文章编号:1671-4598(2016)06-0178-03 DOI:10.16526/j.cnki.11—4762/tp.2016.06.049 中图分类号:TP311.13 文献标识码:A

基于 XML 数据挖掘的 Apriori 算法的研究与改进

张继荣，王向阳
（西安邮电大学 通信与信息工程学院，西安 710061）

摘要：XML 以其诸多优点，迅速成为不同领域间信息表示与交换的标准；大量 XML 数据的出现给数据挖掘带来了新的挑战；挖掘 XML 数据关联规则的大部分工作都是基于 Apriori 算法的研究；对 Apriori 算法的基本方法与效率进行了分析，指出其不足，并提出了改进的 XApriori 算法，该算法基于新的数据结构，利用 Hash 表的存储技术以及对 Apriori 算法的优化来提高查找频繁项集的效率；对 Apriori 算法和 XApriori 算法进行了比较，实验结果表明改进的 XApriori 算法优于 Apriori 算法。

关键词：XML；数据挖掘；关联规则；频繁项集；Apriori

Research and Improvement of Apriori Algorithm for XML Data Mining

Zhang Jirong，Wang Xiangyang
（School of Communication，Xi′an University of Posts and Telecommunications，Xi′an 710061，China）

Abstract：Due to its many advantages，XML has rapidly become as a standard for representing and exchanging information in different fields. A large number of XML data has brought new challenges to data mining. Most of the work for mining XML data association rules is based on Apriori algorithm. The basic methods and efficiency of Apriori are analyzed，pointing out its shortcomings and propose the improved XApriori algorithm. The algorithm is based on the new data structure，the use of Hash table and the optimization of Apriori to improve the efficiency of finding frequent item sets. The experimental results show that XApriori is superior to Apriori.

Keywords：XML；data mining；association rules；frequent item sets；Apriori

0 引言

XML（eXtensible Markup Language，可扩展标记语言）[1]是一种描述数据内容和结构的语言，其具有标准的文本格式，可以在不同平台间，进行数据的表示、存储和交换，XML 的飞速发展产生了大量的 XML 数据，所以，对其进行挖掘[2]已经变得非常重要。对 XML 数据关联规则[3-4]的挖掘概括有以下 3 种不同的方法：1）Wan 等人提出的 XML 挖掘算法是直接使用 XQuery 语言来实现 Apriori 算法的，无需预先处理 XML 文档[5]。2）Daniele Braga 等人提出的方法是通过预处理数据、抽取关联规则和后期处理关联规则 3 个主要步骤来实现 Apriori 算法[6]。3）Ling Feng 等人提出一种不基于 Apriori 算法并且将不同的事务和项的概念映射到 XML 文档树型结构的方法[7]。本文基于 XML 文档并通过改进的 Apriori 算法[4]来进行关联规则的挖掘。XML 挖掘包括对其结构和内容的挖掘，本文是基于 XML 文档的内容来进行关联规则的挖掘，通过 DOM 技术解析、提取出 XML 文档的内容[8]，由于 XML 文档的内容等价于元素的值，而被解析提取之后的 XML 元素值的集合与文本基本等价。所以，对 XML 文档内容的挖掘就转变成对一般文本文档的挖掘。

1 关联规则的基本概念

设所有的项目集合记为 $I=\{i_1, i_2, \cdots, i_m\}$，数据集记为 D（事物数据库），$D=\{T_1, T_2, \cdots, Tn\}$，其中 T_i（$i=1, 2, \cdots, n$）称为事务且 $T_i \subseteq I$，包含 k 个项的项集称为 k－项集[9]。在 XML 数据中，将提取出的每一个元素记为一个事务，每个事务都有自己的 ID，元素的值记为项，各个项都有自己的编号 I_k（$k=1, 2, \cdots, n$），把 XML 数据提取出的事务集合记为 D，项的集合记为 I。以某食品店部分交易记录信息的 XML 文档的数据表示为例来说明。记录信息的部分 XML 文档如下：

```
<? xml version="1.0" encoding="ISO8859"? >
<person id="001">
<food>
<sort>apple</sort>
<sort>tomato</sort>
<sort>fish</sort>
<sort>rice</sort>
</food>
.....
```

