漸近交差法に基づくオンライン頻出系列パターンマイニング

Incremental intersection for frequent sequential pattern mining

山本泰生 [1,2]*　山内夏美 [3]　岩沼宏治 [1]
Yoshitaka Yamamoto[1,2]　Natsumi Yamauchi[3]　Koji Iwanuma[1]

[1] 山梨大学大学院総合研究部
[1]Graduate Faculty of Interdisciplinary Research, University of Yamanashi
[2] 科学技術振興機構，さきがけ
[2]JST, Presto
[3] 山梨大学工学部コンピュータ理工学科
[3]Department of Computer Science, University of Yamanashi

Abstract: We propose a novel method for frequent sequence mining over transaction stream (FSM-TS). Unlike itemset mining, the task of FSM-TS is required to handle *sequences of itemsets* and thus involved in the highly combinatorial explosion of mining objects. The proposed method addresses this problem by extending two key techniques, called *incremental intersection* and *resource-oriented approximation*, in the context of FSM-TS. In this paper, we briefly describe the validness of it and show a preliminary result in the experiment obtained using real datasets.

1 Introduction

FSM-TS is one of the most general mining tasks in streaming data mining. The target data is streaming transactions with variable length L where the well-studied single stream is the specific case of $L = 1$. FIM-TS is used to derive co-occurrences and sequences appeared frequently in transactions, and thus regarded as an extension of such fundamental tasks as frequent itemset mining (FIM) [8] and frequent sequence of items mining (FISM) [7]. This task is widely applicable to streaming data analysis especially for extracting explanatory variables with the discrete structre. Indeed, it is essential to find causalities (episode discovery [1, 5]) in real-time over the input *text data* (ex. twitters, newspaper and SNS). However, compared with other fundamental tasks that have been intensively studied (i.e., FIM and FISM), it is still poorly investigated to establish a reasonable solution for FSM-TS due to the huge search space.

There is a recent work to integrate the compression and appriximation techniques in FIM [3, 6, 9]. The literature presents an one-pass approximation algorithm for obtaining a lossy-compressed form of the frequent itemsets. This method is composed of two key techniques: the exact mining, called *incremental intersection*, for finding the closed itemsets and so-called *resource-oriented* approximation mining. In this paper, we reconstruct them in the context of FSM-TS and propose an one-pass algorithm for efficiently finding the frequent sequences of itemsets. We next show the validness of the output as well as a preliminary result in the experiment. For the space limitation, we omit the proof of theorems and select the expeimental result to be put in the paper.

*連絡先： 〒 400-8510 山梨県甲府市武田 4-4-37 A3-K212
E-mail: yyamamoto@yamanashi.ac.jp

2 Notion and terminologies

We briefly review the notations and terminology in the paper. Let $I = \{x_1, \dots, x_u\}$ be the universe set of items. An *itemset* is a non-empty subset of I. A *sequence* α is an ordered list of itemsets, denoted by $\langle s_1 \cdots s_n \rangle$, where s_i $(1 \leq i \leq n)$ is an itemset $\{a_1, a_2, \dots, a_m\}$. It is abbreviated by $(a_1 a_2 \cdots a_m)$ for simplicity. We call n, s_n and s_1 by the *width*, *head* and *tail* of α, respectively. Let $\alpha = \langle s_1 \cdots s_n \rangle$ and $\beta = \langle t_1 \cdots t_m \rangle$ be two sequences. α is a *subsequence* of β if there exist integers $i_1 < \cdots < i_n$ such that $s_k \subseteq t_{i_k}$ for each k $(1 \leq k \leq n)$. $\alpha \circ \beta$ denotes the concatenation of β to α.

A transaction stream $\mathcal{S}$ is an unbounded single sequence of itemsets (i.e., *transactions*) with variable-length. $\mathcal{S}$ is written as $\langle t_1 \cdots t_N \rangle$ when the output is requested at time n. Given $\mathcal{S}$, we aim at seeking for the frequent subsequences in it. We use so-called *head frequency* measure [1, 4] to define the frequency of a sequence over the stream.

