j, is $\max_{\{B_i\}} \|B_i \otimes P_j\|$. The remaining items in the result pattern, P_j, are the extraneous items. The different item counts in the result patterns are summarized in Table 6. Denoted as P, precision is calculated as follows:

$$\text{Precision } P = \frac{N_{\text{pat}I}}{N_{\text{item}}} \times 100\% = \left(1 - \frac{N_{\text{extra}I}}{N_{\text{item}}}\right) \times 100\% \tag{3}$$

[1] $\|\text{seq}_i\|$ = length of seq_i denotes the total number of items in seq_i.

Table 6
Item counts in the result patterns

Notation	Meaning	Equation
N_{item}	Total # of items	$\sum_{\text{rslt pat}\{P_j\}} \lVert P_j \rVert$
$N_{\text{pat}\,I}$	Total # of pattern items	$\sum_{\text{rslt pat}\{P_j\}} (\max_{\text{base pat}\{B_i\}} \lVert B_i \otimes P_j \rVert)$
$N_{\text{extra}\,I}$	Total # of extraneous items	$\sum_{\text{rslt pat}\{P_j\}} (\lVert P_j \rVert - \max_{\text{base pat}\{B_i\}} \lVert B_i \otimes P_j \rVert)$

4.2. Evaluation at the sequence level

At the sequence level, the three criteria that we measure are (1) the total number of result patterns, (2) the number of spurious patterns, and (3) the number of redundant patterns. To do so, we categorize the result patterns into spurious, redundant, or max patterns depending on the composition of pattern items and extraneous items. We do not report the number of max patterns because it can be easily calculated by $N_{\max} = N_{\text{total}} - N_{\text{spur}} - N_{\text{redun}}$.

Spurious patterns are those that were not embedded into the database, but what the algorithm incorrectly assumed to be sequential patterns in the data. In this evaluation, spurious patterns are defined as the result patterns that have more extraneous items than pattern items. As discussed in the previous section, max patterns are those that recover the most pattern items for a given base pattern. The remaining sequential patterns are redundant patterns. These are result patterns, P_a, which match with a base pattern, B_i, but there exists another result pattern, $P_{\max}$, that matches with the same base pattern but better in the sense that $\lVert B_i \otimes P_{\max} \rVert \geqslant \lVert B_i \otimes P_a \rVert$. Therefore these patterns are redundant data that clutter the results.

4.3. Units for the evaluation criteria

Recoverability and precision are reported as a percentage of the total number of items in the result ranging from 0% to 100%. In comparison, the spurious patterns and redundant patterns are reported as number of patterns. These measures can easily be changed to percentage of the total number of result patterns as needed.

We report the actual number of patterns because the number of spurious patterns can be tiny as a percentage of the total number of result patterns. In fact, by definition we expect that there will be only a few spurious patterns if the algorithm is reasonably good. In such situations, a user would want to see exactly how few spurious patterns are in the result rather than its proportion in the result. For example, one of the experiments on the support model had over 120,000 result patterns of which 16 (0.012%) were spurious patterns.

Unlike spurious patterns, redundant patterns are not incorrect patterns. Sometimes, they can even have additional information, such as suggesting common variations of a strong pattern in the data. The most negative effect of redundant patterns is the confounding effect it can have on understanding the results when there are too many of them. Hence, the exact number of redundant patterns is directly related to the interference factor. For example, it is easy to glean some information and/or ignore 10 redundant patterns of 20 result patterns but not so easy to work through 50% of 120,000 patterns.

The five evaluation criteria are summarized in Table 7.

5. Example

Let Table 8 be the base patterns used to construct a sequence database. The expected frequency $E(F_{B_i})$, the expected length after corruption $E(L_{B_i})$, and the actual length $\lVert B_i \rVert$ of the base patterns are also given.

Table 7
Evaluation criteria

Criteria	Meaning	Level	Unit
Recoverability R	The degree of the base patterns detected (Eq. (2))	Item	%
Precision P	1 – degree of extraneous items in the result patterns (Eq. (3))	Item	%
N_{spur}	# of spurious patterns ($N_{\text{extra}\,I} > N_{\text{pat}\,I}$)	seq	# of patterns
N_{redun}	# of redundant patterns	seq	# of patterns
N_{total}	Total # of result patterns returned	seq	# of patterns

Table 8
Base patterns $\{B_i\}$: $N_{\text{pat}} = 3$, $L_{\text{pat}} = 7$, $I_{\text{pat}} = 2$

ID	Base patterns B_i								$E(F_{B_i})$	$E(L_{B_i})$	$\|B_i\|$
B_1	(PR)	(Q)	(IST)	(IJ)	(U)	(D)	(NT)	(I)	0.566	0.784	13
B_2	(FR)	(M)	(GK)	(C)	(B)	(Y)	(CL)		0.331	0.805	10
B_3	(D)	(AV)	(CZ)	(HR)	(B)				0.103	0.659	8

Table 9
Result patterns $\{P_j\}$

ID	Result pattern P_j								$\|P_j\|$
P_1	(PR)	(Q)	(I)	(IJ)	(IJU)	(U)	(D)	(T)	12
P_2	(GKQ)	(IT)	(IJ)	(D)	(NT)				10
P_3	(P)	(IS)	(U)	(DV)	(NT)				8
P_4	(FR)	(M)	(C)	(BU)	(Y)	(CL)			9
P_5	(F)	(AV)	(CL)	(BSU)	(I)				9

In addition, let Table 9 be the result patterns from a sequential pattern mining algorithm. The length of the result patterns is given in $\|P_j\|$. Then, the evaluation is done in the follows steps:

(1) *Identify total number of result patterns.* $N_{\text{total}} = \|\{P_j\}\| = 5$.
(2) *Map result patterns to base patterns.* Each result pattern, P_j, is mapped to the best matching base pattern B_i such that $\|B_i \otimes P_j\|$ is maximized over all base patterns B_i in Table 10. For result pattern P_5, $\|B_1 \otimes P_5\| = \|(\text{U})(\text{I})\| = 2$, $\|B_2 \otimes P_5\| = \|(\text{F})(\text{C})(\text{B})\| = 3$, and $\|B_3 \otimes P_5\| = \|(\text{AV})(\text{C})(\text{B})\| = 4$. Thus, result pattern P_5 is mapped to base pattern B_3.
(3) *Count the number of pattern items and extraneous items for each result pattern.* In Table 10, the number of pattern items and extraneous items are given in $N_{\text{pat}\,I}$ and $N_{\text{extra}\,I}$. Result pattern P_1 has nine pattern items it shares with B_1 (PR)(Q)(I)(IJ)(U)(D)(T) and three extraneous items (IJU).
(4) *Calculate precision, P.* The total number of extraneous items is $3 + 2 + 1 + 1 + 5 = 12$. The total number of items in the result pattern is $12 + 10 + 8 + 9 + 9 = 48$. Thus, $P = (1 - \frac{12}{48}) \times 100\% = 75\%$.
(5) *Identify spurious patterns*, N_{spur}. If a result pattern, P_j, has more extraneous items than pattern items, it is classified as a spurious pattern. P_5 is a spurious pattern because $4 < 5$. Therefore, $N_{\text{spur}} = 1$.
(6) *Identify max patterns*, $P_{\max}(i)$. Of the remaining result patterns, for each base pattern, B_i, identify the max result pattern such that $\|B_i \otimes P_j\|$ is maximized over all result patterns $P_j(i)$ mapped to B_i. In Table 10, result patterns are sorted by $\|B_i \otimes P_j\|$ for each base pattern. P_1 and P_4 are max patterns for B_1 and B_2, respectively. B_3 does not have a max pattern.
(7) *Identify redundant patterns*, N_{redun}. All remaining result patterns are redundant patterns. P_2 and P_3 are redundant patterns for B_1. $N_{\text{redun}} = 2$.
(8) *Calculate recoverability, R.* For each max pattern, calculate recoverability with respect to B_i, $R(B_i) = \frac{\|B_i \otimes P_{\max}(i)\|}{E(L_{B_i}) \cdot \|B_i\|}$. Truncate $R(B_i)$ to 1 if necessary. Weight and sum over all base patterns.

$$R = E(F_{B_1}) \cdot R(B_1) + E(F_{B_2}) \cdot R(B_2) + E(F_{B_3}) \cdot R(B_3) = 0.566 \cdot \frac{9}{10} + 0.331 \cdot \frac{8}{8} + 0.103 \cdot 0 = 0.84 = 84\%$$

6. Comparison study

In this section, we employ the evaluation method to conduct a comparison study of the traditional support model[2] with an alternative multiple alignment model. The comprehensive evaluation method enables us to understand what patterns are returned from each model under a variety of situations.

[2] Traditionally, one itemset frequent patterns, called *large itemsets*, are considered as part of the results. However, in our evaluation we only consider result patterns with more than one itemset in the sequence as a valid pattern. One itemset patterns are obviously not meaningful sequential patterns and can be dismissed easily.

Table 10
Worksheet: $R = 84\%,\ P = 1 - \frac{12}{48} = 75\%,\ N_{\text{total}} = 5,\ N_{\text{spur}} = 1,\ N_{\text{redun}} = 2$

ID	Base pattern B_i									$\lVert B_i \rVert$	$E(F_{B_i})$	$E(L_{B_i})$	$E(L_{B_i}) \cdot \lVert B_i \rVert$
	Result pattern P_j									$\lVert P_j \rVert$	$N_{\text{pat}\,I}$	$N_{\text{extra}\,I}$	$R(B_i)$
B_1	(PR)	(Q)	(IST)	(IJ)		(U)	(D)	(NT)	(I)	13	0.566	0.784	10
P_1	(PR)	(Q)	(I)	(IJ)	(IJU)	(U)	(D)	(T)		12	9	3	9/10 = 0.9
P_2		(GKQ)	(IT)	(IJ)			(D)	(NT)		10	8	2	Redundant
P_3	(P)		(IS)			(U)	(DV)	(NT)		8	7	1	Redundant
B_2	(FR)	(M)	(GK)	(C)	(B)	(Y)	(CL)			10	0.331	0.805	8
P_4	(FR)	(M)		(C)	(BU)	(Y)	(CL)			9	8	1	8/8 = 1
B_3	(D)	(AV)	(CZ)	(HR)	(B)					8	0.103	0.659	5
P_5	(F)	(AV)	(CL)		(BSU)	(I)				9	4	5	Spurious

6.1. Spurious patterns in random data

In this section, we study empirically under what condition spurious patterns are generated from completely random data. Since there are no base patterns embedded in the data, evaluation criteria recoverability, precision, and number of redundant patterns do not apply. The only important criteria that applies is the number of spurious patterns, N_{spur}, generated by the algorithm. Since there are no base patterns in the data $N_{\text{total}} = N_{\text{spur}}$. We generate random databases with parameters $\lVert I \rVert = 100$, $N_{\text{seq}} = 1000$, $I_{\text{seq}} = 2.5$, and $L_{\text{seq}} = 10, \ldots, 50$.

Both models have a user specified cutoff point. In the support model, the user sets the minimum support min_sup. In the alignment model, the user sets the strength cutoff point θ. To better understand each of the parameters, we first examined the threshold where the first spurious pattern occurs, T_{spur}. In both models, T_{spur} suggests that as the sequences get longer, spurious patterns are more likely to occur (Fig. 2(a)). This is to be expected. However, the significant differences in the values of T_{spur} in the two models should be fully appreciated. In the support model, T_{spur} is the highest point at which a spurious pattern appears in the full database. On the other hand, in the multiple alignment model, T_{spur} is the highest point at which a spurious pattern occurs in any subgroup of the database (any cluster). That is support is based on the full database, whereas strength is based on clusters.

