基于业务路径和频度矩阵的关联规则挖掘算法

胡 波 黄 宁 仵伟强
(北京航空航天大学可靠性与系统工程学院 北京 100191)

摘 要 关联规则挖掘为分析机载网络关联故障及提高排故效率提供了重要方法。分析了经典Apriori算法的局限性,结合机载网络领域知识、矩阵运算和频繁项集性质,提出一种高效的关联规则挖掘算法。应用机载网络故障具有的基于业务路径的关联特征,提出分块挖掘策略,从而实现挖掘过程的噪声隔离。提出频度矩阵和特征向量,结合矩阵特点和频繁项集性质,设计5个扫描策略,从而减少了循环次数和对比运算。与Apriori算法相比,新算法能有效提高频繁项集的搜索速率。

关键词 关联规则,关联故障,业务路径,分块挖掘,频度矩阵
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.12.026

Algorithm for Mining Association Rules Based on Application Paths and Frequency Matrix

HU Bo HUANG Ning WU Wei-qiang
(School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China)

Abstract Association rule mining is an important method to analyze the associated faults of the airborne network and improve the efficiency of faults diagnosis process. This paper analyzed the limitations of the classical Apriori algorithm, and proposed an efficient association rule mining algorithm, which is based on the knowledge of the airborne network, matrix operation and frequent item sets. Due to the association characteristics of the airborne network faults based on the application paths, this paper proposed a mining strategy of block mining, so as to realize the noise isolation in mining process. With the conception of frequency matrix and feature vector, 5 kinds of scanning strategies were proposed, thereby reducing the number of cycles and the comparison operation. Comparing with the classical Apriori algorithm, the new algorithm can effectively improve the search efficiency of frequent itemsets.

Keywords Association rules, Association faults, Application paths, Block mining, Frequency matrix

1 引言

飞机上的各种航电设备通过端系统与交换机连接,形成了由若干个以交换机为中心的星型子网组成的机载网络系统[1]。近半个世纪来,在经历分散、联合、综合、高度综合4个阶段[2-4]后,机载网络的综合化趋势导致设备间关系复杂,相互的影响越来越密切,故障间呈现出复杂的关联关系:一个故障往往引发多个其他故障,这些故障动态依赖,不断关联触发。在工程中,面对同时发生的多个故障,在其关联关系未知的情况下,工程人员只能逐一排查,排故效率低下。

机载网络运行中保存了大量故障数据,蕴含着故障的所有关联信息,但因故障数据库巨大,靠人工逐条分析寻找是不现实的。而关联规则挖掘的主要目的就是从大数据对象中,通过历史信息,挖掘个体之间隐含的或不容易被发现的关联关系。利用关联规则挖掘,可以有效地从故障数据库中找出机载网络故障的关联关系。

然而,当前挖掘算法无法很好地找出故障关联关系。在机载网络故障数据库中,由于故障背后的业务千变万化,导致相比有效信息,噪声信息比重极大,难以形成类似通信网络"告警风暴"那么明显的事件关联现象。而传统关联规则挖掘算法主要是基于经典Apriori算法进行改进,如散列、事务压缩、选样、动态项集等方法[5-8],虽能在一定程度上解决了效率低的问题,但当其应用于类似机载网络故障数据库时,其挖掘效率仍难以满足工程需求。

2 问题分析

2.1 关联规则

关联规则挖掘是指从事务数据库等信息存储中发现有趣的关联关系的过程[9]。R. Agrawal在1993年提出并阐明了关联规则的含义。

设 $I=\{i_1,i_2,\cdots,i_n\}$ 为数据项集,由 n 个不同数据项 i_k $(k=1,2,\cdots,n)$ 构成,设子集 $T\subseteq I$ 为事务,不同事务构成数据库 D。关联规则:$A\Rightarrow B$,支持度为 $Sup(A\Rightarrow B)$,置信度为 $Conf(A\Rightarrow B)$,表示若某事务中发生 A,则会以一定概率发生 B。其中,A 称为前件,B 称为后件,且 $A\subset I$,$B\subset I$,$A\cap B=\varnothing$。

到稿日期:2015-12-08 返修日期:2016-04-11

胡 波(1990—),男,硕士生,主要研究方向为网络可靠性,E-mail:syhxb@buaa.edu.cn;黄 宁(1968—),女,博士,博士生导师,CCF高级会员,主要研究方向为网络可靠性、软件测试、可靠性;许伟强(1986—),男,博士生,主要研究方向为网络可靠性及网络性能故障和规律。

支持度 $Sup(A\Rightarrow B)=\frac{Count(A\cup B)}{|D|}\times 100\%=P(A\cup B)$，表示 AB 同时出现的概率；$Count(A\cup B)$ 表示同时包含 AB 的事务个数；$|D|$ 表示事务库大小。

置信度 $Conf(A\Rightarrow B)=\frac{Sup(A\cup B)}{Sup(A)}\times 100\%=P(B|A)$，表示一个事务在 A 发生的条件下 B 发生的概率。

对于给定的最小支持度 $Sup_{\min}$ 和最小置信度 $Conf_{\min}$，若 $Sup(A\Rightarrow B)\geqslant Sup_{\min}$，且 $Conf(A\Rightarrow B)\geqslant Conf_{\min}$，则 $A\Rightarrow B$ 为强关联规则。若 $Sup(I_k)\geqslant Sup_{\min}$，则 I_k 为频繁项集 L_k。能否快速找出所有频繁项集决定了关联规则挖掘的效率[9]。

2.2 Apriori 算法

Apriori 算法是由 R. Agrawal 等提出的经典布尔关联规则挖掘算法[10]。算法基于层次迭代思想，根据最小支持度 $Sup_{\min}$，由 L_{k-1} 到 L_k 经过连接和剪枝，逐层搜索迭代，直至再没有满足 $Sup_{\min}$ 的项集出现，挖掘出频繁项集，然后基于频繁项集生成强关联规则。算法简单实用，可信度高，适用性强，其过程如图 1 所示。