Definition 1 (Window) *Let $\mathcal{S} = \langle t_1 \cdots t_n \rangle$ be the stream and k be the maximal width of target subsequences to be searched. The window at time i, denoted by $win(i)$, is the subsequence $\langle t_{e(i)} \cdots t_i \rangle$, where*

$$e(i) = \begin{cases} i-k+1 & if\ i \geq k \\ 1 & otherwise \end{cases}$$

Definition 2 (Head frequency) *Let $\alpha = \langle s_1 \cdots s_m \rangle$ be a sequence. The head frequency of α at time n wrt $\mathcal{S}$ is defined as $sup(\alpha, n) = \Sigma_{i=1}^{n} include(win(i), \alpha)$, where the function $include(win(i), \alpha)$ is given as:*

$$\begin{cases} 1 & if\ s_m \subseteq t_i \text{ and } \alpha \text{ is a subsequence of } win(i) \\ 0 & otherwise \end{cases}$$

$w(\alpha, i)$ denotes the set of windows until i including α.

The head frequency focuses on the occurrence under the set-inclusion condition between two head elements (i.e., s_m and t_i). Thus, there is no overlap in the occurrences captured by this measure. Besides, the head frequency ensures anti-monotonicity criterion with respect to the following inclusion relation [4].

Definition 3 *Let $\alpha = \langle s_1 \cdots s_n \rangle$ and $\beta = \langle t_1 \cdots t_m \rangle$ be two sequences. α includes β, denoted by $\alpha \sqsupseteq \beta$, if $s_n \subseteq t_m$ and α is a subsequence of β. α properly includes β, denoted by $\alpha \sqsubseteq \beta$, if $\alpha \sqsupseteq \beta$ but $\alpha \not\sqsubseteq \beta$.*

Lemma 1 *[4] If $\alpha \sqsubseteq \beta$, $sup(\alpha, n) \geq sup(\beta, n)$.*

Let $sup(\alpha, i)$ be the support of α at time i ($i \leq n$). A sequence α is *frequent* if $sup(\alpha, n) \geq \sigma n$ for a minimal support threshold σ ($0 < \sigma \leq 1$). The FSM-TS task is finding the frequent sequences over the stream $\mathcal{S}$.

Anti-monotonicity brings the concept of *closeness* compression. Let $F(i)$ be the frequent sequences at time i. A sequence in $F(i)$ is *closed* if there is no sequence in $F(i)$ that properly includes it. $CF(i)$ denotes the closed frequent sequences. $F(i)$ can be restored from $CF(i)$ without generating any compression error (i.e., lossless compression). Due to memory efficiency, it is desirable to seek for $CF(i)$.

The difficulty lies in how to efficiently obtain it for each time i in online processing. Unlike statistic database, it is required to process streaming transactions, continuously arriving at high speed. In this paper, we develop an incremental way to manage the closed sequences in one-pass approximation setting.

3 Incremental intersection

The key idea is based on the technique of *incremental intersection*, which has been proposed in context of frequent itemset mining (FIM) [2, 10], where the mining object is an itemset in I. The literature [2, 10] has revealed a cumulative and incremental way to update the closed itemsets based on the following theorem:

Theorem 1 *[10] Let $T(i)$ be the closed itemsets at time i, and t_{i+1} be the transaction at $i+1$. $T(i+1) = T(i) \cup \{t_{i+1}\} \cup \{\beta \mid \beta = \alpha \cap t_{i+1}, \beta \neq \emptyset, \alpha \in T_i\}$.*

By Theorem 1, the closed itemsets at the next time can be derived by computing the intersection of each stored itemset with the new transaction. The incremental intersect is applicable to FSM. Figure 1 describes the intuition how the closed sequences $TS(i)$ can be incrementally updated for each time i. Note that our mining objects are sequences of itemsets. Hence, it is necessary to define the intersection of a sequence with the window. Sinec the window is also a sequence, it is equivalent to define the intersection of two sequences. However, it is not obvious to answer what the "intersection of sequences" means. In the context of FIM, for two itemsets α and β, their intersection (i.e., $\alpha \cap \beta$) can be interpreted as the *maximal subset included in both.* Based on this notion, we formalize the intersection of sequences as follows:

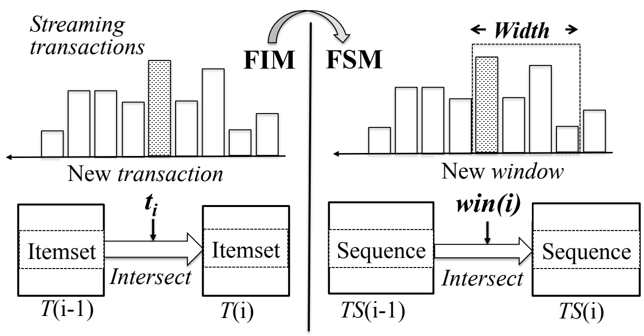

図 1: Incremental intersect for FIM and FSM

Definition 4 *Let α, β and γ be sequences. γ is an intersection between α and β if $\gamma \sqsubseteq \alpha$, $\gamma \sqsubseteq \beta$ and there is no γ' such that $\gamma' \sqsubset \gamma$, $\gamma' \sqsubseteq \gamma$ and $\gamma' \sqsubseteq \gamma$.*

Example 1 *Let $\alpha = \langle(a)(b)(c)\rangle$ and $\beta = \langle(ab)(bc)\rangle$ be two sequences. There are two intersections between α and β, that is, $\langle(a)(c)\rangle$ and $\langle(b)(c)\rangle$.*

The (sequence) intersection in FSM is not necessarily unique, which is unlike to the (itemset) intersection in FIM. $int(\alpha, \beta)$ denotes the set of intersections between α and β. The validity of incremental intersection for FSM is described as follows:

Theorem 2 *Let $TS(i)$ be the closed sequences at time i and $win(i+1)$ be the next window. Then,*

$$\begin{aligned} TS(i+1) &= TS(i) \cup \{win(i+1)\} \cup \\ &\quad \{\beta \mid \beta \in int(\alpha, win(i+1)), \alpha \in TS(i)\}. \end{aligned}$$

4 Finding the intersections

We need to develop an efficient way to compute $int(\alpha, win(i+1))$. At first, it is infeasible to generate every candidate subsequence of α and $win(i+1)$, and then check if it is closed or not. Indeed, there are $O(2^{kL})$ candidates to be checked for the maximal transaction length L and sequence width k. To avoid such generate-and-test approach, we focus the search only on the following *cross table*:

Definition 5 *Let $\alpha = \langle s_1 \cdots s_n\rangle$ and $\beta = \langle t_1 \cdots t_m\rangle$ be two sequences ($n \leq m$). The cross table of α with β is the two-dimensional table T, where for each i ($1 \leq i \leq n$) and j ($1 \leq j \leq m$), the i-th column and j-th row element, denoted by $e(i,j)$, corresponds to $s_i \cap t_j$. For some integers i and j, the area $A(i,j)$ is the set of elements $\{T[u][v] \mid i \leq u < n, j \leq v < m\}$.*

Definition 6 *Let $A(i,j)$ be an area in the cross table. A path wrt $A(i,j)$ is a list $\langle e(i_1,j_1) \cdots e(i_l,j_l)e(n,m)\rangle$ such that $e(i_u,j_u) \in A(i,j)$ ($1 \leq u \leq l$), $i \leq i_1 < \cdots < i_l < n$ and $j \leq j_1 < \cdots < j_l < m$. Given a path p, $S(p)$ denotes the sequence of p obtained by removing the empty elements from p. Let p and q be two paths. We say p includes q if $S(p) \sqsupseteq S(q)$. A path p is maximal wrt $A(i,j)$ if p is a path wrt $A(i,j)$ and there is no path q wrt $A(i,j)$ such that $S(q) \sqsupset S(p)$.*

Example 2 *Consider the following two sequences:*

$$\alpha = \langle(abc)(cd)(a)(cd)(bc)\rangle, \beta = \langle(a)(ac)(c)(abcd)(b).$$

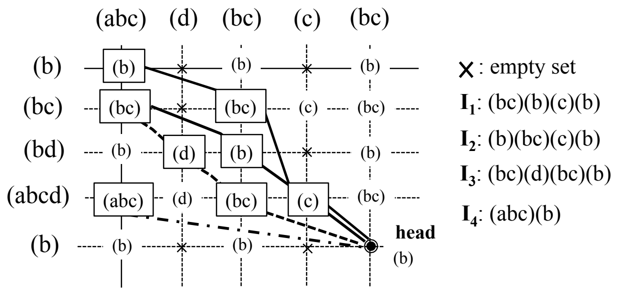

図 2: Example of maximal paths

There are four intersections I_1, I_2, I_3 and I_4 between α and β in Figure 3, each of which corresponds to one maximal path wrt $A(1,1)$.

