基于改进倒排表和集合的最频繁项集挖掘算法*

陈小玉[1]，杨艳燕[1]，刘克成[1]，朱颢东[2]

(1. 南阳理工学院 计算机科学与技术系，河南 南阳 473004；2. 郑州轻工业学院 计算机与通信工程学院，郑州 450002)

摘　要： 最频繁项集挖掘是文本关联规则挖掘中研究的重点和难点，它决定了文本关联规则挖掘算法的性能。针对当前在最频繁项集挖掘方面的不足，将集合论引入倒排表以对其进行改进，然后以此为基础提出了几个命题和推论，并结合最小支持度阈值动态调整策略，提出了一个基于改进的倒排表和集合理论的最频繁项集挖掘算法，最后对所提算法进行验证。实验结果表明，所提算法的规则有效率和时间性能比常用的两个最频繁项集挖掘算法，即 NApriori 和 IntvMatrix 算法都好。

关键词： 最频繁项集；文本关联规则；倒排表；集合理论

中图分类号： TP301　　**文献标志码：** A　　**文章编号：** 1001-3695(2012)06-2135-03

doi:10.3969/j.issn.1001-3695.2012.06.035

Most frequent itemset mining algorithm based on improved inverted list and set theory

CHEN Xiao-yu, YANG Yan-yan, LIU Ke-cheng, ZHU Hao-dong[2]

(1. *Dept. of Computer Science & Technology, Nanyang Institute of Technology, Nanyang Henan* 473004, *China*; 2. *School of Computer Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou* 450002, *China*)

Abstract: Most frequent item sets mining is the focus and the difficulty of text association rules mining, and directly determines the performance of text association rules mining algorithms. Aiming at shortcomings existing in most frequent item sets mining algorithms, this paper improved traditional inverted list, it combined minimum support threshold dynamic adjustment strategy and presented a new most frequent itemset mining algorithm based on improved inverted list and set theory. In addition, it also offered several propositions and deductions which were used to improve the performance of the provided algorithm. Finally, through experiment testing, the provided algorithm is better in effective rate of rules and time performance than NApriori and IntvMatrix which are two frequently-used most frequent itemsets mining algorithms.

Key words: most frequent itemsets; text association rules; inverted list; set theory

频繁项集的规模决定了文本关联规则挖掘算法的性能，从而使得最频繁项集挖掘算法成为文本关联规则挖掘中的核心研究内容之一[1~8]。目前，已有众多学者提出了一些最频繁项集挖掘算法[1~8]。然而，这些算法仅通过指定一个固定数 N 来决定频繁项集的规模，其缺点如下：由于没有采用最小支持度，这就导致挖掘过程中对那些支持度较小的频繁项集进行处理，这必然导致挖掘算法性能的低下。文献[3]借用最小支持度动态调整策略，提出了一个 NApriori 算法，该算法首先指定一个全局最小支持度，然后用该支持度与每步频繁项集的支持度进行比较，如果全局最小支持度较小，就令它为当前步的最小支持度。但是，该算法仍然需要多次扫描事务数据库，时间复杂度较高。文献[4]提出了一个 IntvMatrix 算法，虽然该算法采用倒排矩阵来快速生成频繁项集，但是该算法仍然需要多次扫描倒排矩阵，而且倒排矩阵中充满空元素。为此，本文借用文献[3]中动态调整支持度策略，提出了一种基于改进倒排表和集合的 Top-N 最频繁项集挖掘算法。

1　基础知识

定义 1　Top-N 最频繁项集。按照支持度从大到小的顺序对全部项集进行排序，假设第 N 个项集的支持度为 NS，则[3]

$$\text{Top-}N=\{X\mid \text{support}(X)\geqslant NS\} \tag{1}$$

如果第 $N+1, N+2, \cdots, N+m$ 个项集的支持度也等于 NS，则 Top-N 中的最频繁项集可能多于 N 个。

定义 2　Top-N_k 最频繁。按照支持度从大到小的顺序对把全部 k-项集进行排序，假设第 N 位的 k-项集的支持度为 NS_k，则[3]

$$\text{Top-}N_k=\{X\mid \text{support}(X)\geqslant NS_k \text{ 且 } |X|=k\} \tag{2}$$

命题 1　Top-N_k 的最小支持度 NS_k 不大于 Top-N 的最小支持度 NS[3]。

推论 1　Top-N 与 Top-N_k 的关系为

$$\text{Top-}N\subseteq\{\bigcup_k \text{Top-}N_k\}\text{[3]}$$

依据推论 1 可知，$\{\bigcup_k \text{Top-}N_k\}$ 是 Top-N 的候选项集。因

收稿日期： 2011-11-04；**修回日期：** 2011-12-12　　**基金项目：** 河南省教育厅自然科学研究指导计划项目(2010C520007)

作者简介： 陈小玉(1978-)，女，河南南阳人，讲师，硕士，CCF 会员，主要研究方向为数据挖掘、粗糙集、智能计算(rain 1244@126.com)；杨艳燕(1979-)，女，陕西横山水，讲师，主要研究方向为计算机应用、网络安全；刘克成(1962-)，男，陕西横山人，教授，主要研究方向为计算智能、人工智能；朱颢东(1980-)，男，博士，主要研究方向为软件过程技术与方法、文本挖掘等.