该文档内容经过解析、提取之后，可以表示成表 1 和表 2 的形式。

关联规则[10]是形如 R：X→Y 的逻辑蕴含式，其中 X⊆I，Y⊆I 且 X∩Y=Φ。在数据库 D 中，存在关联规则 X→Y 的支持度记为 Support（X→Y）或 Support（X∪Y），表示同时出现项集 X 和 Y 的比例。若有 Support（X），则存在关联规则 X→Y 的置信度，记为 Confidence（X→Y）= Support（X∪Y）/Support（X），表示包含 X 的事务下同时也包含 Y 的事务的比例。

收稿日期：2015-12-03； **修回日期**：2016-01-05。

作者简介：张继荣（1963-），女，辽宁沈阳人，博士，教授，硕士生导师，主要从事现代通信网方向的研究。

表 1　事务数据

编号	名称
1	I_2
2	I_4，I_5
3	I_1，I_3，I_7，I_9
4	I_1，I_3，I_5，I_9
5	I_1，I_5，I_6，I_9
6	I_4，I_8
7	I_2，I_4
8	I_1，I_5，I_7
9	I_1，I_3，I_4，I_5，I_9
10	I_3，I_5，I_7
11	I_8
12	I_1，I_5，I_8
13	I_1，I_5，I_7

表 2　项目数据

编号	名称
I_1	apple
I_2	pear
I_3	tomato
I_4	peach
I_5	fIsh
I_6	bean
I_7	beef
I_8	bread
I_9	rIce

关联规则的产生，给定最小支持度 minsup，首先找出所有的其支持度不小于 minsup 的频繁项集，再由频繁项集计算出关联规则。如何高效地查找频繁项集是整个算法的重点，也是本文后续工作研究的重点。

在大量的关联规则算法中，基于 Apriori 算法[11]挖掘效率的研究成为了重点。在 1993 年，Agrawal 等人提出了 Apriori 算法。该算法的基本思想：按包含项目数自小至大的、逐层搜索迭代的方法来寻找频繁项集。即用频繁 k－项集产生（k＋1）－项集。具体过程是：第一次对数据库进行扫描，找出所有的频繁 1－项集，记为 L_1；对 L_1 自连接产生 C_2，对 C_2 修剪后，第二次扫描数据库，找出所有的频繁 2－项集，记为 L_2，如此循环，直到没有频繁项集为止。从过程中可看到 Apriori 算法的不足：在面对较大数据时，需要花费较大的开销产生数目巨大的候选项集，C_k 中的项集的数目将会呈现出指数般的增长；另外需要 k 次扫描数据库，k 为最大频繁项集中子项集的项目数。从而带来了很大的 I/O 交互负载并且非常耗费时间。

2　Apriori 改进算法的设计

2.1　数据结构的设计及事务数据的 Hash 表表示

分析 Apriori 算法的不足可知，Apriori 算法效率的提高关键在于：1）减少扫描事务数据库的次数；2）减少候选项集的数目。对此，本文设计了新的数据结构，并利用 Hash 表来表示和存储数据。该方法的基本思想是：对于 Apriori 算法来说，其依赖的数据结构是水平的，本文对其转变成项目事务垂直对应的关系数据结构。对 XML 解析提取出的数据事务和项用 Hash 表来记录[12]，每个项构建一个 Hash 表，Hash 表中的元素为各个事务在对应的项中出现的事务编码，为了方便后续计算项集出现的次数，本文以二进制编码形式来表示事务元素[13]，各事务在对应的项中出现，则相应的表示为 1，否则为 0。将表 1 和表 2 可以表示成 Hash 表的形式，如表 3 所示。

表 3　事务数据 Hash 表

项（Item）	事务代码（Code）
I_1	0011100110011
I_2	1000001000000
I_3	0011000011000
I_4	0100011010000
I_5	0101100111011
I_7	0010000101001
I_8	0000010000110
I_9	0011100010000

