Top-k Sequence Pattern Mining with Non-overlapping Condition

Xin Chai\textsuperscript{a,c}, Dan Yang\textsuperscript{a,c}, Jingyu Liu\textsuperscript{a,c}, Yan Li\textsuperscript{b}, Youxi Wu\textsuperscript{a,b,c}

\textsuperscript{a}School of Computer Science and Engineering, Hebei University of Technology, Tianjin 300401, China
\textsuperscript{b}School of Economics and Management, Hebei University of Technology, Tianjin 300401, China
\textsuperscript{c}Hebei Province Key Laboratory of Big Data Calculation, Tianjin 300401, China

Abstract. Pattern mining has been widely applied in many fields. Users often mine a large number of patterns. However, most of these are difficult to apply in real applications. Top-k pattern mining, which involves finding the most frequent k patterns, is an effective strategy, because the more frequently a pattern occurs, the more likely they are to be important for users. However, top-k mining can only mine short patterns in mining applications with the Apriori property. It is well-known that short patterns contain less information than long patterns. In this paper, we focus on mining top-k sequence patterns of each pattern length. We propose an effective algorithm, named NOSTOPK (non-overlapping sequence pattern mining for top-k). The algorithm calculates the support of a pattern using a Nettree data structure, which has been introduced to tackle various types of pattern matching and sequence pattern mining issues. We find the top k patterns of length \textit{len}, and calculate the supports of the corresponding \(k \times |\Sigma|\) super-patterns of length \(len + 1\) to discover the new top k super-patterns with \(len + 1\). Experimental results demonstrate that the algorithm achieves a better performance than comparable algorithms.

1. Introduction

The core idea of data mining is to mine frequent patterns [1]. To effectively mine frequent patterns have been afforded much consideration by researchers. Sequence pattern mining [2,3] has played an important role in many mining tasks, such as author characteristics [4], prediction HAS QoE [5], sequential classification [6], time series analysis [7], and multivariate temporal data [8]. In order to avoid mining a lot of unrelated patterns, gap-constrained sequence pattern mining algorithms have been proposed. For example, Zhang et al. [9] studied the problem of gap pattern mining with no-condition. Min et al. [10] redefined the problem in [9], where the Apriori property can be used. Wang et al. [11] designed the algorithm MDSP-CGC to avoid improper settings of the gap constraint, where useful patterns cannot be found. However, sequence pattern mining with gap constraints is a tough problem, since not only many strategies are involved in such as pattern matching strategy to calculate the support of a pattern [12] and

\begin{itemize}
  \item 2010 Mathematics Subject Classification. 68T10.
  \item Keywords. Sequence pattern mining; Gap constraint; Top-k; Non-overlapping condition; Nettree.
  \item Received: 10 October 2017; Accepted: 27 October 2017
  \item Communicated by Hari M. Srivastava
  \item Research supported in part by the National Natural Science Foundation of China under Grant 61673159, in part by the Natural Science Foundation of Hebei Province under Grant F2016202145, in part by the Natural Science Foundation of Heilongjiang Province under Grant F20170109, and in part by the Science and the Technology Project of Hebei Province under Grant 15210325.
  \item Corresponding author: Yan Li
  \item Email addresses: chaixin@scse.hebut.edu.cn (Xin Chai), 1694999193@qq.com (Dan Yang), liujingyu@scse.hebut.edu.cn (Jingyu Liu), lywuc@163.com (Yan Li), wuc@scse.hebut.edu.cn (Youxi Wu)
\end{itemize}
As we know, it is difficult to mine frequent patterns, can be consistent with the avoidance of redundancy. For example, the algorithm NOSEP [16], which is designed to mine frequent patterns, can effectively balance the completeness of the mining with the Apriori property. As we know, it is difficult to set the minimum support threshold, minsup, without a priori knowledge. The higher the minsup is, the fewer mined patterns there tend to be. However, some useful patterns may be missed. On the contrary, when minsup decreases, more patterns will be frequent patterns. However, when using all these excessive patterns it is more difficult to draw real life conclusions. Therefore, an important issue is how to effectively compress frequent patterns and mine more valuable and meaningful patterns.

