Software Engineering and Applications 软件工程与应用, 2018, 7(5), 261-272
Published Online October 2018 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2018.75030

Hans汉斯

Intelligent Recommendation for Literature Reading in Digital Library

Huisheng Zhu, Lin Chen, Yu Zhang, Yu Lu

School of Computer Science and Technology, Taizhou University, Taizhou Jiangsu
Email: zhs@fudan.edu.cn

Received: Sep. 20th, 2018; accepted: Oct. 1st, 2018; published: Oct. 8th, 2018

Abstract

Digital library has become a mainstream platform for readers to read literatures with its advantages of large information capacity, fast transmission speed, and less spatio-temporal restrictions. How to provide readers with high guidance quality and good personalized services is a key to transform digital library into smart library. In this paper, an intelligent recommendation algorithm for literature reading in digital library is proposed. After scanning the given literature reading stream only one pass, the algorithm mines frequent episodes by depth-first-search, stores frequent episodes by a shared prefix tree, and compresses the search space by episode monotonically. Experiments show that the proposed algorithm has better spatio-temporal performance and higher mining quality than the classical algorithms. The mining results can reveal the reading habits of readers and predict the reading tendency of readers, which helps the digital library to improve the utilization rate of literature resources and provide readers with better personalized services.

Keywords

Digital Library, Intelligent Recommendation, Frequent Episode, Depth-First-Search, Shared Prefix Tree

数字图书馆下的文献阅读智能推荐

朱辉生，陈　琳，张　禹，陆　宇

泰州学院计算机科学与技术学院，江苏　泰州
Email: zhs@fudan.edu.cn

收稿日期：2018年9月20日；录用日期：2018年10月1日；发布日期：2018年10月8日

文章引用: 朱辉生, 陈琳, 张禹, 陆宇. 数字图书馆下的文献阅读智能推荐[J]. 软件工程与应用, 2018, 7(5): 261-272.
DOI: 10.12677/sea.2018.75030

摘　要

数字图书馆以其信息容量大、传输速度快、时空限制少等优势，已成为读者阅读文献的主流平台。如何向读者提供导向性强、个性化好的服务，是数字图书馆向智慧图书馆转变的关键。鉴于此，提出一种文献阅读智能推荐算法。该算法只需单遍扫描文献阅读流，以深度优先搜索策略来挖掘频繁情节，以共享前缀树来存储频繁情节，以情节单调性来压缩搜索空间。实验表明，与经典算法相比，所提算法具有较好的时空性能和较高的挖掘质量，挖掘结果能够揭示读者的阅读习惯和预测读者的阅读倾向，从而有助于数字图书馆提高文献资源的利用率并向读者提供更好的个性化服务。

关键词

数字图书馆，智能推荐，频繁情节，深度优先搜索，共享前缀树

1. 引言

数字图书馆以其信息容量大、传输速度快、时空限制少等优势，已成为读者阅读文献的主流平台。以中国知网 CNKI 为例，作为目前全球最大的中文数字图书馆，CNKI 已拥有 40 多个国家及地区的众多读者，日均访问量超过 100 万人次。在集中文献资源的同时，如何向读者提供导向性强、个性化好的服务，是数字图书馆向智慧图书馆转变的关键。

文献阅读流是数字图书馆下读者产生的阅读事件组成的事件流，其中蕴藏着丰富的语义模式。文献智能推荐旨在对文献阅读流进行分析，挖掘频繁出现的文献序列(即频繁情节)，从而发现读者的阅读习惯和预测读者的阅读倾向，这将有助于数字图书馆提高文献资源的利用率并向读者提供个性化的服务。

