第35卷 第19期 计 算 机 工 程 2009年10月
Vol.35 No.19 Computer Engineering October 2009
·软件技术与数据库· 文章编号：1000—3428(2009)19—0086—02 文献标识码：A 中图分类号：TP311

会话流中 Top-*k* 闭序列模式的挖掘

彭慧丽[1]，张啸剑[2]
(1. 河南省直广播电视大学教务科，郑州 450008；2. 河南财经学院计算机系，郑州 450002)

摘 要：在会话流中挖掘 Top-*k* 闭序列模式，存在因相关比率 ρ 的大小而导致的内存消耗和挖掘精度之间的冲突。基于 False-Negative 方法，提出 Tstream 算法，制定 2 种约束策略限制 ρ。基于该策略设计加权调和计数函数，渐进计算每个模式的支持度。实验结果证明了该算法的有效性。

关键词：Top-*k* 闭序列模式；加权调和平均数；调节因子

Top-*k* Closed Sequential Pattern Mining in Session Streams

PENG Hui-li[1], ZHANG Xiao-jian[2]
(1. Department of Education, Henan Radio & Television University, Zhengzhou 450008;
2. Department of Computer Science, Henan University of Finance & Economics, Zhengzhou 450002)

【Abstract】 The current methods in session streams for mining Top-*k* Closed Sequential Pattern(Topk_CSP) may lead to a conflict between output precision and memory consumption because of using ρ. This paper proposes TStream algorithm, which is based on False-Negative approach. TStream utilizes two constraint strategies to restrict ρ, and employs a weighted harmonic count function to calculate the support of each pattern progressively. Experimental results show that the algorithm is efficient.

【Key words】 Top-*k* Closed Sequential Pattern(Topk_CSP); Weighted Harmonic Average(WHA); regulatory factor

1 概述

Top-*k* 闭模式挖掘一直是数据流研究中的热点。False-Positive 和 False-Negative 是常用的 2 类模式挖掘方法。由于会话流具有数据流的连续性、无界性等特点，传统的 Top-*k* 闭序列模式(Top-*k* Closed Sequential Pattern, Topk_CSP)挖掘算法已不适用。研究者们针对数据流提出了许多算法挖掘频繁模式[1-2]和 Top-*k* 频繁模式[3]。但这些算法存在如下须解决的问题：(1)利用类似文献[1-2]中的算法通常会产生大量的模式。(2)采用文献[2]中方法挖掘闭模式时，支持度阈值 σ 的设置非常敏感。σ 值太小会导致过多的闭模式；σ 值太大会导致无闭模式产生。(3)目前大多数算法基于 False-Positive 方法挖掘各种模式，如文献[1, 3]中的算法。False-Positive 方法利用相关比率 ρ 控制内存消耗、挖掘精度和查全率。使用较大的 ρ 会降低内存消耗，但精确性降低；使用较小的 ρ 能够提高精度，但内存消耗增加，挖掘效率降低。

为解决上述问题，本文挖掘会话流中的 Topk_CSP，制定 2 个边界参数来限制 ρ，设计 2 个边界的加权调和平均数替代 ρ，并设置 1 个调节因子调节 ρ 值的大小。在此基础上提出一种基于 False-Negative 方法和时间敏感滑动窗的挖掘算法 TStream，有效地挖掘某一会话流上的 Topk_CSP。

2 相关概念和描述

令 $P=\{P_1, P_2,\cdots,P_n\}$是 Web 页面的完全集合。一个会话 S 是一个由时间戳指定顺序的序列。一个序列是由被访问时间标记的 Web 页面组成的。会话流 Ss 是由不断到达的会话组成的动态增长会话集，即 $Ss=\{S_1, S_2,\cdots, S_m,\cdots\}$。时间敏感滑动窗 Tsw 是一个向前滑动的窗口，由一组连续的时间单元组成的集合。设当前滑动窗为 Tsw_Γ，则 $TSsw_\Gamma=<t_{\Gamma-w+1}, t_{\Gamma-w+2},\cdots, t_\Gamma>$，其中，$t_\Gamma$ 为当前时间单元；窗口的大小为 w。在窗口 Tsw 中，当序列 s 满足条件 $C(s, Tsw)\geq\sigma|Sset(Tsw)|$时，$s$ 为频繁序列模式，其中，$Sset(Tsw)$表示在 Tsw 中到达的会话集合；σ 为给定的一个支持度阈值；$C(s, Tsw)$表示 $Sset(Tsw)$中包序列 s 的会话数目。

定义 1 给定 Tsw 和序列模式 s，如果不存在这样的模式 s'，使得条件 $s\subset s'$ 和 $C(s, Tsw)= C(s', Tsw)$同时满足，则 s 为闭序列模式。如果恰好存在(k-1)个闭序列模式 s_i'' (i=1, 2,⋯, k-1)，满足 $C(s_i'', Tsw) >C(s, Tsw)$，则 s 在 Tsw 上是 Topk_CSP。

