Comparative Study of Sequential Pattern Mining Frameworks Support Framework vs. Multiple Alignment Framework

Hye-Chung (Monica) Kum Susan Paulsen Wei Wang
Computer Science, Univ. of North Carolina at Chapel Hill (kum, paulsen, weiwang@cs.unc.edu)

Abstract

Knowledge discovery and datamining (KDD) is commonly defined as the nontrivial process of finding interesting, novel and useful patterns from data. In this paper, we examine closely the problem of mining sequential patterns and propose a comprehensive evaluation method to assess the quality of the mined results. We propose four evaluation criteria, namely (1) recoverability, (2) the number of spurious patterns (3) the number of redundant patterns, and (4) the degree of extraneous items in the patterns, to quantitatively assess the quality of the mined result from a wide variety of synthetic datasets with varying randomness and noise levels. Recoverability, a new metric, measures how much of the underlying trend has been detected. Such an evaluation method provides a basis for comparing different frameworks for sequential pattern mining, which is very essential in understanding the performance of approximate solutions. In this paper, the method is employed to conduct a detailed comparison of the traditional frequent sequential pattern framework with an alternative approximate pattern framework based on sequence alignment. We demonstrate that the alternative approach is able to best recover the underlying patterns with little confounding information under all circumstances including those where the frequent sequential pattern framework fails.

1. Introduction

Knowledge discovery and datamining (KDD) is commonly defined as the nontrivial process of finding interesting, novel, and understandable patterns from data. In any particular datamining problem, the first and most important task is to define patterns operationally. The definition should ultimately lead to useful understandable patterns. Yet, designing a good definition (framework) and evaluating what patterns emerge from the framework is difficult.

In this paper, we focus on the problem of mining sequential patterns. Sequential pattern mining finds interesting patterns in sequence of sets. Mining sequential patterns has become an important datamining task with broad applications. For example, supermarkets often collect customer purchase records in sequence databases in which a sequential pattern would indicate a customer's buying habit.

Sequential pattern mining is commonly defined as finding the complete set of frequent subsequences in a set of sequences [1]. Much research has been done to efficiently find such patterns. But to the best of our knowledge, no research has examined in detail what patterns are actually generated from such a definition. In this paper, we examined the results of the support framework closely to evaluate whether it in fact generates interesting patterns.

To this end, we propose a comprehensive evaluation method that can quantitatively assess how "useful" and "understandable" the results are using the well known synthetic data generator [1]. This synthetic data generator has become a benchmark for evaluating performance. In this paper, we propose to extend this benchmark to evaluate the quality of the mined results. By mapping the mined patterns back to the base patterns (reported by the data generator) that generate the data, we are able to measure how well the methods find the real underlying patterns and whether or not it generates any confounding patterns under a variety of situations with varying randomness and noise levels.

Such a method provides a basis for comparing the results of different sequential pattern mining frameworks. In addition, the method is crucial in understanding the quality of the mined results because often times the frameworks are too complicated to analyze theoretically what patterns will emerge. The evaluation method provides a comprehensive empirical understanding of the results. This is especially important for understanding the performance of approximate solutions.

When the evaluation method was applied to the support framework for sequential pattern mining, it revealed that the framework generates huge number of redundant and spurious patterns in long sequences, which bury the true patterns. Our theoretical analysis of the expected support of short patterns in long sequences confirm that many short patterns can occur frequently simply by chance. Furthermore, in the presence of noise in the data, the support framework cannot detect the underlying patterns well because a sequence supports a pattern if, and only if, the pattern is fully contained in the sequence. Hence, noise in the data can cause the exact matching approach to miss general trends in the sequence database. Many customers may share similar buying habits, but few of them follow exactly the same buying patterns.

Motivated by these observations, we examined an entirely different framework for analyzing sequential data. What would be the proper framework to find the major groups of similar sequences in the database and then to uncover the underlying trend in each group? As a simple extension from association rule mining (mining patterns in sets), the conventional framework does not efficiently detect trends in sequences. However, detecting common underlying patterns (called consensus strings or motifs) in simple sequences (strings) has been well studied in computational biology. The current research employs the multiple alignment framework to detect consensus strings.

In the simple edit distance problem, one is trying to find an alignment of two sequences such that the edit distance is minimum. In multiple alignment, the purpose is to find an alignment over N strings such that the total pairwise edit distance for all N strings is minimal. A good alignment is one in which similar characters are lined up in the same column. In such an alignment, the concatenation of the common characters in each column would represent the underlying pattern. Table 1 shows an example of how the original word "pattern" was recovered from five typos.

In this paper, we extend the multiple alignment framework to sequences of sets and employ the evaluation method to understand what pattern are generated. In contrast to the support framework, the study reveals that the alignment framework, robust to both noise and random sequences in the data, returns a succinct but accurate summary of the base patterns with few spurious or redundant patterns and very low level of extraneous items.

Table 1. Multiple alignment of the word "pattern"

seq_1	P	A	T	T	T	E	R	N
seq_2	P	A			T	E	R	M
seq_3	P			T	T		R	N
seq_4	O	A		T	T	E	R	B
seq_5	P		S	Y	Y	R	T	N
Underlying pattern	**P**	**A**		**T**	**T**	**E**	**R**	**N**

In summary, we make the following contributions: (1) We design an evaluation method to assess the quality of the mined results in sequential data. (2) We propose a novel framework for sequential pattern mining, *multiple alignment sequential pattern mining*. (3) We employ the evaluation method for a comparative study of the support framework and the alignment framework. (4) We derive the expected support of patterns under the null hypothesis to better understand the behavior and limitations of the parameter *min_sup.*

The remainder of the paper is organized as follows. Sections 2 and 3 provide an overview of the two frameworks. Section 4 demonstrates both frameworks through an example. Section 5 details the evaluation method and the comparative study. It also includes the analysis of the expected support in random data. Section 6 provides an overview of related works. Section 7 concludes with a summary.

