基于 MFP－Miner 算法的图书借阅数据关联规则挖掘

Association Rule Mining of Borrowed Books Data Based on MFP－Miner Algorithm

吕志芳 王怀阳 贾吉庆 （中国海洋大学电子工程系 山东青岛 266100）

摘 要： 本研究利用关联规则挖掘的最大频繁模式算法对图书馆历史借阅数据进行快速有效地挖掘，以获取隐含在借阅数据中有用的关联信息，优化图书馆馆藏结构，发掘学科间的隐性联系和学科动向。由于该算法在挖掘过程中不需要产生候选项目集，因而节约了对候选项目集进行计数的时间，从而使算法的效率得到了很大的提高。

关键词： 数据挖掘 关联规则 最大频繁模式 频繁模式树

1 引言

对高校图书馆图书历史借阅信息进行关联规则挖掘可以发现一些比较有趣的规律，例如，最近在信息管理系的借阅记录中得到：有 30% 的竞争情报借阅同时带有会计学、财务管理的借阅，有 80% 的证券分析借阅带有数据挖掘借阅。这让我们联想到可能有读者在进行企业财务竞争情报的研究、证券领域的数据挖掘研究，这都是很有价值的信息[3]。根据这些规律我们可以制订相应决策，把同时被借阅的书籍整理在一起，以优化馆藏结构、馆间分布，这样不仅可以辅助教师的教学工作，甚至可以发掘学科间的隐性联系和学科动向。

关联规则的挖掘算法有很多种，如 Apriori 算法、FP－Growth 算法[5]、Max－Miner 算法、MFP－Miner 算法[6]等等。随着图书量的增多，借阅信息数据库规模越来越大，用传统的挖掘算法已不能快速有效地处理巨大的借阅历史数据库信息，针对此问题，本文提出采用最大频繁模式项集算法（即 MFP－Miner 算法）对数据进行快速有效地处理。

2 MFP－Miner 算法

关联规则数据挖掘（简称关联规则挖掘）是从大量的数据中挖掘出有价值的描述数据项之间相互联系的有关知识[1]。MFP－Miner 算法采用了 Han jiawei 等人提出的频繁模式树[4]结构，是一种新的基于 FP－Tree[2]的挖掘最大频繁项目集的快速算法。该算法的原理是，充分利用 FP－Tree 的特点，在挖掘过程中不需产生最大频繁候选项目集。即在进行最大频繁项目集挖掘前，首先将事务数据库中的每条事务所包含的频繁项目按照支持数降序压缩存储到频繁模式树中。之后，在挖掘过程中的只需对该树进行搜索，而不需扫描数据库。

MFP－Miner 挖掘流程：

（1）扫描数据库，建立 fp－tree。

（2）在挖掘过程中，若发现树中的某个节点 N 的计数不小于 mincount，则从 Htable 中取出所有排列在 N. node－name 前面的项目组成集合 X；然后，通过 N 的同名节点链，找出节点链中所有计数不小于 mincount 的同名节点；最后，遍历每条以同名节点为后缀的路径 P，检查 X 是否存在于 P 中，一旦发现了这样的路径，则可将在当前频繁模式（或条件频繁模式）树中

的挖掘过程终止。

3 用 MFP－Miner 算法对图书借阅历史数据进行关联规则挖掘

下面我们将分析中国海洋大学图书馆的借阅数据，实现基于 MFP－Miner 算法的关联规则挖掘，得出书籍借阅信息中隐含的规律，并对这些规律进行分析，从而能够在一定程度上对图书馆的工作提一些建设性意见。

3.1 数据准备

首先对数据进行预处理，建立事务数据库，即每位读者同时借阅的图书统计表。中国海洋大学图书馆提供的连续五年的借阅数据是以日志形式存在的文本文档，并且都是以月为单位导出的，因此首先应该将各年月的分散数据连接到一起，然后才能对数据进行统一预处理。

数据的连接是在 Visual Basic 环境下进行的，即将 2000 年 2 月至 12 月的数据全部追加到 log2000－1. txt 中，另存为文本文档 log2000. txt。同理依次可以得到 log2001. txt, log2000. txt, log2002. txt, log2003. txt, log2004. txt。最终将这五年的数据再连到一起得到文本文档 fiveyears. txt。

