文章编号:1001 - 9081(2008)09 - 2341 - 04

一种改进的 FP-Growth 算法及其在业务关联中的应用

赵孝敏, 何松华, 李贤鹏, 尹 波
(湖南大学 计算机与通信学院,长沙 410082)
(xm_zhao8312@yahoo.cn)

摘 要:基于 FP-树的 FP-Growth 算法在挖掘频繁模式过程中需要递归地产生大量的条件 FP-树,效率不高,并且不太适合应用在移动通信业务交叉销售等具有业务约束的关联规则挖掘中。因此,提出了基于项目约束的频繁模式树 ICFP-树和直接在此树上进行挖掘的新算法——ICFP-Mine。理论分析和实验结果表明,ICFP-Mine 算法在内存占用和时间开销等方面比 FP-Growth 算法更优越,在移动通信业务交叉销售领域的应用中取得了较好的效果。

关键词:频繁模式;项目约束;ICFP-树;交叉销售
中图分类号: TP301.6 **文献标志码**:A

Improved FP-Growth algorithm and its applications in the business association

ZHAO Xiao-min, HE Song-hua, LI Xian-peng, YIN Bo
(*College of Computer and Communication, Hunan University, Changsha Hunan* 410082, *China*)

Abstract: The FP-Growth algorithm, based on FP-Tree, needs to create a large number of conditional FP-Trees recursively in the process of mining frequent patterns. It is not efficient and not good to apply in mobile communication business cross-selling, in which the association rules mining is business-constraint. Therefore, an items-constraint frequent pattern tree ICFP-Tree and a new ICFP-Mine algorithm which directly mines in the tree were proposed. Theoretical analysis and experimental results show that the ICFP-Mine algorithm is superior to FP-Growth algorithm in memory occupancy and time costs. It has achieved better results in the field of mobile communication business cross-selling applications.

Key words: frequent patterns; items-constraint; ICFP-Tree; cross-selling

0 引言

频繁模式的挖掘是关联规则分析、序列模式分析、相关分析等许多重要数据挖掘任务的关键步骤[1]。传统的大多数频繁模式挖掘算法均是以 Apriori 先验算法[2]为基础的,产生频繁模式时需要生成大量的候选项目集,而且必须多次扫描数据库对候选项目集进行筛选。为了避免生成候选项目集,文献[3]提出了基于 FP-树生成频繁项目集的 FP-Growth 算法,该算法无须生成候选项目集。因此,该方法显著地缩小了搜索空间,有效地避免了组合爆炸,挖掘效率明显提高。但 FP-树结构在挖掘过程中需要递归地创建大量的条件 FP-树,并且挖掘时通常要把整个 FP-树一起放入内存,对于非常大的数据库,这是不可能的。因此该算法存在明显的不足。

在移动通信企业数据业务深度运营中的业务交叉销售应用中,数据库通常是很大的,并且通常不需要把所有的关联规则全部挖掘出来,很多情况下只需要挖掘具有业务约束(例如只需要挖掘与 GPRS 业务相关的所有关联规则)和一定支持度约束(例如挖掘与 GPRS 相关的规则支持度为 50%,但 WAP 与彩信在一起时支持度又变为 40%)的关联规则。基于 FP-树的 FP-Growth 算法能够挖掘出数据库中隐含的所有频繁模式,但对于具有特定导向性的关联规则的挖掘,则不是优化的。

为此,本文提出了基于项目约束的频繁模式树 ICFP-树(Items-Constraint Frequent Pattern-Tree)和直接在此树上挖掘频繁模式的新算法 ICFP-Mine(Items-Constraint Frequent Pattern-Mine)。ICFP-树用来压缩存放所有只含有约束项目的事务的相关信息,ICFP-Mine 算法通过调整 ICFP-树的相关节点信息直接在 ICFP-树上采用深度优先的策略挖掘所需频繁模式,而不需要任何其他附加的数据结构,从而大大提高了挖掘效率。

