文章编号:1001 - 9081(2008)11 - 2964 - 03

一种分布式序列模式挖掘算法

常 鹏[1],陈 耿[2],朱玉全[1]
(1. 江苏大学 计算机科学与通信工程学院,江苏 镇江 212013; 2. 南京审计学院 省级审计信息工程重点实验室,南京 210029)
(yqzhu@ujs.edu.cn)

摘 要:针对分布式环境下的序列模式挖掘问题,提出了一种分布式序列模式挖掘(DSPM)算法。DSPM 以 PrefixSpan 算法为基础,使用抽样检测技术平衡了任务负载,将挖掘任务分解后分配到多台计算机上以多进程、多线程并行执行。另外采用了伪投影技术来降低生成投影数据库的开销。实验结果表明,DSPM 算法能够快速有效地挖掘分布式环境下的全局序列模式。

关键词:数据挖掘;序列模式;分布式;模式增长
中图分类号: TP311.13 **文献标志码**:A

Mining algorithm of distributed sequential pattern

CHANG Peng[1], CHEN Geng[2], ZHU Yu-quan[1]
(1. *School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang Jiangsu* 212013, *China*;
2. *Jiangsu Key Laboratory of Audit Information Engineering, Nanjing Audit University, Nanjing Jiangsu* 210029, *China*)

Abstract: In order to mine sequential patterns in distributed environment, Distributed Sequential Pattern Mining (DSPM) algorithm based on prefixSpan was proposed. Sample dataset was detected to balance the workload. Mining tasks were decomposed and distributed to many other computers. Pesudo-projected techniques were used to reduce the cost and the parallel was advanced by multithreading. The experimental results show that DSPM algorithm can mine global sequential patterns effectively and quickly.

Key words: data mining; sequential pattern; distributed; pattern growth

0 引言

序列模式是由文献[1]提出的一个重要数据库中的知识发现(Knowledge Discovery in Database, KDD)研究课题,在 Web 访问模式分析、网络入侵检测、顾客购物行为分析、疾病治疗、DNA 分析等领域中具有十分广泛的应用前景,其目的是在相应的序列数据库中挖掘出频繁出现的序列模式。

随着网络的快速发展,海量的数据使得传统的单计算机系统在功能和性能上已不能满足对数据处理能力的需要。由网络联接多台计算机所构成的分布式系统已成为当今的主流系统,系统中的数据往往呈分布式状态。对于此状态下的序列数据,各单机系统上执行挖掘算法所得到的序列模式只是针对局部数据有效,无法获得分布式系统中所有数据全局有效的序列模式。对于此问题,一种直接的方法是将分布式环境中所有数据集中在一台计算机上执行挖掘算法,但在具有海量数据的分布式环境中,这将造成巨大的通信开销。而且在某些应用中,出于对客户隐私保护、数据安全性等原因,原始数据的传输亦是不可行。

文献[2]提出了基于前缀投影的序列模式挖掘算法 PrefixSpan,文献[3]提出了共享内存计算机上的序列模式发现问题的处理方法。文献[4]提出了基于树投影技术的两种不同的并行算法来解决分布内存并行计算机的序列模式发现。文献[5-6]中分别提出了快速分布式序列模式挖掘(Fast Distributed Mining of Sequential Pattern, FDMSP)算法和分布式全局序列模式挖掘(Distributed Mining of Global Sequential Pattern, DMGSP)算法来挖掘分布式环境下的全局序列模式。但 FDMSP 算法在各分站点的序列被分配到选举站点时需要传输相当大的数据量,而 DMGSP 算法以产生序列模式(Generalized Sequential Pattern, GSP)算法为基础,时间复杂度较高。文献[7]对分布式环境下的序列模式挖掘算法的并行执行进行了研究,指出了算法研究的重点,但未给出具体的实现方法。

本文以 PrefixSpan 算法为基础,针对分布式环境下的序列模式挖掘问题,提出了一种分布式序列模式挖掘算法 DSPM。该算法使用抽样检测技术平衡了任务负载,将挖掘任务分解后分配到多台计算机上以多进程、多线程并行执行。另外,本算法还采用了伪投影技术来降低生成投影数据库的开销。实验结果表明,DSPM 算法能够快速有效地挖掘出分布式环境下的全局序列模式。

