DOI:10.13718/j.cnki.xsxb.2020.11.012

大数据环境下的高效分布式增量序列挖掘①

南 楠[1], 严英占[2]

1. 岭南师范学院 基础教育学院,广东 湛江 524048; 2. 中国电子科技集团第 54 研究所,石家庄 050081

摘要:本文提出一种基于 MapReduce 架构的高效分布式增量序列模式挖掘算法(Incremental Sequential Pattern Mining, IncSPM),用于解决大数据环境中每当数据增加时就更新序列模式的问题. 该算法利用后向挖掘算法来有效利用先前挖掘生成的序列模式,同时设计同现反转映射(Co-occurrence Reverse Map, CRMAP)数据结构来处理候选序列的组合爆炸问题,最后设计了新的候选生成和早期修剪机制以加快挖掘过程. 用两种真实数据集对本文提出的算法进行了评估,实验表明与其他方法相比,本文算法在执行时间、内存消耗和扩展性方面均有实质性的提高.

关 键 词:大数据挖掘;增量序列模式;后向挖掘;同现反转映射数据结构

中图分类号:TP393 **文献标志码**:A **文章编号**:1000-5471(2020)11-0080-06

序列模式挖掘(Sequence Pattern Mining, SPM)是一个广泛应用的热门数据挖掘任务,用来挖掘频繁出现的有序事件或子序列[1-2],在挖掘 Web 使用模式、分析客户购买行为、挖掘 DNA 序列等方面有很大的应用价值. 由于大数据具有大容量、多样性、高速度、低价值密度和准确性等特征[3-4],每当数据发生变化时,都必须重新运行序列模式挖掘算法,这是因为从旧数据获得的频繁项,在更新的数据中可能变为不频繁项,并且在更新的数据中可能会出现新的频繁项[5]. 如果序列数据库的规模比较大,则有可能会产生大量的候选序列模式,故传统的 SPM 不可扩展[6]. 传统的序列模式增量维护算法不能很好地适应大序列数据库,必须设计分布式挖掘算法来处理大数据[7].

MapReduce 是一种遵循分而治之的策略来处理大数据问题的分布式编程框架,它对程序员隐藏了数据划分和分配、任务调度、机器间通信和容错的内部细节,允许没有任何经验的程序员在分布式系统中有效地处理系统的资源[8]. MapReduce 模型不仅简化了分布式编程的任务,而且实现了对大数据集更好的处理[9]. 后向挖掘算法用于高效序列模式的增量挖掘,该算法 4 个显著的优点:①提供一种简单的方法来检测稳定序列;②引入了唯一的稳定序列性质,即稳定序列的任何扩张也是稳定的;③稳定序列属性,通过跳过对稳定序列的支持计数来提高挖掘速度;④验证了序列新支持计数的过程很简单.

为了使增量挖掘算法可以很好地适应大序列数据库,已经有大量序列模式增量维护的算法研究. 文献[10]提出了一种利用 MapReduce 框架从大数据集中挖掘频繁项集的位向量积算法,该算法运用位向量数据结构来维护压缩的事务,从给定的事务列表中有效地搜索频繁项集,其优点是只需扫描一次数据集. 文献[11]提出了一种用于高效序列模式挖掘的纯数组结构(Array Structure for High-utility Sequential, AHUS)和并行策略,该算法运用高效用序列模式挖掘来发现序列数据库中效用值等于或大于给定最小效用阈值的所有序列模式的任务,运用 AHUS 并行挖掘来同时识别高效用序列模式(High-utility Sequential Pattem, HUSP),具有较好的可扩展性. 文献[12]提出了基于云均匀分布式词汇序列树算法的序列模式挖掘算法. 该算法使用两阶段 MapReduce 框架发现序列模式,无需启动多轮 MapReduce 即可显著提高整体

① 收稿日期:2020-02-16
基金项目:国家自然科学基金项目(61404119);河北省教育厅青年基金项目(QN2016182).
作者简介:南 楠(1983-),女,硕士,讲师,主要从事计算机应用研究.

性能，并在云中的机器之间提供完美的负载平衡，实现了极高的可扩展性，并提供了比现有的云端算法更好的负载均衡.