Thus, in practice min_sup is usually set very low and is almost always less than 10%. In the support model, when T_{spur} is greater than 10% it suggests a significant problem in dealing with spurious patterns. Fig. 2(a) show that in all datasets, T_{spur} is in fact greater than 10%.

In comparison, since the strength cutoff point, θ, is specified against a similar group of sequences, it is set high (20–50%) in practice. The suggested default for consensus patterns is 50%. Thus, $T_{\text{spur}} \leqslant 35\%$ for all databases signifies that the first spurious pattern occurs at a relatively low point. Spurious patterns can clearly be differentiated from real patterns in such circumstances.

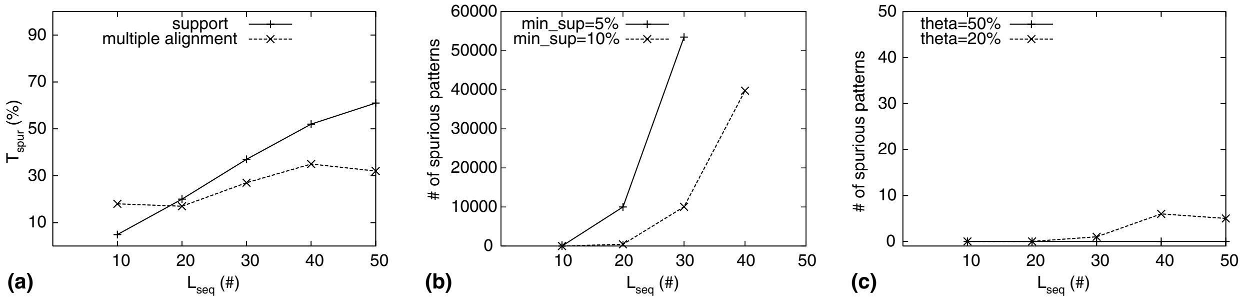


Fig. 2. Comparison results for random data: (a) T_{spur}, (b) support model and (c) alignment model.

Now let us examine the results of each model in detail. The support model has no mechanism to eliminate patterns that occur simply by chance. When sequences are long, short patterns can occur frequently simply by chance [7]. The threshold where the first spurious pattern is generated, T_{spur}, depicts this well empirically. As the sequence becomes longer, T_{spur} increases quickly. When $L_{\text{seq}} = 50$, a simple sequential pattern, $\langle (A)(A) \rangle$, occurs in 610 of 1000 sequences simply by chance. That is 61% of the sequences share the random short pattern. Even when L_{seq} is only 20, the sequential pattern occurs in 200 of 1000 sequences simply by chance. Thus, when min_sup is set to conventional values of less than 10% we can expect the result to include these short spurious patterns. Accordingly, the support model generated many spurious patterns given random data. In fact, Fig. 2(b) demonstrates that the number of spurious patterns increases exponentially with respect to L_{seq}. Clearly, support alone cannot distinguish between significant patterns and random sequences.

In contrast, ApproxMAP handles random data very well. The probability of a group of random sequences aligning well enough to generate a consensus pattern is negligible. Thus, using default parameters ($k = 5$, $\theta = 50\%$), the multiple alignment model found no spurious patterns in any database with $L_{\text{seq}} = 10, \ldots, 50$ (Fig. 2(c)). Although the algorithm generated many clusters, all the clusters were either very small, or not enough sequences in the cluster could be aligned to generate consensus patterns. When sequences are longer, there are a few negligible number of spurious patterns (1, 6, and 5 when $L_{\text{seq}} = 30$, 40, and 50, respectively) generated at the lower cutoff point of $\theta = 20\%$.

6.2. Baseline study of patterned data

Now we move on to investigate the behavior of the algorithms on patterned data. This experiment serves several purposes. First, it evaluates how well the models detect the underlying patterns in a simple patterned database. Second, it enables us to study the behavior of the input parameters. Third, it illustrates how readily the results may be understood. Fourth, it establishes a baseline for the remaining experiments.

To keep the results manageable, we generate a simple database with 1000 sequences built from 10 base patterns. The full parameters of the database is given in Table 11. The first task is to study the input parameters and tune both models to the optimal settings for this dataset.

Table 12 gives the results for the support model for min_sup = 4%, ..., 10%. As expected, when support is decreased recoverability improves. However, the number of redundant and spurious patterns increase as support is decreased. There is not much change in precision with respect to support. Although though the number of extraneous items increases significantly, the number of items in the redundant patterns grows just as quickly. Thus, precision can not properly depict how much extraneous items exist in the results because the result is overwhelmed by the huge number of redundant patterns. In the presence of large numbers of redundant and spurious patterns, the amount of extraneous items mixed in with the sequential patterns hold less significance.

Table 11
Parameters for the IBM data generator in experiment 2

Sequence database	$\|\|I\|\| = 100$	$N_{\text{seq}} = 1000$	$L_{\text{seq}} = 10$	$I_{\text{seq}} = 2.5$
Base pattern	$\|\|A\|\| = 500$	$N_{\text{pat}} = 10$	$L_{\text{pat}} = 7$	$I_{\text{pat}} = 2$

Table 12
Results from patterned data from the support model

min_sup (%)	Recoverability (%)	N_{item}	$N_{\text{extra}\,I}$	Precision (%)	N_{total}	N_{spur}	N_{redun}
4	94.20	6,085,496	197,074	96.76	715,758	249	715,499
5	91.59	1,782,583	66,058	96.29	253,782	58	253,714
6	**91.52**	**881,721**	**26,633**	**96.98**	**128,936**	**16**	**128,910**
7	87.67	543,717	12,213	97.75	82,638	7	82,621
8	83.76	286,418	6279	97.81	47,540	4	47,526
9	80.81	158,035	3465	97.81	28,337	4	28,323
10	76.89	89,535	2063	97.70	17,323	3	17,310

Table 13
Results from patterned data from the alignment model ($\theta = 30\%$)

k	Recoverability (%)	N_{item}	$N_{\text{extra}\,I}$	Precision (%)	N_{total}	N_{spur}	N_{redun}
3	92.36	179	5	97.21	15	0	7
4	91.66	153	2	98.69	13	0	6
5	91.16	136	4	97.06	11	0	4
6	**91.16**	**106**	**3**	**97.17**	**8**	**0**	**1**
7	85.77	100	1	99.00	8	0	2
8	82.86	90	4	95.56	7	0	1
9	85.77	90	0	100.00	7	0	1
10	70.76	82	4	95.12	6	0	1

As the optimal point, we considered min_sup equal to 5% and 6% which had recoverability at over 90% and number of spurious patterns less than 100. For similar recoverability, min_sup = 6% had much less spurious patterns. Thus, for this dataset, we chose min_sup = 6% as the optimal setting.