Top-k pattern mining can greatly reduce the number of frequent patterns, and effectively avoid the coverage of high frequency patterns when using the frequent closed pattern [17] algorithm. Therefore, this method is a more efficient technique for compressing frequent patterns [18]. Traditional top-k algorithms are designed to obtain the top-k of all frequent patterns. Sometimes users want to know the top-k patterns of length L, but traditional algorithms cannot satisfy such requirements. Wu et al. [19] proposed the algorithm MAPBOK, which is designed to obtain the top-k patterns with a support ratio larger than a support threshold $\rho$ for each length. However, MAPBOK deals with no-condition. The latest research shows that sequence pattern mining with the non-overlapping condition can solve the shortcomings of the one-off condition and no-condition, and that it has good prospects [16]. However, the proposed NOSEP algorithm is used to mine all frequent patterns. Therefore, the algorithm NOSTOPK for top-k sequence pattern mining with the non-overlapping condition is proposed here. In this algorithm, the minimum support threshold $\rho$ does not need to be specified. NOSTOPK is convenient, fast, and most importantly is able to satisfy the requirements of users.

The main contributions of this paper are threefold: (1) in algorithm NOSTOPK, the minimum support threshold $\rho$ does not need to be specified, which effectively solves the problem that the minimum support threshold $\rho$ is difficult to set for frequent pattern mining; (2) traditional top-k algorithms are to mine the top-k patterns of all frequent patterns. We deal the issue which can mine the top-k items in each length. Moreover, the required pattern length $L$ and the number of patterns $k$ are determined by users; (3) we propose an effective algorithm NOSTOPK and extensive experimental results verify the feasibility and effectiveness of our algorithm.

2. Problem Definition

**Definition 1.** A collection of different events (namely characters) is called an event set, denoted by $\Sigma$. A sequence $S$ of length $n$ is an ordered list of events, denoted as $S=s_1s_2\cdots s_n$, where $s_i \in \Sigma$. A sequence database is a set of multiple sequences $S$, expressed as $SDB=\{S_1,S_2,\cdots,S_N\}$.

**Definition 2.** The number of characters that can separate two characters in a pattern is called a gap constraint, denoted as $gap=[\text{mingap}, \text{maxgap}]$, where mingap and maxgap are nonnegative integers and represent the minimum and maximum gap constraints, respectively.

**Definition 3.** A pattern $P=p_1[min_1, max_1]p_2\cdots [min_{m-1}, max_{m-1}]p_m$ is called a pattern with gap constraints, where $p_j \in \Sigma$ and $\Sigma$ is the set of events. In particular, given a gap constraint $gap=[\text{mingap}, \text{maxgap}]\ (0 \leq \text{mingap} < \text{maxgap})$, if $min_1=\cdots=min_{m-1}=\text{mingap}$ and $max_1=\cdots=max_{m-1}=\text{maxgap}$, then the pattern $P$ is a pattern with periodic gap constraints.

**Example 1.** Suppose that the pattern $P=p_1[min_1, max_1]p_2[min_2, max_2]p_3=C[0,2]G[0,2]C$ is given, where the gap constraint $[0,2]$ indicates that between each character $C$, $G$, and $C$ there can be zero, one, or two characters. Because $min_1=min_2=0$ and $max_1=max_2=2$, $P$ is a pattern with periodic gap constraints.

**Definition 4.** Given the pattern $P$ of length $m$ and a sequence $S$ of length $n$, if there are $m$ integers $l_1, l_2, \cdots, l_m$ with $1 \leq l_1 < l_2 < \cdots < l_m \leq n$, $min_1 \leq l_{i+1}-l_i-1 \leq max_i$, and $p_1=s_{l_1}, p_2=s_{l_2}, \cdots, p_m=s_{l_m}$, then the location index $\langle l_1, l_2, \cdots, l_m \rangle$ represents an occurrence of $P$ in $S$. 
Definition 6. For two occurrences \( L \) in Example 2. For example, for the occurrence \( \langle \text{occurrence set of } P \rangle\text{set}, which is denoted by \( \text{sup} \) are two non-overlapping occurrences. If all occurrences of an occurrence set are non-overlapping, then the occurrence \( \langle L \rangle \) in Example 2, under the condition of the length constraint \( \text{len} \). Example 4.