2. Support Framework

Table 2. An example of a sequence

Items $\boldsymbol{I}$	Itemsets s_{13}	Sequence seq_1
{A, B, C, D}	(C D)	< (A) (B) (C D) >

Definitions : Let *items* be a finite set of literals $\boldsymbol{I}$={i_1, i_2, i_3, …, i_r}. Then an *itemset* is a set of items from $\boldsymbol{I}$ and a *sequence* is an ordered list of itemsets. A database $\boldsymbol{D}$ is a set of such sequences. We denote a sequence seq_i as $<s_{i1}s_{i2}s_{i3}...s_{im}>$ (concatenation of itemsets) and an itemset s_{ij} as $(i_{j1}\ i_{j2}\ i_{j3}\ ...\ i_{jn})$ where items i_{jk} are from $\boldsymbol{I}$. (Table 2)

seq_2 is a supersequence of seq_1 and seq_1 is a subsequence of seq_2, if and only if seq_1 is derived by deleting some items or whole itemsets from seq_2. Given a sequence database $\boldsymbol{D}$, the support of a sequence *seq*, *sup(seq)*, is the number of supersequences of *seq* in the database.

Problem Statement: Given $\boldsymbol{N}$ sequences and a support threshold, *min_sup*, find all patterns *P* s.t. $sup(P) \geq min\_sup$.

The bottleneck in applying the support framework occurs when counting the support of all possible frequent subsequences in $\boldsymbol{D}$. Thus the two classes of algorithms differ in how to efficiently count support of potential patterns. The apriori based breadth-first algorithms [1, 11] pursue level-by-level candidate-generation-and-test pruning following the Apriori property: any super-pattern of an infrequent pattern cannot be frequent. In contrast, the projection based depth-first algorithms [2, 9, 13] avoid costly candidate-generation-and-test by growing long patterns from short ones. The depth first methods generally do better than the breadth first methods when the data can fit in memory. The advantage becomes more evident when the patterns are long [12].

3. Multiple Alignment Framework

In this section, we present a new framework for finding useful patterns in sequence of sets. The probability that two long sequences are similar is negligible. Thus, if some number of long sequences can be aligned such that certain items occur in certain positions frequently we are able to implicitly find sequential patterns that are statistically significant.

Definitions: The example in section 4 will help you understand the definitions better. The database, $\boldsymbol{D}$, are defined in the same way as in the support framework.

The *global multiple alignment* of a set of sequences is obtained by inserting a null itemset, ( ), either into or at the ends of the sequences such that each itemset in a sequence is lined up against a unique itemset or ( ) in all other sequences. In the rest of the paper, alignment will always refer to a global multiple alignment (Tables 6 and 7).

Given two aligned sequences and a distance function for itemsets, *the pairwise score* between the two sequences is the sum over all positions of the distance between an itemset in one sequence and the corresponding itemset in the other sequence. Given a multiple alignment of N sequences, the *multiple alignment score* is the sum of all pairwise scores. Then the optimum multiple alignment is one in which the multiple alignment score is minimal [6].

$$PS\ (seq_i, seq_j) = \sum distance(s_{ik}, s_{jk})\ \ \text{(for all k)}$$

$$MS\ (N) = \sum PS(seq_i, seq_j)\ \text{(over all } 1 \leq i \leq N \text{ and } 1 \leq j \leq N)$$

Weighted sequences are an effective method to compress a set of aligned sequences into one sequence. A *weighted itemset*, denoted as $ws_j=(i_{j1}:w_{j1},\ ...,\ i_{jm}:w_{jm}):v_j$, is defined as an itemset that has a weight associated with each item in the itemset as well as the itemset itself. Then a *weighted sequence*, denoted as $wseq_i=<(i_{11}:w_{11},\ ...,\ i_{1s}:w_{1s}):v_1\ ...\ (i_{l1}:w_{l1},\ ...,\ i_{lt}:w_{lt}):v_l>:n$, is a sequence of weighted itemsets paired with a separate weight for the whole sequence. The weight associated with the weighted sequence, *n*, is the total number of sequences in the set. The weight associated with the itemset s_j, v_j, represents how many sequences have a nonempty itemset in position *j*. And the weight associated with each item i_{jk} in itemset s_j, w_{jk}, represents the total number of the item i_{jk} present in all itemsets in the aligned position *j* (Tables 6 & 7).

The *strength* of an item, i_{jk}, in an alignment is defined as the percentage of sequences in the alignment that have item i_{jk} present in the aligned position *j*. strength(i_{jk}) = w_{jk}/n * 100%. Clearly, larger strength value indicates that more sequences share the item in the same aligned position.

Given a threshold, θ, and a multiple alignment of $\boldsymbol{N}$ sequences, the *consensus itemset* for position *j* in the alignment is an itemset of all items that occur in at least θ sequences in position *j*. Then a *consensus sequence* is simply a concatenation of the consensus itemsets for all positions excluding any null consensus itemsets. When weights are included it is called a *weighted consensus sequence*.

$$\text{Consensus itemset (j)} = \{\ i_k \mid \forall\ i_k \in \boldsymbol{I} \text{ and strength}(i_k) \geq \theta\ \}$$

Based on item strengths, items in an alignment are divided into three groups: rare items, non-frequent items, and frequent items. The rare items may represent noise and are in most cases not of any interest to the user. The frequent items occur in enough of the sequences to constitute the underlying pattern in the group. The non-frequent items do not occur frequently enough to be part of the underlying pattern but occur in enough sequences to be of interest. The non-frequent items constitute variations on the general pattern. That is, they are the items most likely to occur regularly in a subgroup of the sequences. Using this categorization we make the final results more understandable by defining two types of consensus sequences corresponding to two thresholds: (1) The pattern consensus sequence, which is composed solely of frequent items and (2) the variation consensus sequence, an expansion of the pattern consensus sequence to include non-frequent items (Table 8). This method presents both the frequent underlying patterns and their variations while ignoring the noise. It is an effective method to summarize the alignment because the user can clearly understand what information is being dropped. Furthermore, the user can control the level of summarization by defining the two thresholds for frequent and rare items as desired.

Problem Statement: Given ***N*** sequences, a distance function for itemsets, and strength thresholds for consensus sequences (users can specify different thresholds for each set), the problem of *multiple alignment sequential pattern mining* is (1) to partition the ***N*** sequences into ***K*** sets of sequences such that the sum of the ***K*** multiple alignment scores is minimum, (2) to find the optimal multiple alignment for each partition, and (3) to find the pattern consensus sequence and the variation consensus sequence for each partition.