基于文献[12]的研究，通过设计新的 CRMAP 数据结构和后向挖掘算法，提出了基于 MapReduce 的高效分布式增量序列模式挖掘(IncSPM)算法，用来解决大数据环境中序列模式挖掘的增量维护问题. 文献[12]是一种广度优先搜索算法，它们组合较小的序列产生候选序列，生成可能不会出现在数据库中的候选序列. 本文算法通过后向挖掘算法来有效利用先前挖掘生成的序列模式，引入一种高效的 CRMAP 数据结构来生成出现在输入数据库中有希望的候选对象，以减小搜索空间. 基于 CRMAP 数据结构，设计候选生成规则和早期修剪机制以避免输入数据库中生成错误的候选序列，从而使本文算法具有良好的线性可扩展性. 实验结果显示，本文 IncSPM 在处理时间、内存消耗和可扩展性方面的有实质性的提高.

1　后向挖掘和同现反转映射

1.1　后向挖掘

以往的序列模式挖掘 MapReduce 算法没有处理增量维护问题，不能利用以前挖掘生成的序列模式，每当数据增加时，它们都会在更新的数据集上重新运行算法. 在大数据环境中，通过扫描整个数据集来重新挖掘所有序列模式是不可接受的，本文采用后向挖掘来进行序列模式的高效增量挖掘. 序列 s 的更新数据集(Updated Dataset，UD)中输入序列集合称为 s 的投影 pj_s，序列 s 最后一个项目集中的项集称为序列 s 的结尾 end_s，输入序列 s 的相应属于增量数据集(Increment Sequence Dataset，IncSD)中项目集称为序列 s 的增量 inc_s，更新数据集 UD 中所有输入序列的 inc_s 并集称为 UD 的增量并集，其增量 inc_s 包含末端输入序列的集合被称为序列 s 的末端投影 $end\,pj_s$，属于 pj_s 而不属于 $end\,pj_s$ 的输入序列的集合称为序列 s 的 SD 投影.

在后向挖掘中，长度为 k 的序列在向后方向上被扩展到长度 $k+1$，具有项 j 的序列 $\langle a_k a_{k-1}\cdots a_2 a_1\rangle$ 的项集扩展表示为 $\langle\{j \cup a_k\}a_{k-1}\cdots a_2 a_1\rangle$，同样具有项目 j 的序列 $\langle a_k a_{k-1}\cdots a_2 a_1\rangle$ 的序列扩展被表示为 $\langle ja_k a_{k-1}\cdots a_2 a_1\rangle$. 在生成新的序列 s 之后，如果该序列的末端投影为空，则表明任何输入序列的增量不包含 s，并且 s 是稳定的. 因此，UD 中稳定序列的支持数与原始数据集 SD 中的支持数相同.

1.2　同现反转映射数据结构

同现反转映射(Co-occurrence Reverse Map，CRMAP)数据结构用来解决大多数序列模式挖掘算法的性能瓶颈-评估数据库中不存在的模式所花费的时间，可以处理候选序列的组合爆炸问题.

CRMAP 是将每个项目 j 映射到序列中位于其前面的一组项目的数据结构. 本文定义了两个 CRMAP 结构，即 $\mathrm{CRMAP_i}$ 和 $\mathrm{CRMAP_s}$. $\mathrm{CRMAP_i}$ 以不少于最小支持数(Minimum support，min_sup)序列将每个项目 j 映射相对于项集扩展在其之前的一组项目，$\mathrm{CRMAP_s}$ 以不少于 min_sup 序列将每个项目 j 映射到相对于序列扩展在其之前的一组项目.

引理 1　当且仅当 $x \in \mathrm{CRMAP_i}(y)$ 时，相对于项集扩展 $\langle(x, y)\rangle$ 的长度为 2 的向后扩展序列被认为是频繁的.

证明：根据 $\mathrm{CRMAP_i}$ 的定义，如果项目 x 没有映射到 $\mathrm{CRMAP_i}(y)$，则相对于少于 min_sup 序列的项集扩展而言，x 出现在 y 之前，故 $\langle(x, y)\rangle$ 不频繁.

