基于位置信息的显露序列模式挖掘研究

陈湘涛 肖碧文
(湖南大学信息科学与工程学院 长沙 410082)

摘 要 显露序列因为具有强区分能力,常被用来构建有效的分类器。当前算法大多关注序列模式的支持度或出现次数,而忽略序列模式在序列中的出现位置,这将导致一些重要的信息丢失。为此,提出一种带有局部位置信息的显露序列模式,并给出位置显露序列模式挖掘算法。该算法基于出现次数框架,结合后缀树,省略了候选模式的生成与选择步骤,能够快速有效地挖掘出位置显露序列模式。实验结果表明,采用位置显露序列模式构建的分类器在平均分类准确度上高于传统的显露序列模式挖掘算法。

关键词 出现次数,显露序列,子序列,位置信息
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.07.031

Emerging Sequences Pattern Mining Based on Location Information

CHEN Xiang-tao XIAO Bi-wen
(School of Information Science & Engineering, Hunan University, Changsha 410082, China)

Abstract Owing to the strong ability of distinguishing, emerging patterns have been widely used to build defective classifier. As most of the existing algorithms focus on the support or the occurrences of sequence patterns, and the location of the sequence patterns in a sequence is usually ignored, some important information may be missed. In this paper, we put forward an emerging sequence pattern with local location information, and a mining algorithm of the emerging sequence pattern with location information. Based on the framework of occurrences, combined with the suffix tree, omitting the generation and selection procedure of candidate patterns, this algorithm can quickly and efficiently mine emerging sequence patterns with the location information. The experimental results show that the classifier which is built by emerging sequence patterns with location information is better than the traditional algorithm of mining the emerging sequence patterns on the average classification accuracy.

Keywords Occurrence, Emerging sequence, Subsequence, Location information

1 引言

序列对比是数据挖掘中的一项重要工作[1],有很多学者着重于对比序列模式的研究,如 Ji X N 等[2]提出了最小区分子序列(MDS)的概念和算法。在对比序列的研究中,显露序列由于具有强区分能力而占有举足轻重的地位。Zaane O R 等[3]提出了显露序列的概念,显露序列是指在一个类中频繁出现而在另一个类中非频繁出现的子序列。近年来,随着对显露序列模式的研究加深,显露序列模式被用于较多领域,如序列[4]、分类[5]、聚类[6]等。虽然显露序列具有很强的区分能力,但是之前提出的诸多算法大多是于支持度与置信度框架来评估和挖掘对比子序列。支持度框架的评估方法有可能会丢失一些有效信息,该框架只关心子序列是否出现在当前序列中,而不关心子序列出现的次数,因此 Deng K 等[7]提出了一种基于出现次数框架的模式评判方法,将子序列在序列中出现的次数作为评判指标取代了支持度,使得挖掘子序列的信息更加完善。然而,无论是支持度与置信度框架还是基于出现次数的框架,都只是处于判定显露序列模式存在与不存在的阶段,既不考虑支持度或出现次数的变化规律,也不考虑该显露序列出现的时间或空间情况,而这可能会丢失一些重要的信息。

为了减小模式搜索空间或者提高模式区分能力,序列模式挖掘算法一般会采用相关的约束条件进行辅助挖掘。如 Kemmer A 等[8]采用的全局约束与间隙约束相结合的方法,Wu Y X 等[9]采用的带有通配符的挖掘方法,杨皓等[10]采用的 top-k 与间隙约束相结合的方法等,都能够减小模式搜索空间,去除区分能力低的模式。然而,随着信息需求的增加,各领域对序列模式算法的要求也越来越高,仅仅通过增加约束的序列模式挖掘算法已经研究到了较高的层次,越来越多的学者试图从其他被人忽略的方面寻找突破。Zhang J Y

到稿日期:2016-03-07 返修日期:2016-06-08 本文受湖南省自然科学基金项目(2015JJ2032),湖南省科技计划项目(2014WK2002)资助。
陈湘涛(1974-),男,博士,副教授,CCF 会员,主要研究方向为数据挖掘,E-mail:xtchen2009@sina.cn;肖碧文(1989-),男,硕士生,主要研究方向为数据挖掘。