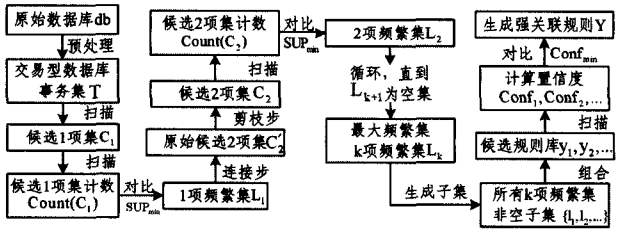

图 1　Apriori 算法流程

其中，连接步将 L_{k-1} 各项组合为候选 k 项集 C_k，剪枝步删去具有非频繁项集子集的 C_k，从而确定 k 项频繁集 L_k。由图 1 可以看出，生成 L_k 至少需要 $3k$ 次对全库 T 扫描（外存）和 k 次连接、剪枝循环，每次循环都需要无差别组合生成候选项集，再对比消除噪声信息。显然，事务库大小和噪声比重直接影响到挖掘效率，组合与消除计算量的大小也直接由事务库决定。

然而，机载网络系统测试业务较多，且数据库噪声比重较大，故障 D 产生不会形成类似通信网络“告警风暴”的集中爆发现象，若将 Apriori 算法应用于机载网络故障关联分析，将导致去噪不彻底、扫描速率慢等问题。此外，受噪声干扰，机载网络关联故障时间相关性不明显，且工程中时间信息不够完善，故通信网络告警相关性分析中常用的时间窗口和滑动步长法也不适用[11]。

目前对通用算法的优化主要包括：针对数据库扫描的优化，如散列技术[5]、压缩项集[6]、事务集编号[12]等；针对连接、剪枝步的优化，如动态项集计数[8]、非频繁项集剪枝及提高连接步效率[13]等；抽样方法，牺牲准确性以提高效率[7]。然而，这些优化算法均无法解决机载网络关联故障事务数较多且相关性不明显的问题，难以带来挖掘效率的显著提高。

3　基于极大业务路径的分块挖掘策略

机载网络故障具有基于业务路径的关联特征，分析该性质，提出极大业务路径的概念，进一步提出分块挖掘策略。

3.1　基于业务路径的关联特征

机载网络故障间存在复杂的动态关联关系，是因为系统运行中以航电业务为流程的各应用间存在动态逻辑连接关系，导致在一定试验应力下某故障发生时，在当前业务路径上关联触发新的故障，且随着业务路径的动态关联，故障呈现出动态依赖和不断传播的特点[1]。业务路径、动态性和关联性定义如下。

（1）业务路径：在一定试验应力下，航电业务流程上各应用逻辑的连接关系。例如视景增强控制业务，在特定设备加电的试验应力下，其业务路径为：视景增强→RDC→综合处理机→显示处理单元。图 2 是加载部分航电业务的某机载网络某时刻的业务路径示意图。

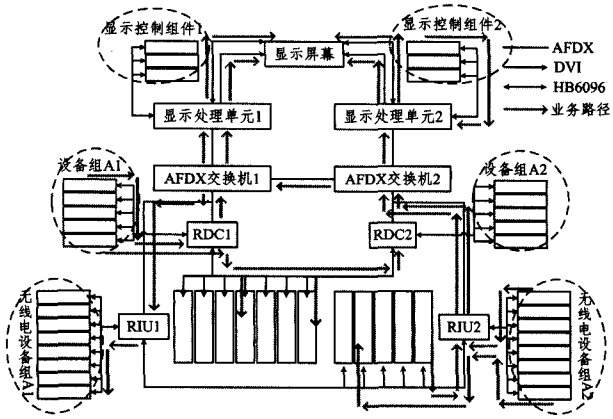

图 2　机载网络业务路径示意图

（2）动态性：同样试验应力下，故障的产生与业务路径密切相关，业务路径的动态性导致故障的动态性。如图 3 所示，某研究所示的故障案例中，某测试业务可通过两条路径实现，路径 1：ADC1→DPU1；路径 2：ADC1→RDC1→DPU1。试验发现，ADC1 同样加电和进行数据发送，测试业务通过路径 1 时，DPU1 会发生视频信号抖动失准的故障，而通过路径 2 时，经过 RDC1 后 DPU1 并未发生此故障。显然，同样试验应力条件下，某些故障只有通过特定业务路径才会呈现出来。

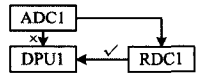

图 3　基于业务路径的故障动态性示意图

（3）关联性：同一测试业务中多个故障可能由同一故障导致，即同样试验应力和同样业务路径的故障之间具有关联性。图 4 示出了一个典型案例：故障 1 表现为总线数据监控设备监控到 AFDX 交换机转发数据出现丢包；故障 2 表现为显示屏幕没有无近地告警设备显示告警信号，两个故障的“业务路径”和“试验应力”信息相同。

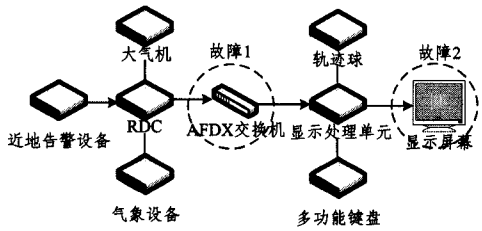

图 4　基于业务路径的故障关联性示意图

两故障设备分别为 AFDX 交换机与显示屏幕，其失效过程为：AFDX 交换机数据丢包，导致业务路径上的数据传输中断，显示设备接收不到有效数据，从而无法进行信息显示。从

该例可以看出，故障数据库中具有同样试验应力条件、业务路径相同或相交的故障，其极有可能是关联触发的。

综合以上分析可知，机载网络故障具有基于业务路径的动态关联特点，其关联规则与故障属性“业务路径”和“试验应力”密切相关，具有如下性质。

性质 1 业务路径不相交、试验应力不同的两个故障，不可能是由直接关联触发的。

性质 2 间接关联触发的两个故障，可通过有限个直接关联触发的故障对发现其关联关系。

工程中具有类似故障特点的网络系统大量存在。利用该性质，可以在连接步中避免噪声的组合。

3.2 故障信息模型

以某研究所实际故障单为例，故障主要包括故障现象、故障设备、业务路径、应力条件等属性，根据挖掘算法需求简化故障属性，构建故障信息模型 F：

$$F=\{S,X,\vec{N},\Delta\}$$

其中，$S=\{S_1,S_2,\cdots,S_i,\cdots\}$ 为故障属性“试验 ID”，$X=\{X_1,X_2,\cdots,X_i,\cdots\}$ 为故障属性“试验应力”，$I=\{A,B,\cdots\}$ 为故障现象，$\vec{N}=\{\vec{N}_1,\vec{N}_2,\cdots,\vec{N}_i,\cdots\}$ 为故障属性“业务路径”，$\vec{N}_i=(n_j,n_k,\cdots)$ 为故障 i 的业务路径，其中 $n_j,n_k\in n$，$n=\{1,2,\cdots\}$ 为故障节点设备。各参数相应序列通过原始数据库中对应字段的不同值按字典排序映射形成。

