文章编号: 1001- 2486(2009) 02- 0090- 04

一种基于混合搜索的高效 Top-K 最频繁模式挖掘算法*

敖富江[1], 杜 静[2], 陈 彬[1], 黄柯棣[1]
(1. 国防科技大学 机电工程与自动化学院, 湖南 长沙 410073; 2. 国防科技大学 计算机学院, 湖南 长沙 410073)

摘 要: 挖掘数据集中的 Top-K 最频繁模式具有重要意义。已有 Top-K 最频繁模式挖掘算法通常采用最频繁的 k 个项目作为初始项目, 并将初始项目中频率最低的项目的支持度作为初始边界支持度。但实际组成 Top-K 最频繁模式的项目数目可能远少于 k, 从而制约了算法的效率。为此, 提出了一种基于混合搜索方式的高效 Top-K 最频繁模式挖掘算法 MTKFP。该算法首先利用宽度优先搜索获得少量的短项集, 并利用短项集确定数目少于 k 的初始项目范围以及较高的初始边界支持度; 然后利用深度优先搜索获得所有 Top-K 最频繁模式。实验表明, MTKFP 算法所获得的初始项目数目至少低于已有算法 70%, 初始边界支持度高于已有算法; MTKFP 算法的性能优于已有最好算法。

关键词: Top-K 最频繁模式; 边界支持度; 混合搜索; FP-Tree
中图分类号: TP391 文献标识码: A

An Efficient Mixed-searching-based Algorithm for Mining Top-K Most-frequent Patterns

AO Fu-Jiang[1], DU Jing[2], CHEN Bin[2], HUANG Ke-Di[1]
(1. College of Mechatronics Engineering and Automation, National Univ. of Defense Technology, Changsha 410073, China;
2. College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: It is significant to mine Top-K most-frequent patterns in dataset. The existing algorithms usually use the k-most frequent items as the initial items, and use the support of item with lowest frequency in initial items as the initial border support. In fact, since the number of items in Top-K most-frequent patterns is much less than k, the efficiency of the existing algorithms is restricted. To solve this problem, an efficient mixed-searching-based algorithm for mining Top-K most-frequent patterns, MTKFP is presented. The algorithm firstly mines some short item sets by breadth-first searching, and uses short item sets to obtain the scope of the initial items (the number of initial items is less than k) and the higher initial border support; then it obtains all Top-K most-frequent patterns by depth-first searching. The experimental results show that the number of initial items of MTKFP is 70% lower than that of existing algorithms, and the initial border support of MTKFP is higher than that of existing algorithms. Hence the performance of MTKFP is superior to that of the best existing algorithm.

Key words: Top-K most-frequent patterns; border support; mixed searching; FP-Tree

频繁模式挖掘是一类重要的数据挖掘应用。传统的频繁模式挖掘算法通常需要用户指定一个最小支持度阈值。但如果阈值太大, 则可能得不到充足的频繁模式; 如果阈值太小, 则将得到大量冗余的频繁模式, 不利于发现有用知识。挖掘 Top-K 最频繁模式仅需要用户指定需要挖掘的模式数目 k, 算法将自动返回最频繁的 k 个模式, 因此是一种较好的解决方法。当前已有的 Top-K 最频繁模式算法主要包括 Top-K FP-growth[1]、ExMiner[2] 和 OExMiner[3]。Top-K FP-growth 算法和 ExMiner 算法的初始项目包含 k 个最频繁的项目(若数据集中的项目总数不足 k, 则包含所有项目), 初始边界支持度设置为初始项目中频率最低的项目的支持度(若数据集中的项目总数不足 k, 则为 0); 然后基于初始项目建立 FP-Tree, 并基于 FP-Tree 采用深度优先搜索挖掘 Top-K 最频繁项集。对于大多数数据集来说, Top-K 最频繁模式中所包含的项目数目远低于 k, 并且最终的 Top-K 边界支持度远高于第 k 个最频繁项目的支持度。因此

* 收稿日期: 2008- 09- 18
基金项目: 国家自然科学基金资助项目(60573057; 60704038)
作者简介: 敖富江(1975—), 男, 博士生。

Top-K FP-growth 算法和 ExMiner 算法所建立的FP-Tree 非常冗余, 空间效率不高; 并且由于初始项目数目过多、初始边界支持度较低, 从而影响算法的时间效率。OExMiner 算法对此做了改进。该算法利用初始 m(m 小于k) 个项目探索Top-K 最频繁项集的最终边界支持度。但如果 m 指定不当, 需要利用更大的 m 重新探索, 因此可能会降低算法效率。为了降低初始项目范围并得到较高的初始边界支持度, 本文提出了一种基于混合搜索方式的Top-K 最频繁模式高效挖掘算法 MTKFP。