Table 3 : Inter itemset distance metric replace

Replace (A, B) =	$\frac{[\text{size}(A)+\text{size}(B) - 2*\text{size}(A \cap B)]}{[\text{size}(A) + \text{size}(B)]}$
$\text{INDEL}_R(A)$ = Replace (A,∅) = 1	0 ≤ Replace ≤ 1

The exact solution to multiple alignment pattern mining is NP-hard, and therefore too expensive to be practical. An efficient approximation algorithm, ApproxMAP (APPROXimate Multiple Alignment Pattern mining), has three steps. First, *k* nearest neighbor clustering is used to approximately 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 *k-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 *k-1* sequences. Third, based on user-defined thresholds the weighted sequence of each partition is used to generate two consensus sequences per partition, the pattern and the variation consensus sequences. To further reduce the data presented to the user, a simple gamma-corrected color-coding scheme is used to represent the item strengths in the patterns [7].

ApproxMAP defines the distance function, *Replace()*, for itemsets (Table 3). *Replace*() is mathematically equivalent to the Sørensen coefficient, an index similar to the Jaccard coefficient except that it gives more weight to the common elements [7]. Thus, it is more appropriate if the commonalities are more important than the differences.

Multiple alignment pattern mining has many practical applications. It is a versatile exploratory data analysis tool for sequential data, because it organizes and summarizes the high dimensional data into something that is viewable by people. Multiple alignment pattern mining summarizes interesting aspects of the data as follows: (1) The partitioning through *k* nearest neighbor clustering and subsequent multiple alignment within a cluster organizes the sequences, (2) the weighted sequences provide a compressed expression of the full database, and (3) the weighted consensus sequences provides a summary of each cluster's pattern at a user specified level. In addition, given the appropriate threshold, consensus sequences are patterns that are approximately similar to many sequences in the database. That is they are *approximate sequential patterns* based on approximate support, defined as *asup(seq)* =|{*seq'*| *seq'*∈ ***D*** & *dist(seq,seq')* ≤ *min_dist*}| [7]. Note that once users have found interesting patterns, they can use the more efficient pattern search methods to do confirmatory data analysis.

Multiple alignment pattern mining is also an effective method for clustering similar sequences. This has the follow-

Table 4. Sequence database

ID	Sequences				
seq1	< (A)	(B C Y)	(D) >		
seq2	< (A)	(X)	(B C)	(A E)	(Z) >
seq3	< (A I)	(Z)	(K)	(L M) >	
seq4	< (A L)	(D E) >			
seq5	< (I J)	(B)	(K)	(L) >	
seq6	< (I J)	(L M) >			
seq7	< (I J)	(K)	(J K)	(L)	(M) >
seq8	< (I M)	(K)	(K M)	(L M) >	
seq9	< (J)	(K)	(L M) >		
seq10	< (V)	(K W)	(Z) >		

Table 5. support=20%

id	pattern	sup	id	pattern	sup
1	(A)	4	24	(I) (L,M)	3
2	(B)	3	25	(J) (K)	3
3	(C)	2	26	(J) (L)	4
4	(D)	2	27	(J) (M)	3
5	(E)	2	**28**	**(J) (L,M)**	**2**
6	(I)	5	29	(K) (K)	2
7	(J)	4	30	(K) (L)	5
8	(K)	6	31	(K) (M)	4
9	(L)	7	32	(K) (L,M)	3
10	(M)	5	33	(I,J) (K)	2
11	(Z)	3	34	(I,J) (L)	3
12	(B,C)	2	**35**	**(I,J) (M)**	**2**
13	(I,J)	2	36	(I) (K) (K)	2
14	(L,M)	2	37	(I) (K) (L)	2
15	(A) (B)	2	38	(I) (K) (M)	2
16	(A) (C)	2	**39**	**(I) (K) (L,M)**	**2**
17	**(A) (D)**	**2**	40	(J) (K) (L)	2
18	**(A) (E)**	**2**	**41**	**(J) (K) (M)**	**2**
19	**(A) (Z)**	**2**	42	(K) (K) (L)	2
20	**(A) (B,C)**	**2**	43	(K) (K) (M)	2
21	(I) (K)	4	**44**	**(I,J) (K) (L)**	**2**
22	(I) (L)	5	**45**	**(I) (K) (K) (L)**	**2**
23	(I) (M)	4	**46**	**(I) (K) (K) (M)**	**2**

Table 6. Cluster 1 (min_strength = 40% = 1.2 < 2 sequences)

seq1	(A)	()	(B, C, Y)	(D)	()	
seq4	(A, L)	()	()	(D, E)	()	
seq2	(A)	(X)	(B, C)	(A, E)	(Z)	
Weighted sequence	(A:3, L:1):3	(X:1):1	(B:2, C:2, Y:1):2	(A:1, D:2, E:2):3	(Z:1):1	3
Consensus seq (w ≥ 2)	*(A)*		*(B, C)*	*(D, E)*		
Wgt Consensus seq (w ≥ 2)	*(A:3):3*		*(B:2, C:2):2*	*(D:2, E:2):3*		*3*

Table 7. Cluster 2 (min_strength = 40% = 2.8 < 3 sequences)

seq_9	(J)	()	(K)	(L, M)	()	
seq_5	(I, J)	(B)	(K)	(L)	()	
seq_3	(A,I)	(Z)	(K)	(L, M)	()	
seq_7	(I, J)	(K)	(J, K)	(L)	(M)	
seq_8	(I, M)	(K)	(K, M)	(L, M)	()	
seq_6	(I, J)	()	()	(L, M)	()	
seq_{10}	()	(V)	(K, W)	()	(Z)	
Weighted seq	(A:1,I:5,J:4,M:1):6	(B:1,K:2,V:1,Z:1):5	(J:1,K:6,M:1,W:1):6	(L:6,M:4):6	(M:1,Z:1):2	7
Con. seq(w≥3)	*(I, J)*		*(K)*	*(L, M)*		
Wgt Con. seq	*(I:5, J:4):6*		*(K:6):6*	*(L:6, M:4):6*		*7*

Table 8. Consensus sequences (100%: 85%: 70%: 50%: 35%: 20%)

Pattern Consensus Seq 1	support = 40% = 1.2 < 2 sequences	**(A) (B, C) (D, E)**
Variation Consensus Seq 1	Not appropriate in this small set	
Pattern Consensus Seq 2	support = 40% = 2.8 < 3 sequences	**(I, J) (K) (L, M)**
Variation Consensus Seq 2	support = 20% = 1.4 < 2 sequences	**(I, J) (K) (K) (L, M)**

ing benefits. First, it enables multiple alignment on a group of necessarily similar sequences to find the underlying pattern (consensus sequence). Second, once sequences are grouped the user can specify the threshold, *min_strength*, specific to each group. This is in contrast to the support framework in which *min_sup* is specified against the whole database. Frequent uninteresting patterns, which are usually grouped into large partitions, will have a higher threshold than infrequent interesting patterns, which tend to be grouped into small partitions. Thus, frequent uninteresting patterns will not flood the results as they do in the support framework.

