序列中的一般化局部序列模式发现*

靳晓明+, 陆玉昌, 石纯一

(清华大学 计算机科学与技术系,北京 100084)
(清华大学 智能技术与系统国家重点实验室,北京 100084)

Discovery of Generalized Local Sequential Patterns

JIN Xiao-Ming+, LU Yu-Chang, SHI Chun-Yi

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)
(State Key Laboratory of Intelligent Technology and System, Tsinghua University, Beijing 100084, China)
+ Corresponding author: Phn: 86-10-62785592, E-mail: xmjin00@mails.tsinghua.edu.cn
http://www.tsinghua.edu.cn

Received 2002-06-04; Accepted 2002-08-23

Jin XM, Lu YC, Shi CY. Discovery of generalized local sequential patterns. *Journal of Software*, 2003,14(5): 970~975.
http://www.jos.org.cn/1000-9825/14/970.htm

Abstract: Previous work on pattern discovery in sequence data mainly considers finding global patterns, where every record in the temporal sequence contributes to support the patterns. However, local patterns, which are frequent only in some time periods, are actually very common in practice and the efficient discovery of it is potentially very useful. This paper presents a method for discovering generalized local sequential patterns with the format ‘if *A* occurs, then *B* occurs within time *T* in subsequence *s*’. The proposed method includes an index structure that supports efficiently locating and counting of the pattern instances and a two-phase method for efficiently mining of local patterns. Experimental results corresponded with the problem definition and verified the superiority of the approach.

Key words: generalized local sequential pattern; temporal sequence; data mining algorithm

摘 要: 已有的时序序列中的模式发现方法主要关注于发现全局的模式,该模式的频繁度量通过扫描序列的所有记录产生.然而,仅在某个时间段中频繁的局部模式在实际中是广泛存在的,对其有效的发现是有意义的.介绍了一种在时序序列中发现一般化局部序列模式的方法.发现的模式具有形式“在子序列 *s* 中,如果 *A* 发生,则 *B* 在时间 *T* 内发生”.提出的方法包括一个支持高效的模式实例定位与计数的索引结构和一个 2 段的局部模式挖

* Supported by the National Natural Science Foundation of China under Grant No.79990580 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.G1998030414 (国家重点基础研究发展规划(973))

JIN Xiao-Ming was born in 1974. He is a Ph.D. candidate at the Department of Computer Science and Technology, Tsinghua University. His current research interests are data mining and database. **LU Yu-Chang** was born in 1937. He is a professor at the Department of Computer Science and Technology, Tsinghua University. His current research areas are data mining and machine learning. **SHI Chun-Yi** was born in 1935. He is a professor and doctoral supervisor at the Department of Computer Science and Technology, Tsinghua University. His current research area is artificial intelligence.

掘算法.试验结果符合问题的定义,并证明了提出方法的优越性.

关键词: 一般化局部序列模式;时序序列;数据挖掘算法

中图法分类号: TP311 **文献标识码:** A

Sequential pattern discovery from temporal sequences is an important subroutine in many data mining applications. Previous work on this problem has mainly considered finding global patterns, where every record in the temporal sequence contributes to the pattern[1~4]. Local patterns found only in subsequences are actually very common in practice and can reveal another kind of useful knowledge compared with global patterns. Therefore, knowing which pattern and in which time period is frequent could be equally if not more useful than simply knowing whether a pattern is frequent. For example, "In the sale records of a supermarket, a customer always buys biscuits followed by soda in the summer, but biscuits followed by milk in winter." Then the analysts observe the correlations of purchase behaviors in different season so that the company can plan sell strategy appropriately according to the season.