1 相关概念

分布式序列模式挖掘问题可形式化描述如下:设分布式环境中存在 n 个通过网络互联的站点 $S_1, S_2, \cdots, S_n$,其中的每个站点都是一台独立的计算机。站点 $S_i(i = 1,2,\cdots,n)$ 上的序列数据集记为 db_i,所有站点上的序列数据集记为 DB,$DB = db_1 \cup db_2 \cup \cdots \cup db_n$,且 $db_i \cap db_j = \varnothing, i \neq j, i \in \{1,2,3,\cdots,n\}, j \in \{1,2,3,\cdots,n\}$。站点上的数据序列记为(*id*, *sequence*),其中 *id* 代表序列标识,*sequence* 即一条原始序列数

收稿日期:2008-05-16;**修回日期**:2008-07-18。 **基金项目**:国家自然科学基金资助项目(60572112);江苏省高技术重大项目资助(BG2007028);江苏省六大人才高峰项目(07-E-025);江苏省教育厅项目(06KJB120051)。

作者简介:常鹏(1984-),男,河北邢台人,硕士研究生,主要研究方向:数据挖掘; 陈耿(1965-),男,江苏无锡人,博士研究生,主要研究方向:数据挖掘、数据库系统及其应用、决策支持系统; 朱玉全(1966-),男,江苏常州人,副教授,博士,主要研究方向:复杂信息系统集成、知识发现、入侵检测。

据。设用户给定的最小支持度为 $minSup$。

定义 1　对分布式系统中的某一个序列 s,数据站点 S_i $(1 \leqslant i \leqslant n)$ 上包含 s 的序列数据总数称为 s 在数据站点 S_i 上的局部支持数,记为 $Count_i(s)$。分布式系统中包含 s 的序列总数称为 s 的全局序列数,记为 $Count(s)$。显然,$Count(s) = \sum_{i=1}^{n} Count_i(s)$。

定义 2　对分布式系统中的某一个序列 s,若 $Count_i(s) \geqslant minCount_i$ 成立,则称 s 为站点 S_i 上的局部序列模式。其中 $minCount_i = | db_i | \times minSup$。

定义 3　对分布式系统中的某一个序列 s,若 $Count(s) \geqslant minCount$ 成立,则称 s 为全局序列模式。其中 $minCount$ 为全局最小支持数,$minCount = | DB | \times minSup$。

局部序列模式具有与全局序列模式相同的最小支持度 $minSup$。

定义 4　长度为 k 的序列模式称为 k- 序列模式。

设 $F_G(k)$ 表示全局 k- 序列模式的集合,对任意 $s \in F_G(k)$,有 $Count(s) \geqslant minSup$ 成立。设 $F_{Li}(k)$ 表示站点 S_i 上局部 k- 序列模式的集合,对任意 $s \in F_{Li}(k)$,有 $Count_i(s) \geqslant minCount_i$ 成立。

性质 1　序列模式的每个非空子序列都是序列模式。一个序列若不是序列模式,则其所有超序列都不是序列模式。

性质 2　对于任意一条全局序列模式 s,则至少存在一个站点 S_i,s 和它的所有子序列都是 S_i 上的局部序列模式。

证明　假设不存在这样的站点 S_i,则 $Count_i(s) < minCount_i$,$(i = 1,2,n)$。因此,DB 中包含 s 的序列总数为 $Count(s) = \sum_{i=1}^{n} Count_i(s) < \sum_{i=1}^{n} \min Count_i = minSup \times (\sum_{i=1}^{n} | db_i |) = minSup \times | DB |$,即 $Count(s) < minSup \times | DB |$,因而,$s$ 不是序列模式,故假设不成立。性质 2 得证。

2　分布式序列模式挖掘算法

2.1　主机的确定

在所有站点 $S_1, S_2, \cdots, S_n$ 中,需要一台计算机执行全局序列模式的汇总和对所有计算机的挖掘进度控制等任务,这台计算机就是主机。主机统计全局支持数,计算全局序列模式式。所有计算机执行模式增长算法,生成局部序列模,并对全局序列模式进一步执行模式增长算法。

本文中主机的确定思想为:使用抽样检测评估各站点计算机的性能与负载状况,指定其中负载最小的计算机作为主机,其余的作为分机。具体如下:

1) 设置采样数据集的大小。各站点计算机随机抽取本地序列数据集的 $x\%$ 序列,x 在 1 与 5 之间,具体取值视数据集的大小而定。

2)对抽取出的序列执行简化的 PrefixSpan 算法,此算法仅用于效率测试,限制了挖掘的深度,也不保存挖掘结果。

3)综合考虑各计算机测试挖掘所耗费的时间及机器配置,选取一台计算机作为主机。

抽样检测能够比较准确地反应出各站点计算机的工作负载,选取工作负载最轻的计算机作为主机,承担计算全局序列模式的任务,减小数据倾斜造成的整体效率下降。

2.2　前缀序列树

所有的序列模式可以构成一棵前缀序列树,树的根节点为 NULL,第 1 层是 1- 序列模式,第 2 层是 2- 序列模式,…,第 k 层是 k- 序列模式,依此类推。对于树中除根节点外的任一节点,其子节点都以它为前缀,通过对父节点的扩展而获得的。

根据性质 1,若一个节点的所有子节点都不是序列模式,则其所有子孙节点都不是序列模式,没有必要对此节点继续扩展,这种节点叫作非活动节点,不再产生子节点。反之称为活动节点。

分布式序列模式挖掘(Distributed Sequential Pattern Mining, DSPM)算法在所有计算机站点构造一个前缀序列树,此树存储全局序列模式和活动节点的伪投影数据库,同时也临时保存了局部候选序列模式和伪投影数据库,如确认为非全局序列模式则从树中删除。

2.3　模式增长算法

在 PrefixSpan 算法中,当挖掘出 k- 序列模式后,深度优先挖掘以 k- 序列模式为前缀的 $(k+1)$- 序列模式,在这个过程中,各 k- 序列模式的模式增长计算相互之间是相互独立的。因此,这个过程可以多个线程并行执行,具体算法描述如算法 1 所示。

算法 1　candiSeq patternGrowth(preSeq(k), pjn)

输入:preSeq(k),某全局 k- 序列模式; pjn,此前缀序列模式的伪投影数据库。

返回值:挖掘出的局部序列模式 candiSeq。

```
1)    if (preSeq == null)              // 无前缀, 挖掘 1-序列模式
2)      for(each oriSeq in D)
3)        cp ← (countSup(oriSeq), countPesudo(oriSeq))
                                       // 计算各项支持数和伪投影
4)        PSTree ←cp
                  // 将候选序列及伪投影数据库插入前缀序列树
5)        locSeqPt = countLocSeq(cp) //计算得到局部序列模式
6)        candiSeq ← locSeqPt
7)    else          // 以 preSeq(k) 为前缀挖掘(k + 1)- 序列模式
8)      for(each oriSeq in pjn)
9)        cp ← (countSup(oriSeq), countPesudo(oriSeq))
                                       // 计算各项支持数和伪投影
10)       cp = join(preSeq( k ), cp) // 连接得到候选序列模式
11)       PSTree ← cp
12)       locSeqPt = countLocSeq(cp)
13)       candiSeq ← locSeqPt
14)   return cadiSeq
```

算法 1 在各站点上执行局部序列模式的挖掘任务。第 1) ~6)行是挖掘局部 1-序列模式的过程,与挖掘以 k- 序列模式为前缀的 $(k+1)$- 序列模式的不同在于,后者需要先根据伪投影数据库计算出挖掘的原始序列,且挖掘出的结果要与前缀序列进行连接。算法返回挖掘出的局部序列模式。

2.4　全局序列模式生成

分布式环境下挖掘全局 k- 序列模式,首先需要获取全局候选 k- 序列模式在所有站点上的支持数。根据局部序列模式和全局序列模式的关系,全局候选 k- 序列模式即为所有站点的局部 k- 序列模式的并集,记为 $C(k)$, $C(k) = F_{L1}(k) \cup F_{L2}(k) \cup \cdots \cup F_{Ln}(k)$。

以全局 1- 序列模式的生成过程为例:所有计算机执行模式增长算法,挖掘出局部 1- 序列模式。所有站点将局部 1- 序列模式 $F_{Li}(1)$ $(i = 1,2,\cdots,n)$ 发送到主机,主机合并所有计算机的局部 1- 序列模式,获得全局候选 1- 序列模式 $C(1)$。主机将 $C(1)$ 发送到包括自己在内的所有计算机,索取 $C(1)$ 中各序列的支持数。各站点计算机将 $C(1)$ 中的序列在本地的支持数发送给主机。主机即可计算候选 1- 序列模式的全局支持数,满足 $minCount$ 的即为全局 1- 序列模式。

在有 n 台计算机的分布式系统中计算出一条全局序列模式的通信次数为 $O(3(n-1))$,但每次通信一般发送多条序列。

2.5 前缀序列树剪枝

每次执行模式增长算法,会将所有的候选序列模式放入前缀序列树中。根据性质 1,非序列模式的超序列不可能是序列模式,当候选序列中的全局序列模式被计算出来后,需要对前缀序列树进行剪枝,将非频繁序列从树中删除。