表 3 产生频繁项集的方法：给定最小支持度计数（minsup），如果项目的二进制代码中 1 的个数不小于 minsup，其相应的项集即为频繁 1－项集。有 2－项集 $\{I_i, I_j\}$，只需在 Hash 表中提取第 i 项和第 j 项的二进制代码，对其作按位逻辑与运算，得到的二进制代码中 1 的个数不少于 minsup 的 2－项集 $\{I_i, I_j\}$ 即为频繁 2－项集。频繁 k－项集以相同的方法得到。该方法不会产生大量的候选项集，不但节省了内存空间，而且在计算支持度时，只需提取出部分的二进制代码，不需对 Hash 表进行整体扫描，从而降低了 I/O 交互负载并且节省了时间，最终使得 Apriori 算法的效率得到了大幅度的提高。

2.2　连接步的改进方法

分析 Apriori 算法的计算过程中发现 Apriori 算法需要重复多次的对两个 k－项集进行判断：1）判断其前 k－1 项是否相同；2）判断最后一项是否不同。对 k－项集的判断运算将影响着 Apriori 算法效率的高低。于是本文对其进行了必要的改进。该方法的基本思想：L_{k-1} 自连接生成 C_k 前，判断任意的两个项集的前 $k-2$ 项是否相同，如果不同，则不进行两个项集的连接运算，因为产生的项集是重复的、非频繁的[14]。设项集按升序排列的 L_{k-1} 中，任意项集 X 和 Y 作连接时，若 $X_{k-2} \neq Y_{k-2}$，则放弃对 X 和 Y 的进行连接运算，从而减少了运算量。

例如在数据库中 D 中，按升序排列的项集 L_2＝｛｛I_5、I_1｝，｛I_5、I_8｝，｛I_4、I_3｝，｛I_4、I_1｝，｛I_3、I_8｝，｛I_1、I_6｝｝。按改进的连接方法生成 C_3，C_3＝L_2×L_2＝｛｛I_5、I_1、I_8｝，｛I_4、I_3、I_6｝｝，如果按 Apriori 算法的连接方法生成 C_3＝L_2×L_2＝｛｛I_5、I_1、I_8｝，｛I_5、I_1、I_4｝，｛I_5、I_8、I_3｝，｛I_5、I_1、I_6｝，｛I_4、I_3、I_1｝，｛I_4、I_1、I_6｝，｛I_4、I_3、I_8｝｝，相比可知减少了冗余项集的产生，两者对数据库的扫描分别需要 2 次和 27 次。可见，连接步的改进可以减少运算量和扫描次数。

2.3 剪枝步的改进方法

判断 C_k 中的项集是否属于 L_k 中的频繁项集前，Apriori 算法对 C_k 进行修剪，以除去 C_k 中那些不可能产生频繁项集的候选项集。将 C_k 中各个项集的 $k-1$ 项子项集与 L_{k-1} 中的项集进行匹对，若不出现相同的项集，则删除 C_k 中产生该子项集的项集。该方法对数据库进行了大量的扫描，中间产生了大量的子项集，从而造成了内存空间的浪费和很大的时间开销，导致了剪枝步的运算效率比级低。为了解决这些问题，本文结合了 Apriori 的一些性质，提出了剪枝步的改进方法。Apriori 的一些基本性质[14]：

性质 1：设一项集为 X_k，X_k 中存在的任意子集记为 Y_k，如果 X_k 是频繁项集，那么 Y_k 也是频繁项集。

性质 2：设一事务为 T_k，C_{k-1} 中的任何项集记为 I_{k-1}，如果 $I_{k-1}\not\subset T_k$，则删除事务 T_k，不会影响 L_k 的产生。

从性质可得：如果在 $I=\{i_1, i_2, \cdots, i_k\}$ 中，存在 $j\in I$ 且 $|L_{k-1}(j)|<k-1$，则 I 不是频繁项集，$|L_{k-1}(j)|$ 为集合 L_{k-1} 中包含元素 j 的子项集的个数。

反证法证明：假设有频繁项集 I_k，由性质 1 知 I_k 中的 k 个非空子项集 X_{k-1} 也是频繁项集，所以所有的子项集 X_{k-1} 都存在于 L_{k-1} 中。又由性质 1 知任意项集 X_{k-1} 的每一个项目 $i\in I$ 在 L_{k-1} 共出现了至少 $k-1$ 次，所以，对任何 $j\in I$ 有 $|L_{k-1}(j)|\geqslant k-1$，结果明显矛盾，故 I 不是频繁项集。