此,在获取 Top-N 最频繁项集时,可以利用 Itemset-loop 算法生成$\{\bigcup_k \text{Top-}N_k\}$,并从中选取支持度最大的前 N 个项集组成 Top-N,此时在一定程度上能够提高算法效率。

命题2 假设 Top-N_k 的支持度阈值为 NS_k,Top-N_{k-1} 支持度阈值为 NS_{k-1},则 $NS_k \geqslant NS_{k-1}$[3]。

由命题2可知,在产生 Top-N 时,当前最小支持度阈值能够逐渐增大,这能够减少候选项集的规模,进而提高算法效率。

2 改进的倒排表及其相关推论

倒排表包括词表和文档表两部分,其中,词表由文档集中的特征词组成,文档表的一行由词表中特征词的文档 ID 组成。例如,表1为一个文档事务数据库,表2是相应的倒排表。

表1 文档事务数据库

ID	items(features)						
1	n	b	a	m			
2	l	q	a		q	l	q
3	k	l	m	n	o		
4	h	c	g	i			
5	k	a	e	i	c		
6	a	b	c	e	i		
7	f	a	h	g	j	p	q
8	b	c	d	a	e		
9	a	c	e	h	g	r	
10	a	b	c	d	e		

表2 相应的倒排表

Items	ID							
a	10	9	8	7	6	5	2	1
b	10	8	6	1				
c	10	9	8	6	5			
d	10	8						
e	10	9	8	6	5	4		
f	7							
g	9	7	4					
h	9	7	4					
i	6	5	4					
j	7							
k	5	3						
l	3	2						
m	3	1						
n	3	1						
o	3							
p	7							
q	7	2						
r	9							

倒排表能够提高1-项集的查找效率,但是很难获得 k-项集($k \leqslant 2$)的支持度。这是由于倒排表孤立了同一事务中的各个项。从表2可以看出,倒排表存在大量空元素,这极大地浪费了内存空间。鉴于上述不足,本文对倒排表进行了改进:

a)词表中各项按其文档频降序排序,并用相应的序号表明其位置。此时,序号、项名、指向文本事务集合的指针三部分组成词表。

b)文档表中每一行为一个集合,其元素由特征所在的文本事务号组成。此时文档表表中各项按照所包含的元素个数从大到小的顺序排列。表3为改进的倒排表。

推论2 设 T_1、T_2 为项集,T_1 所在的全部文档集为 D_1,T_2 所在的全部文档集为 D_2,则对项集 $T = T_1 \cup T_2$,T 所在的全部文档集为 $D = D_1 \cap D_2$。

证明 1) $\forall d \supseteq T$,由 $T = T_1 \cup T_2$ 可得 $T_1 \subseteq T$,所以 $T_1 \subseteq d$。又由 D_1 是包含 T_1 的全部文档集,故 $d \subseteq D_1$。同样道理也有 $d \subseteq D_2$,所以 $d \subseteq D = D_1 \cap D_2$。

表3 改进的倒排表

项号	项名	指针所指向的集合	项号	项名	指针所指向的集合
1	a	{10,9,8,7,6,5,2,1}	10	l	{3,2}
2	e	{10. 9. 8. 6. 5. 4}	11	m	{3,1}
3	c	{10,9,8,6,5}	12	n	{3,1}
4	b	{10,8,6,1}	13	q	{7,2}
5	g	{9,7,4}	14	r	{9}
6	h	{9,7,4}	15	f	{7}
7	i	{6,5,4}	16	p	{7}
8	d	{10,8}	17	j	{7}
9	k	{5,3}	18	o	{3}

2) $\forall d \in D$,由 $D = D_1 \cap D_2$ 可得 $d \in D_1$,因为 $T_1 \subseteq D_1$,所以 $T_1 \subseteq d$,同样道理也有 $T_2 \subseteq d$。因此,$D = D_1 \cap D_2$ 包含 $T = T_1 \cup T_2$。推论得证。