1 FP-Growth 算法及 ICFP-Mine 算法

1.1 相关定义

设 $I = \{i_1, i_2, \cdots, i_m\}$ 是项的集合,D 是数据库事务的集合,事务 T 是项的集合,显然有 $T \subseteq I$。关联规则是形如 $A \Rightarrow B$ 的蕴涵式,其中 $A \subset I, B \subset I$,且 $A \cap B = \varnothing$。支持度是 D 中事务包含 $A \cup B$ 的百分比,即概率 $P(A \cup B)$。置信度是 D 中包含 A 的事务中同时也包含 B 的百分比,即条件概率 $P(B/A)$。提升度(也称相关性)等于置信度除以 B 的支持度,如果它的值小于1,则表示 A 和 B 是负相关的,意味着 A 的出现导致 B 出现的概率下降;如果等于1,表示 A 和 B 是独立的,意味着 A 和 B 没有相关性;若大于1,则表示 A 和 B 是正相关的,意味着 A 的出现导致 B 出现的概率增加。项集是项的集合,包含 k 个项的项集称为 k- 项集。频繁项集是满足最小支持度的项集,否则称非频繁项集。强关联规则是同时满足最小支持度阈值和最小置信度阈值的规则。

定义 包含约束项目的频繁模式集是指此频繁模式集中的所有频繁模式必须包含有用户指定的项目。

收稿日期:2008 - 03 - 12;**修回日期**:2008 - 05 - 19。

作者简介:赵孝敏(1983 -),男,广西桂林人,硕士研究生,主要研究方向:数据挖掘; 何松华(1964 -),男,湖南邵阳人,教授,博士,主要研究方向:数据挖掘、精确制导; 李贤鹏(1983 -),男,安徽芜湖人,硕士研究生,主要研究方向:数据挖掘; 尹波(1983 -),女,湖南株洲人,硕士研究生,主要研究方向:数据挖掘。

1.2 基于FP-树的FP-Growth算法

FP-树的构造以及FP-Growth挖掘算法的具体步骤参见文献[1]。因为FP-Growth算法将挖掘长频繁模式的问题转换成递归地发现一些短模式,然后连接后缀,并且挖掘的顺序是从最不频繁的一项开始。若用FP-Growth算法挖掘具有项目约束的频繁模式,则它必须要把不比约束项目频繁的所有项目挖掘完,才能从挖掘出的频繁模式集中筛选出包含约束项目的频繁模式。由于包含多个约束项目与包含一个约束项目挖掘的方法类似,下面以挖掘包含一个约束项目的例子来说明。

1.2.1 频繁模式树FP-树

频繁模式树FP-树用来压缩存放事务项的相关信息以便有效挖掘频繁模式。

例1 给定一个事务数据库D(表1的头两列),只需挖掘包含一个项目约束(本例取I2)的频繁模式,最小支持度设为2。

表1 AllElectronics某分店的事务数据

TID	项ID的列表	(降序排列后的)频繁项	包含I2的频繁项
T1	I1, I2, I5	I2, I1, I5	I2, I1, I5
T2	I2, I4	I2, I4	I2, I4
T3	I2, I3	I2, I3	I2, I3
T4	I1, I2, I4	I2, I1, I4	I2, I1, I4
T5	I1, I3	I1, I3	无
T6	I2, I3	I2, I3	I2, I3
T7	I1, I3	I1, I3	无
T8	I1, I2, I3, I5	I2, I1, I3, I5	I2, I1, I3, I5
T9	I1, I2, I3	I2, I1, I3	I2, I1, I3

首先,我们给出FP-树的定义如下。

1)由一个根节点(用"null"标记),一组根节点的项前缀子树和一个项头表组成。

2)项前缀子树的每个节点由项目名、节点计数、节点链和父节点指针四部分组成。其中节点计数表示通过该节点的事务的个数,节点链指向FP-树下一个拥有相同项目名的节点(当没有时为空)。