Example 3. The only occurrences with length constraints \( \text{len}=[1,4] \) of \( P \) in \( S \) are \( \langle 1,2,3 \rangle \), \( \langle 1,2,5 \rangle \), \( \langle 1,4,5 \rangle \), and \( \langle 3,4,5 \rangle \), which are \( \langle 1,2,3 \rangle \) and \( \langle 3,4,5 \rangle \).

Example 5. In Example 2, under the condition of the length constraint \( \text{len} = [1,4] \), the largest non-overlapping occurrence set of \( P = A\{0,2\}T\{0,2\}A \) in sequence \( S \) is \( \langle 1,2,3 \rangle \), therefore \( \text{sup}(P, S) = 2 \).

Definition 7. The support of the pattern \( P \) in the sequence database \( SDB \) is the sum of the supports \( \text{sup}(P) \) for each length \( \text{len} \). Denoted by \( \text{sup}(P, SDB) = \sum_{s=1}^{n} \text{sup}(P, S) \).

Definition 8. For top-\( k \) sequence pattern mining with the non-overlapping condition, the goal of solving the problem is to mine the top-\( k \) patterns of each length with gap constraints and length constraints in a sequence \( S \) or sequence database \( SDB \) under the non-overlapping condition.

3. NOSTOPK Algorithm

NOSTOPK, proposed in this paper, is a heuristic algorithm. Using a heuristic algorithm to solve the top-\( k \) pattern mining problem can significantly improve the efficiency and provide results that have a small deviation. Because the result of the heuristic algorithm is only an approximate solution, NOSTOPK can use the Apriori property rather than an Apriori-like property to prune candidate patterns. If a sub-pattern \( P \) is a top-\( k \) pattern with length \( L \), then there is a high probability that its super-pattern \( Q \) is a top-\( k \) pattern with length \( (L + 1) \). Thus, the basic idea of this algorithm is that if the pattern length is \( L \), choose the top-\( k \) patterns to mine, then generate the corresponding \( k \times |\Sigma| \) patterns with length \( (L + 1) \) from these patterns, select the top-\( k \) output patterns, and continue to mine (note that if \( |\Sigma| \) is less than \( k \), all characters are seen as frequent patterns with length 1). An illustrative example is provided as follows.

Example 5. Suppose that we have a sequence \( S = s_1s_2s_3s_4s_5s_6s_7s_8s_9s_{10} = \text{GAATTCATCA} \), with length constraint \( \text{len}=[1,5] \), gap constraint \( \text{gap}=[0,1] \), \( L = 2 \), and \( k = 3 \). By scanning the sequence \( S \), we can obtain the candidate \( C = \{A, C, G, T\} \) with length 1, for which the supports are \( \text{sup}(A) = 4 \), \( \text{sup}(T) = 3 \), \( \text{sup}(C) = 2 \), and \( \text{sup}(G) = 1 \). Because \( k = 3 \), \( \{A, T, C\} \text{ are put into the array TopkArr and output. Now, we generate the candidate patterns with length 2 using the three patterns with length 1. Thus, the candidate C with length 2 is } \{AA, AT, AC, TA, TT, TC, CA, CT, CC\}. \text{ After calculating and sorting the supports, the results are } \text{sup}(AT) = 3, \text{sup}(TA) = 2, \text{sup}(TC) = 2, \text{sup}(CA) = 2, \text{sup}(AA) = 1, \text{sup}(AC) = 1, \text{sup}(TT) = 1, \text{sup}(CT) = 1, \text{and } \text{sup}(CC) = 0 \text{. Because } k = 3, \text{ } \{AT, TA, TC\} \text{ are placed in the array TopkArr (note that there are three patterns whose sup}(P, S) \text{ are 2, and any of these is correct). If } L \text{ is greater than } 2, \text{ we can iterate the above process.}

The NETGAP algorithm [16], which employs Nettree, a specially designed data structure, calculates the support effectively.