例 1 设某网络拓扑如图 5 所示，1－7 为设备节点，研究故障数据库时发现，网络运行过程中 1,2,3,5,6,7 设备节点上常发生 $A-F$ 等 6 种故障。

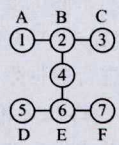

图 5 例 1 的网络拓扑图

根据故障信息模型，量化后的 12 个故障记录如表 1 所列。

表 1 预处理后的故障数据库

故障编号	试验 ID	试验应力	业务路径	故障现象
F_1	S_1	X_1	(1,2,3)	A
F_2	S_1	X_1	(2,3)	B
F_3	S_2	X_2	(1,2,3)	A
F_4	S_2	X_2	(2,3)	C
F_5	S_3	X_3	(2,3)	B
F_6	S_3	X_3	(2,3)	C
F_7	S_4	X_4	(5,6)	D
F_8	S_4	X_4	(5,6,7)	F
F_9	S_5	X_5	(5,6)	E
F_{10}	S_5	X_5	(5,6,7)	F
F_{11}	S_6	X_6	(5,6)	D
F_{12}	S_6	X_6	(5,6)	E

3.3 极大业务路径

根据性质 1 的结论可发现利用业务路径降低噪声比重、减少噪声组合的可行性，如例 2 所示。

例 2 基于表 1 所列的故障数据库，应用 Apriori 算法，以一次试验作为一项事务，形成如表 2 所列的事务集，通过演示由 L_1 生成 L_2 的过程，说明利用性质 1 大幅提高了挖掘效率的可行性。设最小支持度 $Sup_{\min}=2$。

表 2 交易型数据库——事务集

试验 ID	试验应力	业务路径	故障现象
S_1	X_1	(1,2,3)	AB
S_2	X_2	(1,2,3)	AC
S_3	X_3	(2,3)	BC
S_4	X_4	(5,6,7)	DF
S_5	X_5	(5,6,7)	EF
S_6	X_6	(5,6)	DE

(1)扫描事务数据库，生成 L_1。1 项频繁集列表如表 3 所列。

表 3 1 项频繁集列表

L_1	A	B	C	D	E	F
SUP	2	2	2	2	2	2

(2)进行第一次连接步计算，生成候选项集 C_2。2 项频繁集列表如表 4 所列。

表 4 2 项候选集列表

C_2	AB	AC	AD	AE	AF	BC	BD	BE
SUP	2	2	2	2	2	2	2	2

C_2	BF	CD	CE	CF	DE	DF	EF
SUP	2	2	2	2	2	2	2

观察 C_2，根据性质 1，对比表 2 可知，表 4 中灰色的故障组合(如 AD、AE 等)的其业务路径毫无相交，试验应力完全不同，不可能是关联故障，故不可能是频繁项集。这些组合完全是连接步生成的噪声信息。在搜索迭代中，算法每一步都会大量组合生成类似的噪声信息，然后逐个剪枝消除其中的大部分，极大降低了挖掘效率和准确度。根据性质 1 和例 2 进行分析可知，这部分噪声是可以避免的。

综合上述结论，提出基于极大业务路径的分块挖掘策略。

极大业务路径：遍历故障库中所有故障的试验应力和业务路径属性后，若业务路径 R_1-R_n 具有包含关系，且试验应力均为 X_i，其中 R_j 包含其余 $(n-1)$ 条业务路径，则 (X_i,R_j) 为一条极大业务路径。

定义业务路径向量 R_2 包含 R_1 的运算为 $R_1@R_2$：if $(X_1=X_2)$ & $Len(R_1)\leqslant Len(R_2)$，$\exists Part(R_2,Len(R_1))_i=R_1$，则 $R_1@R_2$。其中，$Part(R_i,k)$ 表示向量 R_i 长度为 k 的连续片段组成的集合($k\leqslant Len(R_i)$)，$Part(R_i,k)_j$ 表示该集合中的第 j 项($j=1,2,\cdots,Len(R_i)-k+1$)。

设 $R_1=(1,2,3,4)$，$Len(R_1)=4$，$Part(R_1,3)_2=(2,3,4)$，$Part(R_1,3)=\{(1,2,3),(2,3,4)\}$，若 $R_2=(2,3)$，$R_3=(3,4,5)$，且 $X_1=X_2=X_3$，则 $R_2@R_1$ 成立，$R_3@R_1$ 不成立。

例 3 根据定义 1，可知例 1 的故障数据库在遍历后可生成如下 4 条极大业务路径：

$$\{(X_1,(1,2,3)),(X_2,(2,3)),(X_3,(5,6,7)),(X_4,(5,6))\}$$

事实上，相同应力条件下，两个业务路径包含的故障极可能是直接关联的；相同应力条件下，两个业务路径具有一定重合关系的故障，即业务路径部分相交的故障，有可能是直接关联的，但随着试验集成化，遍历故障库后，极大业务路径更加完善，这种可能性会越来越小。

如图 6 所示，测试路径 1 上 AB 两故障直接关联，随着系统测试集成化，业务路径扩展为路径 2，则路径 2 上的故障 C 与 AB 很可能是有关联关系的。因两路径在同样应力条件下同时接通，且存在两故障关联发生的先例，故不能将具有包含关系的两业务路径分块。

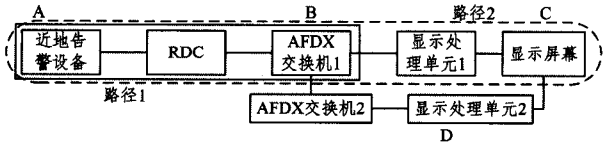

图 6　具有包含关系的业务路径示意图

具有相交关系的两路径也有可能存在故障关联关系，如图 7 所示，路径 1 的故障 AB 和路径 3 的故障 BD 存在一定关联关系，只是两个测试业务当前没有集成，即故障库中没有“近地告警到显示处理器 2”的测试业务路径上出故障的先例，故当前阶段还没有遍历出包含 ABD 的极大业务路径。但是随着试验的集成化越来越高，极大业务路径逐渐完善，最终“部分重合”的情况会越来越少。