3)项头表的每个表目由项名和节点链头两个字段组成,其中节点链头指向FP-树中第一个该项的节点。

根据定义来构造例1的FP-树。首先,第一次扫描数据库,导出频繁项(1-项集)的集合,并得到它们的支持度计数(频繁性)。频繁项的集合按支持度计数降序排列,结果集记做L,这样有L=[I2:7,I1:6,I3:6, I4:2,I5:2]。然后,FP-树的构造如下:首先,创建树的根节点,用"null"标记。第二次扫描数据库D。每个事务中的频繁项按L中的次序排列并对每一个事务创建一个分支。例如,第一个事务"T1: I1,I2,I5",三个项都是频繁项,按L的次序排列得{I2,I1,I5},导致构造树的第一个分支〈(I2:1),(I1:1),(I5:1)〉。该分支具有三个节点,其中I2作为根的子女链接,I1链接到I2,I5链接到I1。第二个事务T2按L的次序包含I2和I4,它导致一个分支,其中I2链接到根,I4链接到I2。然而该分支应当与T1已存在的路径共享前缀I2。这样,我们将节点I2的计数增加1,并创建一个新节点(I4:1),并链接到I2。依此类推,得到FP-树(如图1)。

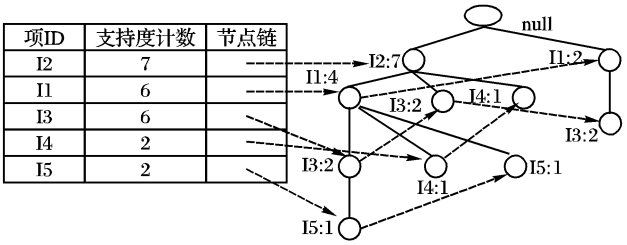


图1 例1中存放压缩的频繁模式信息的FP-树

1.2.2 FP-Growth算法

FP-树的挖掘如下:由长度为1的频繁模式(初始后缀模式)开始,构造它的条件模式基(一个"子数据库",由FP-树中与后缀模式一起出现的前缀路径集组成)。然后,构造它的(条件)FP-树,并递归地在该树上进行挖掘。模式增长通过后缀模式与由条件FP-树产生的频繁模式连接实现。

根据要求,本例只需要挖掘包含I2的所有频繁模式,由于它最频繁,所以就得把所有项目挖掘完才能把包含I2的频繁模式筛选出来。根据算法,我们从L的最后一项(即最不频繁的一项)I5开始。I5出现在图1的两个分支(I5的出现通过沿它的节点链找到):〈(I2,I1,I5:1)〉和〈(I2,I1,I3,I5:1)〉形成。这样考虑I5为后缀,它的两个对应前缀路径是〈(I2,I1:1)〉和〈(I2,I1,I3:1)〉,形成I5的条件模式基。它的条件FP-树只包含单个路径〈(I2:2,I1:2)〉;不包括I3,因为它的支持度小于2。该单个路径产生频繁模式的所有组合:I2 I5:2,I1 I5:2,I2 I1 I5:2。依此类推,挖掘过程和结果总结在表2中,最后再从表2中筛选出包含I2的频繁模式。

表2 通过创建条件(子)模式基挖掘FP-树

项	条件模式基	条件FP-树	产生的频繁模式	包含I2的频繁模式
I5	{(I2 I1:1),(I2 I1 I3:1)}	〈I2:2, I1:2〉	I2 I5:2, I1 I5:2, I2 I1 I5:2	I2 I5:2, I2 I1 I5:2
I4	{(I2 I1:1),(I2:1)}	〈I2:2〉	I2 I4:2	I2 I4:2
I3	{(I2 I1:2),(I2:2),(I1:2)}	〈I2:4, I1:2〉,〈I1:2〉	I2 I3:4, I1 I3:4, I2 I1 I3:2	I2 I3:4, I2 I1 I3:2
I1	{(I2:4)}	〈I2:4〉	I2 I1:4	I2 I1:4

1.3 基于ICFP-树的ICFP-Mine挖掘算法

FP-Growth算法在挖掘频繁模式时需要递归地产生大量的条件模式树,且需要把整个树放入内存,在大型的数据库中很难实现,此外,在移动通信等业务运用中通常只需挖掘具有业务约束的关联规则。

针对这些问题,本文提出了一种具有项目约束的频繁模式树ICFP-树和直接在ICFP-树上采用深度优先进行挖掘的ICFP-Mine算法。

1.3.1 具有项目约束的频繁模式树ICFP-树

ICFP-树与FP-树类似但有区别,区别主要在于ICFP-树只存放包含约束项目的事务数据的相关信息;树中每个节点只由三个域组成:节点名称item-name、节点计数count和父节点指针parent;把事务插入树时事务中项的排放次序是先约束项目再非约束频繁项目,并按支持度递减顺序排列。这样ICFP-树比FP-树不但要少存很多事务项,而且不需要节点链指针和项头表,这样就大大节省了树的存储空间。