4. Example

Table 4 is a sequence database ***D***. Although the data is lexically sorted it is difficult to gather much information from the raw data even in this tiny example.

Table 5 is the result from the support framework. For better readability, we show the maximal sequential patterns in bolded Helvetica font. Note that finding the maximal patterns automatically in sequential data is a non-trivial task.

The ability to view Table 4 is immensely improved by using the alignment framework – grouping similar sequences then lining them up and coloring the consensus sequences as in Tables 6 through 8. Note that the patterns <(A)(BC)(DE)> and <(IJ)(K)(LM)> do not match any sequence exactly.

Given the input data shown in Table 4 (N=10 sequences), ApproxMAP (1) calculates the N*N sequence to sequence distance matrix from the data, (2) partitions the data into two clusters (k=2), (3) aligns the sequences in each cluster (Tables 6 and 7) – the alignment compresses all the sequences in each cluster into one weighted sequence per cluster, and (4) summarizes the weighted sequences (Tables 6 and 7) into weighted consensus sequences (Table 8).

5. Evaluation

We present a method that objectively evaluates the quality of the results produced by any sequential pattern mining method when applied to the output of a well-known synthetic data generator [1]. The evaluation method is a matrix of four experiments – (1) random data, (2) pattern data, and pattern data with (3) varying degree of noise, and (4) varying number of random sequences – assessed on four criteria: (1) recoverability, (2) the number of spurious patterns, (3) the number of redundant patterns, and (4) the level of extraneous items in the result. Recoverability, defined in section 5.2, provides a good estimate of how well the underlying trends in the data are detected. This evaluation method will enable researchers not only to use the data generator to benchmark performance but also to quantify the quality of the results. Such benchmarking will become increasingly important as more datamining methods focus on approximate solutions.

Using this method, we compare the quality of the results from the support framework and the alignment framework. Note that all methods generate the same results for the support framework. In contrast, the exact solution to the multiple alignment pattern mining is NP-hard, and in practice all solutions are approximate. Consequently the results of the alignment framework are method dependent. We use the results of ApproxMAP to represent the alignment framework.

5.1. Synthetic Data

Given several parameters (Table 9), the data generator given in [1] produces the database and reports the base patterns used to generate it. The data is generated in two steps. First, it generates N_{pat} potentially frequent sequential patterns, called base patterns, according to L_{pat} and I_{pat}. Second, each sequence in the database is built by combining base patterns until the size required, determined by L_{seq} and I_{seq}, is met. Along with each base pattern, the data generator reports the expected frequency, $E(F_B)$, and the expected length (total number of items), $E(L_B)$, of the base pattern in the database. 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.

Random data is generated by assuming independence between items both within and across itemsets. The probability of an item occurring is uniformly distributed.

Table 9. Parameters of the synthetic database

N_{seq}	1000	total # of sequences in the DB
L_{seq}	10	avg # of itemsets per sequence in the DB
I_{seq}	2.5	avg # of items per itemset in the DB
N_{pat}	10	total # of base patterns embedded in the DB
L_{pat}	7	avg # of itemsets per base pattern
I_{pat}	2	avg # of items per itemset in the pattern
N_{items}	500	total # of unique items in the DB

5.2. Evaluation Criteria

The effectiveness of a framework can be evaluated in terms of how well it finds the real underlying patterns in the database (the base patterns), and whether or not it generates any confounding information. To the best of our knowledge, no previous study has measured how well the various methods recover the known base patterns in the data generator in [1]. In this section, we propose a new measure, *recoverability*, to evaluate the match between the base patterns and the result patterns. Expanding it, we propose a vector of four criteria – (1) recoverability, (2) the number of spurious patterns, (3) the number of redundant patterns, and (4) the level of extraneous items in the result patterns – which can comprehensively depict the quality of the result patterns.

In order to measure these four criteria, we match the resulting patterns from each method to the most similar base pattern. In the multiple alignment framework, only the pattern consensus sequences are considered. That is, the result pattern, P, is matched with the base pattern, B, if the longest common subsequence between P and B, denoted as $B \otimes P$, is the maximum over all base patterns. The items shared by P and B, $B \otimes P$, are defined as *pattern items*. The remaining items in P are defined as *extraneous items*. The overall *level of extraneous items* in the set of result patterns is the total number of extraneous items over the total number of all items in the full set of result patterns.

Depending on the mix of pattern and extraneous items, the result patterns are categorized into spurious, max, or redundant patterns. *Spurious patterns* are result patterns with more extraneous items than pattern items. Of the remaining, *max patterns* are result patterns whose $\|B \otimes P\|$ is the longest over all result patterns for a given base pattern. Note that the number of max patterns is equal to the number of base patterns detected. The rest of the result patterns, P_1, are *redundant patterns* as there exists a max pattern, P_2, that match with the same base pattern but better. For the alignment framework, we include the number of clusters that generated no patterns as null patterns.

Next we measure how well a framework detects the underlying base pattern by evaluating the quality of the max patterns. 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 strong (frequent) the patterns are in the data. The base patterns are only potentially frequent sequential patterns. The actual occurrence of a base pattern in the data, which is controlled by $E(F_B)$ and $E(L_B)$, varies widely. $E(F_B)$ is exponentially distributed then normalized to sum to 1. Thus, some base patterns with tiny $E(F_B)$ will not exist in the data or occur very rarely. Recovering these patterns are not as crucial as recovering the more frequent base patterns. $E(L_B)$ controls how many items in the base patterns, on average, are injected into one occurrence of the base pattern in a sequence. $E(L_B)$ is normally distributed.