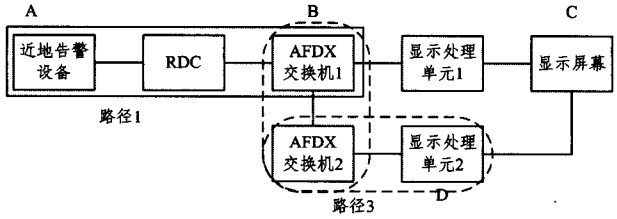

图 7　具有相交关系的业务路径示意图

若数据足够大，路径 1 和路径 3 仍未集成为一条业务路径，则故障 ABD 关联发生的概率是极小的。因为在系统完成几乎所有的常规任务后，两路径仍未集成为极大业务路径，可认为两路径同时接通在常规运行中是小概率事件。

综合以上分析，可以发现极大业务路径具有如下性质。

性质 3　两个直接关联触发的故障，以极大概率发生于同一极大业务路径下，并且随着试验的进行和故障库的增加，这一概率趋向于 100%。

根据性质 3，提出如下分块挖掘策略。

(1)生成事务集后，遍历数据库：

if: $\forall i=1,2,\cdots,n, R_i@R_n$, then: (X_n,R_n)

得到极大业务路径集合：

$\{(X_1,R_1),(X_2,R_2),\cdots,(X_k,R_k)\}$

(2)利用极大业务路径集合，将数据库分割为 k 块，如表 5 所列。

表 5　数据库分割为 k 块

极大业务路径	故障项集
(X_1,R_1)	$\{t_i\}\|_{R_i@R_1}$
…	…
(X_k,R_k)	$\{t_i\}\|_{R_i@R_k}$

(3)各分块挖掘操作隔离进行。事务频度的计算只在同一事务集分块内累加。如 3 个事务集分块中均出现 A，则各自计算频数，不同事务集分块中的项在连接步时不交叉组合。利用极大业务路径生成事务集分块的过程如例 4 所示。

例 4　设某网络拓扑如图 8 所示，1—7 为设备节点，研究故障数据库时发现，网络运行过程中 1,2,3,4,5,7 设备节点上常发生 $A-G$ 等 7 种故障，共 17 次故障记录。

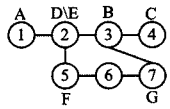

图 8　例 4 的网络拓扑图

算法生成的事务集如表 6 所列。

表 6　例 4 的故障事务集

试验 ID	应力条件	业务路径	故障现象
S_1	X_1	(1,2,3)	AB
S_2	X_1	(1,2,3,4)	ABC
S_3	X_1	(2,3,4)	BC
S_4	X_2	(1,2,3)	AD
S_5	X_2	(1,2,3)	E
S_6	X_1	(1,2,3,4)	BC
S_7	X_3	(1,2,5)	AF
S_8	X_3	(1,2,5,6,7)	AFG

数据库共 8 个事务，算法在连接步时需要全事务集组合交叉。事实上，需要组合的只有表 6 所列的 3 组，只有这 3 组内部才可能产生关联规则。因此，利用极大业务路径法进行分块，生成的事务集分块如表 7 所列。

表 7　分块后的事务集数据库

极大业务路径	故障项集
$(X_1,(1,2,3,4))$	{(AB),(ABC),(BC),(BC)}
$(X_2,(1,2,3))$	{(AD),(E)}
$(X_3,(1,2,5,6,7))$	{(AF),(AFG)}

表 7 中事务集进行搜索迭代时，无需对不同分块间的故障先组合后删减，各块挖掘相互隔离，同时进行。例如，虽然 3 个分块中均包含 A，但在连接步操作时，A 频数分开计算，而不同分块之间的项(如 B、C 与 D、E，以及 F、G 等)在连接步时不组合。对比分块前后，连接步生成 L_2 中所需候选 2 项集 C_2 的个数为 $C_7^2=21>3C_3^2=9$，挖掘效率显著提高。

4　基于频度矩阵的扫描策略

利用极大业务路径将数据库分块后，实现了噪声隔离，能有效提高挖掘效率。在循环扫描过程中，为提高各分块间搜索迭代效率，在分块的基础上提出频度矩阵和特征向量方法，将分块分解为特征向量和频度矩阵，利用矩阵运算特点和频繁项集性质来减少循环次数和对比计算量。

4.1　频度矩阵

特征向量：设某分块事务集 T_i 包含 m 个互不相同的事务 $t_1,t_2,\cdots,t_m$，由 n 个互不相同的项 $I_1,I_2,\cdots,I_n$ 构成，其中 $\{t_i\},\{I_i\}$ 均按字典排序，则称 n 维向量 $(I_1,I_2,\cdots,I_n)$ 为 T_i 的特征向量，记为 $\overrightarrow{It_i}$。本文以 A,B,C 等表示故障项，故 $I_i=A,B,C,\cdots$。

频度矩阵：设某分块事务集 T_i 包含 m 个互不相同的事务 $t_1,t_2,\cdots,t_m$，其特征向量为 $\overrightarrow{It_i}=(I_1,I_2,\cdots,I_n)$，则存在唯一矩阵 $Q_i=(q_{jk})_{n\times m}$，使得：

$$\overrightarrow{It_i}\times Q_i=(I_1,I_2,\cdots,I_n)\begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1m} \\ q_{21} & q_{22} & \cdots & q_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n1} & q_{n2} & \cdots & q_{nm} \end{pmatrix}\Leftrightarrow T_i$$

则称 Q_i 为 T_i 的频度矩阵。其中 $q_{jk}\in N$ 表示 T_i 中第 k 个事务 t_k 的第 j 项 I_j 的频数。

显然，一个事务集分块可分解为一个特征向量和一个频度矩阵。分块后，在搜索迭代中每个分块只需扫描一次特征向量并进行一次矩阵运算即可。

例 5　基于表 7 所列的事务集分块，利用频度矩阵可以将各块分解，最终的事务集分块与特征向量、频度矩阵如表 8 所列。

表 8　各分块频度矩阵及特征向量

事务集	特征向量	频度矩阵
T_1	(A,B,C)	$\begin{pmatrix}1&1&0\\1&1&2\\1&0&2\end{pmatrix}$
T_2	(A,D,E)	$\begin{pmatrix}1&0\\1&0\\0&1\end{pmatrix}$
T_3	(A,F,G)	$\begin{pmatrix}1&1\\1&1\\0&1\end{pmatrix}$

其中：

$$(A,B,C)\begin{pmatrix}1&1&0\\1&1&2\\1&0&2\end{pmatrix}=\begin{pmatrix}ABC\\AB\\BC\times 2\end{pmatrix}$$