1 基本概念

令 $I=\{i_1, i_2, \cdots, i_n\}$ 为一组不同字母的集合, 其中的元素称为项目(*item*)。对于集合 X, 如果 $X \subseteq I$ 并且 $k=|X|$ ($|X|$ 指项集 X 中项目的总数目), 则 X 称为 k 项集。事务 T 是一个项集, 因此对于任意事务 T, 存在 $T \subseteq I$。事务数据集 D 是事务的集合。对于给定项集 X, X 的支持度定义为 $Sup(X)=D_X/|D|$, 其中 D_X 是数据集 D 中包含项集 X 的事务数目, 也称为绝对支持度, $|D|$ 是数据集 D 中事务的总数目。

将所有已挖掘出的项集按照支持度由高到低的顺序排序, 令 θ 为第 k 个项集的支持度, 则称 θ 为边界支持度(*border _sup*)。另外, 称最初指定的边界支持度为初始边界支持度, 称最终的边界支持度为Top-K 边界支持度。所有支持度大于或等于 Top-K 边界支持度的项集组成 Top-K 最频繁模式。

2 MTKFP 算法

令数据集为 D, 数据集中不同项目的数目为 $|I|$, 需要挖掘的Top-K 最频繁项集的数目为 k。MTKFP 算法利用图 1 所示的 GETSHORT 过程获得短项集, 其中 λ是一个正整数阈值(例如 5 或 10 等)。GETSHORT 过程首先获得所有 1- 项集并按照支持度下降顺序排序, 将第 k 个 1- 项集的支持度作为 *border _sup*, 然后取所有支持度大于或等于 *border _sup* 的1- 项集存放于 *itemsets _array* 中, 并且将 L_1 中支持度小于 *border _sup* 的所有项集删除(第 1, 2 行)。

过程 GETSHORT(D, k, λ, $|I|$)
输入: 数据集 D, Top-K 最频繁项集的数目 k, 正整数阈值 λ, 数据集中不同项目的数目 $|I|$
输出: 存储了最频繁的 k 个短项集的数组 *itemsets _array*, 短项集的最大长度 *Maxlength*

```
1  搜索一遍数据集 D, 统计所有 1- 项集 L1 及它们的支持度
2  将 L1 按照支持度下降顺序排序, 选取第 k 个项集的支持度作为 border _sup, 将支持度大于或等于 border _sup 的项集存储于 itemsets _array 中, 并将 L1 中支持度小于 border _sup 的所有项集删除
3  for( i= 2; L1 ≠ Φ; i++ )
4      根据 L(i-1), 利用 Apriori 算法的连接操作获得 Ci
5      搜索一遍数据集 D 获得 Ci 中项集的支持度
6      Li= { c ∈ Ci | c.count ≥ border _sup };
7      将 itemsets _array 中的项集和 Li 中的项集统一按照支持度下降顺序排序, 选取第 k 个项集的支持度作为 border _sup, 将支持度大于或等于 border _sup 的项集存储于 itemsets _array 中
8      if border _sup > 0 && Σ_{m=1}^{i} C(m,|I|) > λk  then
9          Maxlength= i;
10         return { intemset _array, Maxlength};
11     取 Li 为 itemsets _array 中长度等于 i 的所有项集
12 end for
13 Maxlength= i- 2;
14 return itemsets _array, Maxlength;
```