Local pattern discovery has not been well considered in the KDD field. The discovery of temporal association rule[5] and partial periodic pattern[6] seem alike to the problem we consider, but either the mining goal or the formats of the discovered knowledge are essentially different. In our previous work[7], we introduced a problem class that is the discovery of local sequential patterns (LSP). In Ref.[8], a compacted representing model is proposed. In this paper, we address a problem that is the discovery of generalized local sequential patterns (G-LSP) with the format "if A occurs, then B occurs within time T" from a long temporal sequence. The problem has a two-dimensional solution space consisting of patterns and temporal features, therefore it is impractical to use traditional methods for global patterns on this problem directly. Our approach is maintaining a suffix-tree-like index structure to support efficient locating and counting of the instances of local patterns. Based on this index, all G-LSPs are discovered by a two-phase algorithm.

1 Problem Description

A sequence $S=(S(1),S(2),\ldots,S(N))$ is a list of records ordered by position number. Without losing generality, we represent S by a \$-terminated sequence of symbols from an alphabet $\sum=\{a_1,\ldots,a_k\}$, where each symbol uniquely represents a record at a time point. A subsequence, $S[sp,ep]=S(sp),\ldots,S(ep)$, is a continuous part of the original sequence. Given a maximal time duration T and a time period $S[sp,ep]$, we use the pattern format: if A occurs, then B occurs within time T in subsequence $S[sp,ep]$ where A and B are two subsequence, i.e.

$$s[pa,pa+|A|-1]=A\wedge s[pb,pb+|B|-1]=B\wedge s[pb,pb+|B|-1]\in s[pa,pa+T-1],$$

where $s=S[sp,ep]$ and $|X|$ denotes the length of subsequence X, i.e. the number of records in X. We denote the pattern as $A_{(T)}\rightarrow B$ $(S[sp,ep])$.

To evaluate the frequency of local patterns, we use measurements *local frequency* (Lf), *local support* (Lsupp) and *local confidence* (Lconf). The *local frequency* of subsequence A in subsequence s is the number of occurrences of A in s. The *local frequency* of pattern $A_{(T)}\rightarrow B(s)$ is the number of occurrences of the pattern in s. We denote it as:

$$\mathrm{Lf}(A,s)=|\{i\ |s[i,i+|A|-1]=A\}|,$$

$$\mathrm{Lf}\left(A_{(T)}\rightarrow B,s\right)=\left|\left\{pb\left|s\left[pb,pb+|B|-1\right]=B\wedge\exists pa\Rightarrow\begin{pmatrix}s\left[pa,pa+|A|-1\right]=A\wedge\\ \left[pb,pb+|B|-1\right]\subset\left[pa,pa+T-1\right]\end{pmatrix}\right.\right\}\right|.$$

The *local support* of subsequence A and that of the pattern $A_{(T)}\rightarrow B$ in subsequence s are defined as follows:

$$\mathrm{Lsupp}(A,s)=\mathrm{Lf}(A,s)/|s|,$$

$$\mathrm{Lsupp}(A_{(T)}\rightarrow B,s)=\mathrm{Lf}(A_{(T)}\rightarrow B,s)/|s|.$$

The *local confidence* of $A_{(T)}\to B$ in subsequence *s* is the ratio of the *local frequency* of $A_{(T)}\to B$ in *s* over the *local frequency* of *A* in *s*. We denote it as:

$$\text{Lconf}\,(A_{(T)}\to B,s) = \text{Lf}\,(A_{(T)}\to B,s)/\text{Lf}\,(A,s).$$

Given a *maximal distance T*, a *minimal support* ms and *a minimal confidence* mc, if the *local support* of a pattern $A_{(T)}\to B(s)$ is no less than ms and the *local confidence* of that pattern is no less than mc, we consider the pattern as a *generalized local sequential pattern* (G-LSP). Then the mining problem is defined as follows: Given a sequence *S*, find all $\langle A,B,s\rangle$ so that $A_{(T)}\to B(s)$ is satisfied with:

$$\text{Lsupp}\,(A_{(T)}\to B(s)) \geq \text{ms and Lconf}\,(A_{(T)}\to B(s)) \geq \text{mc},$$

where *s* is a subsequence in *S*.