引理 2　当且仅当 $x \in \mathrm{CRMAP_s}(y)$ 时，相对于序列扩展 $\langle(x, y)\rangle$ 的长度为 2 的向后扩展序列被认为是频繁的.

证明：根据 $\mathrm{CRMAP_s}$ 的定义，如果项目 x 没有映射到 $\mathrm{CRMAP_s}(y)$，则相对于少于 min_sup 序列的序列扩展而言，x 出现在 y 之前，故 $\langle(x, y)\rangle$ 不频繁.

2　基于 MapReduce 的 IncSPM 算法

本文所提出的算法由两个 MapReduce 阶段组成. 第一个阶段读取序列的输入分裂并找出频繁 1 项集(1 模式)的支持计数，并通过标志变量来识别 1 模式是否属于增量数据集(IncSD). 第二阶段创建 CRMAP 数据结构，并利用 CRMAP 数据结构通过向后扩展的方式高效地生成候选数据. 此外，后向挖掘中使用的早期修剪属性避免了输入数据库中生成错误的候选序列，从而加快挖掘过程.

2.1 挖掘频繁1序列

本文算法第一阶段用来挖掘频繁1序列. 每个Mapper1读取输入的序列数据集并识别项目 x 是否属于相应序列的增量数据集IncSD，如果项目属于IncSD，则相对于项目 x 将1存储在分布式缓存F中，否则存储为0. 然后，Mapper将项目 x 及其在F中的相应值作为输出〈key，$value$〉.

Reducer1将〈x，$value$〉作为输入，并将项目计数和标志变量初始化为0，其中标志变量用来标识该项是否属于IncSD. Reducer1计算与每个项目相关的值的数量，在收到关于项目的值1之后，Reducer1立即将标志变量更改为1，这有助于在IncSPM的第二阶段找到增量并集，最后将频繁1项集及其对应的计数和标志作为第一阶段的输出，并将其存储在名为Litems的分布式缓存文件中. 本文IncSPM算法的第一阶段流程如图1所示.

2.2 挖掘频繁 k 序列

将第一阶段输出的原始数据集中的频繁1项集、最小支持计数和标志作为第二阶段的输入，然后Mapper2创造增量并集和构造CRMAP数据结构，使用早期修剪属性进行后向挖掘，最后使用Reducer2挖掘频繁 k 序列. 具体流程图如图2所示.

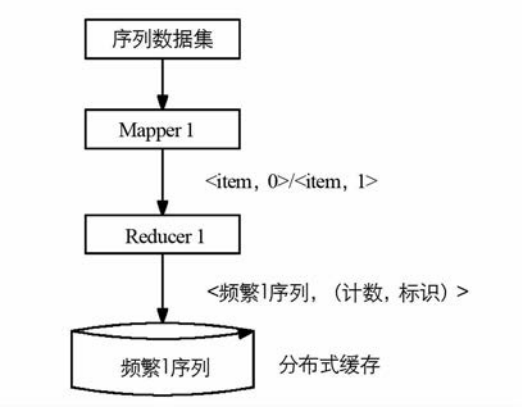

图1 IncSPM算法第一阶段流程图

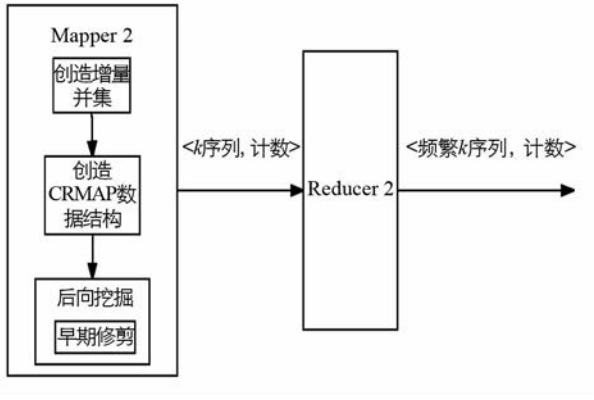

图2 IncSPM算法第二阶段流程图

2.2.1 发现增量并集

第二阶段中的SETUP函数读取分布式高速缓存文件Litems并找到增量并集. 对在分布式缓存文件Litems中的每一个项目，判断其标志是否等于1. 如果等于1，则将该项目加入增量并集，否则将1模式(项目和计数)分配给原始数据集中的频繁顺序模式 L_1^{UD}.