等[11]针对大型生物序列数据库提出了基于马尔科夫链的序列模式挖掘方法；张君雁等[12]利用虚拟属性进行频繁序列模式挖掘，能够很好地解决由基于先验规则等原始定义带来的一些极端问题；陈湘涛等[13]提出了共享显露序列模式，其在面对多数据集协同挖掘时具有很好的时间属性；刘端阳等[14]提出了一种基于逻辑的频繁序列模式挖掘算法，对于支持度特征选择方法提出了一定的质疑，较好地解决了支持度设置问题及挖掘结果可理解性不高的问题。由此可见，传统的序列模式算法即使结合相关约束，也已经不能满足日益增长的序列模式挖掘的需求。

当前存在的序列模式挖掘算法大多数仅关心序列模式出现或者不出现，很少有学者考虑序列模式的支持度的变化情况，这个问题也引起了一些研究者的关注。2009 年，An A 等[15]提出了背离模式的概念，背离模式是指在一个数据集中，模式在两个类中具有相同或相近的支持度，但是却有相反的支持度变化。该模式考虑了支持度在数据集中的变化情况，可以挖掘出支持度相差不大但具有区分能力的子序列模式。背离模式以数据集的前半部分和后半部分的变化情况来区分模式，由于背离模式的定义与挖掘需要固定的数据环境，导致背离模式不能应用于可变的数据集；而且，An A 仅仅考虑了子序列在整个数据集中的位置信息与支持度的变化情况，这也导致背离模式具有很大的局限性；其次，背离度的计算需要多次扫描数据库，且背离模式的挖掘比较复杂。

因此，子序列在整个序列中的出现位置被引入考查因素。

例 1　设定频繁度为 2，间隙为 0，则正类中频繁子序列有{“aa”，“db”，“ab”}，负类中有频繁子序列{“aa”，“ab”，“bd”，“da”}。按照传统的显露序列模式挖掘方法，如果支持度阈值设定为 2，即 sup1－sup2≥2，间隙约束[9]为 0，则可以得到正类的显露序列模式为{“db”}，负类的显露序列模式为{“bd”，“ab”，“da”}。观察频繁子序列模式 aa，它在正类和负类中都为频繁子序列模式，按照显露序列挖掘方法，模式 aa 虽然有高支持度，但仍然要被舍弃。由表 1 可知，子序列 aa 出现在序列 1 的前半段及序列 4 的后半段，显然，将这一信息考虑进来，可以得知模式 aa 在正类中两次都出现在序列的前半段，在负类中两次都出现在序列的后半段。通过这一位置信息，具有高支持度但是不符合显露序列定义的模式 aa 显然是可以被利用的。这正是我们所希望找到的一些被其他研究者所舍弃或者挖掘不到的具有区分能力的模式，因此位置显露序列模式被提出。

表 1　对比序列集

序列 ID	序列	标签
1	aabbcab	正
2	aadb	正
3	acdb	正
4	ababdaa	负
5	bdab	负
6	bdcaab	负

文中将采用基于出现次数的框架来进行模式挖掘。Deng K 等提出的基于出现次数的框架挖掘原理沿用了经典的候选与生成方法，显然这种方法需要先进行候选模式的生成，再进行模式选择，过程复杂且计算量大。我们将对这一方法进行相关改进，使得模式能够直接从后缀树中挖掘出来，省略了候选模式生成这一步。本文关注的另一个重点是在引入位置信息后如何保持显露序列模式挖掘的效率和准确度。现有的很多算法在面临高维数据时，运行时间将大大增加。针对这一问题，在构建后缀树时，按照给定的位置信息将单序列划分为两个子序列，然后再进行建树和模式挖掘，从一定程度上减小了序列的维数。