图 1 挖掘短项集的 GETSHORT 过程

Fig. 1 Procedure GETSHORT for mining short itemsets

对于长度大于 1 的 $i-$ 项集, 根据长度为 $i-1$ 的项集 L_{i-1}, 利用 Apriori 算法[4] 的连接操作获得 $i-$

项集的候选 C_i(第 4 行),然后通过搜索一遍数据集获得 C_i 中项集的支持度,并选取支持度大于 *border _sup* 的 $i-$ 项集 L_i(第 5,6 行);将 *itemsets _array* 中的项集和 L_i 统一排序,并更新 *border _sup*,然后选取支持度大于或等于 *border _sup* 项集存储于 *itemsets _array* 中(第 7 行)。如果 *border _sup* > 0(即 *itemsets _array* 中项集数目大于 k),并且 $\sum_{m=1}^{i} C_{|I|}^{m} > k$($k$($\sum_{m=1}^{i} C_{|I|}^{m}$ 为长度小于等于 i 的项集的最大可能数目,即要求该数目大于用户指定阈值),则返回 *itemsets _array* 和 *itemsets _array* 中项集的最大长度 i(第 8~ 10 行);否则根据 *itemsets _array* 中长度为 i 的项集继续挖掘长度为 $i+1$ 的项集。这里长度为 $i+1$ 的项集仅需要根据 *itemsets _array* 中长度为 i 的项集连接生成,其正确性易用 Top-K 最频繁项集的性质证明。另外,GETSHORT 过程至少需要挖掘 1- 项集和 2- 项集,而 1- 项集和 2- 项集的总数目为 $|I| + C_{|I|}^{2}$。如果存储器满足最大存储 $|I| + C_{|I|}^{2}$ 个项集的需求,则利用数组或 *hash* 表技术,可以通过一遍扫描数据集完成 1- 项集和 2- 项集的挖掘,从而加快挖掘速度;否则利用图 1 中描述的过程处理。

挖掘出短项集之后,根据短项集确定挖掘长项集所需要的项目范围,以及初始 *border _sup*,然后根据深度优先搜索挖掘长项集。MTKFP 算法的全过程如图 2 所示。

算法 MTKFP(D, k)
输入:数据集 D,模式的数目 k
输出:Top-K 最频繁模式

1 *itemsets _array*, *Maxlength* = GETSHORT(D, k)
2 if *itemsets _array* 数组中的项集数目少于 k then return *itemsets _array*
3 将初始边界支持度 *border _sup* 设置为 *itemsets _array* 数组中最小的支持度
4 获取 *itemsets _array* 数组中长度为 *Maxlength* 的所有项集 $L_{Maxlength}$
5 获取 $L_{Maxlength}$ 中项集的所有不同项目,并按支持度下降顺序组成 *F-list*
6 搜索一遍数据集 D,根据 *F-lish* 建立 *FP-Tree*
7 根据建立的 *FP-Tree* 和初始边界支持度 *border _sup*,利用深度优先搜索挖掘长度大于 *Maxlength* 的项集,并与 *itemsets _array* 数组中的项集相结合,获得所有 *Top-K* 最频繁模式

图 2 MTKFP 算法

Fig. 2 MTKFP algorithm

MTKFP 算法首先获取短项集(第 1 行)。如果短项集的数目少于 k,则表明数据集 D 中的项集数目不足 k,此时 *itemsets _array* 数组中的项集即是 Top-K 最频繁项集,因此算法返回(第 2 行)。否则继续挖掘长度大于 *Maxlength* 的项集。由于 GETSHORT 过程已经挖掘了所有长度小于等于 *Maxlength* 的最频繁的 Top-K 短项集,因此若还有其它项集属于 Top-K 最频繁项集,则其长度必然大于 *Maxlength*。令 *itemsets _array* 数组中长度为 *Maxlength* 的项集的集合为 $L_{Maxlength}$,则若长度大于 *Maxlength* 的项集属于 Top-K 最频繁项集,则其只能由 $L_{Maxlength}$ 中项集的所有不同项目组成。同样,其正确性易用 Top-K 最频繁项集的性质证明。因此,MTKFP 算法仅根据 $L_{Maxlength}$ 中项集的所有不同项目生成 *F-list*,并根据该 *F-list* 生成 *FP-Tree*(第 3- 6 行),另外使用 *itemsets _array* 数组中的最小支持度作为初始边界支持度 *border _sup*。最后,根据深度优先搜索挖掘长度大于 *Maxlength* 的项集,并与 *itemsets _array* 数组中项集相结合,获得所有的 Top-K 频繁项集(第 7 行)。

3 实验评测

为了评测 MTKFP 算法,我们在不同数据集下比较了它与 Top-K FP-growth 算法和 ExMiner 算法的不同性能。为此,我们利用 C++ 语言实现了 MTKFP 算法,并根据文献[1]思想实现了 Top-K FP-growth 算法;ExMiner 算法的源码由其作者 Tran Minh Quang[2] 提供。利用 g++ 2.96 编译器编译这些算法。实验用的机器配置为:奔 IV/2GHz 的 CPU,内存大小为 2GB,操作系统为 Redhat Linux 7.3。