依据推论2,连接操作时就不需要再次扫描数据库。例如 a 与 b 连接时只需对相应的集合取交集而无须再次扫描数据库,这也就提高了算法效率。a 与 b 连接结果如表4所示。

表4 a 与 e 连接结果

项号	项名	集合
19	$a \sim b$	{10,5,3,1}

推论3 设 L_x 为 K-频繁项集的集合,如果 L_x 中的项集个数$\leqslant K$,则 L_x 为极大频繁项集。

证明 由经典 Apriori 算法特性"任何强项集的子集必定是强项集"可知,如果 $\exists L_{k+1}$(即 $K+1$ 频繁项集的集合),则 $\forall l_{k+1} \in L_{k+1}$,$l_{k+1}$一定有 $K+1$ 个 k-频繁子集在 L_x 中,因此,如果 L_x 的项集个数$\leqslant K$,则必定不能生成 L_{k+1}。推论得证。

依据推论3,在生成 $k+1$-项候选频繁项集之前,先判断 k-项频繁集中项集的个数是否$\leqslant k$,如果是,则算法可以终止,这样也能提高算法效率。

推论4 设 $\forall l_k \in L_k$(L_k 为 k-频繁集),$\forall \text{Item} \in l_k$,如果 Item在 L_x 中的支持数小于 K,则 l_k 不能用于生成 L_{k+1}。

证明 由经典 Apriori 算法特性"任何强项集的子集必定是强项集"可知,$\forall l_{k+1} \in L_{k+1}$,$l_{k+1}$中必 $\exists K+1$ 个 K-频繁项集 $\in L_x$。明显有 $\forall \text{Item} \in l_k$,在 l_k 的 $K+1$ 个 K-频繁项集中,Item 的支持数$\geqslant K$。所以 $\exists \text{Item} \in l_k$,且 Item 在 L_x 中的支持 $< K$,则 l_k 不能用来生成 L_{k+1}。推论得证。

依据推论4,若某个 k-项频繁集中的一个项在其中出现的次数少于 k,则在产生 $k+1$-项频繁项集时该项集是无用的,此项集可以直接删除,这就减少了候选频繁集的规模,从而提高了算法效率。

3 本文所提算法

3.1 算法思想简述

在文献[3]动态调整支持度策略的基础上,借用文中所给的命题和结论,把倒排表和集合结合起来,从而得到一个 Top-N 最频繁项集挖掘算法。

3.2 算法步骤

输入:文本事务数据库 D,最小支持度 δ_0(初始值为1),频繁项集数 N。

输出:Top-N 最频繁项集。

a)对 D 进行扫描以生成改进的倒排表 W。

b)若 W 中各项的频率 $<N$,则令 $\delta=\delta_0$(δ 为当前最小支持度);否则,令 $\delta=\max\{\delta_0,\delta_N\}$($\delta_N$ 为倒数第 N 个项的支持度),以支持度大于 δ 的前 N 项组成 L_1(L_1 为 1-频繁项集,以改进的倒排表形式表示)。

c)for ($k=2$;count (L_{k-1}) $\geqslant k$;$k++$)

$\{L'_{k-1}=\text{delete}(L_{k-1})$ */依据推论 4 从 L_{K-1} 中删除不能产生频繁 k-项集的 k-1-项集,从而减少 k-项候选集,提高算法效率 */

$C_K=\text{candidate\_gen}(L'_{k-1})$

若 C_K 中元素个数小于 N,则 $\delta=\delta_0$;否则,$\delta=\max\{\delta_0,\delta_{kn}\}$ 作为支持度(其中 δ_{kn} 为倒数第 N 个项的支持度),支持数不小于 δ 的前 N 项组成 L_k(L_k 为 k-频繁项集,以改进的倒排表形式表示)。

d)令 $L=\bigcup_k L_k$,按照从大到小的顺序对 L 中的项集进行排序,并取前 N 个频繁项集输出,算法结束。

4　实例验证

4.1　实验验证

实验数据由 2009 年—2011 年新华网上的 1 200 篇新闻材料组成,对这些新闻材料预处理后,进行频繁项集挖掘实验。

实验中使用 Weka 软件作为实验工具,该软件工具是 New Zealand 的怀卡托大学开发的一款数据挖掘软件,它由 Java 语言编写,由一系列机器学习算法组成。该软件的相关算法使用十分简单,可以直接调用,也镶嵌在代码使用。Weka 软件工具包含如下功能:数据预处理、分类、回归分析、聚类、关联规则、可视化等,它对研究数据挖掘和机器学习十分有用。该软件是免费的,可在网上下载。