Proposition 1 *Let $p = \langle e(i_1,j_1) \cdots e(i_l,j_l)e(n,m)\rangle$ be a maximal path wrt $A(1,1)$. Then, the subpath $\langle e(i_u,j_u) \cdots e(n,m)\rangle$ is also maximal wrt $A(i_u,j_u)$, for every u ($1 < u \leq l$).*

By Proposition 1, we can extend the path p with the new element $e(x,y)$ if and only if (1) p is maximal wrt $A(i_1,j_1)$ where $e(i_1,j_1)$ is the prior tail of p and (2) the extended path is also maximal wrt $A(x,y)$. Every next element of the prior tail $e(i_1,j_1)$ is included either in the following two sets:

$$\begin{aligned} C_h(i_1,j_1) &= \{e(x,y) \mid 1 \leq x < i_1, y = j_1 - 1\}. \\ C_v(i_1,j_1) &= \{e(x,y) \mid x = i_1 - 1, 1 \leq y < j_1 - 1\}. \end{aligned}$$

C_h (*resp.* C_v) is called the *horizontal* (*resp. vertical*) set. Any maximal path can be generated by continuing to select the next tail element from either C_h or C_v. Figure 3 sketches the search process: we first select the element $e(i_l-1, m-1)$ from $C_h(n,m)$ and select the next from $c_v(i_l-1, m-1)$. These stepwise-selected elements are concatenated one by one.

Both horizontal and vertical sets include some elements that are prohibited or redundant to be selected. Let $t = e(i,j)$ be the tail element of the prior path.

Definition 7 (Nearest superset) *$e(u,j)$ ($u < i$) is the nearest superset of t in horizontal, denoted by $n_h(t)$, if $e(u,j) \supseteq e(i,j)$ and there is no u' such that $u < u' < i$ and $e(u',j) \supseteq e(i,j)$. In turn, $e(i,u)$ ($u < j$) is the nearest superset of t in vertical, denoted by $n_v(t)$, if $e(i,u) \supseteq e(i,j)$ and there is no u' such that $u < u' < j$ and $e(i,u') \supseteq e(i,j)$.*

Based on the above notion, we define the selectable elements in $C_v(i,j)$ and $C_h(i,j)$ as follows.

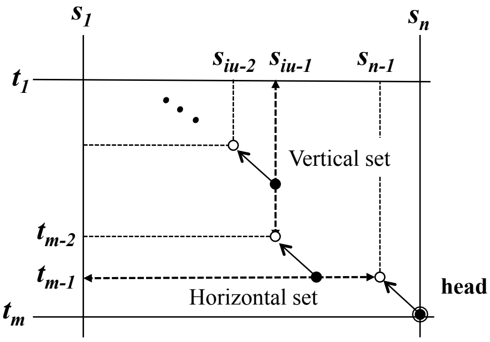

図 3: Overall search process

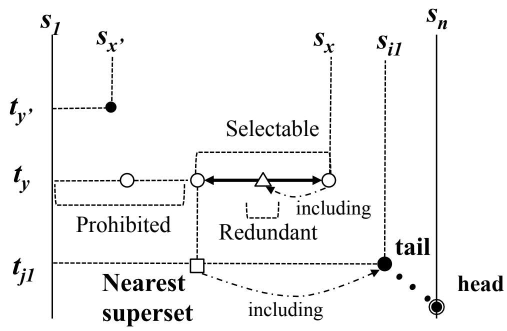

図 4: Pruning techniques

Algorithm 1 Finding the prime paths

Input: $\alpha = \langle s_1 \cdots s_n \rangle$ and $\beta = \langle t_1 \cdots t_m \rangle$
Output: the set U of prime paths (PPs) wrt $A(1,1)$

initialize the candidate path ϕ and set U of PPs
create the cross table T of α with β
if $e(n,m) \neq \emptyset$ **then**
  terminate ▷ the head is empty
else
  $\phi := \langle e(n,m) \rangle \circ \phi$
  if $n == 1$ or $m == 1$ **then**
    add ϕ to U
  else
    compute the non-redundant elements N
    for each element $e(x,y)$ in N **do**
      $\phi := \langle e(x,y) \rangle \circ \phi$
      call *intersect*$(T,\ x,\ y,\ \phi,\ U)$
    end for
  end if
  return U
end if

function INTERSECT$(T,\ u,\ v,\ \phi,\ U)$
  search the nearest supersets n_h and n_v
  if $u == 1$ and n_v does not exist **then**
    add ϕ to U
  else if $v == 1$ and n_h does not exist **then**
    add ϕ to U
  else
    compute the selectable elements $S(u,v)$
    for each element $e(x,y)$ in S **do**
      if the condition in Lemma 5 holds **then**
        $\phi := \langle e(x,y) \rangle \circ \phi$
        call *intersect*$(T,\ x,\ y,\ \phi,\ U)$
      end if
    end for
  end if
  return
end function