Since which pattern exists and in which periods of time a pattern exhibits frequent are both unknown beforehand, previous algorithms for global pattern mining are either inapplicable or have extremely poor time complexity for G-LSP discovery. One method for local patterns is sliding a window through the sequence, and mining for global patterns in each window. This approach has been used in some applications and the time complexity of it is not bad. However it can only find a small part of all the local patterns, of which the valid subsequences are of the same length. A method that finds all local patterns can be derived by applying the previous methods to all the possible subsequence. However, the time complexity of this method is usually poor.

2 Method for G-LSP Discovery

Our method for discovering G-LSPs is: going through the sequence step by step and carrying out a mining process in each step. A mining process includes two phases: candidate generation and G-LSP generation. In the first phase, a suffix-tree-like index structure that we term the local pattern tree (LP tree) is maintained for efficiently pattern counting and the G-LSP candidates are derived. Then in the second phase, each candidate is verified, and G-LSPs are generated.

For the rest of this paper, the following notational conventions will be used: *locus*(*A*) denotes the first node in the indexing tree encountered after *A* is spelled out; *subsequence*(*t*) denotes the subsequence spelled out in the suffix tree by following the path from root to node *t*; *T*(*A*) denotes the sub-tree of which the root is *Locus*(*A*); *T*(*t*) is the sub-tree of which the root is node *t*; and {*Leaf* (*t*)} denotes the set of all leaf nodes of *T*(*t*).

2.1 Index structure

LP tree consists of a standard suffix tree[9], a set of leaf pointers and a leaf chain. A sequence *S* is mapped to a LP tree *L* whose paths are the suffixes of *S*, and whose leaf nodes correspond uniquely to positions within *S*. The data structure of a LP-tree is as follows:

Internal node *t*: (*t.Child*, *t.Ancestor*, *t.Next*, *t.Start*, *t.End*, *t.FirstLeaf*, *t.LastLeaf*) where *t.Start* and *t.End* indicate the starting position and ending position of the subsequence associated with the branch to *t*, *t.Child* stores the pointer to the first child of node *t*, *t.Ancestor* stores the pointer to ancestor node of *t*, *t.Next* stores the pointer to the next node with the same ancestor node of *t*. In order to locate and count the leaf nodes efficiently, all the leaves are linked, forming a leaf chain. For each internal node *t, t.FirstLeaf* and *t.LastLeaf* store the pointer to the first leaf node and the last leaf node of the sub-tree *T*(*t*).

Leaf *t*: (*t.Ancestor*, *t.Next*, *t.Position*, *t.NextLeaf*) where *t.Ancestor* and *t.Next* have the same meaning as that of an internal node, *t.Position* is the position of corresponding suffix in *S*, *t.NextLeaf* stores the pointer to the next leaf node in the leaf chain.

Similar to Suffix tree, the LP tree *L* for sequence *S* has the following properties: 1) Any common subsequence can be spelled out according to the path from the root to an unique internal node. 2) Each internal node except the root has at least two children. 3) The tree has $|S|$ leaves and the number of nodes is at most $2|S|-1$ for each nonempty input sequence. The insertion and construction method for a LP tree is similar to that for a suffix tree. The only difference is that the relative leaf chain pointers need to be updated correspondingly, including the data member *FirstLeaf*, *LastLeaf* and *NextLeaf*. Such operations can be easily done by simple forward and backward scanning in the index.

Given a LP tree, the *local frequency* of subsequence A in any subsequence $S[sp,ep]$ can be calculated by counting the number of leaf nodes of the sub tree rooted at *locus*(A) that satisfies the position restrictions. That is: $\mathrm{Lf}(A,S[sp,ep])=|\{leaf(t_A)|sp\le leaf(t_A).Position\le ep-|A|+1\}|$. Benefiting from maintaining the leaf chain, the number of leaf nodes can be easily and efficiently obtained by a simple traverse in the leaf chain between *t.Firstleaf* and *t.Lastleaf*.

