

改进的 PrefixSpan 算法在 Web 挖掘中的应用

谢清森 杨天奇

(暨南大学,广州 510632)

摘要 针对 PrefixSpan 算法不足,采用修改 Prefix 策略与舍弃非频繁项的方法,减少内存与外存之间频繁地交换,减小在挖掘过程中产生的投影数据库规模,降低构建、扫描投影数据库的时空耗费,从而改进算法。实验结果表明,在长序列模式挖掘中,算法在改进后运行效率比原来提高 35% 以上,更适用于 Web 挖掘。

关键词 Web 挖掘 PrefixSpan 算法 序列模式

中图法分类号 TP391.3; **文献标志码** B

随着 Internet 的飞速发展,各种网络数据量呈指数级的速度增长,如何在浩瀚的数据中找到有价值的信息,是 Web 挖掘要解决的问题^[1]。日志挖掘是 Web 挖掘的一个重要研究方向,而 prefixSpan 在日志挖掘中的效率远胜于其他经典数据挖掘算法。

1 PrefixSpan 算法

1.1 相关概念描述

1.1.1 子序列(Sub-Sequence)

假设序列 $s_1 = \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle, s_2 = \langle \beta_1, \beta_2, \dots, \beta_m \rangle (n \leq m)$, s_1 是 s_2 的子序列,当且仅当存在 $1 \leq j_1 < j_2 < \dots < j_n \leq m$,使得 $\alpha_1 \subseteq \beta_{j_1}, \alpha_2 \subseteq \beta_{j_2}, \dots, \alpha_n \subseteq \beta_{j_n}$ 。

1.1.2 前缀(Prefix)

假设两个给定的序列 s_1 与 s_2 中全部数据项都按字母顺序排列,其中 $s_1 = \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle, s_2 = \langle \beta_1, \beta_2, \dots, \beta_m \rangle (n \geq m)$,若 s_2 是 s_1 的前缀,当且仅当: $\alpha_i = \beta_i, i \leq m; \beta_m \leq \alpha_m; (\alpha_m - \beta_m)$ 中的所有数据项均按字母顺序排序,且均排在 β_m 之后。

1.1.3 投影(Projection)

给定序列 $s_1, s_2 (s_2 \subset s_1)$, s'_1 称为 s_1 对应序列 s_2

的投影,当且仅当

(1) s_2 是 s'_1 的前缀;(2) s'_1 是 s_1 的满足条件(1)的最大子序列。

1.1.4 序列模式(Sequential Pattern)

S 的支持度($\text{support}(S)$)是指在给定的序列数据库 D 中包含 S 的序列个数, $\text{support}(S) = |\{s' \in D) \cap (s' \subseteq S)|$,给定最小支持度 min_sup ,如果每个子序列 S 在序列数据库中的支持度均大于阈值 min_sup ,则称 S 是频繁序列模式,简称序列模式^[2]。

1.2 算法描述

PrefixSpan 算法的主要思想是利用序列前缀划分搜索空间和投影序列数据库,搜索相关符合要求的序列。下面以序列数据库

$D = \{a(abc)(ac)d(cf) >, <(ad)c(bc)(ae) >, <(ef)(ab)(df)cb >, <eg(af)cbc >\}$

及 min_sup 为 2 例描述挖掘过程:

(1) 查找 $\text{length} = 1$ 的序列模式

通过扫描 D 一遍得到 $\text{length} = 1$ 的序列模式 $\langle a > : 4; \dots; \langle f > : 3;$

(2) 分割搜寻空间

将(1)生成的序列模式分割成 6 个前缀,包括 $\text{Prefix} \langle a > ; \dots; \text{prefix} \langle f > ;$

(3) 查找序列模式的子集

按以下步骤建立投影数据库,并以递归方式挖掘序列模式:

- ① 先查找具有 $\text{prefix} < a >$ 的序列模式;
- ② 经由①之后,在序列数据库 S 中的序列有 4 个后缀序列被投影产生;
- ③ 扫描 $\text{projective} < a >$ 得到 $\text{length} = 2$ 的序列模式,递归将其切割并产生各自的投影数据库;
- ④ 当前缀的投影数据库只剩下少于 min_sup 的子序列个数时,退出对该前缀的挖掘^[3]。

2 PrefixSpan 算法改进

PrefixSpan 算法的时间与空间耗费主要是在挖掘过程中构建与扫描投影数据库上。故本文要从这个方面着手研究改进策略,提出 PrefixSpan * 算法。具体的思想是:舍弃非频繁项与修改 Prefix 策略。

2.1 修改 Prefix 策略

方法:挖掘 $L - 1$ 频繁项集子程序均改为当前 Prefix 参数,如当前 Prefix 为 null,则本次直接对原始序列数据库进行挖掘;如当前 Prefix 为非空,则对投影数据库进行挖掘,扫描原始序列,对库中的每个序列先匹配 Prefix,然后记录 Prefix 之后的元素信息,查找以当前 Prefix 长度增加 1 的序列为前缀的频繁序列。