There are two input parameters, k and θ, for ApproxMAP. This makes it more difficult to find the optimal setting for ApproxMAP. Limited by space, we summarize the results from the various experiments we ran to find the optimal settings. In our first experiment we found that $\theta = 30\%$ was a reasonable setting that gave good results for a range of k.

Table 13 give the results from our second experiment with $\theta = 30\%$ while varying k. As expected, as k increases, more sequences are clumped together to form less clusters. This is reflected in the reduction of N_{total}. Initially the clusters merged in this process are those with similar sequences built from the same base pattern. This can be seen for $k = 3,\ldots,6$ where N_{redun} decreases from 7 to 1 and little change occur in recoverability from 92.36% to 91.16%. However, when k is increased beyond 6, small clusters (sequences built from less frequent base patterns) are merged together and we start to loss the less frequent patterns resulting in decreased recoverability. For $k = 6,\ldots,10$, this phenomena occurs where recoverability is decreased from 91.16% to 70.76%. Fig. 3(a) depicts the drop in recoverability at $k = 6$. Fig. 3(b) illustrates that the number of redundant patterns levels off at the same point ($k = 6$). Hence, $k = 6$ is the optimal resolution for clustering this database.

As a last step, we tried to optimize θ for the optimal $k = 6$. To be precise, the optimal point is the one where the most pattern items are found with the least possible extraneous items. In our experiment with k set to 6 while varying θ, we found that the optimal results can be obtained when $k = 6$ and $28\% \geqslant \theta \geqslant 30\%$. Results obtained when $25\% \geqslant \theta \geqslant 35\%$ were very similar to the optimal results as well. The details are given in Table 14.

These experiments on the input parameters revealed interesting behaviors of both models. The experiment on the support model depicts the trade off between recoverability and number of redundant and spurious patterns. The experiment reiterates that support is not a good criteria to detect only the underlying patterns in the data as it can not differentiate between random occurrences and statistically significant patterns. By lowering support, the user can detect more patterns at the cost of more redundant and spurious patterns.

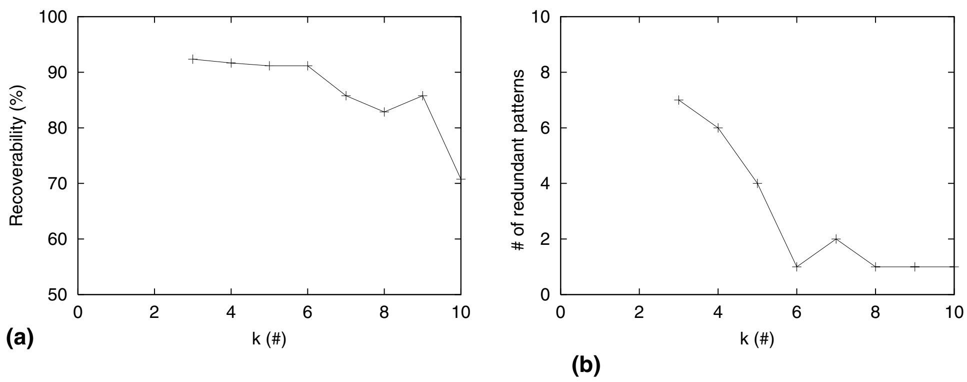


Fig. 3. Effects of k (a) recoverability w.r.t. k and (b) number of redundant patterns w.r.t. k.

Table 14
Results from patterned data from the alignment model ($k = 6$)

θ (%)	Recoverability (%)	N_{item}	$N_{\text{pat}\,I}$	$N_{\text{extra}\,I}$	Precision (%)	N_{total}	N_{spur}	N_{redun}
20	93.10	125	106	19	84.80	8	0	1
25	91.16	108	103	5	95.37	8	0	1
30	**91.16**	**106**	**103**	**3**	**97.17**	**8**	**0**	**1**
35	91.16	102	100	2	98.04	8	0	1
40	88.69	98	98	0	100.00	8	0	1
45	88.69	97	97	0	100.00	8	0	1
50	87.45	96	96	0	100.00	8	0	1

The experiment on the alignment model revealed that ApproxMAP is robust with respect to the input parameters. That is many settings give results comparable to the optimal solution ($k = 5$ or 6; $25\% \geqslant \theta \geqslant 35\%$). More importantly, for a wide range of k and θ the results are at least a sub-optimal solution. For $k = 3$ to 9 and $25\% \geqslant \theta \geqslant 45\%$, all results had recoverability greater than 80% and precision greater than 90%.

Now let us compare the mined results from the optimal runs. The results and the optimal parameters are summarized in Table 15. In this section, we add an important variation of the support model to the comparison experiment. The *MaxSupport* model retrieves only the max sequential patterns from the support model, which reduces the redundancy in the results. We were interested to see just how much redundancy would be reduced. For this experiment, we used the brute force method to find max sequential patterns. It should be noted that there is no known efficient algorithm to find max sequential patterns yet. The only paper we are aware of is [12] for finding closed sequential patterns. More research is required for extending it to find max sequential patterns. The brute force method is too inefficient to be applicable on real data.