自 Manilla [1]等人引入事件流上的频繁情节挖掘问题以来，众多学者对此展开了研究，提出了许多代表性算法。WINEPI [1]、WinMiner [2]和 HUEM-GAO [3]算法都是基于一个情节出现在其中的滑动窗口的个数来统计该情节的支持度，而 MINEPI [1]、EPT [4]、Clo_episode [5]、Ap-epi [6]、UP-Span [7]、DMinEpi [8]和 MELLO [9]等算法则是基于一个情节的最小发生(指这样的一个时间区间，所考虑的情节在其中发生，但不在其任何子区间上发生)次数来计算该情节的支持度。然而，这些算法在挖掘过程中需要多遍扫描事件流，并且产生了大量的候选情节。另外，这些算法在计算一个情节的支持度时都可能包含了该情节多次重叠的发生，从而会导致情节发生的“过计数”问题。例如，设事件流 S = <(A,1), (A,2), (B,3), (D,4), (A,5), (C,6), (B,7), (E,8), (A,9), (C,10), (B,11), (A,12), (C,13)>，窗口大小为 3，则事件流 S 被分为 13 − 1 + 3 = 15 个滑动窗口，即 W1 = <(A,1)>, W2 = <(A,1), (A,2)>, W3 = <(A,1), (A,2), (B,3)>, W4 = <(A,2), (B,3), (D,4)>, W5 = <(B,3), (D,4), (A,5)>等，尽管情节<BD>在 S 上只出现了一次，但由 WINEPI 等算法计算得到的情节<BD>的支持度却为 2(对应着两个重叠的滑动窗口 W4 和 W5)，由 MINEPI 等算法计算得到的情节<ABC>的支持度为 3 (对应着三个重叠的最小区间[2,6], [5,10], [9,13])。

为了解决情节发生的“过计数”问题，学者们基于非重叠发生(一个情节的两次发生是非重叠的，当且仅当其中的一次发生出现在另一次发生之后)计数提出了事件流频繁情节挖掘算法 Discovery Non Over [10]、NONOVERLAPPING [11]和 NONEPI [12]，这些算法为每个情节使用了一个自动机，通过单遍扫描

事件序列来跟踪这些自动机的状态迁移，从而统计出相应情节在事件流上的发生次数。虽然这些算法较好地解决了情节发生的“过计数”问题，但由于非重叠发生未必是最小发生，所以这些算法挖掘的频繁情节并不能很好地描述一个情节中事件类型之间的紧随关系。例如，由算法 NONEPI 得到的情节<ABC>在事件流 S 上的第一次发生是[1,6] (即<(A,1), (B,3), (C,6)>)，并不是最小发生[2,6] (即<(A,2), (B,3), (C,6)>)。

为此，本文提出了一种事件流频繁情节挖掘算法 *MANEPI* (Minimal And Non-overlapping EPIsode)。该算法基于最小且非重叠发生来计算一个情节的支持度，既可以很好地描述一个情节中事件类型之间的紧随关系，又可以避免情节发生的“过计数”问题。该算法采用深度优先搜索策略来挖掘频繁情节，只需要单遍扫描事件流且不产生任何候选情节，大大提高挖掘效率。另外，该算法使用共享前缀树来存储频繁情节，并利用情节单调性压缩频繁情节的搜索空间，加速挖掘过程。理论分析和实验评估表明该算法能有效地挖掘给定事件流上的频繁情节。

2. 预备知识

2.1. 基本概念

定义 1 (事件流)：给定事件类型集 $\varepsilon=\{E_1,E_2,\cdots,E_m\}$，一个事件就是一个二元组$(E,t)$，其中，$E\in\varepsilon$，$t$ 表示该事件的发生时间。定义在 ε 上的事件流 S 是由若干事件按发生时间先后排列的序列，表示为 $S=\langle(E_1,t_1),(E_2,t_2),\cdots,(E_n,t_n)\rangle$，其中 $t_i<t_j\,(1\le i<j\le n)$。

定义 2 (情节)：情节α是由若干事件类型组成的序列，记为 $\alpha=\langle E_1E_2\cdots E_k\rangle$，其中 $E_i\,(1\le i\le k)\in\varepsilon$ 且对于所有的 i 和 j $(1\le i<j\le k)$满足 E_i 总是列在 E_j 之前。情节α中事件类型的个数称为α的长度，记为$|\alpha|$。长度为 k 的情节称为 k-情节。