ICFP-树的构造算法:

输入:事务数据库 D;最小支持度阈值 min_sup;约束项目

输出:ICFP-树

1)扫描 D 一次,只收集包含约束项目的事务中频繁项的集合 F 和它们的支持度。对 F 先约束项目再非约束频繁项目,并按支持度递减顺序排列,结果为频繁项表 L;

2)创建 ICFP-树的根节点,以 root 标记。再次扫描数据库 D,对 D 中包含约束项目的每个事务 Trans,执行:选择 Trans 中的频繁项集并按 L 中的次序排列,设排列后的频繁项表为[p|P],其中 p 为第一个元素,而 P 为剩余元素的表。调用 insert_tree([p|P], T),该过程执行如下:若 T 有子女 N 使得 N. item_name = p. item_name,则 N 的计数加 1;否则创建一个新节点 N,将其计数设为 1,并链接到它的父节点 T。若 P 非空,递归调用 insert_tree(P,N)。

1.3.2　ICFP-Mine 算法

ICFP-Mine 挖掘算法的基本思想就是:由于相同模式可能分布在不同子树中,当分别对每个子树进行挖掘时,无法判断其是否频繁,必须递归地构造其条件模式树才能判断,就如同 FP-Growth 算法。但如果把包含相同模式的这些子树合并成一个树,这样就可以直接判断该子树是否频繁而不需要再构造条件模式树。因此在 ICFP-树构造完之后,就开始深度优先地挖掘频繁模式,边挖掘边调整相关节点信息,把处在不同子树中的相同模式调整到一个子树中去。

ICFP-Mine 算法步骤如下:

输入:ICFP-树以及最小支持度 min_sup

输出:包含约束项目的所有频繁模式

对于以 ICFP-树的一级节点为根的各个子树 T_i, 调用挖掘函数 ICFP-Mine(T_i, *min_sup*):

```
1)   for 所有以 T_i 的子节点 β 为根的子树 sub-Tree
2)     if β. count ≥ min_sup then
3)       {把 β 及前缀路径一起作为一频繁模式放入频繁模式集(其中支持度为 β. count );
4)       Sib-Tree = 以 T_i 的兄弟节点名等于 β 为根的子树;
                                    //找到具有相同前缀路径的分支
5)       Unite( sub-Tree, Sib-Tree);          //合并这两个分支
6)       ICFP-Mine ( β, min_sup );            //递归调用 }
```

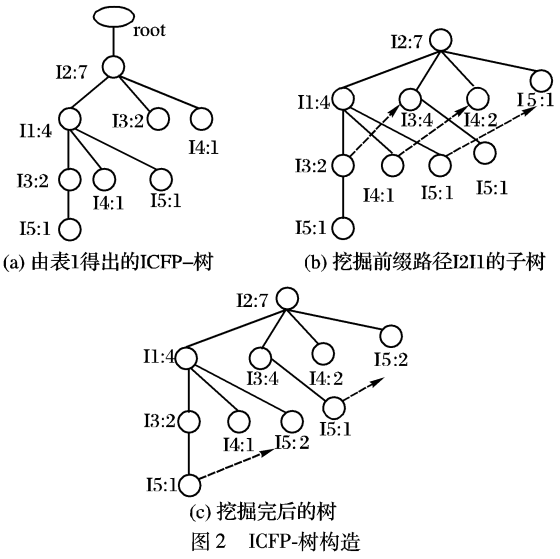
其中函数 *Unite*(*a*, *b*) 的实现如下:

```
1)   if b 为空分支 null then b = a;
2)   else b. count + = a. count
3)   for a 的所有子节点 a_i
4)     { b_i = b 的子节点名字等于 a_i 的节点
5)       Unite( a_i, b_i); }
```