本文第 2 节给出了位置显露序列的定义以及其他相关定义；第 3 节设计了位置显露序列的挖掘算法；第 4 节通过挖掘到的位置显露序列模式的个数、位置显露序列模式的个数变化与分类准确度的变化情况和整体的分类准确度来评价整个算法。

2　相关定义

假设对于序列 $S'=e_{i1}e_{i2}e_{i3}\cdots e_{im}$ 和序列 $S=e_1e_2e_3\cdots e_n$，如果存在整数使得 $1\leqslant i_1\leqslant i_2\leqslant\cdots\leqslant i_m\leqslant n$，且对于所有的 $1\leqslant j<m$，存在 $i_{j+1}=i_j+1$ 成立，则称 S' 是 S 的一个子序列，或 S 包含 S'，记为 $S'\subseteq S$。例如，ab 是序列 abcc 的一个子序列，但 ba 不是。在序列 $S=e_1e_2e_3\cdots e_n$ 中项的出现位置由下标决定，如项 e_1 在整个序列 S 中的出现位置为下标 1。

定义 1(显露序列[3])　设序列 α 是序列集 C_{pos}，C_{neg} 中的子序列，λ 为出现次数阈值，当且仅当以下条件成立时，称子序列 α 是一个显露序列。

给定一个正数 λ(出现次数阈值)和两个序列集合类 C_{pos}，C_{neg}，当且仅当以下条件成立时，子序列 α 是一个显露序列。

$$count(\alpha,C_{pos})-count(\alpha,C_{neg})\geqslant\lambda \tag{1}$$

其中，$count(\alpha,C_{pos})$表示子序列 α 在正类 C_{pos} 中的出现次数，$count(\alpha,C_{neg})$表示子序列 α 在负类 C_{neg} 中的出现次数，$\lambda>0$ 表示出现阈值。

定义 2(子序列局部位置)　子序列局部位置是指子序列 α 在某个序列 S 中的局部位置信息均值。其计算公式如下：

$$Loc=\frac{\sum_{j=1}^{n}l_j}{n*L} \tag{2}$$

其中，n 指子序列 α 的长度，l_j 表示子序列 α 中第 j 个元素在整个序列 S 中的出现位置，L 表示整个序列 S 的长度。

例 2　在表 1 中，模式 aa 出现在序列 1 的前半段和序列 4 的后半段。根据公式，子序列 aa 在序列 1 中的准确位置为 0.21，在序列 4 中的准确位置为 0.93。

定义 3(位置显露序列)　设序列 α 是序列集 C_{pos}，C_{neg} 中的子序列，当以下条件之一成立时，子序列 α 是一个显露序列。

$$(\alpha_{cnt\_C_1}-\alpha_{cnt\_C_2})\geqslant\lambda \tag{3}$$

$$\begin{cases}(\alpha_{cnt\_C_{11}}+\alpha_{cnt\_C_{12}}-\alpha_{cnt\_C_{21}}+\alpha_{cnt\_C_{22}})<\lambda\\(\alpha_{cnt\_C_{11}}-\alpha_{cnt\_C_{21}})\geqslant\beta\\(\alpha_{cnt\_C_{12}}-\alpha_{cnt\_C_{22}})<\beta\end{cases} \tag{4}$$

其中，$\alpha_{cnt\_C_{11}}$ 为子序列 α 在正类中出现在给定区分位置 k 前面的计数；$\alpha_{cnt\_C_{12}}$ 为子序列 α 在正类中出现在给定区分位置 k 后面的计数；$\alpha_{cnt\_C_{21}}$ 为子序列 α 在负类中出现在给定区分位置 k 前面的计数；$\alpha_{cnt\_C_{22}}$ 为子序列 α 在负类中出现在给定区分位置 k 后面的计数；$\lambda>0$ 表示出现次数阈值，$\beta>0$ 为位置阈值。