定义 3 (子情节，超情节)：给定情节α和β，若β中的事件类型均在α中出现，且先后顺序与α中的这些事件类型一致，则称β是α的子情节，或α是β的超情节，记为$\beta\subseteq\alpha$或$\alpha\supseteq\beta$。

定义 4 (前缀)：设情节 $\alpha=\langle E_1E_2\cdots E_k\rangle$，则称情节 $\beta=\langle E_1E_2\cdots E_{k-1}\rangle$ 是α的前缀。

定义 5 (串接)：给定情节α和β，将β的所有事件类型按其顺序列在α后面形成的新情节称为α和β的串接，记为 $concat(\alpha,\beta)$。

定义 6 (发生)：给定事件流 S 和情节 $\alpha=\langle E_1E_2\cdots E_k\rangle$，若 $\langle(E_1,t_1),(E_2,t_2),\cdots,(E_k,t_k)\rangle$ 是从 S 中删除若干事件后得到，且 $t_i<t_{i+1}\,(1\le i\le k-1)$，则称区间 $[t_1,t_k]$ 为α在 S 上的一次发生，t_k-t_1 为该发生的区间长度，t_1 为该发生的起始时间，t_k 为该发生的终止时间。

定义 7 (最小发生)：设 $[t_s,t_e]$ 是情节α在事件流 S 上的一次发生，若 S 上不存在α的另一次发生 $[t'_s,t'_e]$，使得 $t_s<t'_s$ 且 $t'_e\le t_e$，或 $t_s\le t'_s$ 且 $t'_e<t_e$，即 $[t'_s,t'_e]\subset[t_s,t_e]$，则称 $[t_s,t_e]$ 是α在 S 上的一次最小发生。情节α在事件流 S 上的所有最小发生组成的集合记为$\alpha.mo$。

定义 8 (最小且非重叠发生)：设 $[t_s,t_e]$ 和 $[t'_s,t'_e]$ 是情节α在事件流 S 上的两次最小发生，且满足 $t_e<t'_s$ 或 $t'_e<t_s$，则称 $[t_s,t_e]$ 和 $[t'_s,t'_e]$ 是α在 S 上的最小且非重叠发生。情节α在事件流 S 上的所有最小且非重叠发生组成的最大集合记为$\alpha.mano$。

定义 9 (支持度)：集合$\alpha.mano$ 的基数$|\alpha.mano|$称为情节α的支持度，记为$\alpha.sup$。

定义 10 (频繁情节)：给定支持度阈值 *min_sup*，若情节α的支持度大于等于 *min_sup*，则称α是一个频繁情节。

2.2. 引理 1：若$\beta\subseteq\alpha$，则$\beta.sup\ge\alpha.sup$

证明：因为β是α的子情节，则对于$\alpha.mano$ 中α的任意两个最小且非重叠发生 $[t_{is},t_{ie}]$ 和 $[t_{js},t_{je}]$，一

定存在β的两个最小且非重叠发生$[t'_{is},t'_{ie}]$和$[t'_{js},t'_{je}]$，满足$[t'_{is},t'_{ie}]\subseteq[t_{is},t_{ie}]$和$[t'_{js},t'_{je}]\subseteq[t_{js},t_{je}]$，这表明$|\beta.mano|\geq|\alpha.mano|$，即$\beta.sup\geq\alpha.sup$。引理 1 证毕。■

2.3. 定理 1：非频繁情节的任何超情节也是非频繁的，频繁情节的任何子情节也是频繁的

定理 1 也称为情节单调性。

证明：由引理 1 和定义 10 可得证。■

2.4. 问题描述

给定事件流 *S* 和支持度阈值 *min_sup*，问题描述为：设计一个挖掘 *S* 上所有频繁情节的算法，要求：1) 只需要单遍扫描 *S*；2) 挖掘过程中不产生候选频繁情节；3) 尽可能少的时空需求。