观察矩阵可以发现，每行相加即 $Count(C_1)$，第二、三列即 $Count(C_2)$，第一列即 $Count(C_3)$。因此频度矩阵既保留了频度信息，又能体现各项间的关联信息，在扫描中无需遍历数据库及逐个进行对比运算只需根据矩阵运算，即可完成扫描。

频度矩阵具有如下性质。

性质 4　(1)频度矩阵第 i 行之和等于特征向量第 i 项的频度：

$$\sum_{j=1}^{m}q_{ij}=Count(I_i)$$

(2)频度矩阵归一化处理：

$$Q_i=\begin{pmatrix}q_{11}&q_{12}&\cdots&q_{1m}\\q_{21}&q_{22}&\cdots&q_{2m}\\\vdots&\vdots&\ddots&\vdots\\q_{n1}&q_{n2}&\cdots&q_{nm}\end{pmatrix}$$

$$=\left(\lambda_1\begin{pmatrix}e_{11}\\e_{21}\\\vdots\\e_{n1}\end{pmatrix},\lambda_2\begin{pmatrix}e_{12}\\e_{22}\\\vdots\\e_{n2}\end{pmatrix},\cdots,\lambda_m\begin{pmatrix}e_{1m}\\e_{2m}\\\vdots\\e_{nm}\end{pmatrix}\right)$$

$$q_{ij}=\lambda_j e_{ij},e_{ij}=\begin{cases}0\\1\end{cases},\lambda_j\in N_+$$

其中，频度矩阵第 j 列系数 λ_j 等于第 j 列对应事务的频度：

$$\lambda_j=Count(t_j),t_j=(I_1,I_2,\cdots,I_n)\times\begin{pmatrix}e_{1j}\\e_{2j}\\\vdots\\e_{nj}\end{pmatrix}$$

如例 5 中事务集分块 T_1：

$$\begin{pmatrix}1&1&0\\1&1&2\\1&0&2\end{pmatrix}=\left((1\begin{pmatrix}1\\1\\1\end{pmatrix},1\begin{pmatrix}1\\1\\0\end{pmatrix},2\begin{pmatrix}0\\1\\1\end{pmatrix}\right)$$

$Count(ABC)=1,Count(AB)=1,Count(BC)=2$

可以看出，利用频度矩阵及其运算特点，可以大幅简化循环扫描中的对比计算。

4.2　频繁项集性质

在频繁项集挖掘中，存在如下性质。

性质 5　任何频繁项集的非空子集也是频繁的，非频繁项集的超集也是非频繁的[10]。

推论 1　若某事务集分块中所有项目的频度均低于最小支持度，则该事务集分块中没有频繁项集，可直接删除该事务集分块。

推论 2　若某事务集分块中存在某项目的频度低于最小支持度，则包含该项目的任意事务均是非频繁的，可直接将该项目从事务集分块中删除。

性质 6　若某事务长度小于 k，则该事务不支持 L_{k+1}[10]。

推论 3　若某事务集分块的特征向量长度小于 $k+1$，则该事务集分块的最大频繁项集为 L_k，在之后的循环中可直接删除该事务集分块。

推论 4　若某事务集分块中存在某事务长度小于 $k+1$，则在该事务集分块生成 L_k 之后的循环中可直接删除该事务。

4.3　扫描策略

(1)利用频繁项集相关性质，通过最小支持度压缩频度矩阵和特征向量，从而压缩事务集分块。

扫描策略 1：根据推论 1，若某事务集分块的频度矩阵每行之和均小于 $Sup_{\min}$，则直接删掉该事务集分块：

$$\text{if:}\max_i(\sum_{j=1}^{m}q_{ij})<Sup_{\min},q_{ij}\in Q_k,\text{then:delete }T_k$$

如例 5 事务集分块 T_2 的频度矩阵各行之和均小于 2，因此可直接删除第二分块，压缩数据库，提高扫描效率：

$$\begin{pmatrix}1&0\\1&0\\0&1\end{pmatrix}\begin{matrix}1+0<2\\\Rightarrow 1+0<2\Rightarrow\text{delete }T_2\\0+1<2\end{matrix}$$

扫描策略 2：根据推论 2，若某事务集分块中有一项的频度低于最小支持度，即频度矩阵某行之和小于 $Sup_{\min}$，则在该分块特征向量中删除该项，在频度矩阵中删除该行，从而压缩该事务集分块：

$$\text{if:}\exists i\leqslant n,\sum_{j=1}^{m}q_{ij}<Sup_{\min},\text{then:delete }I_i,q_{ij}\mid_{j\leqslant m}$$

如例 5 中事务集分块 T_3 的频度矩阵第三行之和小于 2，因此压缩该分块：在特征向量中删除 G，在频度矩阵中删除第三行，T_3 可压缩为 T_3'，从而提高扫描效率：

$$T_3\Leftrightarrow(A,F,G)\begin{pmatrix}1&1\\1&1\\0&1\end{pmatrix}$$

$$\begin{pmatrix}1&1\\1&1\\0&1\end{pmatrix}\begin{matrix}1+1=2\\\Rightarrow 1+1=2\Rightarrow\text{delete }G,q_{31},q_{32}\\0+1<2\end{matrix}$$

$$T_3'\Leftrightarrow(A,F)\begin{pmatrix}1&1\\1&1\end{pmatrix}$$

扫描策略 3：根据推论 3，若某事务集分块的特征向量长度小于 $k+1$，则在生成 L_k 后直接删掉该事务集分块：

$$\text{生成 }C_{k+1}:\text{if:}\max_j(\sum_{i=1}^{n}e_{ij})<k+1,e_{ij}\in Q_s,\text{then:delete }T_s$$

扫描策略 4：根据推论 4，若某事务集分块频度矩阵分解后的某列之和小于 $k+1$，则在生成 L_k 后，直接删掉该列：

$$\text{生成 }C_{k+1}:\text{if:}\exists j\leqslant m,\sum_{i=1}^{n}e_{ij}<k+1,\text{then:delete }q_{ij}\mid_{i\leqslant n}$$

(2)传统算法在计算 $Count(C_k)$ 时需循环扫描事务集，引入频度矩阵后，可利用矩阵特性，跳过对比扫描工作，直接用乘积获得结果。

定义函数 $jo(x)$：

$$jo(x)=\begin{cases}0, & \text{if } x=2k+1,k=0,1,2,\cdots\\x, & \text{if } x=2k,k=0,1,2,\cdots\end{cases}$$