We designed a weighted measure, *recoverability*, which can more accurately evaluate how well the base patterns have been recovered. Specifically, given a set of base patterns, B, and a set of result patterns, P, we define the recoverability as,

$$R = \sum_{\text{base pat } B} E(F_B) \cdot \min \left\{ \begin{array}{c} 1 \\ \left(\frac{\max_{\text{rslt pat } P}\{\|B \otimes P\|\}}{E(L_B)} \right) \end{array} \right.$$

$\max\{\|B\otimes P\|\}$ is the number of items in the max pattern shared with the base pattern B. Since $E(L_B)$ is an expected value, sometimes the actual observed value, $\max\{\|B\otimes P\|\}$ is greater than $E(L_B)$. In such cases, we truncate the value of $\max\{\|B\otimes P\|\}/E(L_B)$ to one so that recoverability stays between 0 and 1. When recoverability of the mining is high, major portions of the base patterns have been found.

In summary, a good framework would produce high recoverability with small numbers of spurious and redundant patterns and a low level of extraneous items.

5.3. Spurious patterns in random datasets

5.3.1. Analysis of expected support on random data

Implicit in the use of a minimum support for identifying frequent sequential patterns is the desire to distinguish true patterns from those that appear by chance. Yet many subsequences will occur frequently by chance, particularly if the subsequence is short, the data sequence is long, and the items appear frequently in the database. When *min_sup* is low, the support of these sequences can exceed *min_sup* simply by chance. Conversely, an arbitrarily large *min_sup* may miss rare but statistically significant patterns.

We can derive the expected support, *E{sup(seq)}*, for a subsequence under the statistical null hypothesis that the probability of an item appearing at a particular position in the sequence is independent of both its position in the sequence and the appearance of other items in the sequence. The expected support (measured as a percentage) for a subsequence under our null hypothesis is $\Pr\{seq_i$ appears at least once$\}=1-\Pr\{seq_i$ never appears$\}$.

First let's consider simple sequences, strings. Then the probability that A will appear at least one time in a string of length L, where $p(A)$ is the probability of item A appearing in the sequence at any particular position, is

$$E(\text{ sup }(A)) = 1- \Pr\{A \text{ never appears}\} = 1- [\,1- p(\text{A})\,]^{L} \qquad (1)$$

Now consider the $\Pr\{A..B\}$ where A and B arbitrarily represent two items, which need not be distinct. To calculate $\Pr\{\sim(A..B)\}$, i.e. the probability of never seeing an A followed by a B, divide all possible outcomes into $L+1$ mutually exclusive sets: A first appears in position j, denoted as first(A, j), for $j=1..L$ and A never appears. Then,

$$\Pr\{\text{first}(A, j)\}=\Pr\{A \text{ first appears in pos } j\}=[1- p(A)]^{j-1}p(A) \quad (2)$$

Next, we condition the probability of $\sim(A..B)$ as follows:

$$\Pr\{\sim(A..B)\}=\sum \Pr\{\sim(A..B)|\text{ first}(A, j)\} \cdot \Pr\{\text{first}(A, j)\} + \Pr\{\sim(A..B)|\, A \text{ never appears}\} \cdot \Pr\{A \text{ never appears}\} \quad (3)$$

for $j=1..L$. Where in general,

$$\Pr\{\sim(A..B)|\text{ first}(A, j)\} = \Pr\{\sim(A..B)|\, A \text{ first appears in pos } j\} \quad (4)$$
$$=\Pr\{B \text{ never appears in a sequence of length } L\text{-}j\} = [1\text{-}p(B)]^{L-j}$$

By substituting the proper equations into (3) and recognizing that $\Pr\{\sim(A..B)|\, A$ never appears$\}= 1$ we have

$$\Pr\{\sim (A..B)\} = p(A)\sum_{j=1}^{L} \left\{[1 - p(B)]^{L-j}[1 - p(A)]^{j-1}\right\}+[1 - p(A)]^{L}$$

Then, $E(\text{ sup }(AB)) = 1\text{-}\Pr\{\sim(A..B)\}$.

Due to space limitations, the extensions are discussed briefly. First, the expected support of longer subsequences can be calculated recursively. Second, the expected support of sequences of sets can be calculated by noting that the probability of seeing a set of n arbitrary items in the same itemset is $p(i_1)*p(i_2)\ldots*p(i_n)$. Then we can derive the expected support by substituting the probability of an arbitrary element, *p(A),* with $p(i_1)*p(i_2)\ldots*p(i_n)$, in any of the equations above.

The quantities $p(A)$, $p(B)$, etc., are not known, however it is beyond the scope of this paper to evaluate its distribution. We merely propose as a first step that the choice of *min_sup* as well as the interpretation of the results should be guided by $E(sup(seq_i))$. In Frigure 1, we calculate the expected support of <(A)(B)> with respect to L. The expected support grows linearly with respect to L.

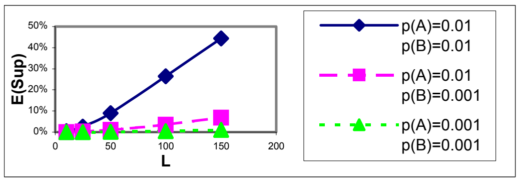

Figure 1. E(sup) w.r.t. L

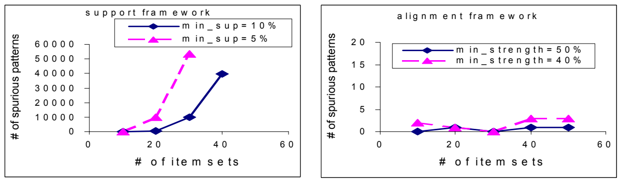

Figure 2. # of spurious patterns in random data

5.3.2. Empirical analysis

We generated random data sets with parameters N_{seq}, I_{seq}, N_{items} as given in Table 9 and varied L_{seq} to test empirically how many spurious patterns are generated from random data.

The support framework has no mechanism to eliminate patterns that occur simply by chance. As seen in the theoretical analysis, when sequences are long, short patterns can occur frequently simply by chance. Consequently, support alone cannot distinguish between significant patterns and random sequences. To combat this problem in association rule mining, [4] has used correlation to find statistically significant associations. Unfortunately, the concept of correlation does not extend easily to sequences.

Consistent with the theoretical analysis, the support framework generates many spurious patterns given random data (Figure 2). The number of spurious patterns increases exponentially with respect to L_{seq}. This follows naturally from Figure 1 which shows that *E(sup)* of length 2 sequences increase linearly with respect to L. Thus, the total of all patterns, $1\leq L_{pat}\leq \max_L$, should grow exponentially. When *min_sup*=

5% and L_{seq}=30, there are already 53,571 spurious patterns. In real applications, since *min_sup*<<5%, and N_{seq}> 30, there will be many spurious patterns mixed in with true patterns.