Definition 9. Nettree is a tree-like data structure. It has root nodes, leaf nodes, children, parents, depth, and other concepts of the tree, but also has its own characteristics, such as that it can have multiple root nodes. In addition to the root node, other nodes can have multiple parent nodes. A node may have multiple paths to root nodes. It uses \( n^j_i \) to indicate the pattern in which layer \( j \) is labeled as \( i \). Thus, the nodes of the same tag can appear in different levels [15, 16, 20].
An illustrative example is presented to show the principles of NETGAP.

**Example 6.** Given a sequence $S = s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11} s_{12} s_{13} s_{14} s_{15} = AATTCATCAGCCATG$ and a pattern $P = p_1 p_2 p_3 p_4 = [0, 3] T [0, 3] C [0, 3] G$, the Nettree shown in Figure 1 (a) can be created according to the pattern and sequence.

Figure 1: The Nettree for $P$ in $S$

(1) In Figure 1 (a), $n_5^2$, which is a third level leaf, can be pruned according to NETGAP, because there is no path to reach a fourth level leaf via node $n_5^2$. Then, $n_3^1$ can also be deleted after pruning node $n_5^2$. Similarly, nodes $n_2^{14}, n_1^{13}$, and $n_9^1$ are pruned. The final Nettree is shown in Figure 1 (b).

(2) It is easy to obtain the first occurrence $⟨1, 4, 8, 10⟩$ in the updated Nettree, marked in light grey in Figure 1 (b). Then, all four light grey nodes are pruned according to the non-overlapping condition. After pruning the four nodes, $n_2^1$ is also pruned, because there is no path to reach a fourth level leaf via node $n_2^1$. By iterating this process, the second occurrence $⟨6, 7, 11, 15⟩$, marked in dark grey, can be found. Therefore, we know that there are two non-overlapping occurrences, $⟨1, 4, 8, 10⟩$ and $⟨6, 7, 11, 15⟩$, for $P$ in $S$. Hence, $\text{sup}(P, S) = 2$.

The algorithm NOSTOPK is presented as follows.

**Algorithm 1.** NOSTOPK

Require: Sequence S or sequence database SDB, len = [minlen, maxlen], gap = [a, b], L, k
Ensure: The top-k patterns of each length in array TopkArr
1: Scan sequence S or sequence database SDB, calculate the support of each character and sort them, get the top-k characters, store them in array TopArr;
2: level ← 1;
3: while level ≤ L - 1 do
4: C = gen_candidate (level - 1); // generate candidate set and store the result in C
5: for each cand in C do
6: pro.value ← NETGAP(cand); // NETGAP is a function that calculates the support of a pattern
7: if (pro.value <> 0) then
8: queue.push(pro);
9: end if
10: while !queue.empty() && i ≤ k do
11: Store the top-k patterns in array TopArr;
12: end while
13: end for
14: level++;
15: end while
16: return TopkArr;
Table 1: Protein sequence datasets

<table>
<thead>
<tr>
<th>Sequence Database</th>
<th>Number of Sequences</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDB1</td>
<td>507</td>
<td>91875</td>
</tr>
<tr>
<td>SDB2</td>
<td>338</td>
<td>62985</td>
</tr>
<tr>
<td>SDB3</td>
<td>169</td>
<td>32503</td>
</tr>
<tr>
<td>SDB4</td>
<td>590</td>
<td>109424</td>
</tr>
<tr>
<td>SDB5</td>
<td>400</td>
<td>73425</td>
</tr>
<tr>
<td>SDB6</td>
<td>200</td>
<td>37327</td>
</tr>
</tbody>
</table>

4. Experimental Results and Analysis

As a benchmark dataset, the data used in this paper has been investigated in previous studies, such as [16] and [19], and consists of six protein sequences. The first three protein sequences SDB1, SDB2, and SDB3 can be obtained from ASTRAL95.1.161, and the last three SDB4, SDB5, and SDB6 can be obtained from ASTRAL95.1.171. The features of the protein sequences are shown in Table 1. All experiments are run on a computer with the Intel Core i3-2350M 2.30GHz CPU, 6 GB RAM, and Windows 7 operating system. Dev-C++ is used to develop all algorithms.