Definition 8 (Selectable elements) *Let $e(i,j)$ be the tail element t. $e(x,y)$ in $C_h(i,j)$ (resp. $C_v(i,j)$) is prohibited wrt t if there is the nearest superset $n_h(t) = e(u,j)$ (resp. $n_v(t) = e(i,u)$) of t and $x < u$ (resp. $y < u$). $e(x,y)$ in $C_h(i,j)$ (resp. $C_v(i,j)$) is redundant wrt t if there is an element $e(x',y)$ (resp. $e(x,y')$) such that $x < x' < i$ (resp. $y < y' < j$) and $e(x,y) \supseteq e(x',y)$ (resp. $e(x,y) \supseteq e(x,y')$). $e(x,y)$ in $C_h(i,j)$ (resp. $C_v(i,j)$) is selectable wrt t if $e(x,y)$ is neither prohibited nor redundant.*

Definition 9 (Prior path) *Let p be a path wrt $A(i,j)$ whose tail is $e(i_p,j_p)$. p is prior in horizontal (resp. vertical) if there is no path q whose tail is $e(i_q,j_q)$ such that $S(q) = S(p)$, $i_p > i_q$ (resp. $j_p > j_q$) and $j_p = j_q$ (resp. $i_p = i_q$). We say that p is prior, if p is prior both in horizontal or vertical.*

We call by a *prime path* (PP) such a path that is prior and maximal wrt $A(i,j)$. It is sufficient to seek only for the prime paths (PPs) wrt $A(1,1)$. Let p be a prime path wrt $A(u,v)$ whose tail is $e(u,v)$, and $S(u,v)$ be the set of selectable elements wrt $e(u,v)$. We clarify the condition that the extended path $\langle e(x,y) \rangle \circ p$ becomes prime. Let $e(u_p,v_p)$ be the non-empty element in p lastly appeared before $e(u,v)$.

Lemma 2 *Let $e(u,v)$ be the tail of the prime path p. For every $e(x,y) \in S(u,v)$, $\langle e(x,y) \rangle \circ p$ is prime wrt $A(x,y)$, provided that the following is satisfied:*

Case(1) *both $e(x,y)$ and $e(u,v)$ are non-empty;*

Case(2) *if $e(x,y)$ is empty and $e(u,v)$ is non-empty, then $e(x,v) \not\supseteq e(u,v)$ and $e(u,y) \not\supseteq e(u,v)$;*

Case(3) *if $e(x,y)$ is non-empty and $e(u,v)$ is empty, then $e(x,v) \not\supseteq e(x,y)$ and $e(u,y) \not\supseteq e(x,y)$;*

Case(4) *if $e(x,y)$ and $e(u,v)$ are empty, $e(x,v_p) \not\supseteq e(u_p,v_p)$ and $e(u_p,y) \not\supseteq e(u_p,v_p)$.*

Algorithm 1 sketches the procedure for finding the prime paths between two sequences α and β.

Theorem 3 *Let A be the area of the cross table of $\alpha = \langle s_1 \cdots s_n \rangle$ with $\beta = \langle t_1 \cdots t_m \rangle$. The output of Algorithm 1 corresponds to the prime paths wrt $A(1,1)$.*

5 RO approximation in FSM

One crucial drawback of incremental intersection lies in the huge memory consumption: $|TS(i)|$ always increases as the time i passes over. Besides, $|int(\alpha, win(i+1))|$ is not necessarily unique: many intersections can be generated. This makes the increase of $|TS(i)|$ more exhaustive. There are recent works [3, 6, 9] to embed the (counter-based) approximation technique into incremental intersection for FIM. This technique is used to maintain the frequency of a mining object α at each time i, denoted by $c(\alpha, i)$, allowing to include some error count Δ such that $c(\alpha, i) - \Delta \leq sup(\alpha, i) \leq c(\alpha, i)$ holds.