3. 频繁情节挖掘算法 MANEPI

3.1. 共享前缀树

由于许多频繁情节具有相同的前缀，为节省存储空间，算法采用共享前缀树存储频繁情节。共享前缀树 *SPT* (Shared Prefix Tree)是一棵有向根树，树中的每个结点 N_α表示为一个四元组(*label, mo, mano, children*)，其中：*label* 为结点 N_α对应的事件类型，根结点的 *label* 为空，从根结点深度遍历到结点 N_α的路径上所有 *label* 的串接表示频繁情节α；*mo* 为结点 N_α对应的频繁情节α的最小发生集，根结点的 *mo* 为空；*mano* 为结点 N_α对应的频繁情节α的最小且非重叠发生集，根结点的 *mano* 为空；*children* 为指向结点 N_α的孩子结点的指针，且这些指针按照所指孩子结点的 *label* 的字典序排列，所有叶结点的 *children* 为空。不难看出，*SPT* 中非根结点对应的频繁情节是其所有孩子结点对应的频繁情节的前缀，共享前缀树由此而得名。

3.2. 算法 MANEPI 描述

算法 *MANEPI* 的基本思想是：首先单遍扫描给定事件流，从中发现所有频繁 1-情节；然后对已发现的一个频繁 *i*-情节α和一个频繁 1-情节 *e* 进行串接，形成一个长度为 *i*+1 的新情节 *concat*(α,*e*)，这个过程称为以α为前缀的一次情节增长。若 *concat*(α,*e*)的支持度小于给定的支持度阈值，则 *concat*(α,*e*)为非频繁情节，由定理 1 可知，*concat*(α,*e*)的任何超情节也是非频繁的，因而就无须以 *concat*(α,*e*)为前缀进行情节增长；反之，若 *concat*(α,*e*)的支持度大于等于给定支持度阈值，则 *concat*(α,*e*)为频繁情节，算法接着以 *concat*(α,*e*)为前缀递归地进行下一次情节增长，整个算法的终止条件是没有任何情节可以增长。下面是 *MANEPI* 的伪代码。

AlgorithmMANEPI(S, min_sup)

Input: *S*: an event stream = <(E_1,t_1), (E_2,t_2), ⋯, (E_n,t_n)>;

min_sup: a support threshold

Output: *SPT*: a tree which stores all frequent episodes within *S*

1: Create *SPT* with simply a root node

2: Scan *S* once to find all frequent 1-episodes and Sort them in lexicographical order

3: For each frequent 1-episode *e*do

4: Create node N_e as a child of the root

5: For each frequent 1-episode *e*do

6: *MineGrow* (N_e)

Procedure Mine Grow (N_α)

Input: N_α: the node to be expanded

Objective: find all frequent episodes with prefix α

1: For each 1-episode e do

2: Let $\beta = concat(\alpha,e)$

3: Let $\beta.mo = ComputeMO(\alpha,e)$

4: Let $\beta.mano = ComputeMANO(\beta)$

5: If $\beta.sup \geq min\_sup$

6: Create node N_β as a child of N_α

7: $MineGrow(N_\beta)$

3.3. 情节增长

MANEPI 的核心是情节增长 *MineGrow*，该过程的关键是如何计算一个新情节的支持度，具体而言，就是如何根据已发现的频繁 i-情节α和频繁 1-情节 e 来发现新情节 $concat(\alpha,e)$的所有最小且非重叠发生组成的最大集合。直观上，可以基于$\alpha.mano$ 和 $e.mano$ 来构造 $concat(\alpha,e).mano$。然而，由于在某些情形下α可能存在多个$\alpha.mano$，无论选择哪一个$\alpha.mano$，都有可能导致 $concat(\alpha,e)$的一些最小且非重叠发生丢失，从而也就不能发现所有的频繁情节。

例如，设 S = <(A,1), (A,2), (B,3), (D,4), (A,5), (C,6), (B,7), (E,8), (A,9), (C,10), (B,11), (A,12), (C,13)>，α = <AA>，e = <B>，$\beta = concat(\alpha,e)$，表 1 列出了基于 S 上所有可能的$\alpha.mano$ 进行情节增长的两种情形。