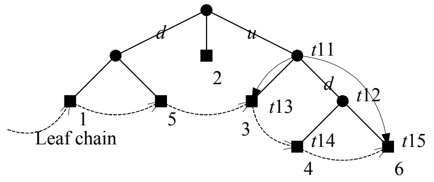

Fig.1 LP tree for S="dsuududds$", S[6,10] inserted

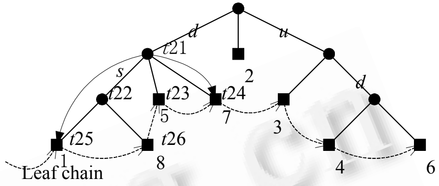

Fig.2 LP tree for S="dsuududds$", S[8,10] inserted

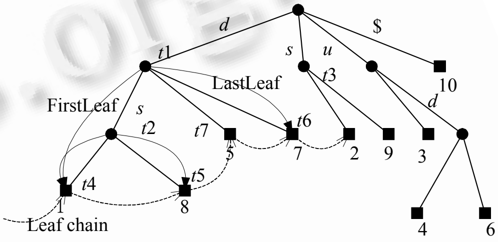

Fig.3 The whole LP tree for S="dsuududds$"

Example 1. The construction processes of the LP tree for the sequence 'dsuududds$' are shown in Fig.1~Fig.3. Here only parts of all the leaf pointers and the leaf chain are shown for the sake of clearness. Suppose we are interested in the 'd' and 'ds' in subsequence 'dsuud', then the *local support* can be calculated as follows:

Locus ('*d*')=$t1$, *locus* ('ds')=$t2$,

$leaf(t1)=\{t4,t5,t6,t7\}$, $leaf(t2)=\{t4,t5\}$,

$\mathrm{Lf}(d,\text{'}dsuud\text{'})=|\{leaf(t1)|1\le leaf(t1).Position\le 5-|\text{'}d\text{'}|+1\}|=|\{t4,t7\}|=2$,

$\mathrm{Lf}(ds,\text{'}dsuud\text{'})=|\{leaf(t2)|1\le leaf(t2).Position\le 5-|\text{'}ds\text{'}|+1\}|=|\{t4\}|=1$.

2.2 Generation of G-LSP candidates

In the candidates generation phase, all subsequences that are frequent enough in the possible periods are found, i.e. to find all $\langle B,s\rangle$ so that Lsupp $(B,s)\ge$ms. Note that, if $\mathrm{Lsupp}(A_{(T)}\rightarrow B(s))\ge$ms, since $\mathrm{Lsupp}(A_{(T)}\rightarrow B(s))=\mathrm{Lf}(AB,s)\le\mathrm{Lf}(B,s)=\mathrm{Lsupp}(B,s)$, there is $\mathrm{Lsupp}(B,s)\ge$ms. This enable us first find all the frequent subsequences B, which we term *G-LSP candidates*, and then mine for G-LSPs by searching the records in front of it.

The method for generating G-LSP candidates is: After a leaf node t is inserted, search the nodes in the path of traveling from node t to root. For each node t_1 in the travel path, search all the leaf nodes $(p_1,p_2,\ldots,p_n)$ in $T(t_1)$ to find all the subsequences $D=S[p_m.Position,t.Position]$ such that the Lsupp $(subsequence(t_1),D)\ge minimal\ support$. These $\langle subsequence(t_1),D\rangle$ are outputted as G-LSP candidates.

Note that, let $(p_1,p_2,\ldots,p_n)$ denote the starting positions of subsequence A, we need only consider the subsequences whose starting positions are p_m instead of scanning all the possible subsequences. This reduces the time complexity dramatically without losing expected candidate.

2.3 Discovery of G-LSPs

After a candidate is generated, a simple subsequence counting method is used to generate G-LSPs. Given the candidate $\langle CB,S[sp,ep]\rangle$, the detail process is as follows:

1) Retrieve all the positions of the instances of CB in $S[sp,ep]$ by going through the leaf nodes of *subtree*(CB)

in the LP-tree. Denote the results by $(p_1,p_2,\ldots,p_N)$.