扫描策略 5：根据性质 4，在计算某事务集分块的 $Count$

$(s_i)(s_i \in C_k)$时，将 s_i 按特征向量分解为频度向量，将频度向量与频度矩阵乘积可得到 $Count(s_i)$，无需循环对比：

$$\forall s_i \in C_k, s_i=(I_1, I_2, \cdots, I_n)\begin{pmatrix} g_{i1} \\ g_{i2} \\ \vdots \\ g_{im} \end{pmatrix}$$

$$Count(s_i)$$

$$=\frac{1}{2} jo\left(\begin{pmatrix} g_{i1} \\ g_{i2} \\ \vdots \\ g_{in} \end{pmatrix}^{\mathrm{T}}\begin{pmatrix} e_{11} & e_{12} & \cdots & e_{1m} \\ e_{21} & e_{22} & \cdots & e_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ e_{n1} & e_{n2} & \cdots & e_{nm} \end{pmatrix}\right)\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix}$$

$$=\frac{1}{2}\lambda_1 jo\left(\begin{pmatrix} g_{i1} \\ g_{i2} \\ \vdots \\ g_{in} \end{pmatrix}^{\mathrm{T}}\begin{pmatrix} e_{11} \\ e_{21} \\ \vdots \\ e_{n1} \end{pmatrix}\right)+\cdots+\frac{1}{2}\lambda_m jo$$

$$\left(\begin{pmatrix} g_{i1} \\ g_{i2} \\ \vdots \\ g_{in} \end{pmatrix}^{\mathrm{T}}\begin{pmatrix} e_{1m} \\ e_{2m} \\ \vdots \\ e_{nm} \end{pmatrix}\right)$$

如例 5 事务集分块 T_1，计算 $Count(AC)$，$AC \in C_2$：

$$AC \Leftrightarrow (A,B,C)\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

$$Count(AC)=\frac{1}{2} jo\left(\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^{\mathrm{T}}\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}\right)\begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$$

$$=\frac{1}{2}(1\times jo(2), 1\times jo(1), 2\times jo(1))$$

$$=\frac{1}{2}(1\times 2, 1\times 0, 2\times 0)=1$$

可以看出，利用该策略，在循环扫描时直接进行乘积运算，无需逐个扫描频繁集的频数，能有效提高挖掘效率。

5 BFM 算法

综合基于极大业务路径的分块挖掘策略和基于频度矩阵的搜索迭代策略，提出 BFM (Blocking Frequency Matrix)挖掘算法。

5.1 算法流程

算法首先遍历数据库，生成极大业务路径，并利用极大业务路径将数据库分块。根据频度矩阵的定义，将各分块的故障项集分解为特征向量和频度矩阵。然后利用扫描策略 1 和策略 2 进行搜索前的简化，根据分块矩阵的行和，通过扫描策略 1 删减不满足条件的分块，通过扫描策略 2 将分块矩阵降阶。

首先生成各分块的 1 项频繁集，开始循环搜索迭代。利用扫描策略 3 直接跳过不满足条件的分块循环，利用扫描策略 4 将频度矩阵进一步降阶优化。优化后的分块进行连接步和剪枝步生成候选项集，通过扫描策略 5 计算候选项集频数，进而生成频繁集。分块内循环在当前频繁项集为空集时结束，分块间循环在所有分块完成后结束，最终返回所有频繁项集。BFM 算法流程如图 9 所示。

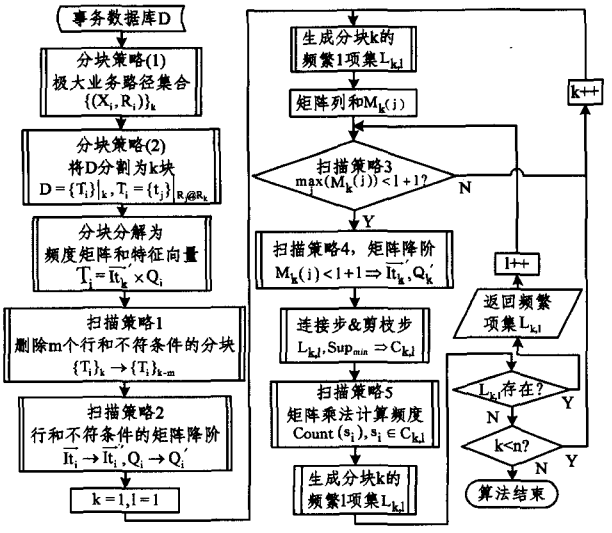

图 9 BFM 算法流程

5.2 算法描述

输入：事务数据库 D，最小支持度 Sup_{min}

输出：所有的频繁项集 L

1. FOR(i=1;$XR_i \neq \emptyset$;i++){
2. XR_i=find_i_maxpath(D);//生成第 i 条极大业务路径
3. T_i=find_i_block(XR_i,D);//生成第 i 个事务集分块
4. It_i=find_i_featurevector(T_i);//生成第 i 个事务集分块的特征向量
5. Q_i=find_i_frequentmatrix(It_i, T_i);//生成第 i 个事务集分块的频度矩阵
6. [n_i,m_i]=size(Q_i);//计算第 i 个频度矩阵的尺寸
7. }
8. N=NUM(XR_i);//计算极大业务路径个数
9. FOR(k=1;k≤N;k++){
10. IF($\max_i(\sum_{j=1}^{m_k} Q_k(i,j)) < Sup_{min}$){delete Q_k, It_k;}//扫描策略 1，利用 Q_k 行和简化{T_k}
11. ELSE{
12. IF($\sum_{j=1}^{m_k} Q_k(i,j) < Sup_{min}$){
13. delete $It_k(i)$ from It_k;delete $Q_k(i,j)|_{j\leq m_k}$ from Q_k;
14. }//扫描策略 2，利用 Q_k 行和简化 T_k
15. }
16. }
17. FOR EACH $Q_k(i,j)${
18. IF($Q_k(i,j)=0$){$E_k(i,j)=0$;}
19. ELSE{$E_k(i,j)=1$;$\lambda_k(j)=Q_k(i,j)$;}//将 Q_k 归一化为矩阵 E_k 及相应因子 λ_k
20. }
21. FOR(k=1;k≤N;k++){//开始分块 k 的搜索迭代
22. $L_{k,1}=\{I_i \in It_k \mid \sum_{j=1}^{m_k} Q_k(i,j) \geq Sup_{min}\}$;//利用 Q_k 行和生成 1 项频繁集
23. $M_k(j)=\sum_{i=1}^{n_k} E_k(i,j)$;//计算 E_k 列和
24. FOR(l=2;$L_{l-1} \neq \emptyset$;l++){//开始 l 项频繁集的搜索迭代
25. IF($\max_j(M_k(j)) < l$){BREAK;}//扫描策略 3，利用 E_k 列和跳过分块 k 的扫描
26. ELSE{.
27. FOR EACH $M_k(j)${
28. IF($M_k(j) < l$){delete $Q_k(i,j)|_{i\leq n_k}$ from Q_k;}//扫描策略 4，利用