The recoverability for all models is good at over 90%. However, in the support model it is difficult to extract the 10 base patterns from over 128,000 results. In fact, the generated patterns from the support model are almost 129 times the number of sequences in the given database ($N_{\text{seq}} = 1000$ and $N_{\text{total}} = 128{,}936$). Majority of the result patterns are redundant patterns ($N_{\text{redun}} = 128{,}910$) which are either subsequences of a longer pattern or a small variation on it. They hold no additional information and instead bury the true patterns. In addition, there are more spurious patterns ($N_{\text{spur}} = 16$) than the 10 base patterns in the data. Even if we were to find only the max-patterns, there are still 20,992 max-patterns. That is still close to 21 times that of the number of sequences in the given database. Again most are redundant patterns: 20,974 are redundant patterns and 9 are spurious patterns. The results still hold too much redundancy due to the variations that exist in the basic pattern and noise in the data. Furthermore, although the precision seems to be reasonable at 96.98%, this accounts for 26,633 extraneous items. The precision seems good because there are so many redundant items in the results. Consequently, when the redundancy is reduced in the max support model, precision decreases to 87.77%.

In contrast, the alignment model returned a very succinct but accurate summary of the base patterns. Table 16 gives, in one small table, the full results. All the consensus patterns, PatConSeq$_i$, with the matching base patterns, BaseP_i, are shown along with expected frequency and length of the base patterns. The sequences are sorted by the expected frequency of the base patterns. In this small database, manual inspection clearly shows how well the consensus patterns match the base patterns used to generate the data. Each consensus pattern found was a subsequence of considerable length of a base pattern. Clearly, the eight consensus sequences provide a good overview of the 1000 data sequences. The consensus patterns do not cover the three weakest base

Table 15
Comparison results for patterned data

Model	Optimal settings	R (%)	N_{item}	$N_{\text{extra}\,I}$	P (%)	N_{total}	N_{spur}	N_{redun}
Alignment	$k = 6$, $\theta = 28\%, \ldots, 30\%$	91.16	106	3	97.17	8	0	1
Support	min_sup = 6%	91.52	881,721	26,633	96.98	128,936	16	128,910
MaxSupport	min_sup = 6%	91.15	122,144	14,939	87.77	20,992	9	20,974

Table 16
Consensus patterns and the base patterns in a small data set

$E(F_B)$	$E(L_B)$	$\|P\|$	Type	Patterns
0.21	0.66	13	ConPat_1	⟨(15,16,17,66)(15)(58,99)(2,74)(31, 76)(66)(62)⟩
		14	BasePat_1	⟨(15,16,17,66)(15)(58,99)(2, 74)(31,76)(66)(62)(93)⟩
0.161	0.83	19	ConPat_2	⟨(22,50,66)(16)(29,99)(94)(45,67)(12,28,36)(50)(96)(51)(66)(2,22, 58)⟩
		15	ConPat_3	⟨(22,50,66)(16)(29,99)(94)(45,67)(12,28,36)(50)(96)(51)⟩
		22	BasePat_2	⟨(22,50,66)(16)(29,99)(94)(45, 67)(12,28,36)(50)(96)(51)(66)(2,22,58)(63,74,99)⟩
0.141	0.82	11	ConPat_4	⟨(22)(22)(58)(2,16,24,63)(24,65,93)(6)⟩
		14	BasePat_3	⟨(22)(22)(58)(2,16,24,63)(24, 65,93)(6)(11,15,74)⟩
0.131	0.9	11	ConPat_5	⟨(31,76)(58,66)(16,22,30)(16)(50,62,66)⟩
		15	BasePat_4	⟨31,76)(58,66)(16,22,30)(16)(50, 62,66)(2,16,24,63)⟩
0.123	0.81	13	ConPat_6	⟨(43)(2,28,73)(96)(95)(2,74)(5)(2)(24,63)(20)⟩
		14	BasePat_5	⟨(43)(2,28,73)(96)(95)(2,74)(5)(2)(24, 63)(20)(93)⟩
0.121	0.77	8	ConPat_7	⟨(63)(16)(2,22)(24)(22,50,66)⟩
		9	BasePat_6	⟨(63)(16)(2,22)(24)(22,50, 66)(50)⟩
0.054	0.6	16	ConPat_8	⟨(70)(58)(22,58,66)(22,58)(74)(22,41)(2,74)(31,76)(2,74)⟩
		13	BasePat_7	⟨(70)(58,66)(22)(74)(22,41)(2, 74)(31,76)(2,74)⟩
0.014	0.91	17	BasePat_8	⟨(20,22,23,96)(50)(51,63)(58)(16)(2, 22)(50)(23,26,36)(10,74)⟩
0.038	0.78	7	BasePat_9	⟨(88)(24,58, 78)(22)(58)(96)⟩
0.008	0.66	17	BasePat_{10}	⟨(16)(2,23,74,88)(24,63)(20, 96)(91)(40,62)(15)(40)(29,40,99)⟩

patterns. But, recoverability is still quite good at 91.16% because in general, the consensus patterns recover major parts of the base patterns with high expected frequency in the database. In addition, there were only one redundant pattern, no spurious patterns, and three extraneous items.

It is interesting to note that a base pattern may be recovered by multiple consensus patterns. For example, ApproxMAP forms two clusters whose consensus patterns approximate base pattern BaseP_2. This is because BaseP_2 is long ($\|\text{Base}P_2\| = 22$, $E(L_B) = 18$) and has a high expected frequency (16.1%). Therefore, many data sequences are generated using BaseP_2 as a template. However, two sequences using the same long base pattern as the template are not necessarily similar to each other because sequences are generated by removing various parts of the base pattern and combining them with other items. As a result, the sequences generated from a long base pattern can be partitioned into multiple clusters. One cluster with sequences that have most of the 22 items from BaseP_2 (PatConSeq_2) and another cluster with sequences that are shorter (PatConSeq_3). The one which shares less with the base pattern is classified as a redundant pattern in the evaluation method.