2.2.2 构造CRMAP数据结构

本文构造CRMAP数据结构生成出现在输入数据库中有希望的候选对象，以减少算法的运行内存.

$CRMAP_i$ 构造：根据定义，$CRMAP_i$ 是项目与其前面项目列表(共现列表，CL)相对于项集扩展的映射. 序列从最后一个项集扫描到第一个项集，以便找到项集中每个项目的CL. 如果 $CRMAP_i$ 中存在对应于 $item_i$ 的条目，则从 $item_i$ 的 $CRMAP_i$ 中检索 $item_i$ 的CL，通过检索到的CL检查共现项 $item_j$ 来找到它的计数. 如果项目 j 先前在CL中不存在，则它包含在CL($item_i$)中；如果 $CRMAP_i$ 中对应于项 i 的条目不存在，则通过包括同现项 $item_j$ 来创建项 i 的CL. $item_i$ 的 $CRMAP_i$ 用相应的CL更新.

$CRMAP_s$ 构造：根据定义，$CRMAP_s$ 是项目与其前面项目列表相对于序列扩展的映射，除了相对于序列扩展找到项的CL之外，创建 $CRMAP_s$ 的步骤与创建 $CRMAP_i$ 的步骤相同.

2.2.3 后向挖掘算法

后向挖掘算法的输入为序列、序列第一项的 $CRMAP_i$ 和 $CRMAP_s$、序列的末端映射和SD映射，输出为更新的数据集UD中的频繁序列模式. 1模式的末端投影用于寻找其向后延伸的末端投影，通过读取序列的 $CRMAP_i$ 和 $CRMAP_s$ 来推断序列可能的后向扩展，如果序列的 $CRMAP_i$ 不存在，则不存在项目集扩展. 从序列的末端投影找到这些生成序列中的每个序列的末端投影，扫描序列的SD投影以找到从其延伸的序列的SD投影，通过将 end_{pj} 的大小和 SD_{pj} 的大小相加获得这些序列的支持计数. 当所有扩展的支持计数满足最小支持阈值，它们被包括在频繁序列列表中并且调用递归调用.

2.2.4　生成候选序列

在创建 CRMAP 数据结构后，生成候选集. 为了减少候选者数量，本文基于后向挖掘设计了两个候选生成规则，以避免产生太多输入数据库中不存在的假数据，从而高效生成候选序列.

定义 1　对于长度为 $k-1$ 的给定序列模式 $s=\langle a_{k-1}a_{k-2}\cdots a_2a_1\rangle$，将属于 $\mathrm{CRMAP_i}(y)$ 的所有 x 扩展生成项集候选 $C_k=\langle\{xa_{k-1}\}a_{k-2}\cdots a_2a_1\rangle$，其中 y 是 a_{k-1} 中的第一项.

定义 2　对于长度为 $k-1$ 的给定序列模式 $s=\langle a_{k-1}a_{k-2}\cdots a_2a_1\rangle$，将属于 $\mathrm{CRMAP_s}(y)$ 的所有 x 扩展生成序列候选 $C_k=\langle\{xa_{k-1}\}a_{k-2}\cdots a_2a_1\rangle$，其中 y 是 a_{k-1} 中的第一项.

2.2.5　生成早期修剪序列

序列模式挖掘中最耗时的操作是计算支持计数. 如果在计算支持计数之前对生成的候选者进行修剪，则可以提高挖掘速度. 本文基于CRMAP数据结构定义了3个早期修剪属性，根据3个定义的属性对长度大于等于 3 的序列进行早期修剪.

定义 3　如果 C_{k+1} 中包括的新项目不存在于 C_k 的第一项集中任何项目的 $\mathrm{CRMAP_i}$ 中，则可以修剪任何项目集扩展候选 C_{k+1}.

定义 4　如果 C_{k+1} 中包括的新项目不存在于 C_k 的第二项集中任何项目的 $\mathrm{CRMAP_s}$ 中，则可以修剪任何项目集扩展候选 C_{k+1}.