在 sql server 2000 环境下，提取挖掘中需要用到的属性。针对图书馆借阅数据库，与该课题相关的属性是操作日期、读者记录号、书目记录号；针对书目库，相关的属性是书目记录号及中图分类号。删除读者一次只借阅一本书的记录，建立事务数据库。在事务数据库中，通过查询发现，读者一天内同时借阅的书籍与其专业有一定的关联，并且同一类书籍之间被同时借阅的概率比较大，因此这里挖掘中图分类号为 TP 类的书籍（自动化技术，计算机技术）。事务数据库表如表 1。

3.2 算法实现

在发现频繁项目集的算法研究中，计算项目集的支持度是发现频繁项目集的过程中最耗时的操作，因此降低候选项目集的数量和减少扫描数据库的次数成为减少开销的有效手段之一。由于最大频繁项目集的集合中隐含了所有频繁项目集，每个最大频繁项目集包含多个频繁项目集，所以对同一个数据库来说，在相同的最小支持度下，发现的最大频繁项目集的数量要远小于频繁项目集的数量，因此 MFP－Miner 算法的计算量要远远小于 mafia 等算法的计算量。具体算法如下[6]：

表 1　事务数据库

ReaderID	time	IID	TID	COUNT
1000	2001-8-31 0:00:0	t198	1	1
1000	2001-8-31 0:00:0	t028	1	1
1001	2001-6-15 0:00:0	t028	2	1
1001	2001-6-15 0:00:0	t070	2	1
1010	2001-3-2 0:00:00	t089	3	1
1010	2001-3-2 0:00:00	t105	3	1
1012	2001-3-12 0:00:0	t094	4	1
1012	2001-3-12 0:00:0	t132	4	1
1017	2000-10-18 0:00:	t070	5	1
1017	2000-10-18 0:00:	t094	5	1
1026	2000-9-11 0:00:0	t198	6	1
1026	2000-9-11 0:00:0	t129	6	1
1026	2001-1-12 0:00:0	t094	7	1
1026	2001-1-12 0:00:0	t135	7	1
1026	2001-5-31 0:00:0	t070	8	1
1026	2001-5-31 0:00:0	t106	8	1
1027	2001-6-12 0:00:0	t105	9	1
1027	2001-6-12 0:00:0	t129	9	1
1031	2000-12-4 0:00:0	t028	10	1

表 2　挖掘结果

TID	IID	COUNT
1	t186	855
1	t198	855
2	t171	855
2	t198	855
3	t070	855
3	t198	855
4	t186	495
4	t195	495
5	t070	495
5	t195	495
6	t193	487
6	t198	487
7	t186	487
7	t193	487
8	t171	102
8	t193	102
9	t070	102
9	t171	102
9	t193	102

输入：最小支持度 minsup，在此 minsup 下构造的 FP－Tree T；

输出：事务数据库 D 中满足 minsup 要求的最大频繁项目集的集合 MFS。

(1) MFS = NULL;

(2) 调用 MFP－Max(T,MFS,null)；

(3) return MFS;

Procedure MFP－Max(T,MFS,x) /＊发现 T 中所有以 x 为后缀的最大频繁项目集，并存放到 MFS 中＊/

(1) if T 只包含单个路径 P then begin

(2) $m=\{a_1 \cup a_2 \cup \cdots \cup a_n \mid a_i \in P\} \cup x$ 并且 m. support = a_n. support;

(3) if m 不是 MFS 中某项目集的子集 then

(4) MFS = MFS∪m;

(5) end

(6) else begin

(7) for k = HTable. length to 1 do begin

(8) $m = a_k \cup x$ 且 m. support = HTable[k]. support;/＊a_k = HTable[k]. item－name＊/.

(9) 构造 m 的条件频繁模式基和条件 FP－Tree Tm;

(10) if Tm≠NULL then /＊Tm 中除了根节点之外还包含有子节点＊/

(11) 调用过程 MFP－Max(T_m,MFS,m)；

(12) else if m 不是 MFS 中某项目集的子集 then

(13) MFS = MFS∪m;/＊把 m 放入 MFS 中，同时删除 MFS 中是 m 子集的项目集＊/

(14) if a_k 的同名节点链中含有节点计数不小于 mincount 的节点 then begin

(15) $m=\{ak-1 \cup ak-2 \cup \cdots \cup a1\}$；/＊取出项目头表中 a_k 以前的项目＊/

(16) if m 存在于某条以 a_k 的同名节点为后缀的路径中 then

(17) break;