1.3.3　举例说明

仍以例 1 给出的事务数据库为例(表 1 的前两列),同例 1 一样只挖掘包含 I2 项的频繁模式,最小支持度仍为 2。ICFP-树的构造跟 FP-树的构造类似,只是把不包含 I2 的事务(T5 和 T7)去掉,因为所有事务必须包含 I2,所以 I2 的支持度肯定最大,表 L = [I2:7,I1:4,I3:4, I4:2,I5:2],具体构造过程不再详述,最后所得的 ICFP-树,如图 2(a)所示。

ICFP-树构建完之后,用 ICFP-Mine 挖掘算法进行挖掘。按照深度优先,先挖掘 I2 子树的第一个分支,即前缀为〈I2, I1〉的三个分支,这三个分支要分别与 I1 的兄弟节点分支合并(如图 2(b)),继续往下挖掘,前缀路径为〈I2,I1,I3〉下的分支〈I5〉要与前缀路径为〈I2 I1〉下(即此 I3 节点的兄弟节点分支)的〈I5〉分支合并,这样前缀路径为〈I2 I1〉的子树挖掘完,得频繁模式{〈I2 I1:4〉,〈 I2 I1 I3:2〉,〈I2 I1 I5:2〉}。同理挖掘其他子树,前缀路径为〈I2 I3〉下的〈I5〉分支要与前缀路径为〈I2〉下的〈I5〉分支合并,得频繁模式{〈I2 I3:4〉,〈I2 I4:2〉,〈I2 I5:2〉},到此,ICFP-树就挖掘完了(如图 2(c))。我们把这里挖掘出的频繁模式与表 2 中包含 I2 的频繁模式进行比较,我们发现是一样的,且这里的挖掘速度要比 FP-Growth 算法快很多。



(a) 由表1得出的ICFP-树　(b) 挖掘前缀路径I2I1的子树　(c) 挖掘完后的树

图 2　ICFP-树构造

1.3.4　ICFP-树相对于 FP-树的优点

通过分析,本文提出的挖掘具有项目约束的 ICFP-树与经典的 FP-树相比具有以下优点。

1)树的存储空间。由于 ICFP-树只存放包含约束项目的事务数据的相关信息,树中每个节点只有三个域:节点名称、节点计数和父节点指针。这样 ICFP-树比 FP-树不但要少很多事务项信息,而且不需要节点链指针和项头表,从而大大节省了树的存储空间。

2)挖掘速度。基于 ICFP-树的 ICFP-Mine 算法采取深度优先地边挖掘边调整节点信息的策略,因此不需要递归地产生大量的条件模式树且 ICFP-树比 FP-树小很多,并且不需要把所有的频繁模式都挖掘出来然后再筛选出包含约束项目的频繁模式,因此 ICFP-Mine 算法要比 FP-Growth 算法快得多。

1.4　算法实现与比较

用 VC++6.0 在内存为 512 MB,CPU 为 P4 3.06 GHz 双核,操作系统为 Windows XP 的机子上实现了 ICFP-树的构造算法和 ICFP-Mine 挖掘算法并进行了性能测试,在同样实验环境下与基于 FP-树的 FP-Growth 算法进行了比较。数据集采用 http://www. ics. uci. edu/ ~ mlearn/ML Summary. html 上提供的与移动通信数据库类似的相对密集型的蘑菇数据库来进行实验。该数据库有 8 124 条记录,记录了蘑菇的 23 种属性。在进行算法性能比较时,两种算法在最小支持度一样时,每一次实验以蘑菇的一种属性做为项目约束条件,即固定一个最小支持度需做 23 次实验,然后取 23 次实验的时间平均值,然后再调整最小支持度的大小,重新实验。

图 3 显示了具有一个项目约束时两种算法运行时间随最小支持度阈值变化的测试结果,从图中可以看出,本文算法在不同支持度时,运行时间都明显优于 FP-Growth 算法;而且随支持度阈值的减小,本文算法的增长趋势也较平缓。因此本文算法在挖掘具有项目约束的频繁模式时相对于 FP-Growth