Analogous to the previous FIM work, it is considerable to introduce the notion of so-called *resource-oriented* (RO) approximation [8] to incremental intersection for FSM. In brief, the RO approximation enables to fix the memory resource to be consumed using a *size* constant r ($r > 0$). Only r counters are used to maintain the frequencies of mining objects. The key idea is simple: $TS(i)$ is updated with $win(i+1)$, and if $|TS(i)| > r$, we delete the "ignorable" mining objects with lower frequencies so that the reduced $TS(i)$ stores at most r objects. In the following, $TS(i)$ is composed of the data entries each of which is a tuple of form $\langle \alpha, c(\alpha, i), \Delta(\alpha, i) \rangle$ where α is the stored sequence, $c(\alpha, i)$ is the frequency of α at time i, and $\Delta(\alpha, i)$ is the (maximal) error count in $c(\alpha, i)$. $\Delta(i)$ denotes the maximal error count in $TS(i)$. Algorithm 2 describes the update procedure for $TS(i)$ with the RO approximation. Note that buf is the buffer size, corresponding to the maximal number of candidate entries to be stored in memory. In case that $|C|$ is greater than buf, we immediately stop to generate new intersections. In this case, we check if the stored sequence α is included in the window (Lines 36-42). If not, it is sufficient just to increment both $c(\alpha, i)$ and $\Delta(\alpha, i)$, instead of computing $int(\alpha, win(i+1))$. We call this *intersect-skip*.

The RO-approximation appears in Lines 29-34. In case that $|TS(i)|$ is greater than the size constant r, we remove the entry with the minimal frequency from $TS(i)$ one by one. Then, the frequency of the removed entry is maintained as $\Delta(i)$. Consequently, Algorithm 2 achieves the completeness for finding the frequent sequences, described as follows:

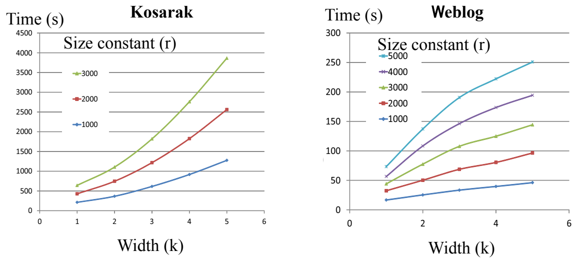

図 5: Executing time (sec) wrt k and r

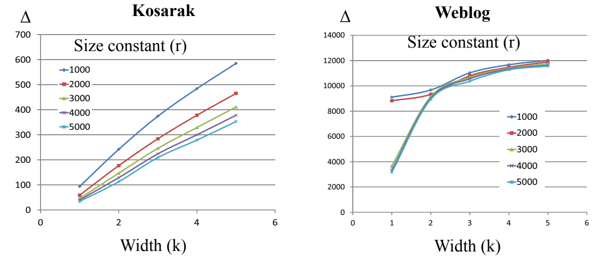

図 6: Error count Δ wrt k and r

Theorem 4 *Let $\mathcal{S} = \langle t_1 \cdots t_n \rangle$ be the stream, σ the minimal support threshold and $F(n)$ the frequent sequences over $\mathcal{S}$ wrt σ. If $\Delta(n) \geq \sigma n$, for every $\alpha \in F(n)$, $TS(n)$ stores an entry for β such that $\beta \sqsupseteq \alpha$ and $sup(\beta, n) \leq sup(\alpha, n) \leq sup(\beta, n) + \Delta(\beta, n)$,*

6 Preliminary experiment

We have implemented the proposed method by C and evaluated its performance using two kinds of real datasets (Kosarak in FIMI and Weblog in [8]). Figures 5 and 6 show the total execution time (sec) and total error count in accordance with varying the sequence width k and the size constant r, respectively.

7 Concluding remarks

This paper studies frequent sequence mining from transaction stream (FSM-TS). We apply the incremental intersect to FSM-TS, which is then integrated into the resource-oriented approximation. By Theorem 4, it is possible to restore the frequent sequences from the output, allowing the approximation error count ($\Delta(n)$ in maximal). Hence, we succeed in lossy (but accuracy-preserved) compression for the frequent sequences. It is required to furthermore investigate the scalability wrt the width k and the size constant r.