剪枝过程可表述如下:当主机向各台计算机发送全局 k-序列模式时,同时会发送这组序列模式的前缀序列 preSeq。各计算机在前缀序列树中查找 preSeq 代表的节点,将其所有子节点与接收到的全局 k- 序列模式进行比较,把非全局 k- 序列模式所代表的节点从前缀序列树中删除。

2.6 算法 DSPM 描述

算法 DSPM 在所有站点 $S_i(i = 1,2,\cdots,n)$ 上执行,由 2 个主过程组成,它们分别是 host 过程和 child 过程。host 过程仅在主机执行,负责生成全局序列模式。child 过程负责生成局部序列模式,并相应数据传输。child 过程在所有计算机上执行,主机执行此过程略有不同,仅数据的传输不通过网络进行。接收器在系统初始化时被初始化,负责接收数据和指令,建立新线程,执行相应算法。

2.6.1 child 过程

```
1)   receive(gloSeq(k))                          // 全局 k- 序列模式
2)     prun PSTree
                    // 剪枝,从前缀序列树中删除非频繁序列模式
3)     for(each k - s in gloSeq(k))
4)     pjn ← getPjn(PSTree, k - s)
5)     locSeq(k + 1) ← patternGrowth(k - s, pjn)
6)     send(locSeq(k + 1))                       // 发送局部序列模式
7)     wait;                                          // 此线程等待
8)     receive(gloSeq(k))                        // 全局候选序列模式
9)     send(count_i(k - s))                           // 发送支持数
10)  return
```

在算法开始时,挖掘 1-局部序列模式,仅执行第 5)~9)行的逻辑。第 1)行接收到主机发送的一组有相同前缀的全局序列模式,根据前缀执行第 2)行,从前缀序列树中执行剪枝。第 3)~9)行步对接收到的每个序列模式执行模式增长算法,与 host 过程协作计算出全局序列模式。

2.6.2 host 过程

```
1)   C(k) ← receive(locSeq(k))
                                  // 接收到分机发送的局部序列
2)   if(! all locSeq(S_i) arrive)
                                  // 判断是否所有站点都挖掘完毕
3)     return;
4)   send(C(k))          // 向所有站点发送全局候选序列模式
5)   receive(count_i(k - s))                    // 接收所有支持数
6)   if(! all count_i(k - s) arrive)
7)     return;
8)   F_G Tree ← gloSeq(k)                  // 计算出全局序列模式
9)   if(F_G Tree stop growth)              // 判断挖掘过程是否结束
10)    send(over);
11)    outPut F_G Tree;
12)  send(gloSeq(k))      // 把全局序列模式发送到所有站点
13)  return
```

F_GTree 是保存全局序列模式的数据结构,组织形式与前缀序列树相同,但仅存储全局序列模式及其支持度,并标记是否为活动节点。若所有节点都为非活动节点,则说明挖掘过程执行完毕。即第 7)~9)行。

3 实验结果

文献[5-6]的实验中使用 DGSP 算法与提出的算法进行比较。DGSP 算法即首先将各站点的数据集中到一个站点,再执行 GSP 算法进行挖掘,总的挖掘时间包括数据传输和执行时间。由于 PrefixSpan 算法优于 GSP 算法,且本文所提 DSPM 算法以 PrefixSpan 算法为基础,本章将 DSPM 算法与 PrefixSpan 算法进行了比较。

为了验证算法 DSPM 的执行情况,在以 10 MPS 传输速率连接的局域网内的两台计算机上进行了测试。计算机配置都为 T2050 双核1.6 GHz、1 GB内存。数据集为使用 DataFactory 生成并经过处理的序列数据集,处理后的数据集总大小为 12.7 MB,一台机器上数据集大小为 6.4 MB,另一台机器数据集大小为 6.3 MB。序列总数目为 160 000 条序列。序列的长度最小为 1,最大为 20,平均长度为 10。选取多个不同的最小支持度分别测试这两个算法的执行效率。

图 1 为 PrefixSpan 算法和 DSPM 算法的执行效率比较图。实验中发现,负载平衡对算法的运行效率有着重要影响,而在大数据量或低网络传输速度的情况下网络通信耗费的时间有着更大的影响。