图 3 和图 4 分别比较了数据集 T10I4D1000kN1000k 和数据集 connect4 下三个算法的运行时间。其中 T10I4D1000kN1000k 数据集采用 IBM 数据生成器[4] 生成,该数据集中的不同项目的数目 N 是 1000k。

connect4 是一个真实、稠密的数据集, 共有 67557 个事务, 事务集中不同项目的数目为 128, 下载于[5]。分别比较了 k 为 1000 到 7000 时三个算法的运行总时间。其中 MTKFP 算法的 λ参数设置为 10。

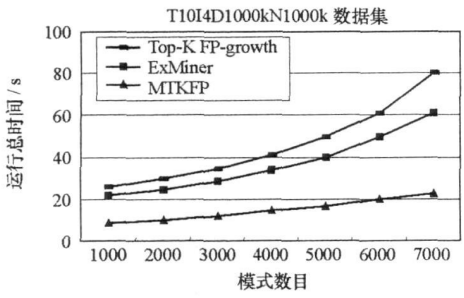

图 3 T10I4D1000kN1000k 数据集下的运行时间

Fig. 3 Running time comparisons for T10I4D1000kN1000k

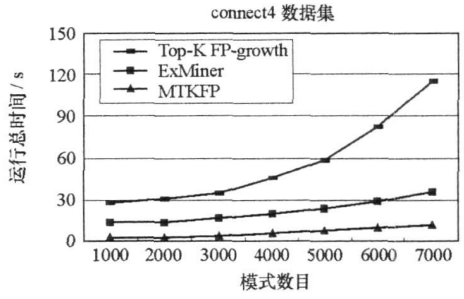

图 4 Connect4 数据集下的运行时间

Fig. 4 Running time comparisons for connect4

根据图 3 和图 4, MTKFP 算法的总运行时间远低于其它两个算法。算法的效率与初始项目数目和初始边界支持度 *border _sup* 紧密相关。表 2 和表 3 中分别给出了 T10I4D1000kN1000k 数据集下和 connect4 数据集下, MTKFP 算法与其它两个算法(即 ExMiner 算法和 Top-K FP-growth 算法) 的初始项目总数和初始边界支持度(绝对支持度) 的比较。在两个数据集下, MTKFP 算法的初始项目总数至少低于其它两个算法 70%, 而 MTKFP 算法的初始边界支持度高于其它两个算法。

表 2 T10I4D1000kN1000k 数据集下的初始参数比较

Tab. 2 Original parameters comparisons for T10I4D1000kN1000k dataset

内 容 \ k		1000	2000	3000	4000	5000	6000	7000
初始项目数目	others	1000	2000	3000	4000	5000	6000	7000
	MTKFP	236	559	851	1190	1499	1815	2117
初始 border _sup	others	1118	889	759	668	596	532	482
	MTKFP	1317	1105	986	908	839	786	745

表 3 Connect4 数据集下的初始参数比较

Tab. 3 Original parameters comparisons for connect4 dataset

内 容 \ k		1000	2000	3000	4000	5000	6000	7000
初始项目数目	others	128	128	128	128	128	128	128
	MTKFP	21	25	29	32	34	37	38
初始 border _sup	others	0	0	0	0	0	0	0
	MTKFP	60909	56016	51448	46359	41578	37650	33648

4 结 论

本文提出了挖掘 Top-K 最频繁模式的 MTKFP 算法。该算法是一种综合了宽度优先与深度优先搜索的混合算法。实验表明, 这种方式所获得的初始项目总数和初始边界支持度均优于已有算法, 并且 MTKFP 算法的速度快于同类最好算法。

参 考 文 献:

[1] Hirate Y, Iwahashi E, Yamana H. TF^{2}P-growth: An Efficient Algorithm for Mining Frequent Patterns without Any Thresholds[C]// Proc. of ICDM04. Brighton, UK, October, 2004.

[2] Quang T M, Oyanagi S, Yamazaki K. ExMiner: An Efficient Algorithm for Mining Top-K Frequent Patterns[C]// LNAI 4093, Springer-verlag, Berlin, Heidelberg, 2006, 436- 447.

[3] Quang T M, Oyanagi S, Yamazaki K. Mining the K-most Interesting Frequent Patterns Sequentially[C]// Proc. of Intelligent Data Engineering and Automated Learning (IDEAL06), 2006, 620- 628.

[4] Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules[C]// Proc. Of the 20th Intl. Conf. on Very Large Databases (VLDB' 94), Santiago, Chile, Sept, 1994, 487- 499.

[5] Goethals B. The FIMI Repository[EB]. http://fimi.cs.helsinki.fi, 2003.