例 3　在表 1 中，假设出现次数阈值 $\lambda=2$，位置阈值 $\beta=2$，且以序列的中间位置 $k=0.5$ 为区分位置。

由表 1 可知，子序列模式 bd 在正类中出现 0 次，在负类中出现 2 次，满足定义 1，因此子序列模式 bd 是一个显露序列；对于子序列模式 aa，它在两个类中的出现次数都为 2，根据定义 1，子序列模式 aa 并非是显露序列，但是，若考虑区分位置 $k=0.5$，则可以得出如表 2 所列的位置信息。

表 2　子序列 α 的位置信息

序列 ID	准确位置
1	0.21
2	0.37
4	0.93
6	0.87

由表 2 可知，子序列在正类中出现准确位置小于 0.5 的计数为 2，在负类中出现准确位置小于 0.5 的计数为 0；反之，在正类中出现位置大于 0.5 的计数为 0，在负类中出现位置大于 0.5 的计数为 2。因此，根据定义 3，子序列模式 aa 是一个位置显露模式。

3　序列模式挖掘

基于上述位置显露序列的定义，本文给出位置显露序列模式挖掘算法，并将其用于分类器的设计。该算法采用基于出现次数的框架来进行模式挖掘。Deng K 等提出的基于出现次数的框架挖掘原理沿用了经典的候选与生成方法，本文将对这一方法进行相关改进。

3.1　带有位置信息的后缀树的构建

由于后缀树[16]的构建仅仅需要线性时间，并能够准确、快速地表示出序列中的所有子序列，且其空间复杂度比字典树要小很多，因此，与 Deng K 等一样，本文将采用后缀进行序列模式的挖掘。

后缀树的构造过程如下：后缀树有一个空的根节点和作为根节点的孩子节点的后缀树集。每一个后缀子树的节点都包含 4 个域：cnt_C11，cnt_C12，cnt_C21 和 cnt_C22，它们分别表示当前节点的模式在正类中位置 k 之前出现次数的计数、正类中位置 k 之后出现次数的计数、负类中位置 k 之前出现次数的计数和负类中位置 k 之后出现次数的计数。

假定给定区分位置 $k=0.5$，则后缀树的构建分为 3 步（见算法 1）。

第一步　初始化树为一棵仅有根节点的树，其计数为 0，孩子节点为空；

第二步　将训练数据集中的每一条序列按照给定的位置划分为两个子序列片段，对每一个子序列片段，产生其所有的后缀子串；

第三步　对每一个产生的后缀子串，调用树的插入算法（见算法 2）将其插入到初始的后缀树。

在算法 1 的第二步中，对每一条原始序列先进行子序列片段的分割，再构建后缀树，这将大大减小数据集的维度。

算法 1　Construct_Tree(T)

Input：sequence dataset D

Output：Tree T

Algorithm：

1. for each sequence in dataset do
2. According to the given position 0.5
   divided its into two subsequences
3. Create all of them suffixes
4. For each suffix x do
5. Insert_Tree(x，class_label)

算法 2　Insert_Tree(x，class_label)

Input：sequence s，Tree T

Output：T with added x in the Tree

Algorithm：

1. match symbols in x with symbols represented in the edges of the tree starting from the root
2. case 0：exactly match to update corresponding counters only
3. case 1：a leaf node is met do
   Update corresponding counters，create a new child node and associate its edge
   with the unmatched part of x
4. case-1：a mismatch occurs do
   Update corresponding counters，split the node，then create a new child node

3.2　位置显露序列模式挖掘

位置显露序列模式的挖掘算法如算法 3 所示。该算法采用深度优先搜索策略，对于根节点的所有孩子节点：

1）判断其在两个类中的出现次数计数是否大于给定的最小出现次数阈值（最小出现次数剪枝）；

2）判断当前节点在两个类中出现次数计数的差值是否满足给出的显露序列模式定义；

3）判断当前节点在两个类中按给定位置出现次数计数的差值是否满足给出的位置显露序列模式定义；

4）判断该节点是否为叶子节点，如果不是则进行迭代步骤。

根据算法 3，所有正类和负类的序列模式将被直接挖掘出来。

LESs 的生成算法伪码如算法 3 所示。

算法 3　LES_Gen(node * root，int λ，int β，prefix)