In contrast, the probability of a group of random sequences aligning well enough to generate a consensus sequence is negligible. Thus, using default values, the multiple alignment framework found few spurious patterns – from 0 to 3 (Figure 2). Furthermore, all the patterns found have only one item, which can easily be dismissed. Although the algorithm generated many clusters (80 to 100), all the clusters were either very small, or not enough sequences in the cluster could be aligned to generate meaningful consensus itemsets.

5.4. Baseline study of patterned database

Table 10. Pattern data

	parameters	recover	noise	max	spur	redundant
Sup framework	Min_sup=5%	91.6%	5.2%	10	6,648	247,266
Align framework	k=6&Min_strgh=30%	92.5%	0%	8	0	2

This experiment serves several purposes. First, it evaluates how well the frameworks detect the underlying patterns in a simple patterned dataset. Second, it illustrates how readily the results may be understood. Third, it establishes a baseline for the remaining experiments. We generate 1000 sequences from 10 base patterns (Table 9). We tuned both frameworks to the optimal parameters.

The recoverability for both frameworks was good at over 90% (Table 10). However, in the support framework it is difficult to extract the 10 base patterns from results that include 247,266 redundant patterns and 6,648 spurious patterns. Furthermore, there were 93,043 extraneous items in total, which constituted 5.2% of all items in the result patterns.

In comparison, the alignment framework returned a very succinct but accurate summary of the base patterns. There were only two redundant patterns, no spurious patterns, and no extraneous items. Table 11 shows all the pattern consensus sequences, *$PatConSeq_i$*, the variation consensus sequences, *$VarConSeq_i$*, with the matching base patterns, *$BaseP_i$*. The two weak base patterns missed are at the bottom. In this small dataset, manual inspection clearly shows how well the consensus sequences match the base patterns used to generate the data. Each consensus pattern found was a subsequence of considerable length of a base pattern. The 20 consensus sequences provide a good overview of the 1000 data sequences.

Table 11. Patterns detected when k=6; PatSup=30%, VarSup=20%, 5 seq ≤ DB support ≤ 100 seq

$BaseP_i$ (E(freq):E(len))	LEN	**Pattern <100: 85: 70: 50: 35: 20>**
$PatConSeq_1$	13	< (15 16 17 66) (15) (58 99) (2 74) (31 76) (66) (62) >
$VarConSeq_1$	18	< (15 16 17 66) (15 22) (58 99) (2 74) (24 31 76) (24 66) (50 62) (93) >
$BaseP_1$ (0.21:0.66)	14	[15,16,17,66] [15] [58,99] [2,74] [31,76] [66] [62] [93]
$PatConSeq_2$	9	< (16) (99) (94) (45 67) (50) (96) (51) (66) >
$VarConSeq_2$	13	< (16 22) (29 99) (94) (22) (45 67) (50) (96) (51) (66) (15) >
$PatConSeq_3$	13	< (22 50 66) (16) (29 99) (94) (45 67) (12 28 36) (50) >
$VarConSeq_3$	13	< (22 50 66) (16) (29 99) (94) (45 67) (12 28 36) (50) >
$PatConSeq_4$	19	<(22 50 66)(16)(29 99)(94)(45 67)(12 28 36)(50)(96)(51)(66)(2 22 58) >
$VarConSeq_4$	21	<(22 50 66)(16)(29 99)(58 94)(2 45 67)(12 28 36)(50)(96)(51)(66)(2 22 58)>
$BaseP_2$ (0.161:0.83)	22	[22,50,66][16][29,99][94][45,67][12,28,36][50][96][51][66][2,22,58][63,74,99]
$PatConSeq_5$	11	< (22) (22) (58) (2 16 24 63) (24 65 93) (6) >
$VarConSeq_5$	15	< (22) (22) (22) (58) (2 16 24 63) (24 65 93) (6) (11 15 74) >
$BaseP_3$ (0.141:0.82)	14	[22] [22] [58] [2,16,24,63] [24,65,93] [6] [11,15,74]
$PatConSeq_6$	11	< (31 76) (58 66) (16 22 30) (16) (50 62 66) >
$VarConSeq_6$	11	< (31 76) (58 66) (16 22 30) (16) (50 62 66) >
$BaseP_4$ (0.131:0.90)	15	[31,76] [58,66] [16,22,30] [16] [50,62,66] [2,16,24,63]
$PatConSeq_7$	13	< (43) (2 28 73) (96) (95) (2 74) (5) (2) (24 63) (20) >
$VarConSeq_7$	16	< (22 43) (2 28 73) (58 96) (95) (2 74) (5) (2 66) (24 63) (20) >
$BaseP_5$ (0.123:0.81)	14	[43] [2,28,73] [96] [95] [2,74] [5] [2] [24,63] [20] [93]
$PatConSeq_8$	8	< (63) (16) (2 22) (24) (22 50 66) >
$VarConSeq_8$	9	< (63) (16) (2 22) (24) (22 50 66) (50) >
$BaseP_6$ (0.121:0.77)	9	[63] [16] [2,22] [24] [22,50,66] [50]
$PatConSeq_9$	11	< (70) (58 66) (22) (74) (22 41) (2 74) (31 76) >
$VarConSeq_9$	16	< (70) (58 66) (22) (22 58) (74) (22 41) (2 66 74) (31 76) (2 74) >
$BaseP_7$ (0.054:0.60)	13	[70][58,66][22][74][22,41][2,74][31,76][2,74]
$PatConSeq_{10}$	15	< (20 22 23 96) (50) (51 63) (58) (16) (2 22) (50) (23 26 36) >
$VarConSeq_{10}$	8	< (20 22 23 31 76 96) (50 66) (51 63) (58) (16) (2 22) (50) (23 26 36) >
$BaseP_8$ (0.014:0.91)	17	[20,22,23,96][50][51,63][58][16][2,22][50][23,26,36][10,74]
$BaseP_9$ (0.038:0.78)	7	[88][24,58,78][22][58][96]
$BaseP_{10}$ (0.008:0.66)	17	[16][2,23,74,88][24,63][20,96][91][40,62][15][40][29,40,99]

Table 12. Consensus pattern detected from the systematic interaction of two base patterns

$BaseP_1$	(15 16 17 66)	(15)	(58 99)	(2 74)	(31 76)	(66)	(62)	(93)
ConSeq	(15 16 17 22 66)	(2 15 22)	(22 58 99)	(2 58 74)	(2 16 24 31 63 74 76)	(24 65 66 93)	(6 62)	
$BaseP_3$	(22)	(22)		(58)	(2 16 24 63)	(24 65 93)	(6)	(11 15 74)

Many of the redundant patterns in the support framework are either subsequences of a longer pattern or a small variation on it. They hold no additional information and instead bury the real patterns. Although, finding max or closed frequent itemset has been researched [3, 8], these methods to not extend easily to sequences.