实验中使用 MATLAB 作为计算工具,它是当今国际上科学界(尤其是自动控制领域)最具影响力、也是最有活力的软件。它提供了灵活的程序设计流程、强大的科学运算、高质量的图形可视化与界面设计、便捷地与其他程序和语言接口的功能,目前已广泛应用于工程计算、控制设计、信号处理与通信、图像处理、信号检测、金融建模设计与分析等领域。

实验主要目的就是比较本文算法与文献[3]提出的 NApriori 算法和文献[4]提出的 IntvMatrix 算法的性能差异。为此,进行如下两个实验:a)比较这三种算法在不同规模的频繁项集上进行挖掘时所消耗的时间。从实验结果中选取差别比较明显的数据画了性能对比图,结果如图 1 所示(纵轴代表所花费的时间,单位为 s;横轴代表所挖掘的频繁项集数目);b)比较这三种算法在所选择数据集上挖掘出的规则数目以及有效规则数,结果如表 5 所示。

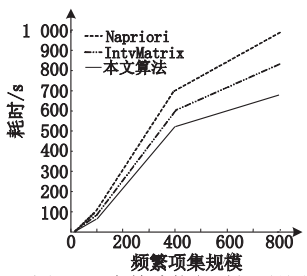

图1　三个算法执行时间对比结果

表 5　三个算法规则有效率实验结果对比

算法	挖掘出的规则数	有效规则数	有效率/%
本文算法	459	454	98.91
IntvMatrix	529	438	82.80
NApriori	701	441	62.91

4.2　实验分析

表 5 表明,本文算法规则有效率较高,IntvMatrix 算法规则有效率次之。由于 NApriori 算法产生了大量无意义的规则,规则有效率最低。图 1 表明,在频繁项集规模相同的情况下,本文算法时间性能明显优于 IntvMatrix 和 NApriori 算法。

经分析,产生上述结果的原因如下:

a)本文算法对传统倒排表进行了改进并引入了集合理论,这大大提高了算法的检索速度。

b)在本文算法中,利用改进的倒排表以及集合来组织文档词,使得算法整个过程仅需扫描一次事务数据库,这大大提高了算法性能。

c)依据推论 1,如果 k-项候选集中存在 k-1 子集包含非频繁项集,则该 k-项候选集被删除,从而减少了候选频繁集的个数,这能够提高算法性能。

d)依据推论 2,在实现频繁项集连接及事务连接时,只需要利用集合论求交集即可,而不需要再次扫描事务数据库,这也大大提高了算法性能。

e)依据推论 3 可知,若 L_{k-1} 中频繁项集的个数 $<k$,则算法不再产生 L_k,此时算法可以提前结束,这也能够进一步提高算法性能。

f)根据推论 4,如果 L_{k-1} 中存在项数 $<k-1$ 的项集,就可以将该项集删除,这在生成 k-项候选集时就能大大降低其规模,从而在一定程度上解决候选项集瓶颈问题,这也能够提高算法性能。

5　结束语

针对目前在频繁项集挖掘方面的不足,本文利用集合论对传统倒排表进行改进并提出几个命题和推论,以此为基础并结合最小支持度阈值动态调整策略提出了一种新的最频繁项集挖掘算法。从对比实验来看,本文所提算法在规则有效率和时间性能这两方面,均优于 NApriori 和 IntvMatrix 算法,使本文算法在文本关联规则挖掘中有一定的应用价值。

参考文献:

[1] 张文煜,周满元. 数据流中一种基于滑动窗口的前 K 个频繁项集挖掘算法[J]. 计算机应用研究,2011,28(7):2519-2521.

[2] 赖军,李双庆. 挖掘滑动时间衰减窗口中网络流频繁项集[J]. 计算机应用研究,2011,28(3): 895-898.

[3] FU A W C, KWONG R W, TANG Jian. Mining N-most interesting itemsets[J]. ISMIS,2000,12(4): 41-48.

[4] 战力强, 刘大昕. 频繁项集快速挖掘算法研究[J]. 哈尔滨工程大学学报,2008,29(3):266-271.

[5] 周勇,韩君,程春田. 滑动窗口中近期数据流频繁项集挖掘[J]. 计算机工程与设计,2011,32(4):1307-1310.

[6] WU Fan, CHIANG S W, LIN J R. A new approach to mine frequent patterns using item-transformation methods[J]. Information Systems,2007,32(7):1056-1072.

[7] 花红娟,张健,陈少华. 基于频繁模式树的约束最大频繁项集挖掘算法[J]. 计算机工程,2011,37(9):78-80.

[8] 黄红星,王秀丽,黄习培. 挖掘最大频繁项集的改进蚁群算法[J]. 计算机工程与应用,2011,47(13):161-165.