Input：T node of Tree，minimum occurrence threshold z，minimum location threshold min，prefix which is the sequence path from root to current node

Output：The set of Location Emerging sequences LESs

Algorithm：

1. for each child of root T do

```
2.  while (child->cnt_C11+child->cnt_C12) ‖ (child->cnt_
    C21+child->cnt_C21)>θ
    //最小出现次数剪枝
3.    if (child->cnt_C11+child->cnt_C12)-(child->cnt_C21+
      child->cnt_C21)>λ
4.      {
5.        ES=prefix
6.        ES+=child->strdata
7.        insert ES to ESs
8.      }
9.    else if (child->loc_cnt_C11-child->loc_cnt_C12)>β and
      (child->cnt_C21-child->cnt_C22)< 1
10.     {
11.       ES=prefix
12.       ES+=child->strdata
13.       Insert ES to LESs
14.     }
15.   end if
16.   if current node is not the leaf node do
17.   LES_Gen(current node,int z,int min,prefix)
18.   end if
19. end while
20. end for
```

3.3 剪枝策略

本文采用最大前缀非频繁剪枝[2]策略进行剪枝。其过程如下：

对于给定的最小出现阈值 θ，若子序列 s 满足：($child->cnt\_C11+child->cnt\_C12)<\theta$ 或者($child->cnt\_C21+child->cnt\_C21)<\theta$，则将 s 的所有孩子节点剪枝掉。

当子序列 s 在正类或负类中的出现次数小于最小出现阈值时，树中 s 所有后代的出现次数均不可能大于或等于最小出现阈值。因此，当算法搜索到这样的节点时就无需再对其孩子节点进行搜索，从而进行剪枝。

4 实验及分析

为了评估位置信息在挖掘过程中的影响程度，将从以下几个方面进行实验。

1)位置阈值对位置显露序列模式数目的影响；

2)位置显露序列模式数目对分类准确度的影响；

3)LES_miner 算法与其他相关算法的整体分类准确度对比。

4.1 实验环境及数据

本文使用的数据集是 UCI 机器学习上的 UNIX 用户指令数据集[17]，它包含了 8 个 UNIX 计算机用户的 9 套用户指令数据。实验环境为：Intel Core I3-3240 CPU(3.40GHz)，4GB 内存。

4.2 位置显露序列模式的数目与位置阈值的关系

由实验数据得出，当出现次数阈值为 0.025 时，算法的分类效果最稳定，故选定出现次数阈值为 0.025。在此基础上查位置显露序列模式数目与位置阈值变化之间的联系。从图 1 可以看出，随着位置阈值的增大，大部分对比序列集产生的位置显露序列模式的数目下降趋势明显。

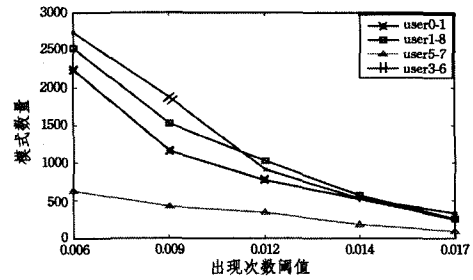

图 1 位置显露序列模式数目与位置阈值的关系

4.3 分类准确度与位置阈值的关系

实验考查位置显露序列模式在分类过程中的作用，即随着位置显露序列模式的数量变化，分析分类准确度的变化情况。给定出现次数阈值为 0.025，考查位置显露序列模式数目与分类准确度的关系。从图 2 和图 3 可以看出，位置阈值从 0.004 变化到 0.01 时，分类的准确度下降明显，结合图 1，此时位置显露序列模式个数下降趋势明显，由此可知，位置显露序列模式数目对分类性能产生了重要影响。