由上述性质对剪枝步进行改进，方法的基本思想是：删除集合 L_{k-1} 中所有满足 $|L_{k-1}(j)|<k-1$ 的频繁项集，j 为所有的项目，最后得到一个新的频繁项集 l_{k-1}，由 l_{k-1} 自连接生成拥有 k 项候选项集的集合 c_k。

举例说明：设 $L_3=\{\{I_1、I_2、I_3\}, \{I_1、I_2、I_5\}, \{I_1、I_2、I_4\}, \{I_1、I_2、I_6\}, \{I_1、I_2、I_7\}, \{I_2、I_3、I_4\}, \{I_2、I_3、I_5\}, \{I_2、I_3、I_6\}, \{I_2、I_3、I_7\}\}$，由上述修剪方法可得 $|L_3(1)|=5$，$|L_3(2)|=9$，$|L_3(3)|=5$，$|L_3(4)|=|L_3(5)|=|L_3(6)|=|L_3(7)|=2$，故删掉 L_3 中包含项目 I_4，I_5，I_6，I_7 的频繁项集得到 $l_3=\{\{I_1、I_2、I_3\}\}$，得新的 c_4 为空集，显然 L_4 也为空集。显然改进的 Apriori 算法只需扫描一次数据库，可以提前删除不可能包含于频繁项集的项，从而大幅度的减少匹配候选频繁项集的时间开销。可见较之 Apriori 算法的效率有了很大的提升。

2.4 算法描述

第一步：对 XML 文档进行解析、提取出事务和相应的项，并且以新的数据结构存储在 Hash 表中，构成事务集合数据库 D；

第二步：输入预定的最小支持度，并扫描数据库，在扫描过程中计算各个 1-项集的支持度，通过对各个 1-项集相对应的二进制代码中 1 的个数的统计计算，删除小于最小支持度的项集，得到集合 L_1；

第三步：连接步，为了产生 $k-$项集 L_k，运用改进的连接步的设计思想产生 $k-1$ 项的项集 L_{k-1}（项集中的项按在 D 中出现频率的大小进行生序排列）；

第四步：剪枝步，计算 L_{k-1} 中所有项目的频度，通过改进的剪枝步的设计思想来产生一个新的更小的 $k-1$ 项频繁项集的集合 l_{k-1}，再通过自连接生成 C_k；

第五步：将 C_k 中不满足给定最小支持度的项集删除掉，形成 L_k；

通过循环迭代计算，重复第三步到第五步，直到不产生新的频繁项集为止，其中各个项集的支持度为生成各个项集的连接项集的二进制代码按位与运算后代码中 1 的个数。

2.5 对算法进行实例数据的演示

这里以图表 3 的 XML 文档解析提取出的数据为例，每一个项目都以二进制代码的形式表示是否出现在 13 个事务中，给定最小支持度计数为 2。XApriori 算法的实例计算过程如表 4 所示。

表 4 XApriori 算法实例运算过程

L_1

Item	Sup	Item	Sup	Item	Sup
$\{I_1\}$	7	$\{I_4\}$	4	$\{I_8\}$	3
$\{I_2\}$	2	$\{I_5\}$	8	$\{I_9\}$	4
$\{I_3\}$	4	$\{I_7\}$	4		

C_2 ↓

Item	Sup	Item	Sup	Item	Sup
$\{I_2, I_8\}$	0	$\{I_8, I_9\}$	0	$\{I_4, I_1\}$	1
$\{I_2, I_3\}$	0	$\{I_8, I_1\}$	1	$\{I_4, I_5\}$	2
$\{I_2, I_4\}$	1	$\{I_8, I_5\}$	1	$\{I_7, I_9\}$	1
$\{I_2, I_7\}$	0	$\{I_3, I_4\}$	1	$\{I_7, I_1\}$	3
$\{I_2, I_9\}$	0	$\{I_3, I_7\}$	2	$\{I_7, I_5\}$	3
$\{I_2, I_1\}$	0	$\{I_3, I_9\}$	3	$\{I_9, I_1\}$	4
$\{I_2, I_5\}$	0	$\{I_3, I_1\}$	3	$\{I_9, I_5\}$	3
$\{I_8, I_3\}$	0	$\{I_3, I_5\}$	3	$\{I_1, I_5\}$	6
$\{I_8, I_4\}$	1	$\{I_4, I_7\}$	0		
$\{I_8, I_7\}$	0	$\{I_4, I_9\}$	1		