定义 5　如果 C_{k+1} 中包括的新项目不存在于 C_k 的第一项集中任何项目的 $\mathrm{CRMAP_s}$ 中，则可以修剪任何序列扩展候选 C_{k+1}.

2.2.6　具体算法实现

本文第二阶段 MapReduce 平台中的 *SETUP* 函数初始化 map 或 reduce 阶段中使用的资源，每个 Mapper 读取其输入分割并修剪序列中不频繁的项目，随后创建 $\mathrm{CRMAP_i}$ 和 $\mathrm{CRMAP_s}$，找到每个 1 模式 x 的投影和末端投影. 检查每个 1 模式以确定其是否属于增量并集，以此找到稳定的 1 模式并扩展它们以找到它们的频繁超序列. 如果发现 1 模式不稳定，则为每个不稳定的 1 模式调用后向挖掘函数，在调用时向该函数传递 5 个参数：不稳定的 1 模式 x、x 关于序列扩展的同现表、x 关于项集扩展的同现表、x 的末端投影和 x 的 SD 投影，其中 x 的 SD 投影是 x 的投影和 x 的末端投影之间的差.

Reduce2 将 k 和值 v 作为输入，并将项目计数初始化为 0. Reducer2 计算与每个项目相关的值的数量，在收到关于项目的值之后，将项目计数增加. 如果项目计数大于最小支持计数与 $|UD|$ 之积，则输出 k 和项目计数，并将它们写入 Hadoop 分布式文件系统 HDFS 的文件里.

3　实验结果与分析

所有实验均是在具有 1 个主节点和 8 个数据节点的 Hadoop 集群上进行，单个节点在具有 Intel Xeon CPU 2.5GHz/6-Core 处理器和 16 GB 内存的机器上执行，每个节点都运行在装有 Hadoop 1.2.1 的 CentOS 6.5 服务器上，所有算法均使用 JDK 1.8.0_31 实现. 采用两个真实数据集 Kosarak 和 Twitter[12] 进行实验，Kosarak 数据集包含 990 002 个序列和 41 270 个项目，Twitter 数据集包括 12 053 495 条推文、510 603个用户和 1 434 862 个不同的项目.

图 3 和图 4 给出了本文算法与 SPAMC-UDLT[12] 算法在不同数据集上的性能评估. 可以看出执行时间随着最小支持度的增加而减少，本文算法的执行时间优于 SPAMC-UDLT 算法，这是因为本文算法简单地将末端投影和 SD 投影的大小相加来计算支持计数. 此外，使用 CRMAP 数据结构新的候选生成减少了搜索空间的大小，并且在后向挖掘期间应用的早期修剪属性在很大程度上减少了错误候选的数量.

图 5 和图 6 给出了各算法在不同数据集上的内存使用情况，可以看出内存消耗随着最小支持阈值的增加而减少，本文算法的内存消耗优于 SPAMC-UDLT 算法，这是因为 IncSPM 利用共生信息生成候选序列，从而减少了候选序列的数量. 此外，本文算法的早期修剪属性通过修剪错误的候选对象在很大程度上减少了内存使用.