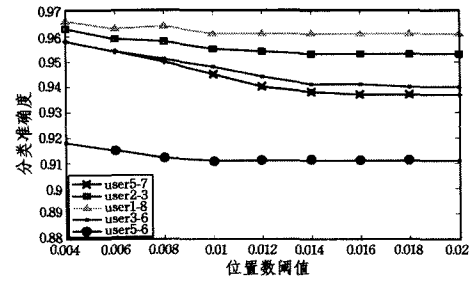

图 2 分类准确度与位置阈值的关系

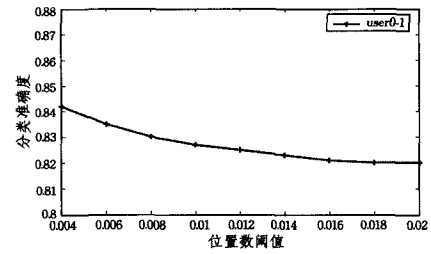

图 3 分类准确度与位置阈值的关系

4.4 分类性能比较

为了验证本文所提方法的有效性，选用两个相关的算法进行对比实验：最小区分子序列(MDSs)算法和显露序列模式挖掘算法(ESs)。MDSs 算法采用基于支持度的评价框架进行模式挖掘，本实验采用的间隙约束为 1；ESs 算法采用基于出现次数的评价框架，没有考虑子序列的位置信息。

UNIX 用户数据的整体分类准确度对比如表 3 所列。

表 3 UNIX 用户数据的整体分类准确度对比

datasets	MDSs	ESs	LESs
user 0-1	0.776	0.818	**0.845**
user 1-8	0.942	0.960	**0.963**
user 2-3	**0.971**	0.968	0.963
user 3-6	0.923	0.931	**0.958**
user 5-6	0.874	0.909	**0.918**
user 5-7	0.910	0.920	**0.938**

从表 3 可知，本文提出的算法比其他两个算法的平均分类精度高。

4.5 理论时间复杂度分析

总结最小区分子序列(MDSs)挖掘算法与显露序列模式(ESs)挖掘算法，序列模式的挖掘分为 3 步：1)扫描数据集，投影到相应的树结构；2)遍历树，得到候选模式；3)通过匹配，选择出符合要求的模式。与其相比，本文算法省略了候选模式生成步骤，直接进行特征选择，理论时间复杂度如表 4 所列。由表 4 可知，理论上本文算法的时间复杂要低于其他两个算法的时间复杂度，且由于采用了后缀树，本文算法的空间复杂也大大低于 MDSs 算法。其中，n 表示数据集规模，v 表示树的边数，m 表示子序列长度。

表 4　理论时间复杂对比

算法	投影树	候选生成	特征选择	总时间
MDSs	$O(n^2)$	$O(n+v)$	$O(nm)$	$O(n^2+nm)$
ESs	$O(n)$	$O(n+v)$	$O(nm)$	$O(n+nm)$
LESs	$O(n)$	—	$O(n+v)$	$O(n+v)$

结束语　本文贡献如下：1)提出了位置显露序列模式(LESs)，即一种在整体序列中考虑了子序列模式出现的位置信息的新序列模式，与之前算法中的显露序列模式相比，当子序列模式在两个类中具有相同的出现次数时，位置显露序列模式仍然具备相当强的分类能力。2)提出位置显露序列模式(LESs)的挖掘算法，该算法可以直接从后缀树中挖掘出位置显露序列，既不用像背离模式那样多次扫描数据库，也不用像显露序列(ESs)模式采用先生成候选模式再进行模式选择的方式，位置显露序列模式挖掘(LESs)可以省略候选生成与出现次数检测这两个步骤；并且，在构建后缀树的同时，位置显露序列模式挖掘算法还能一定程度地适应高维度的数据集。3)与背离模式相比，所提算法能够适用于动态的数据集。与最小区分子序列(MDSs)和显露序列(ESs)相比，位置显露序列挖掘算法的平均分类准确度更高。

然而，位置显露序列模式挖掘算法还存在一些问题：子序列模式的区分位置需要人为指定，如本文给定的区分位置 0.5。因此，寻找一种新的自适应方法来进行动态的位置信息挖掘将是未来研究的一个方向。

参 考 文 献

[1] AGRAWAL R, SRIKAN R. Mining sequential patterns [C]// Eleventh International Conference on Data Engineering. IEEE Xplore, 1995: 3-14.