L_2 ↓

Item	Sup	Item	Sup	Item	Sup
$\{I_3, I_7\}$	2	$\{I_4, I_5\}$	2	$\{I_9, I_5\}$	3
$\{I_3, I_9\}$	3	$\{I_7, I_1\}$	3	$\{I_1, I_5\}$	6
$\{I_3, I_1\}$	3	$\{I_7, I_5\}$	3		
$\{I_3, I_5\}$	3	$\{I_9, I_1\}$	4		

C_3 ↓

Item	Sup	Item	Sup
$\{I_3, I_7, I_9\}$	1	$\{I_3, I_9, I_5\}$	2
$\{I_3, I_7, I_1\}$	1	$\{I_3, I_1, I_5\}$	2
$\{I_3, I_7, I_5\}$	1	$\{I_7, I_1, I_5\}$	2
$\{I_3, I_9, I_1\}$	3	$\{I_9, I_1, I_5\}$	3

L_3 ↓

Item	Sup	Item	Sup
$\{I_3, I_9, I_1\}$	3	$\{I_9, I_1, I_5\}$	3

C_4 ↓

Item	Sup
$\{I_3, I_9, I_1, I_5\}$	3

→

L_4 ↓

Item	Sup
$\{I_3, I_9, I_1, I_5\}$	3

3 算法分析与测试

对比 Apriori 算法，XAprior 算法有以下优点：

（下转第 188 页）

Water Transport (Academic Version), 2002, 6 (2): 108 - 110.
[3] ATCS Jeffrey J. Application of trend analysis methodologies on built-in-test (BIT) (and non-BIT) systems in a operational U. S. navy fighter/attack squadron [A]. 2005 IMTC - Instrumentation and Measurement Technology Conference [C]. Ottawa, Canada, 2005.
[4] 曾天翔. 电子设备测试性及诊断技术 [M] 北京：航空工业出版社，1996.
[5] 温熙森，徐永成，易晓山，等. 智能机内测试理论及应用 [M]. 北京：国防工业出版社，2001.

(上接第 180 页)

1) XApriori 算法只对 XML 文档数据库扫描一次，在后续计算只需要索引出相应项集的二进制代码，随着算法的运行，需要处理的事务和项集在不断地减小，从而节省了大量的时间，也大大的降低了 I/O 交互负载。

2) XApriori 算法的数据结构比较简单，设计了项目事务垂直对应的关系数据结构，对每一个项目进行了二进制编码，存储在 Hash 表中。在后续计算中使用了集合，使得计算比较节省时间。

3) XApriori 算法只在开始扫描数据库时对每个事务和项目进行了处理，在后续计算只需要索引出相应项集的二进制代码，不需对所有项集进行处理。例如支持项目 A 的事务的集合为 T_A，支持项目 B 的事务的集合为 T_B，则同时支持项目 A、B 的事务为两个集合交集（交集是两个集合的二进制代码进行按位逻辑与运算后的二进制代码所代表的事务集合，不再对事务中的项目进行匹配），这就使得寻找两个集合的交集时无需进行循环扫描。XApriori 算法避免了对所有项集进行大量的模式匹配计算和循环扫描，提高了时间效率。

4) XApriori 算法的连接步和剪枝步可以提前删除不可能包含频繁项集的候选项集，减少了冗余，使得算法的效率得到了较大的提高。

为了测试 XApriori 算法，用 Eclipse 分别实现了 Apriori 和 XApriori 算法，使用了 XML 文档解析提取出的关系型数据，该数据的事务数中为 1 000，平均每个事务拥的项集数为 10。该实验在内存为 2 GB，CPU 主频为 2 200 MHz，操作系统为 Windows7 的计算机上进行。实验结果如图 1 所示，可以看出，XApriori 算法有了较大的提高。