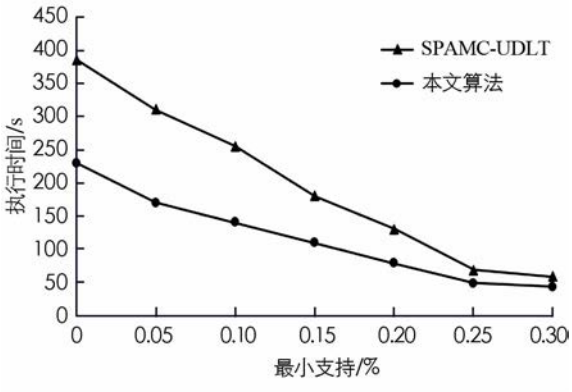

图3 Kosarak数据集下不同算法的性能比较

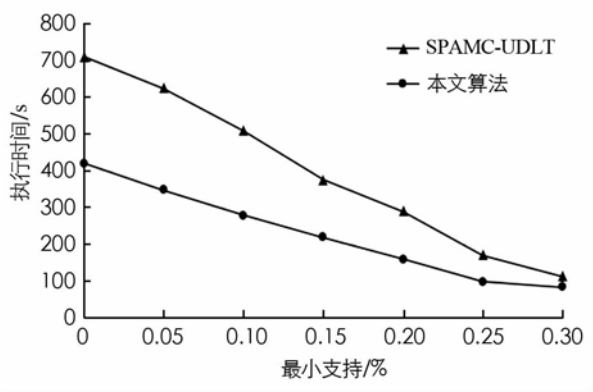

图4 Twitter数据集下不同算法的性能比较

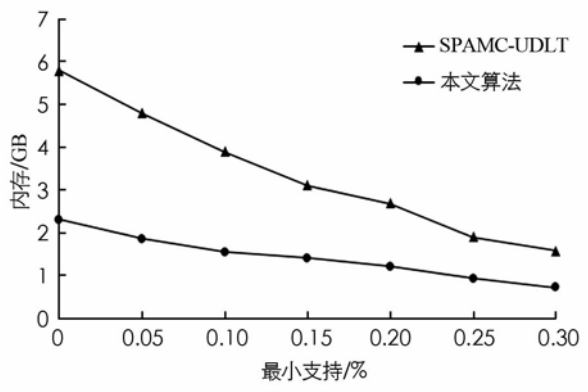

图5 Kosarak数据集下不同算法的使用内存比较

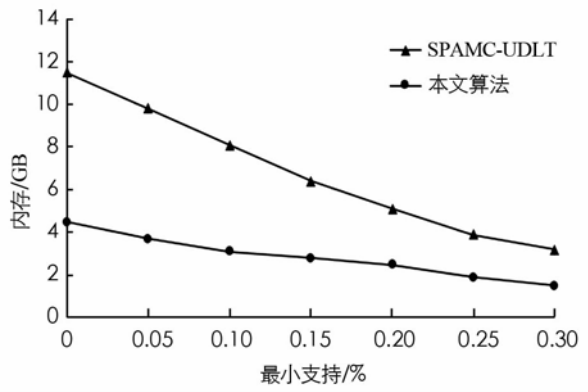

图6 Twitter数据集下不同算法的使用内存比较

图7给出了最小支持在0.1%～0.3%的情况下，本文算法相对于序列数量变化的可扩展性. 从图7可以看出，本文算法具有良好的可扩展性，这是因为本文设计的2个候选生成规则和3个早期修剪属性可以有效地避免在输入数据库中出现过多的错误候选生成，从而致使候选序列高效生成.

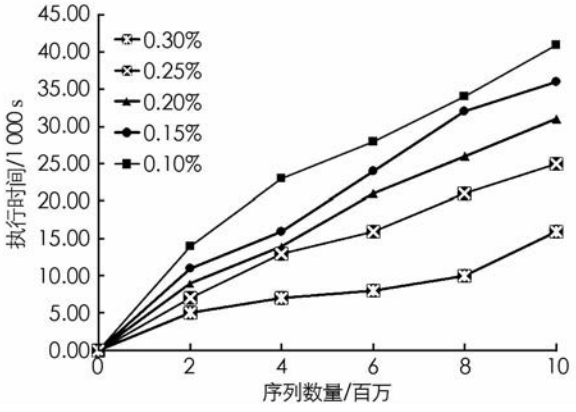

图7 本文算法相对于序列数量变化的可扩展性

4 结 语

为了使增量挖掘算法能够很好地适应大序列数据库，本文利用MapReduce分布式平台，提出了一种高效的增量序列模式挖掘(IncSPM)算法，用来处理大数据环境中的序列模式增量维护问题. 该算法通过结合候选序列的后向挖掘来避免挖掘更新数据库中支持度不变的序列，同时引入高效的CRMAP数据结构来生成出现在输入数据库中有希望的候选对象，以减小搜索空间. 基于CRMAP设计高效的候选生成规则和早期剪枝属性以避免输入数据库中生成错误的候选序列，加快挖掘过程. 实验表明，本文算法在执行时间、内存和可扩展性方面均有明显的提高. 未来的工作是解决在数据被删除和修改时更新序列模式的问题，并尝试在其他大数据处理框架(如Apache Spark)上实现所提出的算法.