In contrast, the two redundant patterns in the alignment framework have useful information. Sequences in the partition that generated *PatConSeq*$_3$ (Table 11) were separated out from the partition that generated *PatConSeq*$_4$ because they were missing most of the six items at the end of *PatConSeq*$_4$. Similar information can be gathered from the pattern consensus sequence *PatConSeq*$_2$. Thus, when the redundant patterns in the alignment framework are subsequences of a longer pattern, it alerts the user that there is a significant group of sequences that do not contain the items in the longer pattern.

5.5. Robustness with respect to noise

We evaluate the robustness of the frameworks with respect to varying degrees of noise in the data. Noise is introduced into the patterned data (section 5.4) using a corruption probability α. Items in the database are randomly changed into another item or deleted with probability α. This implies that 1-α is the probability of any item remaining constant. Hence, when α=0 no items are changed, and higher values of α imply a higher level of noise [12].

Table 13. Effect of noise

parameters	1-a	recover	noise	max	spurious	redundant	null
Min_sup = 3% (30 seqs)	5%	88.50%	10.10%	10	12,733	257,792	na
	15%	71.50%	18.00%	10	3,250	50,320	na
	25%	55.80%	20.50%	10	970	15,375	na
	35%	43.30%	24.60%	10	512	7,425	na
k=2 Min strgth =30%	5%	92.40%	10.10%	8	1	7	18
	15%	88.30%	9.50%	8	0	6	14
	25%	81.30%	3.60%	7	0	8	31
	35%	77.30%	4.60%	8	0	10	44

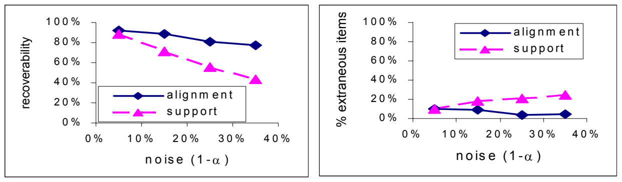

Figure 3. Effect of noise

The results show that the support framework is vulnerable to random noise injected into the data. As seen in Table 13 and Figure 3, as the corruption factor, α, increases, the support framework detects less of the base patterns and incorporates more extraneous items in the patterns. When the corruption factor is 35%, the recoverability degrades significantly to 43% even when *min_sup* has been reduced to 3%=30 sequences. Such results are expected since the framework is based on exact match. Note that even with recoverability at 43%, the framework returns 7,947 patterns that have 24.6% extraneous items in them.

In comparison, the alignment framework is robust to noise in the data. Despite the presence of noise, it is still able to detect a considerable number of the base patterns (i.e. recoverability is 77.3% when corruption factor is 35%). Furthermore, although the number of extraneous items in the consensus sequence has increased relative to the data without noise, the number of extraneous items is not directly related to the level of noise injected (Table 13 and Figure 3). This is because the random noise injected into the data does not correspond to the extraneous items in the consensus sequences. Rather, the extraneous items in the consensus sequences tend to be segments of a second base pattern. In the presence of noise, a smaller cluster is more likely to form around sequences that were constructed from a systematic interaction between two base patterns. In such cases, a new base pattern highly shared by a small group of sequences emerges and ApproxMAP detects it. The experiment with 1-α = 5% had 20 extraneous items in total of which 14 came from the consensus pattern, mapped to BaseP$_1$, given in Table 12. A closer examination reveals that when the extraneous items (in gray) are put together, the new pattern is highly similar to a second base pattern, BaseP$_3$. In fact, of the 14 extraneous items, only three are true extraneous items and 11 are from BaseP$_3$ (darker gray). Thus, the true level of extraneous items in the result is 9/198=4.5%. The simple model of matching a consensus pattern to only one base pattern results in items that belong to the second base pattern being categorized as extraneous items, when in fact they are not.

5.6. Robustness w.r.t. random sequences

This experiment is designed to test the effect of random sequence added to patterned data. We generated random data as in the first experiment with the same parameters as the patterned data in the second experiment and added them together. The main effect of the random sequences are the weakening of the patterns as a percentage of the database.

Consequently, in the support framework the recoverability and the number of spurious and redundant patterns are reduced when *min_sup* is maintained at 5%. Level of extraneous items in the patterns increase as we add more random sequences. On the other hand, if we maintain *min_sup*=50 sequences, obviously the recoverability can be maintained. The tradeoff is that spurious patterns, redundant patterns, and extraneous items all increase slightly (Table 14 and Figure 4).

In ApproxMAP, the parameter that controls which patterns are detected is *k* in the kNN clustering step. The user input parameter *k*, can be adjusted to control the resolution of clustering. Since *k* defines the neighbor space, larger *k* values will tend to merge more points (resulting in a smaller number of large clusters) while smaller values of *k* will tend to break up clusters. Thus, *k* controls at what level of detail the data is partitioned. The benefit of using a small *k* is that ApproxMAP can detect patterns that occur rarely. The cost is that it will break up clusters representing frequent patterns to generate multiple consensus sequences that are similar.

In this experiment, when *k* is maintained at 6, the added random sequences had no effect on the clusters formed or how the patterned sequence were aligned. Each cluster just picks up various amounts of the random sequences which are aligned after all the patterned sequences are aligned. In effect, the random sequences are ignored when it cannot be aligned with the patterned sequences in the cluster. The rest of the random sequences formed separate clusters that generated no patterns, as in the first experiment. Nonetheless, the consensus sequences were shorter, reducing recoverability, since the random sequences increased the cluster size, and thus weakened the signature in the cluster (Figure 4).