实验结果表明,数据倾斜和网络传输是影响分布式环境下序列模式挖掘的两个最重要的因素。在绝大多数情况下,DSPM 算法优于 PrefixSpan 算法。

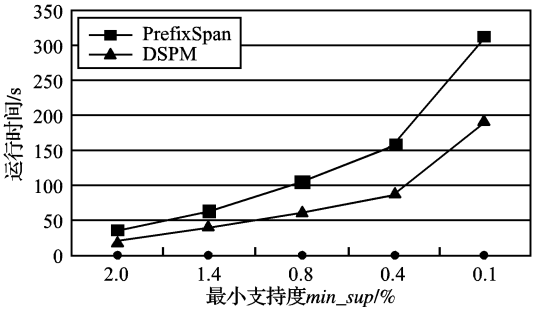

图 1 PrefixSpan 和 DSPM 算法执行效率比较

4 结语

本文对在分布式环境中挖掘序列模式问题进行了研究,提出了 DSPM 分布式序列模式挖掘算法,以 PrefixSpan 算法为核心,对挖掘任务进行分解,并采用了抽样检测、伪投影等技术提高了算法效率。实验结果验证了 DSPM 算法的有效性和性能,也指出了下一步重点研究的问题所在:如何减轻数据倾斜的影响和减小通信开销。今后的研究将针对这两方面的优化进行,并与对网络点击流的分布式挖掘[8]等实际应用相结合。

参考文献:

[1] AGRAWAL R, SRIKANT R. Mining sequential patterns[C]// Proceedings of the 11th International Conference on Data Engineering. Taipei: [s.n.], 1995: 3-14.

[2] HAN J, PEI J, MORTAZAVI-ASL B, *et al.* PrefixSpan-Mining sequential patterns efficiently by prefix-projected pattern growth[C]// Proceedings of the 17th International Conference on Data Engineering. Heidelberg, DE: [s.n.], 2001: 215-224.

[3] ZAKI M J. Parallel sequence mining on shared-memory machines [J]. Journal of Parallel and Distributed Computing, 2001, 6(1): 401-426.

[4] GURALNIK V, GARG N, KARYPIS G. Parallel tree projection algorithm for sequence mining[C]// Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing. London, UK: Springer-Verlag, 2001: 310-320.

(下转第 2974 页)

次实验结果平均,并使用平均绝对偏差[9](Mean Absolute Error, MAE)来评价推荐的精确度。*MAE* 值越小说明推荐精度越高,*MAE* 计算式如下:

$$MAE = \frac{1}{|T|}\sum_{u,i\in T}|p_{ui} - r_{ui}| \quad (5)$$

其中 T 是测试集,$|T|$ 是测试集的基数,p_{ui} 是用户 u 对项目 i 的预测评分,r_{ui} 是用户 u 对项目 i 的实际评分。

实验中,使用皮尔逊相关系数[10] 作为发现最近邻的相似度计算方法。固定最近邻个数为 50,背景信息相似用户数 M 从 10 变化到 50,步长为 10。实验结果如图 3 所示。

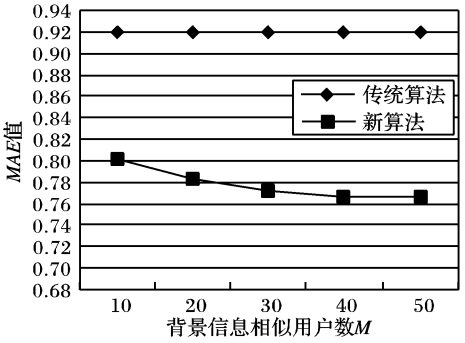

图 3 两种算法的推荐精度比较

从图 3 中可以看到,在相同的最近邻取值下,新算法与传统协同过滤算法相比明显具有更低的 *MAE* 值,并且随着 M 的增大,新算法的 *MAE* 值逐渐减小,当 M 取值为 50 时,新算法大约可以提高 15% 的推荐精度。同时,通过计算得到,当 M 取值为 50 时,经过填充后的用户—项评分矩阵中的零值已不到整个矩阵的 3%,可见利用新算法可以有效地解决矩阵稀疏的问题,从而带来推荐质量的明显提高。

为了验证 1.3 节对算法复杂度的描述,实验基于五组实验数据分别使用两种算法对 10 个随机目标用户进行了在线推荐,记录两种算法所花费的时间,并将五次试验结果平均,实验中只对原始评分数据进行了一次离线预处理,耗时 1.474 s。实验结果如图 4 所示,对比可见,新算法的在线推荐过程在很大地提高了推荐精度的同时仅仅比传统协同过滤算法多花了大约 8 ms 的时间。虽然新算法的预处理相对一次在线推荐来说在时间消耗上大了很多,但是它只在系统空闲时离线计算一次,同时,预处理使后续的每一次推荐都大大提高了精度。因此,无论从新老算法的时间耗用对比,还是从预处理复杂度和推荐精度提升的权衡来看,新算法都是可行的、有效的。