6.3. Robustness with respect to noise

Typically, real data have high levels of noise. For example, in a sales marketing database, many customers may share similar buying habits, but few of them follow exactly the same buying patterns. A good sequential pattern mining algorithm should be able to detect the general trend shared across these similar sequences in the presence of noise. In this section, we evaluate the robustness of the models with respect to varying degree of noise added to the patterned data used in Section 6.2.

The support model is on exact match. That is, a sequence in the database supports a pattern if, and only if, the pattern is fully contained in the sequence. Results show that such an exact match-based model is vulnerable to noise in the data. As seen in Fig. 4, as the corruption factor, $1 - \alpha$, increases, the support model detects less of the base patterns (recoverability decrease) and picks up more extraneous items (precision decrease). When the corruption factor is 30%, recoverability degrades significantly to 25.89%. Even when min_sup is lowered to 2%, the recoverability is only 65.01% when the corruption factor is 30%. Note that even with recoverability at 25.89%, the model returns 2070 patterns that include 495 extraneous items.

In comparison, the alignment model is robust to random noise in the data. The goal in the alignment model is to report the general underlying trend in the data through sequence alignment. In the process of lining similar sequences, ApproxMAP tunes into those items that are shared across sequence while ignoring random noise not shared by sequences. Hence, despite the presence of noise, as shown in Fig. 4, ApproxMAP is still

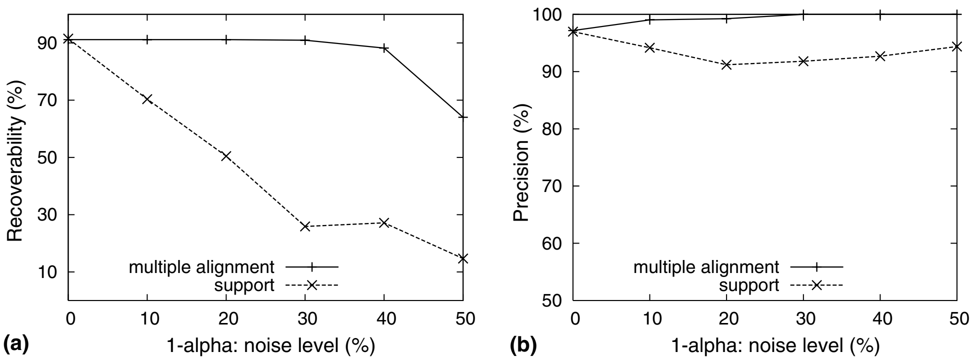


Fig. 4. Effects of noise under optimal settings (a) recoverability w.r.t. $1 - \alpha$ and (b) precision w.r.t. $1 - \alpha$.

able to detect a considerable number of the base patterns (i.e. recoverability is 90.95% when corruption factor is 30%) with high precision and no spurious patterns.

6.4. Robustness with respect to outliers

It is safe to assume that real data has high levels of outliers along with noise. A good mining algorithm should be able to detect the patterns regardless of the outliers in the data. In order to test the effect of outliers on the algorithm, we added random sequences to the patterned dataset used in Section 6.2.

The main effect of the outliers is the weakening of the patterns as a percentage of the database. Consequently, in the support model the recoverability along with the number of spurious patterns, the number of redundant patterns, and the number of extraneous items is decreased when min_sup is maintained at 6%. On the other hand, when min_sup is maintained as 60 sequences, obviously the recoverability can be maintained. The tradeoff is that the number of spurious patterns, redundant patterns, and extraneous items all increase.

Similarly, in ApproxMAP when θ is kept the same at 30% the results decline as the number of outliers increase. However, the longer underlying patterns can be easily detected by adjusting θ to compensate for the outliers in the data. In summary, with a slight decrease in θ we can recover all of the base patterns detected in the simple patterned data with only minor decrease in precision. The detailed results can be found in the Appendix A.

6.5. Discussion

In the absence of noise in the data, the support model can find the underlying patterns in the data. However, the real patterns are buried under the huge number of redundant and spurious patterns. In the presence of noise in the data, the ability to detect the underlying patterns quickly degrades but a large number of redundant and spurious patterns still remain.

These empirical results are inline with the theoretical analysis of the support model and clearly depicts the inherent limitations. First, support along cannot distinguish between statistically significant patterns and random occurrences [7]. Many short patterns can occur frequently simply by chance along. Consequently, the support model returns more spurious patterns than the base patterns in the data. Second, by definition of finding the complete set of frequent subsequences, the support model has much redundancy in the results. Furthermore, even when the results contain only the max sequential patterns the redundancy from variations of the underlying pattern still remains. This is evident in the numerous redundant patterns in both the support model and the max support model. And finally, the exact match-based support model can not detect the underlying patterns in the presence of noise in the data.

In comparison, the evaluation results revealed that the alignment model returns a succinct but accurate summary of the base patterns with few redundant patterns and no spurious patterns. ApproxMAP is able to summarize the 1000 data sequences into 8 consensus patterns. Furthermore ApproxMAP is robust with respect to the two input parameters k and θ. Many settings give results comparable to the optimal solution while a wide range of k and θ give at least a sub-optimal solution. In addition, the experiments demonstrate that the alignment model is robust to both noise and outliers in the data.

Again these empirical results support the theory well. The likelihood of random sequences lining up to produce patterns is slim. Furthermore, the alignment process allows for approximate matches and ignores both noise and variations in the data. By lining up similar sequences and detecting the general trend, the multiple alignment model effectively finds consensus patterns that are approximately similar to many sequences. Such an approach dramatically reduces the redundancy among the derived patterns and virtually eliminates all spurious patterns.

This empirical study suggests that the alignment model will give a good summary of the sequential data in the form of a set of common patterns in the data. In contrast, the support model does not try to summarize. In fact, it tends to return many more patterns than the number of sequences in the original data. This suggests the need for more post processing. It seems appropriate to consider the output of the support model as summary statistics of the sequential data which can be used in subsequent mining processes rather than the end results. Indeed in association rule mining, the output of the support counts are fed into the confidence rule to find association rules between items. Application of such confidence rules to sequential data can be vague and more complex [10]. In order for the support model to be of real use, more research is needed on the final post processing steps of sequential pattern mining.

