基于 Web 日志的频繁访问页面挖掘研究

The Study of Frequent Access Pages Mining Based on Web Log WANG Tao - wei

王涛伟 （浙江万里学院 计算机与信息学院 宁波 315100）

摘要：挖掘最大频繁项目集是多种数据挖掘应用中的关键问题。在经典 Apriori 算法基础上给出了基于 SQL 的 Apriori 算法。对 Web 日志挖掘进行数据预处理的基础上，利用算法挖掘最大频繁访问页面集。实验结果表明算法的效率较好，并有助于促进网站的建设。

关键词：Web 日志挖掘 数据预处理 关联规则 最大频繁项目集

关联规则是数据挖掘中一个重要的研究课题，而生成最大频繁项目集是关联规则挖掘的关键技术。Agrawal 等人于 1993 年首次提出布尔型关联规则问题以及相应的 Apriori 算法[1]，其后诸多的研究人员对关联规则的挖掘问题进行了大量的研究，以提高挖掘效率[2-9]。本文提出了基于 SQL 的 Apriori 算法，给出了 Web 日志挖掘的数据预处理过程，并在此基础上对算法进行了实验。

1 关联规则的描述

1.1 关联规则挖掘问题的形式化描述

定义 1 关联规则：设 I 是 Web 日志的一条记录，即 $I=\{i_1,\ldots,i_m\}$，其中 $i_j(1\leq j\leq m)$ 是某用户访问 Web 网站的数据，如时间、IP 地址、请求的方法、被请求文件的 URL 等。$T_i\in I$ 为 I 的一个子集。事务数据库 $D=\{T_1,T_2,\cdots,T_n\}$ 是关于 T_i 的集合，且 $X\in I,Y\in I,X\cap Y=\Phi$，则记录 $X\Rightarrow Y$ 为在集合 D 中 X 与 Y 的关联规则。

定义 2 支持度：如果 $X\Rightarrow Y$ 在 T 中的 s% 成立，则称 $X\Rightarrow Y$ 的支持度为 s%，即

$s\% = (|\{t|t\text{中含有}X,Y\}|/|T|)\cdot 100\%$，支持度 s% 表示 $X\Rightarrow Y$ 中出现的程度。

定义 3 可信度 c%：表示 D 中 c% 的包含 X 的事务同时也包含 Y。

$c\% = (|\{t|t\text{中含有}X,Y\}|/|\{t|t\text{中含有}X\}|)\cdot 100\%$

定义 4 项目的集合称为项目集（Itemset），包含 k 个项的项目集称之为 k - 项目集。如果项目集满足最小支持度，则它称之为 k - 频繁项目集（Frequent Itemsets）。

关联规则挖掘算法的基本问题是：给定一个交易数据库 D，找出所有满足最小支持度和最小可信度的关联规则。该问题可以分解为以下两个子问题：

（1）求出 D 中满足最小支持度的所有频繁项目集（或称大项目集，Large Itemsets）；

（2）利用第 1 步产生的频繁项目集，生成满足最小可信度的所用关联规则。

问题 2 的解决方法较为简单，对每个频繁项目集 L 计算其所有非空子集，对每个非空子集 a，考察规则 $a\Rightarrow(L-a)$，如果该规则的可信度大于最小可信度，则输出此规则。

问题 1 的求解是关联规则挖掘的关键部分，即如何高速地求出频繁项目集。Apriori 算法[4]被认为是最经典的算法。该算法利用了大项目集的基本原理：若项目集 X 是大项目集，则 X 的任一子集也必定是大项目集；反之若 X 有一子集不是大项目集，则 X 也肯定不是。

1.2 Apriori 算法描述

输入：事务数据库 D；最小支持度阈值 min_sup。

输出：D 中的频繁项集 L_k。（其中 L_k 表示：k - 频繁项目集，C_k 表示：k - 项目集）

（1）L_1 = find_frequent_1 - itemsets(D)；

（2）for ($k=2;L_{k-1}\neq\Phi;k++$){

(3) C_k = apriori_gen(L_{k-1}, min_sup);

(4) for each transaction t ∈ D{ //扫描 D,对每个事务操作

(5) C_t = subset(C_k, t); //subset 函数找出事务中是候选的所有子集

(6) for each candidate c ∈ C_t

(7) c.count + +; //对每个候选项集计数

(8) }

(9) L_k = {c ∈ C_k | c.count ≥ min_sup }

(10) }

(11) return L = $\cup_k L_k$;

procedure apriori_gen(L_{k-1}, min_sup)

(1) for each itemset $l_1 \in L_{k-1}$

(2) for each itemset $l_2 \in L_{k-1}$

(3) if($l_1[1] = l_2[1]$) ∧ ($l_1[2] = l_2[2]$) ∧ … ∧ ($l_1[k-2] = l_2[k-2]$) ∧ ($l_1[k-1] < l_2[k-1]$) then {