For the top-\(k\) mining problem with the non-overlapping condition, we use the ratio calculation formula with a weight to measure the precision of the top-\(k\) mining results. The equation is given in the text as Eq. (1).

\[
\text{Precision} = \frac{\left( \sum_{i=1}^{d} (i \times a_i) \right)}{\left( \sum_{i=1}^{d} (i \times b_i) \right)}
\]

where \(a_i\) and \(b_i\) are the numbers of correct top-\(k\) frequent patterns and total top-\(k\) frequent patterns with length \(i\), respectively, and \(d\) is equal to the length \(L\). In general, \(b_i\) is equal to \(k\), but when \(c\) is less than \(k\), \(b_i\) becomes \(c\), where \(c\) is the number of length \(i\) frequent patterns. According to Definition 8, suppose we want to find top \(k\) patterns with length \(L\) that means we will find \(L \times k\) patterns and NOSTOPK mines the same amount of patterns. Therefore, the number of false positives is the same as that of false negatives. Hence, the precision and the recall are the same.

4.1. Running Time Evaluation

In order to illustrate how \(L\), \(k\), and the length of a sequence affect the running time of NOSTOPK, the results of protein sequences mined with different values for \(L\), \(k\), and the lengths of sequences are shown in Figure 2. (Note: \(L\) is the layer number and the maximum length of generated patterns.)

![Figure 2: The running time on protein sequences](image)

(a) For top-10  (b) For top-20  (c) For top-30

As we can see from Figure 2, the longer the sequence is, the greater \(L\) and \(k\) are, and the longer the running time is.
(1) When $L$ and $k$ are the same, the longer the length of the sequence is, the longer the running time is.

(2) When the lengths of sequences and $k$ remain the same, the greater the $L$ is, the greater the number of patterns at each layer is, and the number of candidate patterns increases. Thus, the running time increases. For example, for the top-30 patterns on each layer mined on SDB1, shown in Figure 2 (c), the running time is 62.1s when $L = 4$, while the running time is 126.3s when $L = 6$, which obtains more than two layers of the patterns, but the running time is more than twice as long.

(3) Clearly $k$ also has a significant effect on the running time. The running time will be longer as $k$ increases, because the number of candidate patterns also increases. We can clearly see that SDB4 is the longest protein sequence. When mining the top-10 on it, the running time for $L=6$ in Figure 2 (a) is 27.4s, the running time for the top-20 mining under the same condition in Figure 2 (b) is 96.6s, and the running time for the top-30 under the same condition in Figure 2 (c) is 143.9s.

4.2. Running Time Contrast Evaluation

The algorithm NOSTOPK is compared with NOSEP-k (NOSEP for top-k), based on the algorithm NOSEP [16], to verify its efficiency. The algorithm NOSEP-k mines the top-k patterns in all frequent patterns with support greater than or equal to $\rho$ at each length. The main difference between NOSTOPK and NOSEP-k is that the candidate set is generated differently. NOSEP-k generates the candidate set by the pattern growth, generating candidate patterns using frequent patterns with support greater than or equal to $\rho$. However, NOSTOPK generates a candidate set using the top-k patterns, which improves efficiency considerably.

A comparison of the running times of the two algorithms for the top-10, top-20, and top-30 on the protein sequences is given in Figure 3.

(1) As the sequence length and $k$ increase, the running times of NOSTOPK and NOSEP-k steadily increase with a little fluctuation.