2) Retrieve the subsequences $s_n=S[p_n-T+|CB|,\ p_n-1]$, and search through $(s_1,s_2,\ldots,s_N)$ to find all subsequence A that fulfill:

$$sp' = sp - T + |B|\,,\ \mathrm{Lsupp}(A_{(T)} \to B, S[sp', ep]) = \frac{\sum_{n=1}^{N} \mathrm{Lf}(A, s_n)}{ep - sp' + 1} \geq \mathrm{ms}\,,$$

$$\text{i.e. } \sum_{n=1}^{N} \mathrm{Lf}(A, s_n) \geq \mathrm{ms} \cdot (ep - sp' + 1).$$

The number of possible A is at most $(T-|CB|)(T-|CB|+1)N/2$. For each A, $\sum\mathrm{Lf}(A,s_n)$ can be easily calculated by searching Leaf (locus(A)) in the LP-tree or counting through $(s_1,s_2,\ldots,s_N)$.

3) For the resulting subsequence A of step 3, calculate the local frequency of A in subsequence $S[sp,ep]$ by going through the leaf nodes of *subtree*(A) in the LP-tree. If $A_{(T)}\to B(S[sp',ep])$ is a G-LSP,

$$\mathrm{Lconf}(A_{(T)} \to B, S[sp', ep]) = \frac{\mathrm{Lf}(A_{(T)} \to B)}{\mathrm{Lf}(A)} = \frac{\sum_{n=1}^{N} \mathrm{Lf}(A, s_n)}{\mathrm{Lf}(A, S[sp', ep])}.$$

Then if

$$\sum_{n=1}^{N} \mathrm{Lf}(A, s_n) \geq mc \cdot \mathrm{Lf}(A, S[sp', ep]),$$

the pattern $A_{(T)}\to B(S[sp',ep])$ is outputted as a G-LSP.

2.4 Overall method

The overall mining method for G-LSPs is:

1) Go through the sequence S and insert each subsequence $S\,[n,\ldots,|S|]$ into a LP tree.

2) After a leaf node t is inserted, find all the G-LSP candidates $\langle subsequence(t_1),D\rangle$.

3) For the candidates $\langle CB,D\rangle$, find all subsequence A that fulfill $A_{(T)}\to B(D')$ is a G-LSP.

4) Repeat the process 1~3 until the whole sequence has been scanned.

The storage complexity of a LP tree is the same as that of a suffix tree, which is about O(|S|)[2]. Compared to traditional mining algorithms in which the core operations are pattern generation and tuple counting, our method costs more storage for the LP tree. The new prospects for the data, which result from our algorithm, may justify the added expense. And, it also reduced the mining time expense, which is a crucial factor in the KDD problem.

3 Experimental Results

Synthetic dataset was generated using the following parameters: L: length of the sequence; PT: maximal time duration, i.e. T of potential G-LSP $A_{(T)}\to B(s)$; PN: The number of potential G-LSPs; $PL1$: length of A in potential G-LSP $A_{(T)}\to B(s)$; $PL2$: length of the B in potential G-LSP $A_{(T)}\to B(s)$; PD: average duration of local patterns, i.e. $|s|$; PS: support of potential G-LSPs; PC: confidence of potential G-LSPs. The dataset is generated as follows: First, generate a sequence with length L, where each record is randomly selected from the symbol set $\{a\ldots z\}$. Second generate both the starting position and the ending position randomly for each pattern. And the corresponding length is normally distributed with PD means. Third, according to PS, PC and PT, generate positions of the instances of the corresponding pattern in the sequence, where the positions are uniformly distributed. Then fill in the letters of this pattern at the positions.