(4) $c = l_1 \infty l_2$;

(5) if has_infrequent_subset(c, L_{k-1}) then

(6) delete c;

(7) else add c to C_k;

(8) }

(9) return C_k;

procedure has_infrequent_subset(c, L_{k-1})

(1) for each (k-1)-subset s of c //求(k-1)-项目集的所有(k-2)-项子集

(2) if s ∉ L_{k-1} then

(3) return true;

(4) return false;

apriori_gen( ) 做两个动作:连接和剪枝。在连接部分(第 1~4 步),L_{k-1} 与 L_{k-1} 连接产生可能的候选项。剪枝部分(第 5~7 步)利用大项目集的基本原理,删除具有非频繁子集的候选。has_infrequent_subset( ) 测试频繁项集的所有子集是否是频繁。

2 基于 SQL 的 Apriori 算法

2.1 算法的思路

数据挖掘的很多原始数据是以表的形式存放在数据库中的,利用 SQL 语句直接对数据库中的数据进行操作可以省去把数据从数据库中提取再用程序控制实现的过程。SQL 语言只利用其提供的 9 个动词就可满足用户从建库到查询、数据维护和简单统计等对数据库的操作需求,其主要原因之一是它的命令语句中包含了一些使用简单而高效的操作子句和函数。例如 SQL 查询语句的 GROUP BY 语句可以选择数据库的一个子集合,将它投影到一组属性的子集合上,并对相同的记录进行合并和计数。对于给定的属性集合表,以及查询限定的条件,SQL 的 GROUP BY 很容易返回对满足条件子句 HAVING 的所有记录的计数。这种 SQL 计数查询正好可以用在关联规则算法的支持度计数过程上,而无须再用烦琐的程序来循环控制。另外频繁项目集产生过程中的连接操作可以由连接查询方便地实现。

2.2 算法的描述和分析

利用大项目集的基本原理和 SQL 语句的特点,算法描述如下。

引入算法的三个重要标记:

2.2.1 数据源(Source sets):用以生成 i 维项目集的 i 维数据集,记为 S[i];

2.2.2 结果集(Result sets):即 i 维大项目集(L[i]),记为 R[i];

2.2.3 中间集(Middle sets):利用 i 维大项目集对 i 维数据源筛选后剩下的数据集合,记为 M[i];然后用以合并生成 i+1 维的数据源 S[i+1]。

(1) 输入:数据源,即 S[1];最小支持度阈值 min_sup。

(2) R[1] = Generate_ResultSets(S[1],1);

(3) i=1;

(4) While(R[i] ≠ Φ) {

(5) M[i] = Generate_MiddleSets(S[i],R[i],i)

(6) i=i+1;

(7) S[i] = Generate_SourceSets(M[i-1],M[1],i);

(8) R[i] = Generate_ResultSets(S[i],i)

(9) }

(10) 输出:R[i]的各条记录;

函数说明:

① procedure Generate_ResultSets(S[i],i) //生成 i 维的结果集

insert into ResultSets[i]

Select item1, item2, …… itemi, count(*) from S

[i]

Group by item1,item2,……itemi

Having count(*) ≥ min_sup

Order by item1,item2,…,itemi;

② procedure Generate_MiddleSets(S[i],R[i],i)

//生成 i 维的中间集

insert into MiddleSets[i]

Select S[i]. id,S[i]. item1,……S[i]. itemi from S[i],R[i]

where S[i]. item1 = R[i]. item1 and ……and S[i]. itemi = R[i]. itemi

order by S[i]. id;

③ procedure Generate_SourceSets(M[i-1],M[1],i) //生成 i 维的数据集

insert into SourceSets[i]

Select M[i-1]. id,M[i-1]. item1,……M[i-1]. itemi-1,M[1]. item1 from M[i-1],M[1]

Where M[i-1]. id = M[i]. id and M[1]. item1 > M[i-1]. itemi-1

Order by M[i-1]. id;

算法分析：

(1) 输入原始数据源，也就是 1 维数据源 S[1]；

(2) 使用函数 Generate_ResultSets 产生 1 维结果集 R[1]，通过利用 item1 对 S[1]进行分组查询得到；

(3) 使用函数 Generate_MiddleSets 从 i 维数据源 S[i]中筛选出含有结果集 R[i]的那一部分数据源，筛选的方法是利用 S[i]和 R[i]的对应项相等；

(4) 使用函数 Generate_SourceSets 将 i-1 维中间集 M[i-1]和原先的 1 维中间集 M[1]组合生成 i 维数据源 S[i]，方法是使用 M[i-1]和 M[1]的 id 相等，而且 M[1]的项目不会和 M[i-1]的项目重复(由于数据库已经按 item 排好序，所以只须 M[1]. item1 > M[i-1]. itemi-1)。