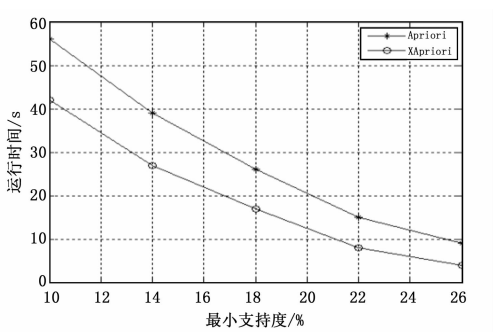

图 1 时间和支持度关系图

4 结束语

本文基于 XML 文档内容关联规则的挖掘，针对 Apriori 算法的缺陷，本文设计了一种高效的基于项目事务垂直对应关系的数据结构并且利用 Hash 表来存储、表示提取出的数据，并通过对 Apriori 算法的连接步和剪枝步进行了优化改进，提出了一种改进的 XApriori 算法，该算法应用于 XML 文档元素值之间关联规则的产生。Apriori 算法在运算中需要反复的扫描数据库，造成了非常大的 I/O 交互负载；在产生新的候选项集时需要反复的对上一级频繁项集进行对比与匹配，造成了运算效率的低下。而 XApriori 算法在整个过程中只需要扫描一次数据库，有效地降低了 I/O 交互负载；在生成候选项集的过程中，利用了逻辑运算，使得连接剪枝步运算大大的简化，逻辑运算对于计算机来说，使运算效率更高；在生成候选项集前，可以提前删除不可能包含频繁项集的候选项集，从而规避了冗余运算，使得算法的效率得到了较大的提高。

经过实验证明，XApriori 算法有效地提高了 Apriori 算法的运行效率。

参考文献：

[1] 源艳芬，梁慎青. 简单介绍可扩展标记语言 XML [A]. 电脑知识与技术，2010，20 (6)：5523 - 5526.
[2] 李 巍. 半结构化数据挖掘若干问题研究 [D]. 吉林：吉林大学，2013.
[3] 李 露，郑 琪. 数据挖掘原理与实践 [M]. 北京：电子工业出版社，2011.
[4] 熊 平. 数据挖掘算法与 Clementine 实践 [M]. 北京：清华大学出版社，2011.
[5] 苏 勇，王 燕. 基于 XQuery 的 XML 文档的关联规则挖掘 [J]. 计算机工程与科学，2011，5 (10)：91 - 95.
[6] Braga D, Campi A, Ceri S. Discovering interesting Information in XML Data with association rules [A]. Proceeding of the 14th International Conference [C]. 2003, (2454): 21 - 30.
[7] Feng L, Dillon T. An XML - enabled association rule framework [A]. Proceeding of the 14th International Conference [C]. 2003, (2736): 88 - 97.
[8] 蔚晓娟. 基于 DOM 的 XML 解析与应用 [J]. 计算机技术与发展，2007，17 (4)：86 - 89.
[9] J. Han, M. Kamber. 数据挖掘概念与技术 [M]. 范 明，孟小峰，译. 北京：机械工业出版社，2007.
[10] 周艳山. 数据挖掘中关联规则的研究与应用 [D]. 哈尔滨：哈尔滨工业大学，2005.
[11] 胡吉明，鲜学丰. 挖掘关联规则中 Apriori 算法的研究与改进 [J]. 计算机技术与发展，2006，16 (4)：99 - 101.
[12] 张 婕，张 燕，李广水. 基于 Hash 表的多谓词约束下频繁项集挖掘 [J]. 微电子学与计算机，2011，28 (10)：56 - 59.
[13] 黄根平，陈海勇，等. 数据集成中 XML 模式和关系模式映射模型研究 [J]. 信息工程大学学报，2009，10 (4)：529 - 531.
[14] 崔贯勋，李 梁，王柯柯，等. 关联规则中 Apriori 算法的研究与改进 [J]. 计算机应用，2010，30 (11)：2952 - 2955.