[2] JI X N, BAILEY J, DONG G. Mining minimal distingushing subsequence patters with gap constraints[J]. Knowledge and Information Systems, 2007, 11(3): 259-286.

[3] ZAANE O R, YACEF K, KAY J. Finding top-n emerging sequences to contrast sequence sets[OL]. https://core.ac.uk/display/15962069.

[4] DENG K, ZAÏANE O R. Contrasting sequence groups by emerging sequences[C]// International Conference on Discovery Science. DBLP, 2009: 377-384.

[5] KOBYLŃSKI Ł, WALCZAK K. Jumping emerging substrings in image classification[M]// Computer Analysis of Images and Patterns. Berlin: Springer Berlin Heidelberg, 2009: 732-739.

[6] SAEED K E K, LEE H G, KIM W J, et al. Using emerging subsequence in classifying protein structural class[C]// Fuzzy Systems and Knowledge Discovery. Piscataway, NJ: IEEE Computer Society, 2009: 349-353.

[7] DENG K, ZAÏANE O R. An Occurrence based Approach to Mine Emerging Sequences[C]// International Comference on Data Warehousing & Knowledge Discovery. DBLP, 2010: 275-284.

[8] KEMMAR A, LOUDNI S, LEBBAH Y, et al. A global Constraint for mining Sequential Patterns with GAP constraint[J]. Eprint arXiv. org, 2016, 9255: 226-243.

[9] WU Y X, WANG L L, REN J D. Mining sequential patterns with periodic wildcard gaps[J]. Applied Intelligence, 2014, 41(1): 99-116.

[10] YANG H, DUAN L, HU B, et al. Mining top-k distinguishing sequential patterns with gap constraint[J]. Journal of Software, 2015, 26(11): 2994-3009.
杨皓，段磊，胡斌. 带间隔约束的 Top-k 对比序列模式挖掘[J]. 软件学报，2015，26(11)：2994-3009.

[11] ZHANG J Y, YANG C H. Sequential Pattern Mining Based on Markov Chain[C]// International Conference on Information Technology in Medicine and Education. IEEE, 2015: 234-238.

[12] ZHANG J Y, MIN F. Frequent Sequence Pattern Mining Algorithm Adopting Dummy Characters[J]. Journal of Chengdu University(Natural Science Edition), 2013, 32(2): 134-137. (in Chinese)
张君雁，闵帆. 采用填充字符的频繁序列模式挖掘算法[J]. 成都大学学报(自然科学版)，2013，32(2)：134-137.

[13] CHEN X T, WANG J, DING P J. Mining shared emerging sequences from multiple datasets[J]. Journal of Central South University(Science andTechnology), 2015(11): 4091-4099. (in Chinese)
陈湘涛，王晶，丁平尖. 面向多数据集的共享显露序列模式挖掘[J]. 中南大学学报(自然科学版)，2015(11)：4091-4099.

[14] LIU D Y, FENG J, LI X F. A logic-based frequent sequential pattern mining algorithm[J]. Computer Science, 2015, 42(5): 260-264. (in Chinese)
刘端阳，冯建，李晓粉. 一种基于逻辑的频繁序列模式挖掘算法[J]. 计算机科学，2015，42(5)：260-264.

[15] AN A, WAN Q, ZHAO J, et al. Diverging Patterns: Discovering Significant Frequency Change Dissimilarities in Large Databases [C]// ACM Conference on Information and Knowledge Management. ACM, 2009: 1473-1476.

[16] BELAZZOUGUI D. Linear time construction of compressed text indices in compact space[C]// arXiv. 2014: 148-193.

[17] BACHE K, LICHMAN M. UCI machine learning repository [EB/OL]. [2016-09-08]. http://archive.ics.uci.edu/ml/datasets.html.