7. Conclusion and future work

7.1. Conclusion

As a first step towards building a general benchmark for sequential pattern mining, we propose an evaluation method that can quantitatively assess how well the models can find known common patterns in the data. We first extend the well-known IBM synthetic data generator to generate variety of situations varying randomness and noise. Then, we develop a comprehensive set of evaluation criteria to use in conjunction with the IBM synthetic data. Together the five criteria: (1) recoverability, (2) precision, (3) the total number of result patterns returned, (4) the number of spurious patterns, and (5) the number of redundant patterns measure how much of the underlying patterns are found and whether or not it generates any spurious patterns or extraneous items.

Such a method provides a basis for comparing the results of different sequential mining models. We emphasize that the purpose of the benchmark is not to determine which sequential method is best for all domains. Rather, the evaluation depicts what information is being returned from the various models under a variety of circumstances. Namely, the evaluation method provides a comprehensive understanding of the resulting patterns empirically.

This evaluation method will enable researchers not only to use synthetic data to benchmark performance in terms of speed, but also to quantify the quality of the results. Such benchmarking will become increasingly important as more data mining methods focus on approximate solutions.

7.2. Future work

Known common patterns in the data are only one type of target patterns. For a more comprehensive benchmark for sequential pattern mining, much work is needed to design good synthetic data with different types of target patterns and matching evaluation criteria. Here we briefly discuss two commonly studied target patterns, frequent patterns and outlier patterns.

Frequent patterns can be defined as subsequences that occur frequently in a given database. Although most widely studied, the fundamental question as to what should be considered frequent in a given database has not been researched much. In order to evaluate frequent pattern mining effectively, the target pattern, frequent sequential patterns, must be objectively defined first. More precisely, given a database, how can we determine an objective support threshold rather than a user given threshold. Designing appropriate evaluation criteria for frequent patterns should come after objectively defining the target pattern.

Outlier patterns are the problem of detecting rare sequences in the data such as in fraud detection. We can expand the current framework to evaluate outlier pattern detection since many of the base patterns are in fact practical outliers. Specifically, base patterns with small $E(F)$ are outliers that occur rarely in the database. Thus, we can consider these base patterns as the target patterns. All but one of the evaluation criteria proposed in this paper apply to outlier pattern detection. Recoverability is not applicable as it gives more emphasis on finding commonly occurring patterns. Comparable criteria need to be designed for outlier pattern detection.

Furthermore, methods to evaluate mining algorithms using real data need more attention. Access to real sequential data is virtually impossible. Most real data available in data repositories are too small for mining sequential data. Even the unusually large *Gazella* data set from KDD-CUP 2000 only has an average of less than two itemsets per sequence [5]. Hence, the dataset cannot be used to mine sequential patterns. More effort is needed to provide real sequential data publicly.

Appendix A. Detailed results

(See Tables A.1–A.6).

Table A.1
Effects of noise: the support model (min_sup = 6%)

$1-\alpha$ (%)	Recoverability (%)	N_{item}	$N_{\text{extra}\,I}$	Precision (%)	N_{total}	N_{spur}	N_{redun}
0	91.52	881,721	26,633	96.98	128,936	16	128,910
10	70.37	114,630	6696	94.16	24,591	5	24,576
20	50.44	20,582	1814	91.19	5907	3	5895
30	25.89	6047	495	91.81	2070	0	2063
40	27.14	2335	171	92.68	906	0	899
50	14.67	1013	57	94.37	440	0	435

Table A.2
Effects of noise: the alignment model ($k = 6$, $\theta = 30\%$)

$1-\alpha$ (%)	Recoverability (%)	N_{item}	$N_{\text{extra}\,I}$	Precision (%)	N_{total}	N_{spur}	N_{redun}
0	91.16	106	3	97.17	8	0	1
10	91.16	104	1	99.04	9	0	2
20	91.16	134	1	99.25	12	0	5
30	90.95	107	0	100.00	9	0	2
40	88.21	95	0	100.00	9	0	2
50	64.03	68	0	100.00	8	0	3

Table A.3
Effects of outliers: the support model (min_sup = 6%)

N_{outlier}	min_sup (%)	Recoverability (%)	N_{item}	$N_{\text{extra}\,I}$	Precision (%)	N_{total}	N_{spur}	N_{redun}
0	60/1000 = 6.0	91.52	881,721	26,633	96.98	128,936	16	128,910
200	72/1200 = 6.0	87.67	476,480	10,628	97.77	73,936	7	73,919
400	84/1400 = 6.0	83.76	224,885	4961	97.79	38,614	4	38,600
600	96/1600 = 6.0	77.98	111,687	2603	97.67	21,094	3	21,081
800	108/1800 = 6.0	67.48	53,765	1402	97.39	11,407	2	11,396

Table A.4
Effects of outliers: the support model (min_sup = 60 sequences)

N_{outlier}	min_sup (%)	Recoverability (%)	N_{item}	$N_{\text{extra}\,I}$	Precision (%)	N_{total}	N_{spur}	N_{redun}
0	60/1000 = 6.0	91.52	881,721	26,633	96.98	128,936	16	128,910
200	60/1200 = 5.0	91.52	882,063	26,726	96.97	129,061	16	129,035
400	60/1400 = 4.3	91.52	882,637	26,867	96.96	129,231	16	129,205
600	60/1600 = 3.8	91.52	882,960	26,961	96.95	129,346	16	129,320
800	60/1800 = 3.3	91.52	883,219	27,048	96.94	129,439	17	129,412

Table A.5
Effect of outliers: the alignment model ($k = 6$, $\theta = 30\%$)