E_k 列和简化 T_k

29. }

30. }

31. $C_{k,1}$=apriori_gen($L_{k,(l-1)}$,Sup_{min});//通过连接步和剪枝步，生成 l 项候选集

32. FOR EACH $s_i \in C_{k,1}$ {

33. g_i=find_vector(s_i,It_k);//将事务 s_i 转化为向量

34. $Count(s_i)=\frac{1}{2}jo(g_i^T \cdot E_k) \cdot \lambda_k^T$;//扫描策略 5，利用矩阵运算求解 s_i 的频度

35. }

36. $L_{k,l}=\{s_i \in C_{k,l} | Count(s_i) \geqslant Sup_{min}\}$;//返回 l 项频繁集

37. }

38. RETURN $L_k=\bigcup_l L_{k,l}$;//返回分块 k 的所有频繁项集

39. }

40. RETURN $L=\bigcup_k L_k$;//返回数据库中所有的频繁项集

5.3 实例仿真

为进一步验证 BFM 算法的效率，选取某研究所某型飞机 2009－2011 年的故障数据库作为实验数据，在内存为 8GB、CPU 为 Intel Core i5-3337u 1.87GHz、操作系统为 64 位 Windows10 的计算机上通过 Matlab 编程实现了 Apriori 算法和 BFM 算法。

设最小置信度为 0.3，部分仿真结果示例如表 9 所列。

表 9 关联规则结果

关联规则	置信度
A31→A67	0.601
A43→A90	0.442
…	…

其中，A31 指“飞管信号异常”；A67 指“AFDX 加载信号失败”，相应业务路径为“飞管系统→AFDX1→远程 1→惯导”。对比排故记录中的失效过程说明，挖掘出的“A31→A67”是有效且可信的。

如图 10 所示，BFM 算法的效率要高于 Apriori 算法，且随着支持度的提高，优势先越来越明显，随后逐渐消失。这是因为随着最小支持度增大，其逐渐超出数据库中多数项目的频度，而 BFM 算法分块后第一次扫描判断即过滤掉了绝大多数分块和项集，相比 Apriori 算法，其效率优势会变得非常明显；而随着最小支持度超出所有项目频度，两种算法一次迭代即判定所有频繁项集，由于 BFM 进行了分块运算后才进行迭代，相比 Apriori 算法在开始直接迭代，算法优势逐渐消失。工程中，机载网络关联故障的挖掘支持度一般要求在 20%以内，此时 BFM 算法效率要远远高出 Apriori 算法。

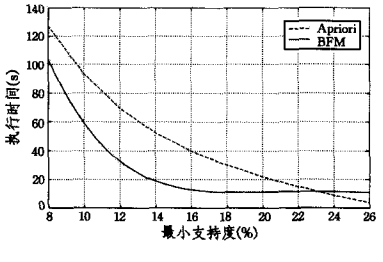

图 10 不同支持度下执行时间的对比

如图 11 所示，BFM 算法产生的频繁项集个数要少于 Apriori 算法，且随着支持度提高，频繁项集的个数差距越来越小。这是因为随着最小支持度增大，搜索迭代的循环次数越来越少，循环中产生的噪声组合也越来越少，BFM 算法“去噪”的优势越来越不明显。

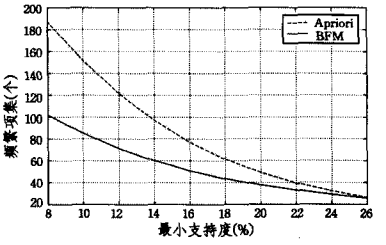

图 11 不同支持度下频繁项集个数的对比

综上，BFM 算法能支持机载网络关联故障挖掘，且比传统算法具有更高的效率。

结束语 本文首先分析了 Apriori 算法效率主要受限于外存扫描和噪声组合。利用机载网络故障基于业务路径的关联特征，找到避免噪声组合的分块策略，算法在搜索迭代之前就过滤掉了大部分噪声信息。在此基础上，进一步将各块分解为特征向量和频度矩阵，结合矩阵运算的特点和频繁项集的性质，找到大幅降低外存扫描的循环策略，极大地减少了算法循环次数和对比计算量。综合以上策略，设计了 BFM 算法。理论和实验结果表明，BFM 算法能有效提高频繁项集的挖掘速率。

参 考 文 献

[1] Liu Hao. The Management System of the Dynamic and Coupling Airborne Network Faults[D]. Beijing: Beihang University, 2014 (in Chinese)
刘浩. 动态耦合关联的机载网络故障管理系统[D]. 北京：北京航空航天大学，2014

[2] Chen Jing. The Future Structure of PAVE PALLAR System Avionics[J]. Aeronautical Computing Technology, 1986(1): 37-60 (in Chinese)
陈静. “宝石柱”航空电子系统未来的结构[J]. 航空计算技术，1986(1)：37-60

[3] Yao Gong-yuan, Wu Jian-min, Chen Ruo-yu. The Growth of Avionics Integration Technologies and Trends of The Modularization[J]. Avionics Technology, 2002, 33(1): 1-10(in Chinese)
姚拱元，吴建民，陈若玉. 航空电子系统综合技术的发展与模块化趋势[J]. 航空电子技术，2002，33(1)：1-10

[4] Xiong Hua-gang, Zhou Gui-rong, Li Qiao. A Survey on Avionics Bus and Network Interconnections and Their Progress[J]. Acta Aeronautica et Astronautica Sinica, 2006, 27(6): 1135-1144 (in Chinese)
熊华钢，周贵荣，李峭. 机载总线网络及其发展[J]. 航空学报，2006，27(6)：1135-1144

[5] Park J S, Chen M S, Yu P S. An effective hash-based algorithm for mining association rules[J]. ACM SIGMOD Record, 1995, 24(2): 175-186