改进前:时间耗费: $\sum_{i=1}^n (t_i + t'_i) + T$, 空间耗费: 投影数据库所占的内存空间;改进后:时间耗费: $\sum_{i=1}^n (t_i + t'_i)$, 新增空间耗费:0; (n 为原始数据库的序列个数; t_i 为第 i 个序列匹配 Prefix 的时间耗费; t'_i 对第 i 投影数据库记录元素频度的时间耗费, T 为创建数据库的时间耗费)

2.2 舍弃非频繁项

以所有的 $L - 1$ 项为前缀生成投影数据库,根据相关概念,对非频繁项挖掘不可能产生频繁项,故把非频繁项删除,仅保存频繁项,生成频繁项集,并以频繁项为前缀挖掘 L 频繁项^[4,5]。

改进前:

$$T_{\text{projectedDB}} = \sum_{i=1}^n \sum_{j=1}^l (m_j + n_j) \quad (1)$$

改进后:

$$T'_{\text{projectedDB}} = \sum_{i=1}^n \sum_{j=1}^l (m_j + \lambda_{f-\text{min_sup}} n_j) \quad (2)$$

(式中 $\lambda_{f-\text{min_sup}} = \begin{cases} 0 \cdots (f < \text{min_sup}) \\ 1 \cdots (f \geq \text{min_sup}) \end{cases}$, f 为频繁度, min_sup 为最小支持度; m_j 为计算频度的时间耗费, n_j 为建立投影库的时间耗费)

$$\nabla_t = \sum_{i=1}^n \sum_{j=1}^l (\lambda_{f-\text{min_sup}} n_j) \quad (3)$$

分析(3)式,当 min_sup 增大时, $\lambda_{f-\text{min_sup}} = 0$ 的概率增高, $T'_{\text{projectedDB}}$ 减小,而 ∇_t 增大,即:当最小支持度增大时算效率的提高更加显著。

3 改进后的算法伪码描述

初始条件:序列数据库 D ; 最小支持数 min_sup

输出结果: D 中所有符合要求的 K -序列模式

前缀 Prefix(min_sup) 算法:

(1) $K = 1$;

(2) 从数据库中查找长度为 K 的频繁序列 f_k ;

(3) 划分搜索空间,分别挖掘含有以这些频繁序列为 prefix 的长度为 $K + 1$ 的频繁序列,如果挖掘结果为空,则停止;

(4) $K = K + 1$, 转第(2)步;

(5) 记录所有挖掘到的频繁序列 f_k 。

投影 Project($f_k, \text{min_sup}$) 算法:

(1) 读取 f_k , 查找到 f_k 的所有后缀 $\text{postfix}(f_k)$;

(2) 查找 $\text{postfix}(f_k)$;

if($f_k \geq \text{min_sup}$) 保存 $\text{postfix}(f_k)$;

else 删除 $\text{postfix}(f_k)$;

(3) 将(2)得到的 $\text{postfix}(f_k)$ 入库,生成投影库 Projected(postfix) database

子程序 PrefixSpan * (s, l, Pls) 算法:

参数: s : 一个序列模式; l : 序列模式 s 的长度;

Pls : 如果 s 为空,则为 s , 否则为 s 的投影数据库。

(1) 扫描 Pls , 查找满足下述要求且长度为 1 的序列模式 s' :

s' 可合并到 s 的最后一个元素中并为序列模式;

$\langle s' \rangle$ 可作为 s 的最后一个元素并为序列模式;

(2) 对每个生成的序列模式 s' , 将 s' 添加到 s 形成的序列模式 s_k , 并输出 s_k ;

(3) 对每个 s_k , 调用 $\text{Project}(f_k, \text{min_sup})$ 算法构造 s_k 的投影数据库 $p|s_k$, 并调用子程序 $\text{PrefixSpan}^*(s_k, l+1, p|s_k)$ 继续挖掘^[3]。

4 PrefixSpan* 算法应用于 Web 挖掘

4.1 实验数据

本实验数据源可在 <http://www.almaden.ibm.com> 下载, 根据网络日志属性的重要程度将数据处理成如下格式: { protocol、src_host、dest_host、timestamp、duration、src_bytes、dest_bytes、state、flags }, 抽取其中 10h 左右的数据, 存入数据库 D_1 中。

4.2 实验结果

验测试环境: CPU: 3.0 G; 内存: 2.0 G; 操作系统 win xp。结果如图 1 与图 2 所示, 其中图 1 为 PrefixSpan* 算法与 PrefixSpan 挖掘 D_1 的运行时间比较, 图 2 为 PrefixSpan* 算法与原算法相比运行效率的提高程度, 显然算法效率提高 35% 以上, 并随着 min_sup 的增大其效能优势更趋明显。