3 约束方法和加权调和计数函数

由于会话流的自身特性，挖掘其中的 Topk_CSP 会出现一定的误差，主要分为面对挖掘精度和面对查全率 2 种。目前，许多基于 False-Positive 方法的算法采用 ρ 来控制这 2 个误差。然而，使用 ρ 会在挖掘精度、内存消耗和查全率之间产生矛盾。

3.1 约束方法

为了解决上述矛盾，给出 2 个边界参数 λ_1 和 λ_2 来约束 ρ，并且制定约束策略来满足用户的挖掘目的。

(1)如果 $\rho<\lambda_1$，则触发第 2 种误差。一个很小的 ρ 值能生成大量的候选模式，导致内存消耗增加，挖掘效率降低，因此，$\rho>\lambda_1$。

(2)如果 $\rho>\lambda_2$，则触发第 1 种误差。由于很大的 ρ 值将会

基金项目：河南省科技厅基金资助项目“非线性降维技术在商业智能中的应用”(082300410110)
作者简介：彭慧丽(1981－)，女，硕士，主研方向：数据挖掘，数据模型；张啸剑，硕士
收稿日期：2009-01-25 **E-mail：**xjzhang82@yahoo.com

产生精度很低的输出结果，因此 $\rho<\lambda_2$。

3.2 加权调和计数函数

最大化($\rho\approx\lambda_2$)或最小化($\rho\approx\lambda_1$)均会加剧上述 2 种误差，因此，设计参数 λ_1 和 λ_2 的加权调和平均数(WHA)来代替 ρ，即 $\rho=WHA(\lambda_1, \lambda_2)$，

$$WHA(\lambda_1, \lambda_2) = (1+\xi^2)\lambda_1\lambda_2 /(\lambda_1 + \xi^2\lambda_2) \quad (1)$$

而在大多数基于 False-Positive 方法的算法中，ρ 等于 ϵ/σ，其中，ϵ 为误差参数。采用 $WHA(\lambda_1, \lambda_2)$替代 ρ，则等式 $\rho=\epsilon/\sigma$ 将发生变化，

$$WHA(\lambda1, \lambda2)= \epsilon/\sigma \quad (2)$$

$$\varepsilon = \sigma \times (1+\xi^2)\lambda_1\lambda_2 /(\lambda_1 + \xi^2\lambda_2) \quad (3)$$

式(1)和式(3)中的参数 ξ 是一个调节因子，通过调整参数 ξ 的值来调节上述 2 种误差以及克服 ρ 引起的问题。

定义 2 根据式(3)，序列模式 s 在一个时间单元 t_i 上的潜在支持度计数定义如下：

$$\hat{C}(s,t_i)=\begin{cases}0 & C(s,t_i)<\varepsilon\,|Sset(t_i)|\\ C(s,t_i) & \text{其他}\end{cases} \quad (4)$$

因此，序列模式 s 在当前滑动窗 $Tsw_\Gamma=<t_{\Gamma-w+1}, t_{\Gamma-w+2},\cdots, t_\Gamma>$ 上的累积支持度计数定义如下：

$$\hat{C}(s, Tsw_\Gamma) = \sum \hat{C}(s,t_i) \quad (5)$$

其中，$t_i\in Tsw_\Gamma, \Gamma-w+1\le i\le\Gamma$。

定义 3 给定参数 λ_1 和 λ_2，$Tsw_\Gamma=<t_{\Gamma-w+1}, t_{\Gamma-w+2},\cdots, t_\Gamma>$为当前的滑动窗。令$<t_{\Gamma-R+1}, t_{\Gamma-R+2},\cdots, t_\Gamma>$为当前窗口 Tsw_Γ 中最近出现的 R 个时间单元，命名为 Tsw_R，大小为$|Sset(Tsw_R)|$，$1\le R\le w$。则加权调和计数函数(WHC)定义如下：

$$WHC(R)=\lceil\sigma\,|Sset(Tsw_R)|\times WHA(\lambda_1,\lambda_2)\rceil \quad (6)$$

可知，在 Tsw_R 中的序列模式 s，如果 $\hat{C}(s, Tsw_R)\ge WHC(R)$，则 s 在 Tsw_Γ 中为潜在频繁序列模式。否则 s 应从 Tsw_Γ 删除。

定义 4 给定 Tsw_R 和潜在序列模式 s，如果不存在这样的模式 s'，使得 $s\subset s'$和 $\hat{C}(s, Tsw_R)=\hat{C}(s', Tsw_R)$条件同时成立，则 s 为潜在闭序列模式。如果恰好存在(k-1)个潜在闭序列模式 s_i'' (i=1, 2,⋯, k-1)，满足 $\hat{C}(s_i'', Tsw_R)>\hat{C}(s, Tsw_R)$，则潜在闭序列模式 s 在 Tsw_R 上为潜在 Top-k 闭序列模式(PTk_CSP)。

4 TStream 算法

TStream 算法包括 2 个子程序：(1)Ttree 构造 Top-k 树；(2)Mtree 对 Top-k 树进行维护。在子程序 Ttree 中函数 *leftcheck* 检测某模式是否为 PTk_CSP。