N_{outlier}	N_{PatSeq}	Recoverability (%)	N_{item}	$N_{\text{extra}\,I}$	Precision (%)	N_{total}	N_{spur}	N_{redun}
0	1000	91.16	103	3	97.17	8	0	1
200	1000	91.16	100	2	98.04	8	0	1
400	1000	88.33	97	0	100	8	0	1
600	1000	82.87	92	0	100	8	0	1
800	1000	72.88	81	0	100	7	0	1

Table A.6
Effect of outliers: the alignment model ($k = 6$)

N_{outlier}	N_{PatSeq}	θ (%)	Recoverability (%)	N_{item}	$N_{\text{extra}I}$	Precision (%)	N_{total}	N_{spur}	N_{redun}
0	1000	30	91.16	103	3	97.17	8	0	1
200	1000	25	91.16	103	4	96.26	8	0	1
400	1000	22	91.16	103	6	94.50	8	0	1
600	1000	18	91.16	103	10	91.15	8	0	1
800	1000	17	91.16	103	11	90.35	8	0	1

References

[1] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: Proceedings of the International Conference on Very Large Databases (VLDB), 1994, pp. 487–499.

[2] R. Agrawal, R. Srikant, Mining sequential patterns, in: Proceedings of the International Conference on Data Engineering (ICDE), 1995, pp. 3–14.

[3] J. Ayres, J. Flannick, J. Gehrke, T. Yiu, Sequential pattern mining using a bitmap representation, in: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2002, pp. 429–435.

[4] V. Guralnik, G. Karypis, A Scalable Algorithm for Clustering Sequential Data, in: Proceedings of the International Conference on Data Mining (ICDM), 2001, pp. 179–186.

[5] R. Kohavi, C. Brodley, B. Frasca, L. Mason, Z. Zheng, KDD-CUP 2000 organizers' report: peeling the onion, in: Proceedings of the SIGKDD Explorations, vol. 2, 2000, pp. 86–98.

[6] H.C. Kum, J. Pei, W. Wang, D. Duncan, ApproxMAP: Approximate mining of consensus sequential patterns, in: Third SIAM International Conference on Data Mining (SDM), San Francisco, CA, 2003, pp. 311–315.

[7] H.C. Kum, S. Paulsen, W. Wang, Comparative study of sequential pattern mining models, in: IEEE ICDM Workshop on The Foundation of Data Mining and Discovery, Maebashi, Japan, December 2002.

[8] C.E. Metz, Basic principles of ROC analysis, in: Seminars in Nuclear Medicine, vol. 8, 1978, pp. 283–298.

[9] J. Pei, J. Han, et al., Mining sequential patterns by pattern-growth: the PrefixSpan approach, in: IEEE Trans. Knowl. Data Eng. 16(11), 2004, pp. 1424–1440.

[10] M. Spiliopoulou, Managing interesting rules in sequence mining, in: Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases, 1999, pp. 554–560.

[11] R. Srikant, R. Agrawal, Mining sequential patterns: generalizations and performance improvements, in: Proceedings of the 6th International Conference on Extending Database Technology (EDBT), March 1996, pp. 3–17.

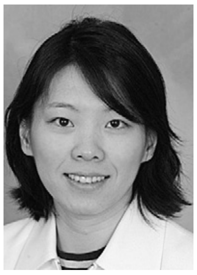
[12] X. Yan, J. Han, R. Afshar, CloSpan: mining closed sequential patterns in large datasets, in: Third SIAM International Conference on Data Mining (SDM), San Francisco. CA, 2003, pp. 166–177.

[13] J. Yang, P.S. Yu, W. Wang, J. Han, Mining long sequential patterns in a noisy environment, in: Proceedings of ACM International Conference on Management of Data (SIGMOD), Madison, WI, June 2002, pp. 406–417.

[14] M.J. Zaki, Efficient enumeration of frequent sequences, in: 7th International Conference Information and Knowledge Management, November 1998, pp. 68–75.

Hye-Chung Kum is an assistant research professor at the University of North Carolina at Chapel Hill. She received her M.S. and Ph.D. degree in Computer Science from the University of North Carolina at Chapel Hill (UNC-CH) in 1996 and 2004. Dr. Kum also minored in social work and received an MSW from UNC-CH in 1998. Her current research interests include data mining, mining data streams, data warehousing, and application of data mining and data warehousing technology for social welfare policy analysis and better program management.

Joong Hyuk Chang received the B.S. and M.S. degree in Computer Science from Yonsei University, Seoul, Korea, in 1996 and 1998, and also received the Ph.D. degree in Computer Science from Yonsei University in 2005. His current research interests include mining data streams, query processing over data streams (including data stream management systems), data mining and knowledge in large-scale data sets, anomaly intrusion detection, and bioinformatics.



Wei Wang is an assistant professor in the Department of Computer Science and a member of the Carolina Center for Genomic Sciences at the University of North Carolina at Chapel Hill. She received a M.S. degree from the State University of New York at Binghamton in 1995 and a Ph.D. degree in Computer Science from the University of California at Los Angeles in 1999. She was a research staff member at the IBM T.J. Watson Research Center between 1999 and 2002. Dr. Wang's research interests include data mining, bioinformatics, and databases. She has filed seven patents, and has published one monograph and more than 70 research papers in international journals and major peer-reviewed conference proceedings. Dr. Wang received the IBM Invention Achievement Awards in 2000 and 2001. She was the recipient of a UNC Junior Faculty Development Award in 2003 and an NSF Faculty Early Career Development (CAREER) Award in 2005. She was named a Microsoft Research New Faculty Fellow in 2005. Dr. Wang is an associate editor of the IEEE Transactions on Knowledge and Data Engineering and an editorial board member of the ACM Transactions on Knowledge Discovery in Data, Journal of Data Management, and International Journal of Data Mining and Bioinformatics. She serves on the program committees of prestigious international conferences such as ACM SIGMOD, ACM SIGKDD, VLDB, ICDE, EDBT, ACM CIKM, IEEE ICDM, and SSDBM.