另外 Apriori 算法是每次都扫描原始的数据集，而基于 SQL 的 Apriori 算法除第一次是扫描原始数据集，而其余执行都是扫描 1 维中间集 M[1]得到数据源集。由于 M[1]是大于最小支持度的 1 维频繁项目集，用 M[1]可以代替原始数据集 S[1]。这样可以有效的缩小搜索空间，从而提高算法的效率。

(5) 使用函数 Generate_ResultSet 产生 i 维结果集 R[i]，方法利用 item1,item2,…,itemi 对 S[i]分组查询得到。

3 Web 日志挖掘

一般将 Web 挖掘分为 Web 内容挖掘、Web 结构挖掘和 Web 日志挖掘三类。Web 日志挖掘的主要目标是从 Web 的访问日志中提取感兴趣的模式，整个 Web 日志挖掘的过程分为三个阶段，即数据准备阶段、挖掘算法实施阶段和规则模式应用阶段，如图 1 所示。对于 Web 日志的挖掘，目前已经进行了大量的研究，也出现了一些商业的挖掘工具如：WebTrends，AWStats 等。通过 Web 日志挖掘，发现用户的访问兴趣和模式，协助管理者优化站点结构，提高站点效率，对于电子商务网站的商业决策具有重要的意义。

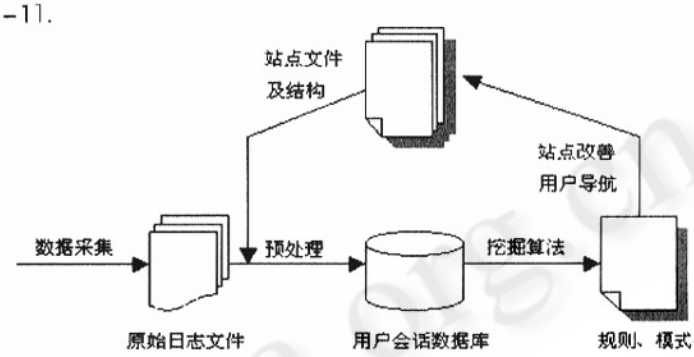

图 1　Web 日志挖掘过程

4 Web 日志的数据预处理

Web 日志挖掘算法实施首先要对日志文件进行数据预处理，包括数据净化，用户识别，会话识别，路径补充。典型的 Web 日志主要包括以下信息：日期(date)、时间(time)、客户 IP 地址(c-ip)、服务器 IP 地址(s-ip)、请求的方法(cs-method)、被请求文件的 URL(cs-uri-stem)、协议状态(cs-status)、引用页的 URL(cs(Referer))及用户代理(cs(User-Agent))等。

数据净化是指删除掉 Web 日志中与挖掘算法无关的数据，如清除掉状态代码为"4xx"和"5xx"的记录，URL 后缀为. gif(. Gif),. jpg(. jpeg,. JPG,. JPEG),.

bmp(.BMP),.map 等的文件。

由于本地缓存和代理服务器的存在,必须尽量从日志文件中识别每一个用户。一般使用一些启发式规则,如 IP 地址相同,但是代理不同,则认为不同的用户等。

会话识别的目的将用户的访问记录分为单个的会话,表示用户对站点的一次连续浏览行为。如认为用户对两页面的请求时刻的差值超过了一定的界限,就认为用户又开始了一个新的会话。

路径补充就是确定访问日志中是否有重要的请求没被记录。当前请求的页与上一次请求的页面之间没有超文本链接,如用户是通过搜索引擎结果点击进入的,则可以利用引用页确定请求来自哪一页。

经过数据预处理,Web 日志的数据被转换为数据库中的事务记录。对于最大频繁访问页面的挖掘,只需要记录号 Tid 和 URI 资源两项,即经过数据预处理后的日志表中为 Tid 字段和 cs－uri－item 字段内容,如表 1 所示。

表 1　预处理后的日志表

Tid	Cs－uri－Item
100	/default.asp
100	/dh/default.htm
200	/xy‖.htm
200	/yx|s/yx|s.htm

对用户最大频繁访问页面的挖掘,我们利用基于 SQL 的 Apriori 算法挖掘用户最大频繁访问页面集。

5　挖掘最大频繁访问页面实验结果

挖掘之前对一级子页面为网站主页 http://www.zjwu.net/default.asp 上链接的 15 个分类页面,如"学校概况"、"二级网站"、"网上教学"等。一级页面链接的下一级为二级子页面,共 42 个,以此类推,建立页面映射表,如 http://www.zjwu.net/default.asp 映射为页面 a1,对预处理过的日志表通过页面映射表进行映射,以便于算法的实现。