Top-k 树是一种字典序列树，类似于前缀树。由 3 个部分组成：(1)Top-k 树由 2 部分组成：一个带有标记为 Ø 的根节点和一个页面前缀子树集合。(2)Top-k 树中除根节点之外的每个节点由 3 个域组成：*page*, *tid*, $\hat{C}(s, Tsw)$，其中，*page* 记录模式 s 中的最后一个页面；*tid* 记录 t_i 的 *id*，在 *id* 时刻 s 被插入到树中；$\hat{C}(s, Tsw)$表示 s 在窗口 Tsw 中支持度。(3)hash 表用来检测一个模式 s 是否是 PTk_CSP。使用 PTk_CSP 的 $WHC(R)$作为哈希地址。

设 *Fset*, *Cset* 分别是当前时间单元 t_Γ 和当前窗口 Tsw_Γ 上的 PTk_CSP 集合。k 为一个整数。TStream 算法代码如下：

```
Subroutine 1 Ttree (Ss, σ, k, λ1, λ2)
Create root of Top-k Tree T;
foreach ti, ti ⊂ Tswfirst =< t1, t2,⋯, tj > do
mine all PTk_CSP from Sset(ti);
if leftcheck (si, si ⊂ PTk_CSP) = false then
    foreach sibling sm of si do
      create a new child of form (si ∪ m, i, 1);
    foreach child si' of si do
      Ttree (si', σ, k, λ1, λ2);
    if (si ⊄ T) and (Ĉ(si, ti)=Ĉ(si', ti)) and (Fset (ti, WHC(R))>k)
    then
      Fset =Fset ∪ {si};
      if (si ⊂ T) then
        add Ĉ(si, ti) to Ĉ(si);
      if(Ĉ(si)<WHC(i-tid(si)+1))or(Ĉ(si,Tsw_R)=Ĉ(si'',Tsw_R),
si ⊂ si'') or(Fset(ti,WHC(R))< k) then
        delete si from T;
        Call Mtree (T, si, σ, k, λ1, λ2, w);
Subroutine 2 Mtree (T, Ss, σ, k, λ1, λ2, w)
foreach incoming tΓ- (tΓ ⊂ TswΓ) do
  mine all PTk_CSP from Sset(tΓ);
  if leftcheck (si, si ⊂ Topk_CSP) = false then
    create a new child of form (si ∪ m, Γ, 1);
    foreach child si' of si do
      Mtree (T, si', σ, k, λ1, λ2, w);
      if (si ⊄ T) and (Ĉ (si, tΓ) = Ĉ (si', tΓ)) and (Fset (tΓ, WHC
(R))> k) then
        update Ĉ (si) of si;
        Fset = Fset ∪ {si};
  if((Γ-tid(si)+1<w)and(Ĉ(si)<WHC(Γ-tid(si)+1)))or((Γ-tid (si)+1 ≥
w) and (Ĉ(si)<WHC(w))) or (Fset (tΓ, WHC(R))< k) then
      delete si from T;
  if Ĉ (si, TswΓ)≥ σ |Sset(TswΓ)| then
      Cset = Cset ∪ {si};
  foreach expiring tΓ-w+1 (tΓ-w+1 ⊂ TswL)
    mine all PTk_CSP from Sset(tΓ-w+1);
    foreach child si' of si do
      Mtree (si');
      same as lines 7-13 in Subroutine 2;
    if si ⊂ Lset and (Γ-tid (si) +1≥w)then
      eliminate si from T;
  Output Topk_CSP on demands;
```

5 算法性能分析

TStream 算法采用 C 语言编写，GCC 编译。机器配置是 2.8 GHz Pentium 处理器，1.0 GB 内存及 Fedora Core 6.0。采用 BMS-WebView-1 和 BMS-WebView-2 数据集。BMS-WebView-1 由 59 602 个会话组成，平均会话大小为 7 个~13 个页面。BMS-WebView-2 由 537 083 个会话组成，平均会话大小为 10 个页面。每个 Tsw_i 包含 20 个时间单元，每个时间单元 t_i 包括近似 100 k 个会话。实验从内存消耗、挖掘精度和查全率 3 个方面测试 TStream 性能。

图 1 显示了在数据集 BMS-Web View-1 上，TStream 算法和基于 Lossy Counting 算法的 Top-k 闭序列模式挖掘算法(TKCLC)在不同 ξ 值下的内存消耗比较。

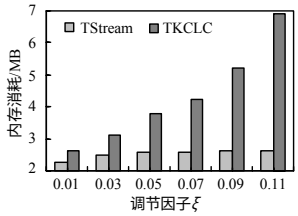

图 1 调节因子变化时的内存消耗

设 λ_1=0.001, λ_2= 0.999, σ=0.010, k=100。结果表明，ξ 从 0.01 变化到 0.11 时，TKCLC 算法的内存消耗增加较快，而

（下转第 90 页）