算法优势明显。

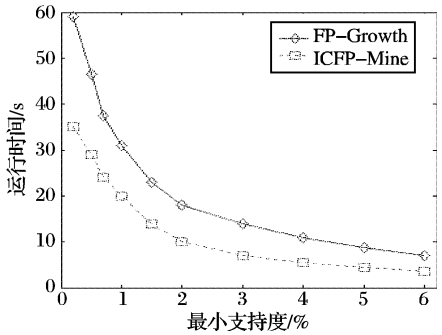


图3 最小支持度-运行时间

2 基于关联规则算法的移动通信业务交叉销售

本文将基于ICFP-树的ICFP-Mine挖掘算法应用于移动通信业务交叉销售的过程分为以下几个步骤。

1)数据准备。从移动通信业务数据库中的定单表、帐单表、用户表等相关表中经过数据清洗抽取出客户的自然属性、业务属性和消费属性放入数据挖掘集市的相关表中。

2)使用基于项目约束的频繁模式树ICFP-树的ICFP-Mine挖掘算法,并且在此基础上加入序列模式GSP算法[4-5],挖掘出业务的关联性与时间的关系。

3)根据频繁模式产生强关联规则并在关联规则的兴趣度衡量上加入提升度分析,筛选出企业感兴趣的有价值的规则。

4)根据挖掘出的关联规则和序列模式结果,企业的销售人员对客户进行有针对性的主动营销和交叉营销。

3 算法应用

本次应用的需求是只要挖掘出包含GPRS或者彩信的所有关联规则,而且此需求必须在一天内响应。由于客户记录太多,常用的FP-Growth挖掘算法虽然能挖掘出所有的规则,但时效性难以满足需求。而采用本文提出的ICFP-Mine挖掘算法虽然只输出以“彩信”、“GPRS”为头的规则,但时效性很好。本次应用的数据是通过对某市移动通信企业的2007年6~8月三个月的数据业务定单表、账单表和用户表通过数据预处理而来的,处理的规则是只要用户在这三个月中任意一天定制了或使用了某一业务,就算此用户有此业务。由于移动通信的增值与数据业务太多,本次应用只选取具有代表性的13种增值业务(分别是:GPRS、WAP、短信、彩信、彩铃、梦网邮箱、全球呼、手机杂志、移动秘书、移动沙龙、一机多号、手机定位、一卡双号)和各种业务的定制或者第一次使用的时间以及客户号作为输入数据的属性。在满足要求的客户中抽取10万个客户记录,共26种属性作为输入数据。

3.1 挖掘移动业务关联规则

把应用数据加载到算法程序中,指定最小支持度为5%,根据算法输出的频繁模式再由产生关联规则的方法[1]并加入置信度40%的限制产生出强关联规则,最后加入提升度概念之后筛选出以下关联规则加以分析。

规则说明:

1)从第1条规则可以得出,客户是否使用彩信与客户是否使用彩铃无关(提升度为1.08,接近1);

2)从第2、3、4条规则可以得出,客户是否使用彩信业务与客户是否使用过GPRS、WAP有密切的相关(提升度分别达到1.95,1.85),特别是GPRS业务支持度达到40.35%,置信度达81.88%;而在客户同时使用GPRS和WAP的情况下与单独使用过GPRS和WAP中一种的情况下对比,提升度稍有增加;

3)从第5条规则可以得出,在客户同时使用全球呼和彩铃的情况下,客户使用彩信的可能性比较大(置信度达71.12%),并且提升度也还不错;

4)从第6条规则可以得出,在客户同时使用彩信和彩铃的情况下,客户使用GPRS的可能性很大(置信度达96.81%),并且提升度很高。

表3 关联规则结果表

规则号	规则内容	支持度/%	置信度/%	提升度/%
1	[彩铃] ⇒ [彩信]	20.88	45.42	1.08
2	[WAP] ⇒ [彩信]	5.34	77.62	1.85
3	[GPRS] ⇒ [彩信]	40.35	81.88	1.95
4	[GPRS] +[WAP] ⇒ [彩信]	5.24	88.27	2.11
5	[全球呼] +[彩铃] ⇒ [彩信]	8.37	71.12	1.61
6	[彩信] +[彩铃] ⇒[GPRS]	20.21	96.81	1.96

3.2 挖掘移动业务序列模式

把应用数据载入到序列模式GSP算法程序中,抽取出如下典型的序列模式(见表4)。

表4 序列模式结果表

序列模式	模式说明
[彩信] +[彩铃]⇒[GPRS][0.15,0.76, 32]	客户在申请了彩信和彩铃之后,大约在一个月后会再申请GPRS业务,支持度达15%,置信度达76%