We also intend to apply the method to real text data to detect causalities in it.

Algorithm 2 Incremental intersection

Input: $TS(i)$, $win(i+1)$, $\Delta(i)$, r, buf
Output: $TS(i+1)$

initialize C ▷ the candidate entries to be stored
$e := get(win(i+1), TS(i))$
if e is null **then** ▷ no entry for $win(i+1)$
  add $\langle win(i+1), \Delta(i), \Delta(i) \rangle$ to $TS(i)$
end if
for each sequence α stored in $TS(i)$ **do**
  if $|C| > buf$ **then** ▷ *but* is the buffer size
    $skip(\alpha, win(i+1))$ ▷ intersect-skip
  else
    compute $int(\alpha, win(i+1))$ ▷ Algorithm 1
    for each $\beta \in int(\alpha, win(i+1))$ **do**
      $e_\beta := get(\beta, C)$ ▷ the entry e_β for β
      if e_β is null **then**
        add $\langle \beta, c(\alpha, i) + 1, \Delta(\alpha, i) \rangle$ to C
      else if $c(\beta, i) < c(\alpha, i) + 1$ **then**
        $c(\beta, i) := c(\alpha, i) + 1$
        $\Delta(\beta, i) := \Delta(\alpha, i)$
      end if
    end for
  end if
end for
for each sequence γ stored in C **do**
  $e_\gamma := get(\gamma, TS(i))$ ▷ the entry e_γ for γ
  if e_γ is null **then** ▷ no entry for γ in $TS(i)$
    add the entry for γ to $TS(i)$
  else
    replace e_γ with the entry for γ
  end if
end for
while $|TS(i)| > r$ **do**
  $m := getMin(TS(i))$
  ▷ m is a sequence with the minimal frequency
  $\Delta(i) := c(m, i)$ ▷ updating $\Delta(i)$
  $delete(TS(i), m)$ ▷ deleting the entry for m
end while
return $TS(i)$ ▷ corresponding to $TS(i+1)$

function SKIP(α, β)
  if $\alpha \sqsubseteq \beta$ **then**
    increment $c(\alpha, i)$ by one
  else
    increment both $c(\alpha, i)$ and $\Delta(\alpha, i)$ by one
  end if
end function

参考文献

[1] A. Achar, S. Laxman and P. S. Sastry: A unified view of the apriori-based algorithms for frequent episode discovery, *J. of Knowl. and Info. Sys.*, 31, 223–250 (2012).

[2] C. Borgelt, X. Yang, R. N. Cadenas, P. C. Saez and A. P. Montano: Finding closed item sets by intersecting transactions, *Proc. of Int. Conf. on EDBT*, 367–376 (2011).

[3] S. Fukuda, K. Iwanuma and Y. Yamamoto: An online frequent closed itemset mining, *Proc. of SIG-FPAI-B404-01*, 1-6 (in Japanese, 2015).

[4] K. Iwanuma, R. Ishihara, Y. Takano and H. Nabeshima: Extracting frequent subsequences from a single long data sequence: a novel anti-monotonic measure and a simple on-line algorithm, *Proc. of ICDM*, 186–193 (2005).

[5] K. Iwanuma: On interestingness measures for sequential text pattern mining, *Trans. of JSAI*, 27, 136–145 (2012).

[6] K. Iwanuma, Y. Yamamoto and S. Fukuda: An on-line approximation algorithm for mining frequent closed itemsets based on incremental intersection, *Proc. of Int. Conf. on EDBT*, to appear (2015).

[7] L. F. Mendes, B. Ding and J. Han: Stream sequential pattern mining with precise error bounds, *Proc. of ICDM*, 941–946 (2008).

[8] Y. Yamamoto, K. Iwanuma and S. Fukuda: Resource-oriented approximation for frequent itemset mining from bursty data streams, *Proc. of Int. Conf. on SIGMOD'14*, 165-179 (2014).

[9] Y. Yamamoto and K. Iwanuma: Online-pattern mining for high-dimensional data streams, *Proc. of Int. Conf. on Big Data*, 2880–2882 (2015).

[10] S.-J. Yen, C.-W. Wu, Y.-S. Lee, V. S. Tseng and C.-H. Hsieh: A fast algorithm for mining frequent closed itemsets over stream sliding window, *Proc. of Int. Conf. on Fuzzy Systems*, 996–1002 (2011).