However, as seen in Table 15 and Figure 4, we can easily find the longer underlying patterns by adjusting either *min_strength* or *k* to compensate for the random sequences in the data. The *min_strength* can be lowered to pick up more patterned items. The tradeoff would be that more extraneous items could be picked up as well. A better adjustment would be to use a smaller *k* value to pick up the less frequent patterns. In this experiment, the small adjustment in *k* from 6 to 3 does not breakup any of the existing clusters. Instead, it just creates many more small clusters of no patterns. The increase in number of spurious and redundant patterns and the percentage of extraneous items are negligible.

Table 14. Support framework results

Rand Seq	Min_sup	Recover	Noise	Max	Spurious	Redundant
250	63 seq (5%)	86.20%	4.10%	10	2,615	113,375
500	75 seq (5%)	80.30%	3.90%	10	1,387	61,123
750	88 seq (5%)	74.90%	4.70%	10	861	30,572
1000	100 seq (5%)	68.20%	6.30%	10	699	16,893
250	50 seq (4%)	91.59%	5.22%	10	6,653	247,167
500	50 seq (3.30%)	91.59%	5.22%	10	6,658	247,222
750	50 seq (2.90%)	91.59%	5.23%	10	6,670	247,313
1000	50 seq (2.50%)	91.59%	5.24%	10	6,691	247,765

Table 15. Mulitple alignment framework results

Rand Seq	k	Min_stgth	Recover	Noise	Max	Spur	Redundant	Null
250	6	20%	81.90%	1/98	8	0	1	1
500	6	20%	82.00%	3/101	8	0	1	6
750	6	20%	79.60%	0/96	8	0	1	7
1000	6	20%	79.60%	0/93	7	0	1	12
250	3	30%	92.03%	1/127	8	0	3	3
500	3	30%	91.94%	1/125	8	0	3	15
750	3	30%	91.94%	2/127	9	1	2	29
1000	3	30%	90.61%	2/126	9	1	2	70

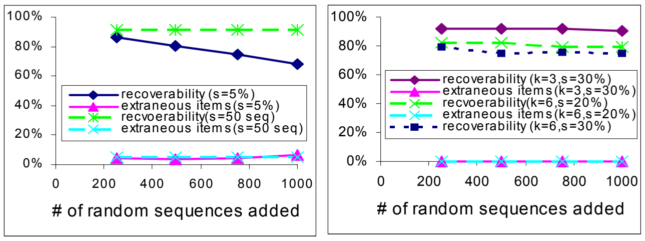

Figure 4. Effect of random sequences

5.7. Scalability

Methods based on the support framework has to build patterns one at a time. Thus, the lower bound on the time complexity with respect to the patterns is exponential. In contrast, the methods based on the alignment framework find the patterns directly through multiple alignment. Thus, the time complexity does not depend on the number of patterns. Instead, computation time is dominated by the clustering step, which has to calculate the distance matrix and build the *k* nearest neighbor list. This inherently makes the time complexity $O(N_{seq}^2 L_{seq}^2)$. However, unlike the support framework there is potential for improving on the time complexity by using a sampling based clustering algorithm and stopping the sequence to sequence distance calculation as soon as it is clear that the two sequences are not neighbors. Thus, not surprisingly when the patterns are relatively long, the multiple alignment algorithms tend to be faster than support based algorithms that are exponential in nature to the size of the pattern.

6. Related Work

Many papers have proposed methods for finding frequent subsequences [1, 2, 9, 11, 13]. There are two works in particular, that extend the support framework to find more useful patterns. [10] extends the framework to find rules of the form "if A then B" using the confidence framework. [12] presents a probabilistic model to handle noise in mining strings. However, it cannot be easily generalized to sequence of sets, and it does not address the issue of generating huge number of patterns that share significant redundancy.

There is a rich body of literature on string analysis in computer science as well as computational biology that can be extended to this problem domain. In particular, multiple alignment has been studied extensively in computational biology [5, 6] to find common patterns in a group of strings (ordered lists of characters). In this paper, we have generalized string multiple alignment to find patterns in ordered lists of sets.

7. Conclusion

Designing good operational definitions for patterns is the first step in mining useful and interesting patterns. Sequential patterns are commonly defined as frequent subsequences based on exact match. Noting some inherent problems in the framework, we present an entirely different framework, multiple alignment sequential pattern mining. We do a comparative study of the two frameworks using a novel evaluation method on the well-known synthetic data generator [1]. The method can comprehensively depict the quality of the result patterns that emerge from certain definitions empirically. The results demonstrate that the alignment framework is able to best recover the underlying patterns with little confounding information under all circumstances including those where the support framework fails.

8. Reference

[1] R. Agrawal and R. Srikant. "Mining Sequential Patterns". In *Proc. IEEE ICDE*, pp 3-14, March 1995.

[2] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. "Sequential pattern mining using a bitmap representation.". In *Proc. ACM SIGKDD*, pp. 429-435, July 2002.

[3] R. Bayardo. "Efficiently Mining Long Patterns from Databases". In *Proc. ACM SIGMOD*, pp 85-93, May 1998.

[4] S. Brin, R. Motwani, and C. Silverstein. "Beyond market baskets: generalizing association rules to correlations." In *Proc. ACM SIGMOD*, pp265-276, 1997.

[5] Osamu Gotoh. Multiple sequence alignment: Algorithms and applications. In *Adv. Biophys.*, V36, pp159-206. 1999.

[6] Dan Gusfield. *Algorithms on strings, trees, and sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, England. 1997.

[7] H.C. Kum, J. Pei, W. Wang, and D. Duncan. ApproxMAP: Approximate Mining of Consensus Sequential Patterns. Technical Report TR02-031. UNC-CH. Oct 2002.

[8] D. Lin and Z. Kedem. "Pincer-search: a new algorithm for discovering the maximum frequent set." In *Proc. 6th Intl. Conf Extending Database Technology (EDBT)*, 1998.

[9] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.C. Hsu, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth", In *Proc IEEE ICDE,* April 2001.

[10] Myra Spiliopoulou. Managing interesting rules in sequence mining. *Proc. Euro. Conf. on Principles and Practice of Knowledge Discovery in Databases*, pp554-560, 1999.

[11] R. Srikant and R. Agrawal. "Mining Sequential Patterns: Generalizations and Performance Improvements". In *Proc. of EBDT.* pp. 3-17. March 1996.

[12] J. Yang, W. Wang, P. Yu, J.Han. "Mining long sequential patterns in a noisy environment." *SIGMOD 2002.*

[13] M. J. Zaki. "Efficient enumeration of frequent sequences." In *Proc. 7th CIKM* , Nov 1998.