In the experiments, minimal support and minimal confidence are set to be 80% of the corresponding PS and PC. The parameters for generating datasets are given in Table 1 together with the results. Our method successfully discovered all the potential G-LSPs. There are some additional patterns that had also been found. Some of them are parts of the expected patterns, e.g. $A_{(T)}\to B$ of expected pattern $AB_{(T)}\to C$. These redundant patterns can be easily phased out in the post-recessing stage. Furthermore, there are a few unexpected patterns that are not related to the patterns we generated. They come from the randomness of the method of generating experimental data.

Parameters								Discovered expected patterns
	PT	*PN*	*PL*1	*PL*2	*PD*	*PS*	*PC*	
500	5	2	2	2	50	0.2	0.5	2
500	5	10	1	1	50	0.2	0.5	10
500	10	10	2	2	70	0.2	0.5	10
1K	10	10	1	1	70	0.3	0.8	10
1K	10	20	2	2	70	0.3	0.8	20
10K	10	10	1	2	70	0.3	0.8	10
10K	15	20	2	2	70	0.3	0.8	20

We also implemented the naïve frequency counting algorithm that scanned every possible subsequence (introduced in section 1), and carried out experiments using generated sequence with varying length for the purpose of performance comparison. In the experiments, both the method introduced in this paper and the naïve algorithm were used. The results verified the superiority of our method to the naïve one in terms of both the absolute value and the scaling speed of the execution time.

4 Conclusions

This paper presents a problem class: discovery of generalized local sequential patterns (G-LSP) in a long temporal sequence. The problem has a two-dimensional solution space consisting of patterns and temporal features, previous algorithms are either inapplicable or have extremely poor time complexity for this problem. In this paper, we propose an index structure and a two-phase method. Using this method, prerequisite node searching and counting are accelerated by using leaf pointers and leaf chain during the mining process, and all G-LSPs are discovered after one scan of the sequence. We evaluated the behavior of our problem and the performance of our algorithm on synthetic datasets. The results correspond with the definition of our problem. In addition, experiments using sequences of various lengths verified the superiority of our method to the naïve one.

Discovery of G-LSPs is useful in its own right as a tool for the analysis of temporal sequence data. In future, we intend to use it as a subroutine in other KDD applications such as segmentation of sequences, exploration by feature, and mining of second order knowledge[10].

References:

[1] Srikant R, Agrawal R. Mining sequential patterns: Generalizations and performance improvements. In: Proceedings of the 5th International Conference on Extending Database Technology. France, 1996.

[2] Wang K. Discovering patterns from large and dynamic sequential data. Special Issues on Data Mining and Knowledge Discovery, Journal of Intelligent Information Systems, 1997,9(1):8~33.

[3] Li Y, Wang XS, Jajodia S. Discovering temporal patterns in multiple granularities. International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining. Lyon, France. 2000.

[4] Kam P, Fu AWC. Discovering temporal patterns for interval-based events. In: Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2000). UK, 2000.

[5] Chen X, Petrounias I. An integrated query and mining system for temporal association rules. In: Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2000). UK, 2000. 327~336.

[6] Han J, Dong G, Yin Y. Efficient mining of partial periodic patterns in time series database. In: Proceedings of the 15th International Conference on Data Engineering. IEEE Computer Society, 1999. 106~115.

[7] Jin X, Wang L, Lu Y, Shi C. Indexing and mining of the local patterns in sequence database. In: Proceedings of the IDEAL 2002. Springer-Verlag, 2002. 68~73.

[8] Jin X, Lu Y, Shi C. Distribution discovery: Local analysis of temporal rules. In: Proceedings of the PAKDD 2002. Taibei: Springer Verlag, 2002. 469~480.

[9] Weiner P. Linear pattern matching algorithms. In: Proceedings of the 14th IEEE Annual Symposium on Switching and Automata Theory. 1973.

[10] Spiliopoulou M, Roddick JF. Higher order mining: Modeling and mining the results of knowledge discovery. In: Proceedings of the 2nd International Conference on Data Mining Methods and Databases, Data Mining II. 2000.