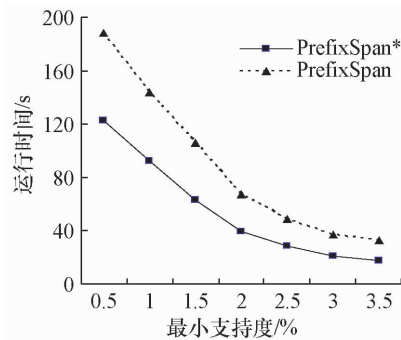


图 1 挖掘 D_1 运行时间比较

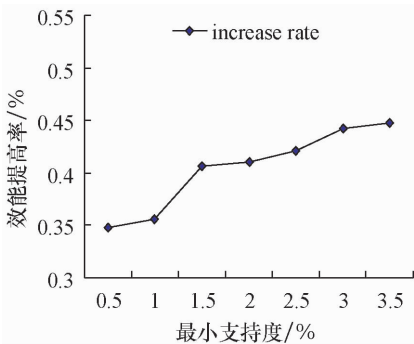


图 2 PrefixSpan* 运行效率提高程度

5 结束语

本文探讨了 PrefixSpan 算法, 并采用舍弃非频繁项与修改 Prefix 策略的方法对算法进行了改进, 使算法效率比原来提高 35% 以上, 能更好的满足海量 Web 数据挖掘的要求。但 Web 挖掘仍需要更深层次的研究和性能上的提升。

参 考 文 献

- 1 Kaushik A. 精通 Web analytics. 北京: 清华大学出版社, 2008
- 2 Han Jiawei, Kamber M. Data mining concepts and Techniques. 北京: 机械工业出版社, 2007
- 3 Pei Jian, Han Jiawei, Mortazavi-Asl B, et al. PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. Proc of 2001 Int'l Conf. on Data Engineering. : IEEE Press, 2001: 215—224
- 4 Ding Bolin, David L O, Han Jiawei, et al. Efficient mining of closed repetitive gapped subsequences from a sequence database. Department of Computer Science, University of Illinois at Urbana-Champaign, 2009
- 5 Pei J, Han J, Wang W. Mining sequential patterns with constraints in Large databases. Proc of the 11th Int Conf on Information and Knowledge Management, McLean, Virginia, 2002. New York, ACM Press : 18—25

Application of Improved PrefixSpan Algorithm in Web Mining

XIE Qing-sen, YANG Tian-qi

(Jinan University, Guangzhou 510632, P. R. China)

[Abstract] Taking account of insufficiency of PrefixSpan algorithm, which is widely applied to data mining, The algorithm by reducing frequency of exchanging is trys to optimize between the memory and the external memory in the Prefix part, and reducing the size of the projection database by discarding the non – frequent items which created in the process of sequence patterns mining. The result of test demonstrates that the operating efficiency is enhanced more than 35%. The conclusion comes to shows that the improved algorithm is applicable to the Web Mining.

[Key words] Web mining PrefixSpan algorithm sequence pattern

(上接第 7175 页)

生成频繁集过程中避免了生成非频繁项目集的超集,另外,在计算支持数时利用位运算避免了扫描原始数据库。实验表明相对于原算法,在时间开销和空间开销上都要优越得多。

参 考 文 献

1 Mobasher B,Cooley R. Srivastava J. Automatic person alization based on Web usage mining. Communications of the ACM,2000;43(8):

142—151

2 Huszerl G, Majzik I. Modeling and analysis of redundancy management in distributed object oriented systems by using UML statecharts. In:Proceedings of 27, Euromicro Conference. 2001: 200

3 Srikantr A. Fast algorithms for mining association rulers//Proceeding of the 20th Int 7 Conference on Very Large Databases, Santiago, Chile, Santiago de Chile: Morgan Kanfmarm,1994:487—499

4 Lou Lanfang, Pan Qingxian. An improved algorithm based on sets operation for mining frequent items. Journal of Shandong University (Natural Science),2008;43(11),54—57

An Optimized Algorithm for Mining Frequent Itemsets Based on Set and Bit Operation

YANG Ni-ni

(Liaoning Shihua University, Fushun 113001, P. R. China)

[Abstract] Generating frequent itemsets is a critical step in association rule mining. Through the analysis of Apriori algorithm, a new algorithm for mining frequent itemsets based on set and bit operation is proposed. In this algorithm, digital view is used to express the transaction who used each item, and bit operating is used in digital view to calculate the number of support of each itemset. The problem of repeatedly scanning the database in Apriori algorithm is solved and operating efficiency is improved in the new algorithm.

[Key words] data mining association rule frequent itemsets