[6] Singh J, Ram H, Sodhi D J S. Improving Efficiency of Apriori Algorithm Using Transaction Reduction[J]. International Journal of Scientific and Research Publications, 2013, 3(1): 1-4

[7] Toivonen H. Sampling large databases for association rules [C] // VLDB. 1996, 96: 134-145

(下转第 162 页)

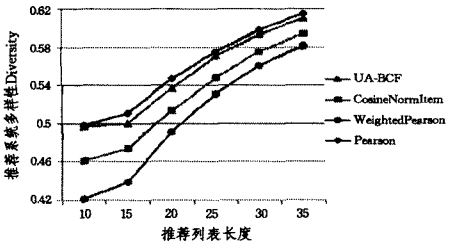

图 5　推荐多样性指标 Diversity 对比结果

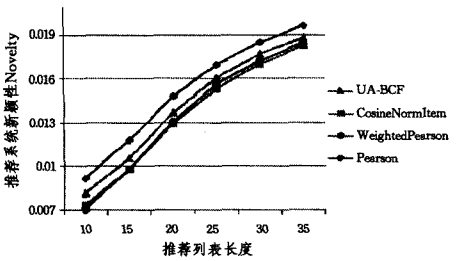

图 6　推荐新颖性指标 Novelty 对比结果

结束语　本文利用用户活跃度和项目流行度信息对现有项目相关性度量方法进行了改进，在保证推荐精度的情况下提高了推荐系统的多样性和新颖性。所提相关性度量方法的基本思想是，对于任意给定的两个项目，若存在较多的仅对两个项目之一有评分的记录，两个项目的相关性会降低。相关性下降程度取决于这些记录对应的用户活跃度和项目流行度。用户活跃度越高，项目相关性下降程度越大，项目流行度差异越小，项目相关性下降程度越大。基于电影数据集 ML100K 的实验结果表明，所提算法可以作为现有基于项目最近邻的协同过滤算法的有效补充。在对多样性和新颖性要求较高的推荐环境中，该算法具有较大的优势。在本文模型中，惩罚权重对相关性度量的效果具有重要作用，如何对惩罚权重进行改进以获得更高的推荐精度是本文后续研究的主要方向。

参 考 文 献

[1] Linden G, Smith B, York J. Amazon. com recommendations: Item-to-item collaborative filtering [J]. IEEE Internet Computing, 2003, 7(1): 76-80

[2] 项亮. 推荐系统实践[M]. 人民邮电出版社, 2012

[3] Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithms[C] // Proceedings of the 10th International Conference on World Wide Web. ACM, 2001: 285-295

[4] Ekstrand M D, Ludwig M, Konstan J A, et al. Rethinking the recommender research ecosystem: reproducibility, openness, and LensKit[C]//Proceedings of the Fifth ACM Conference on Recommender Systems. ACM, 2011: 133-140

[5] Resnick P, Iacovou N, Suchak M, et al. GroupLens: an open architecture for collaborative filtering of netnews[C] // Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work. ACM, 1994: 175-186

[6] Zhang Yin, Zhang Bin, Gao Ke-ning, et al. Autonomy Oriented Personalized Tag Recommendation[J]. Acta Electronica Sinica, 2012, 40(12): 2353-2359(in Chinese)
张引，张斌，高克宁，等. 面向自主意识的标签个性化推荐方法研究[J]. 电子学报, 2012, 40(12): 2353-2359

[7] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems[J]. Computer, 2009 (8): 30-37

[8] Breese J S, Heckerman D, Kadie C. Empirical analysis of predictive algorithms for collaborative filtering[C] // Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann Publishers Inc., 1998: 43-52

[9] McLaughlin M R, Herlocker J L. A collaborative filtering algorithm and evaluation metric that accurately model the user experience[C]//Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2004: 329-336

[10] Ahn H J. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem[J]. Information Sciences, 2008, 178(1): 37-51

[11] Marlin B M, Zemel R S. Collaborative prediction and ranking with non-random missing data[C] // Proceedings of the third ACM Conference on Recommender Systems. ACM, 2009: 5-12

[12] Little R J A, Rubin D B. Statistical analysis with missing data [M]. John Wiley & Sons, 2014

[13] Järvelin K, Kekäläinen J. Cumulated gain-based evaluation of IR techniques[J]. ACM Transactions on Information Systems (TOIS), 2002, 20(4): 422-446

[14] Zhou T, Kuscsik Z, Liu J G, et al. Solving the apparent diversity-accuracy dilemma of recommender systems[J]. Proceedings of the National Academy of Sciences, 2010, 107(10): 4511-4515

[15] Vargas S, Castells P. Rank and relevance in novelty and diversity metrics for recommender systems[C]//Proceedings of the fifth ACM Conference on Recommender Systems. ACM, 2011: 109-116

(上接第 152 页)

[8] Brin S, Motwani R, Ullman J D, et al. Dynamic itermset counting and implication rules for market basket data[J]. ACM SIGMOD Record, ACM, 1997, 26(2): 255-264

[9] Han Jia-wei, Kamber M. Data Mining: Concepts and Techniques [M]. Beijing: China Machine Press, 2001: 130-160(in Chinese)
韩家炜，坎伯. 数据挖掘：概念与技术[M]. 北京：机械工业出版社，2001：130-160

[10] Agrawal R, Srikant R. Fast algorithm for mining association rules[C] // Proc. 20th Int. Conf. Very Large Data Bases (VLDB). 1994, 1215: 487-499

[11] Van V S, Via J, Santamana I. A Sliding-Window Kernel RLS Algorithm and Its Application to Nonlinear Channel Identification [C]//Proceedings of ICASSP 2006. Toulouse, France, 2006

[12] Wei Ling, Wei Yong-jiang, Gao Chang-yuan. Improved Apriori Algorithm Based on Bigtable and MapReduce[J]. Computer Science, 2015, 42(10): 208-210(in Chinese)
魏玲，魏永江，高长元. 基于 Bigtable 与 MapReduce 的 Apriori 算法改进[J]. 计算机科学，2015，42(10)：208-210，243

[13] Xu Zhang-yan, Liu Mei-ling, Zhang Shi-chao, et al. Three Optimized Methods of Apriori Algorithm[J]. Computer Engineering and Applications, 2004, 40(36): 190-193(in Chinese)
徐章艳，刘美玲，张师超，等. Apriori 算法的 3 种优化方法[J]. 计算机工程与应用，2004，40(36)：190-193