以浙江万里学院 http://www.zjwu.netWeb 服务器上 04－10－29 08:10:12 到 04－10－29 10:58:10 两个多小时内的日志文件做实验,最后经过数据预处理后的数据库表中共有 13800 条记录。用基于 SQL 的 Apriori 算法和 Apriori 算法挖掘用户频繁页面集。用 VC++6.0 在内存 128M,CPU 为 PentiumⅣ1.6GHz,操作系统为 Windows 2000 Server,数据库管理系统为 SQL Server2000 的 PC 上,对基于 SQL 的 Apriori 算法进行实验。表 2 显示了在不同的最小支持度(分 3 档:3%,2%,1%)、不同记录数(7200 条,13800 条)挖掘的最大频繁页面集和所用的时间。

表 2　算法挖掘的结果

记录数	min_sup=1% 执行时间	最大频繁页面集	min_sup=2% 执行时间	最大频繁页面集	min_sup=3% 执行时间	最大频繁页面集
7200	23s	{a1,b2,b4,b5} {a1,b4,b5,c5} {a1,b6,b7,c6,c7}	12s	{a1,b4,b5} {a1,b6,c6,c7}	8s	{a1,b4} {a1,b6,c6,c7}
13800	39s	{a1,b4,b5,c5} {a1,b6,b7,c6,c7}	21s	{a1,b4,b5} {a1,b6,c6,c7}	13s	{a1,b4,b5} {a1,b4,c7}

其中 a1,b2,b4,b5,b6,b7,c5,c6,c7 分别对应页面:主页,学校概况,新闻中心,二级网站,网上教学,教务管理,机构设置,网上课堂,教学通知。图 2 显示了对 7200 条记录在不同的最小支持度(分 5 档:5%,4%,3%,2%,1%)基于 SQL 的 Apriori 算法和 Apriori 算法生成频繁项目集所用的时间比较。

从表 2 的挖掘结果可以知道网站中哪些是用户最频繁访问的页面集,这些结果对分析站点的使用情况,改善站点的结构和内容都具积极的指导作用。基于 SQL 的 Apriori 算法直接利用 DBMS 处理关系表,实现过程不断采用临时表,不断缩小搜索空间,充分利用了 SQL 语言的特点,实现简单,从图 2 可知比直接利用 Apriori 算法执行效率高。但是 Apriori 算法的主要缺点是多次扫描事务数据库和产生的候选项目集个数太多,当存在长模式时,效率太低。而采用 FP－Tree 存储结构的的 DMFIA[6],FPST[7] 和 FIUA[8] 算法在效率上较之 Apriori 算法有很大的提高。

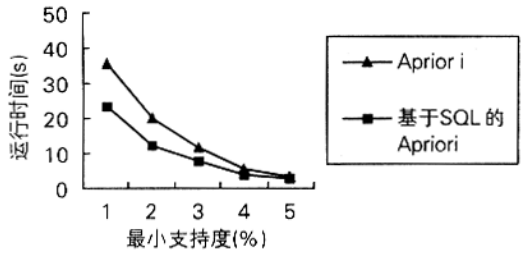

图 2　Apriori 和基于 SQL 的 Apriori 算法比较

参考文献

1 R Agrawal et al. Mining association rules between sets of items in larger databases[C]. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Washington, DC, 1993, 2: 207 ~ 216.

2 Han J W., Pei J., Yin Y.. Mining partial periodicity using frequent pattern tree[R]. In CS Tech, Rep, 99 – 10, Simon Fraser University, 1999.

3 Han J, Pei B. Mortazavi – Asl: Frequent Pattern – projected Sequential pattern Mining[C]. Proceed – ings of the 2000 Int. Conf. KDD'00 , Boston, MA, 2000.

4 Han J. W., Pei J., Yin.. Mining frequent patterns without candidate generation[A]. In: Proceedings of the 2000 ACM – SIGMOD International Conference on Management of Data [C]. Dallas, 2000, 1 ~ 12.

5 路松峰、卢正鼎，快速开采最大频繁项目集[J]，软件学报，2001，12(2)：23 ~ 297.

6 宋余庆、朱玉全、孙志挥、陈耿，基于 FP – tree 的最大频繁项目集挖掘及更新算法[J]，软件学报，2003，14(9)：1586 ~ 1592.

7 惠晓滨、张凤鸣、虞键飞等(2003)，一种基于栈变换的高效关联规则挖掘算法，计算机研究与发展，40(2)：330 – 335。

8 朱玉全、孙志挥、季小俊(2003)，基于频繁模式树的关联规则增量式更新算法，计算机学报，26(1)：91 – 96。

9 吉根林、杨明、宋余庆、孙志挥，最大频繁项目集的快速更新[J]，计算机学报，2005，28(1)：128 ~ 135.