3 结语

本文针对传统协同过滤算法面对的稀疏矩阵问题提出了结合用户背景信息的推荐算法。该算法充分地利用了已有的用户数据和领域知识,对用户背景信息的相似度建模,预先填充用户—项评分矩阵,实验表明该方法能够有效地提高推荐精度,并且不会带来性能上的瓶颈。

对于不同的推荐系统,在已有信息的基础上如何对用户的相似度进行建模对于新算法的实现以及整个系统的推荐质量有很大的影响。例如,选择哪些有用的属性,如何构建分类属性的语义层次树,这将涉及特征子集选择以及分类、聚类等数据挖掘技术。同时,可以结合信息获取技术在已有数据的基础上进一步获得更多的有利于推荐的信息,而这些问题也是我们下一步研究工作的主要内容。

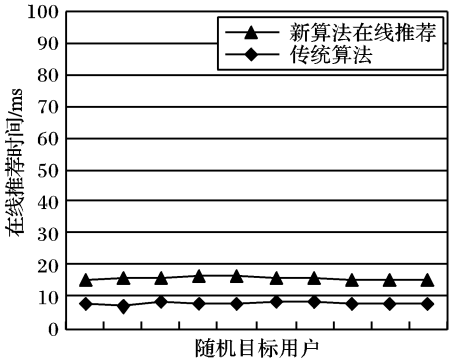

图 4 两种算法在线推荐的时间对比

参考文献:

[1] 曾春, 邢春晓, 周立柱. 个性化服务技术综述[J]. 软件学报, 2002, 13(10): 1952 – 1961.

[2] MOONEY R J, ROY L. Content-based book recommending using learning for text categorization[C]// Proceedings of the 5th ACM Conference on Digital Libraries. San Antonio: ACM Press, 2000: 195 – 204.

[3] MELVILLE P, MOONEY R J, NAGARAJAN R. Content-boosted collaborative filtering for improved recommendations[C]// Proceedings of the 18th National Conference on Artificial Intelligence. Menlo Park, CA: American Association for Artificial Intelligence, 2002: 187 – 192.

[4] GOLDBERG D. Using collaborative filtering to weave an information tapestry[J]. Communications of the ACM, 1992, 35(12): 61 – 70.

[5] 秦国, 杜小勇. 基于用户层次信息的协同推荐算法[J]. 计算机科学, 2004, 31(10): 138 – 140.

[6] 蔡登, 卢增祥, 李衍达. 信息协同过滤[J]. 计算机科学, 2002, 29(6): 1 – 4.

[7] HAN J, KANMBER M. Data mining: Concepts and techniques [M]. 2nd ed. San Francisco: Morgan Kaufmann, 2006.

[8] RESNICK P, IACOVOU N, SUCHAK M, *et al*. Grouplens: An open architecture for collaborative filtering of Netnews[C]// Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work. Chapel Hill: ACM Press, 1994: 175 – 186.

[9] SARWAR B, KARYPIS G, KONSTAN J, *et al*. Item-based collaborative filtering recommendation algorithms[C]// Proceedings of the 10th International World Wide Web Conference. New York: ACM Press, 2001: 285 – 295.

[10] BREESE J S, HECKERMAN D, KADIE C. Empirical analysis of predictive algorithms for collaborative filtering[C]// Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. San Francisco: Morgan Kaufmann Publishers, 1998: 43 – 52.

(上接第 2966 页)

[5] 邹翔, 张巍, 刘洋. 分布式序列模式发现算法的研究[J]. 软件学报, 2005, 16(07): 1262 – 1269.

[6] 龚振志, 胡孔法, 达庆利, 等. DMGSP: 一种快速分布式全局序列模式挖掘算法[J]. 东南大学学报, 2007, 16(04): 574 – 579.

[7] ZHOU LI-JUAN, QIN BAI, WANG YU, *et al*. Research on parallel algorithm for sequential pattern mining[C]// Proceedings of the SPIE Conference on Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security. [S. L.]: SPIE, 2008: 69 – 73.

[8] CHEN JIN-LIN, COOK T. Mining contiguous sequential patterns from Web logs[C]// Proceedings of the 16th International Conference on World Wide Web. New York: ACM Press, 2007: 1177 – 1178.