(2) As $L$ increases, the running time of NOSEP-k increases more than that of NOSTOPK, which remains short with little fluctuation. When $L = 4$, the running time of NOSEP-k is over hundred seconds than that of NOSTOPK. The gap increases when $L = 5$, where the running time of NOSEP-k can be over 3000s greater than that of NOSTOPK. The longest running time for NOSTOPK was for the top-30 mining on SDB4 in Figure 2 (c), requiring about 144s, while NOSEP-k required almost 16,000s, which is 100 times greater than the running time of NOSTOPK. The difference in running time between the two algorithms is greatest when $L = 6$. The shortest time for NOSEP-k is between 2000s and 3000s, and the longest is between 15000s and 16000s, which is more than four hours, while NOSTOPK is completed within 144s.

Thus, the NOSTOPK algorithm has a high efficiency.

4.3. Precision evaluation

In order to illustrate how $L$, $k$, and the length of a sequence affect the precision of NOSTOPK concisely, the results of protein sequences mined with different $L$ and $k$ values and lengths of sequences are shown in Figure 4.

(1) We can see from Figure 4 that the precision is higher when $k$ is larger, while the precision is lower when $L$ is larger.
(2) Regardless of the lengths of sequences and how large $k$ is, the precision is almost the lowest when $L=6$ in Figure 4. This is because when $L=6$, there are significantly more candidate patterns than when $L=5$. As protein sequences contain more characters, while NOSTOPK generates patterns of length 6, only the top-$k$ patterns of length 5 are used. Thus, there will be significantly fewer generating patterns, and there is a certain deviation from the correct patterns. We use the average precision to reflect this effect. Taking the mining from SDB1 to SDB6, for example, when $L=4$ the average precision of NOSTOPK is 98%, when $L=5$ the average precision is 95.6%, and when $L=6$ the average precision is 92%. Since when $L=4$, we consider the instances for Top-10, Top-20, and Top-30 on Datasets from SDB1 to SDB6, there are 18 instances and 1380 patterns in total will be discovered according to Definition 8. Similarly, we know that we will discover 1740 patterns and 2100 patterns in total for $L=5$ and $L=6$, respectively. The experimental results show that 18, 64, and 140 patterns are lost in NOSTOPK when $L=4$, $L=5$, and $L=6$, respectively. Therefore, the smaller $L$ is, the higher the precision is. However, as $k$ increases, the overall precision also increases. For example, when $k=10$ the average precision of NOSTOPK is 93%, when $k=20$ it is 96%, and when $k=30$ it reaches 97%. We know that we will find 900 patterns, 1800 patterns, and 2520 patterns, while the lost patterns are 70, 71, and 81 for $k=10$, $k=20$, and $k=30$, respectively. Thus, the larger $k$ is, the higher the precision is. Therefore, the effect of $k$ on the precision is very important. In summary, the algorithm NOSTOPK achieves a higher precision within a short period of time.

To improve the precision, an effective method is enlarging the searching space. We mine top $e \times k$ patterns with length $j$ and output top $k$ patterns in them at first. Then we use $e \times k$ patterns to generate the candidate patterns with length $j+1$ and find top $e \times k$ patterns. We iterate this process to tackle this issue. We can see that NOSTOPK is the special case with $e=1$. To report how $e$ affects the precision and the running time, the results with $L=4$ and Top 10 are shown in Figure 5.

From Figure 5, the large the $e$ is, the higher the precision is, but the longer the running time is. The reason lies that the larger the $e$ is, the more the number of candidate patterns will be and the higher the precision of the results is. For example, when $e=1.5$, the precision on SDB1 to SDB4 all are 100%, but the running time is about twice greater than that when $e=1$. 

![Figure 4: Comparison of the precision on protein sequences](image)

![Figure 5: The precision and running time for Top-10 and L=4 mining on protein sequences](image)
5. Conclusion

In this paper, we analyze the shortcomings of existing algorithms, and propose a top-$k$ sequence pattern mining algorithm with the non-overlapping condition, named NOSTOPK. This algorithm does not produce a large number of superfluous candidate patterns, and will not lose the patterns that the user is interested in. Furthermore, this algorithm is more effective in satisfying the needs of users than traditional top-$k$ algorithms. The efficiency of NOSTOPK is verified by a large number of experiments on protein sequences.

References