(18) end

(19) end

(20) end

这里选用 C＋＋语言作为编程工具，在 VisualC＋＋6.0 编程环境中实现了算法 MFP－Miner。在算法的实现过程中，发现随着支持度的增加，规则的数目在随之减少，因此需要经过较多的数据进行多次试验，才能得到比较准确的信息。

3.3 实验结果及分析

在此试验中对 TP 类书籍的借阅数据进行关联分析，设置的最小支持度为 0.015，即最小支持数为 100，挖掘出的结果如表 2。

对照检索表，可以发现有 5.22% 的读者同时借阅计算技术、计算机技术类，软件工具类及程序语言、算法语言类书籍，同时借阅计算技术、计算机技术类，软件工具类的读者有 87.41% 的可能性借阅程序语言、算法语言类；有 3.12% 的读者同时借阅计算技术、计算机技术类，Windows 操作系统类及程序语言、算法语言类书籍，同时借阅计算技术、计算机技术类，Windows 操作系统类的读者有 87.41% 的可能性借阅程序语言、算法语言类；……

基于以上关联规则，在图书馆藏书的布局上，可以将常被同时借阅的书籍整理在一起，这样既可以方便师生借阅，也可以在一定程度上辅助老师的教学。其他类书籍的借阅信息也可以如此进行关联分析，从而可以得出更全面的数据。

比较 MFP－Miner 算法与其它算法的效率，在这里选定最常用的 Mafia 算法进行比较，在不同的最小支持度下分别运行 Mafia 和 MFP－Miner 算法，在同样的测试环境下，测试算法的执行时间。这里我们仅给出算法本身的执行时间，不包括数据的输入输出时间。我们在测试中省去了对输入输出时间的测试，而仅测试算法本身的执行时间。测试时间比较如图 1 所示。

从图 1 中可以看出，MFP－Miner 算法的执行效率约为 Mafia 的两倍。这说明用 MFP－Miner 算法对借阅数据进行关联规则挖掘效率更高。

4 结束语

由于 MFP－Miner 算法频繁模式树中存放的是数据库中每条事务包含的频繁项目，其所占用的存储空间要远小于数据库的存储空间，因而节约了算法中所需的搜索时间和内存空间。另外，算法 MFP－Miner 根据频繁模式树的特点，在挖掘过程中不需要产生候选项目集，这不仅节约了内存空间，也节约了对候选项目集进行计数的时间，从而使算法的效率得到了很大的

提高。

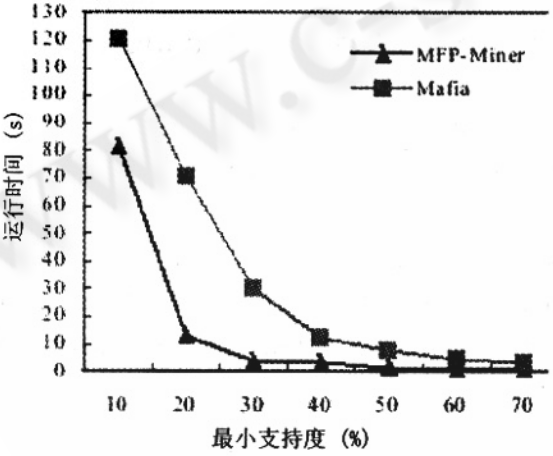

图 1 MFP－Miner 算法与 Mafia 算法的运行时间比较

由于图书馆借阅数据每日有更新，数据库不断改变，要尽快适应师生借阅需求，就需要不断地更新挖掘的结果，如果快速有效地更新规则，有待于我们进一步研究。

参考文献

1 Jiawei Han ,Micheline Kamber. Data Mining Concepts and Techniques，北京：高等教育出版社.

2 Han J, Jian P, Yiwen Y. Mining frequent patterns without candidate generation. In: Proceedings of the 2000 ACM SIGMOD International Conference Management of Data. Dallas, 2000. 1～12.

3 刘晓东，数据挖掘在图书馆工作中的应用，情报杂志，2005(8).

4 战立强、刘大昕、张健沛，一种基于模式树的频繁项集快速挖掘算法，计算机工程与应用，2007，43(11).

5 冯洁、陶宏才，典型关联规则挖掘算法的分析与比较，计算机技术与发展，2007(03)121～124.

6 杨君锐、赵群礼，基于 FP－Tree 的最大频繁项目集更新挖掘算法，华中科技大学学报(自然科学版)，2004，32(11)：88～90.