3.3 结论分析

通过分析以上挖掘出来的关联规则和序列模式,我们可以得出以下几点结论和销售建议。

1)主动业务推送建议。可以在客户开通彩铃或者GPRS或者WAP业务时,主动推销彩信业务或者主动帮客户开通彩信业务。

2)主动促销建议。可以在开通或使用了彩信和彩铃业务的客户中,在大约一个月后主动向这些客户推销GPRS业务,这样交叉销售的成功率就会高很多。也可以推断出这部分客户是对新业务比较感兴趣的,比较时尚的年轻客户。

3)商品搭售或捆绑销售建议。可以将全球呼、彩铃和彩信业务进行捆绑销售,这样成功率高,也使客户能同时体验更多的服务,为以后的新业务向这些客户促销打下基础。

4 结语

理论和实验都证明,本文提出的基于项目约束的频繁模式树ICFP-树的ICFP-Mine挖掘算法,在挖掘具有项目约束的频繁模式时明显优于基于频繁模式树FP-树的FP-Growth算法,适用于移动通信等大型数据库中的业务关联规则的挖掘需求。算法在应用中不仅挖掘出了有价值的业务关联规则,而且还挖掘出了某些业务比较好的促销时间,这为企业制定出良好的业务交叉销售策略提供了有力的技术支持,可以大大提高交叉销售的成功率,从而提高企业的市场竞争力。

(下转第2348页)

要步骤是查找不同次采样中相同的样本，复杂度也是 $O(k^2T^2)$；异概捕获算法则包括构造样本捕获概率的 Logistic 函数、最优化条件似然函数和利用样本采样概率估计索引量，这三个步骤的复杂度分别为 $O(k^2T^2)$、$O(kT^2)$ 和 $O(kT^2)$，故总的复杂度是 $O(k^2T^2)$。因此四种算法在样本数量 kT 相同的情况下复杂度相当，均为 $O(k^2T^2)$，可以由相同样本数量情况下算法的表现对四种算法进行对比。

3.5 结果与讨论

在采样-再采样和高频再采样算法的实验中，本文研究了在样本集大小从 300 到 1 500 的不同情况下两种算法的表现。采样-再采样算法的表现很差，且绝对误差比浮动很大，随着样本集大小的变化为 50% ~400%，这与 Shokouhi 等的实验结果[7]是一致的，因此认为采样-再采样算法并不适合估算索引量大小。而高频采样算法的绝对误差比稳定在 32% ~36%之间，在样本集大小为 1 000 时达到最小值 32%。

在不同的捕获次数 T 下，多重捕获-再捕获算法和异概捕获算法的绝对误差比如图 2 所示。从图中可以看出，多重捕获-再捕获算法的绝对误差比不随捕获次数变化而变化，稳定在 41%，而异概捕获算法的绝对误差比随着捕获次数的增多而降低，表现要好于多重捕获-再捕获算法，捕获次数越多，表现越好。这是可以预计的，因为在多重捕获-再捕获算法中，通过增加捕获次数来减小单次捕获造成的随机误差，但是其边际效益随着捕获次数增加迅速减少，达到一定捕获次数之后，随机误差消除，剩下的即无法通过多次捕获消除的系统误差；而捕获次数越多，被捕获的文档也就越多，在异概捕获算法中获得的信息就越多，条件似然函数也就越逼近全似然函数，因此异概捕获算法的表现也就越好。除了表现优于多重捕获-再捕获算法，这也是异概捕获算法的优势之一，可以通过调节捕获次数即效率来调节算法的精确度即表现。

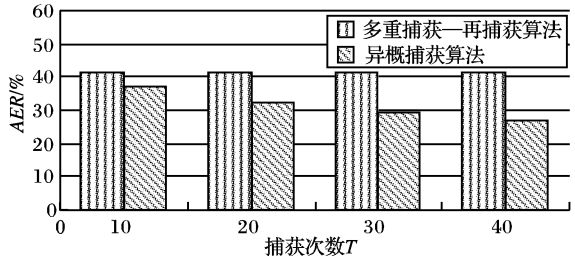


图 2 多重捕获-再捕获与异概捕获的绝对误差比对比

综上所述，异概捕获算法表现最好，且不需要节点返回查询结果数。多重捕获-再捕获算法同样不需要返回结果数，但高频再采样算法在节点能准确返回结果数的情况下表现优于多重捕获-再捕获算法。采样-再采样算法不适合于估算索引量大小。