参考文献：

[1] GAN W S, LIN J C W, FOURNIER-VIGER P, et al. A Survey of Parallel Sequential Pattern Mining [J]. ACM Transactions on Knowledge Discovery from Data, 2019, 13(3): 1-34.

[2] FOURNIER-VIGER P, LIN J C W, KIRAN R U, et al. A survey of sequential pattern mining [J]. Data Science and

Pattern Recognition, 2017, 1(1): 54-77.
[3] WANG Y C, KUNG L, WANG W Y C, et al. An Integrated Big Data Analytics-enabled Transformation Model: Application to Health Care [J]. Information & Management, 2018, 55(1): 64-79.
[4] O'HALLORAN K L, TAN S, PHAM D S, et al. A Digital Mixed Methods Research Design: Integrating Multimodal Analysis with Data Mining and Information Visualization for Big Data Analytics [J]. Journal of Mixed Methods Research, 2018, 12(1): 11-30.
[5] TARUS J K, NIU Z D, KALUI D. A Hybrid Recommender System for E-learning Based on Context Awareness and Sequential Pattern Mining [J]. Soft Computing, 2018, 22(8): 2449-2461.
[6] 邵　梁，何星舟，尚俊娜. 基于 Spark 框架的 FP-Growth 大数据频繁项集挖掘算法 [J]. 计算机应用研究，2018，35(10)：2932-2935.
[7] SALETI S, SUBRAMANYAM R B V. A Novel Mapreduce Algorithm for Distributed Mining of Sequential Patterns Using Co-occurrence Information [J]. Applied Intelligence, 2019, 49(1): 150-171.
[8] WANG J Z, HUANG J L. On Incremental High Utility Sequential Pattern Mining [J]. ACM Transactions on Intelligent Systems and Technology, 2018, 9(5): 1-26.
[9] GAUTAM J V, PRAJAPATI H B, DABHI V K, et al. Empirical Study of Job Scheduling Algorithms in Hadoop MapReduce [J]. Cybernetics and Information Technologies, 2017, 17(1): 146-163.
[10] SALETI S, SUBRAMANYAM R B V. A Novel Bit Vector Product Algorithm for Mining Frequent Itemsets from Large Datasets Using MapReduce Framework [J]. Cluster Computing, 2018, 21(2): 1365-1380.
[11] DINH D T, LE B, FOURNIER-VIGER P, et al. An Efficient Algorithm for Mining Periodic High-utility Sequential Patterns [J]. Applied Intelligence, 2018, 48(12): 4694-4714.
[12] CHEN C C, SHUAI H H, CHEN M S. Distributed and Scalable Sequential Pattern Mining through Stream Processing [J]. Knowledge and Information Systems, 2017, 53(2): 365-390.

Efficient Distributed Incremental Sequence Mining in Big Data Environment

NAN　Nan[1],　YAN Ying-zhan[2]

1. *Basic Education College, Lingnan Normal University, Zhanjiang Guangdong 524048, China;*
2. *No. 54th Institute, China Electronics Technology Group, Shijiazhuang 050081, China*

Abstract: An efficient distributed incremental sequential pattern mining algorithm (Incremental Sequential Pattern Mining, IncSPM) based on MapReduce architecture is proposed to solve the problem of updating sequential patterns whenever data increases in big data environment. With this algorithm, the backward mining algorithm is used to utilize effectively the sequence patterns generated by previous mining, and simultaneously design a Co-occurrence Reverse Map (CRMAP) data structure to deal with the combined explosion problem of candidate sequences. Finally, new candidate generation and early pruning mechanism are designed to speed up the mining process. The proposed algorithm is evaluated on two real datasets, and experiments show that compared with other methods, the algorithm proposed in this paper has a substantial improvement in execution time, memory consumption and scalability.

Key words: big data mining; incremental sequential pattern; backward mining; co-occurrence reverse map data structure

责任编辑　夏　娟