4 结语

本文提出了高频再采样算法和异概捕获算法以解决在非合作情况下分布式搜索引擎在资源库选择中估算资源库索引量大小的问题。高频再采样算法在随机采样后利用文档频率较高的关键词进行再采样，能有效地解决采样-再采样算法中随机关键词在样本集和整个文档集中的出现频率近似相等的假设与真实情况不符的问题，使得估算精度得到较大的提高；而异概捕获算法解决了多重捕获-再捕获算法中所有文档获取概率相同的假设与真实情况矛盾的问题，基于文档被捕获概率不同的假设，利用 Logistic 函数和条件似然方法，估算资源库索引量的大小。从实验结果看，高频再采样和异概捕获算法比采样-再采样和多重捕获-再捕获算法有较明显的提高。

参考文献：

[1] BrightPlanet Corporation. The deep Web: Surfacing hidden value [EB/OL]. [2008 - 01 - 24]. http://www.brightplanet.com/resources/details/deepweb.html.

[2] MENG W Y, YU C, LIU K L. Building efficient and effective metasearch engines[J]. ACM Computing Surveys, 2002, 34(1): 48 - 89.

[3] XU J, LI X. Learning to rank collections[C]// Proceedings of the 30th ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2007: 765 - 766.

[4] GRAVANO L, CHANG C C K, GARCIA-MOLINA H. STARTS: Stanford proposal for internet meta-searching[C]// Proceedings of the 1997 ACM SIGMOD Conference on Management of Data. New York: ACM Press, 1997: 207 - 218.

[5] SI L, CALLAN J. Relevant document distribution estimation method for resource selection[C]// Proceedings of the 26th ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2003: 298 - 305.

[6] LIU K, YU C, MENG W. Discovering the representative of a search engine[C]// Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management. New York: ACM Press, 2002: 652 - 654.

[7] SHOKOUHI M, ZOBEL J, SCHOLER F, *et al.* Capturing collection size for distributed non-cooperative retrieval[C]// Proceedings of the 29th ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2006: 316 - 323.

[8] 刘力平. 捕获再捕获与捕获移出模型的概念、方法和应用[J]. 数学进展, 2004, 33(5): 527 - 539.

[9] ALHO J M. Logistic regression in capture-recapture models[J]. Biometrics, 1990, 46: 623 - 635.

[10] HUGGINS R. On the statistical analysis of capture experiments[J]. Biometrika, 1989, 76: 133 - 140.

[11] HORVITZ D G, THOMPSON D J. A generalization of sampling without replacement from a finite universe[J]. Journal of the American Statistical Association, 1952, 47: 663 - 685.

(上接第 2344 页)

参考文献：

[1] HAN J W, KAMBER M. 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, 2001: 156 - 161.

[2] AGRAWAL R, SRIKANT R. Fast algorithms for mining association rules[C]// Proceedings of 1994 International Conference on Very Large Data Bases(VLDB'94). Santiago: [s. n.], 1994: 487 - 499.

[3] HAN J, PEI J, YN Y. Mining frequent patterns without candidate generation[C]//Proceedings of 2000 ACM-SIGMOD International Conference on Management of Data. Dallas: [s. n.], 2000: 1 - 12.

[4] AGRAWAL R, SRIKANT R. Mining sequential patterns[C]//Proceedings of the 11th International Conference on Data Engineering. Avignon: [s. n.], 1995: 3 - 10.

[5] SRIKANT R, AGRAWAL R. Mining sequential patterns: generalizations and performance improvements[C]// Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96). London: Springer-Verlag, 1996: 3 - 17.

[6] 范明, 李川. 在 FP-树中挖掘频繁模式而不生成条件 FP-树[J]. 计算机研究与发展, 2003, 40(8): 1216 - 1222.