从表 1 中可以看出，无论选择哪一个$\alpha.mano$，都不能得到正确的$\beta.mano$，因为两种情形下的$|\beta.mano|$都等于 1，但事实上$|\beta.mano|$ = |{[1,3], [5,11]}| = 2，若 min_sup = 2，则丢失了频繁情节β。可见，挖掘频繁情节时，若在 SPT 中只保存结点的 mano，则不足以进行正确的情节增长，可能会导致部分频繁情节的丢失。为此，算法将情节增长分为两步实施：第一步，通过单遍扫描$\alpha.mo$ 和 $e.mo$ 得到 $concat(\alpha,e).mo$，这三个最小发生集均是唯一的；第二步，通过单遍扫描 $concat(\alpha,e).mo$ 得到最早出现的 $concat(\alpha,e).mano$。因为 $concat(\alpha,e).mo$ 保存了 $concat(\alpha,e)$的所有最小发生，所以由 $concat(\alpha,e).mo$ 得到的最早出现的 $concat(\alpha,e).mano$ 不会丢失 $concat(\alpha,e)$的任何最小且非重叠发生。由于$\alpha.mo$ 和 $e.mo$ 已将给定的事件流分割成有限个区间，所以基于这些有限的区间进行的情节增长避免了对给定事件流的多遍扫描。另外，*SPT* 中每个结点的 *mano* 只是保存了指向该结点 *mo* 中相应最小发生的指针。下面是情节增长(包括两个子过程)的伪代码。

SubProcedure Compute MO(α,e)

Input: α: a frequent i-episode with its *mo*;

e: a frequent 1-episode with its*mo*

Output: $concat(\alpha,e).mo$: the *mo*of $concat(\alpha,e)$

1: Let$concat(\alpha,e).mo = \phi$

2: Let i = 1

3: For each occurrence $[t'_s, t'_s] \in e.mo$ do

4: Starting from i, find the first index j of $\alpha.mo$s.t. $[t_{js}, t_{je}] \in \alpha.mo$ and $t_{je} < t'_s$ and $t_{(j+1)e} > t'_s$

5: Append $[t_{js}, t'_s]$ to $concat(\alpha,e).mo$

6: Let $I = j$

7: Return $concat(\alpha,e).mo$

Table 1. Two cases of episode growth
表 1. 情节增长的两种情形

α.mano	e.mano	β.mano
{[1,2], [9,12] }	{[3,3], [7,7], [11,11]}	{[1,3]}
{[2,5], [9,12]}	{[3,3], [7,7], [11,11]}	{[2,7]}

SubProcedure Compute MANO(β)

Input: β: an episode with its *mo*

Output: β. *mano*: the earliest *mano* of episode β

1: Let β. *mano* = {1}

2: Let $i = 1$

3: Let $j = i + 1$

4: While $j < |\beta.\ mo|$ do

5: Find the first index k of $\beta.mo$ after js.t. $[t_{is}, t_{ie}] \in \beta.\ mo$ and $[t_{ks}, t_{ke}] \in \beta.\ mo$ and $t_{ie} < t_{ks}$

6: $\beta.mano=\beta.\ mano\ \cup\ \{k\}$

7: Let $i = k$

8: Let $j = i + 1$

9: Return $\beta.mano$

3.4. 算法 MANEPI 运行实例

设事件流 S = <(A,1), (A,2), (B,3), (D,4), (A,5), (C,6), (B,7), (E,8), (A,9) (C,10), (B,11), (A,12), (C,13)>以及 $min\_sup = 2$。首先，*MANEPI* 通过单遍扫描 S，得到所有的频繁 1-情节<A>、<B>和<C>，并在 *SPT* 中生成根结点的孩子结点 $N_{<A>}$、$N_{<B>}$和 $N_{<C>}$；然后进行第一次情节增长，将已发现的频繁情节<A>与频繁 1-情节<A>串接后形成新情节<AA>，通过调用 *ComputeMO* 和 *ComputeMANO* 得到<AA>.sup = |{[1, 2], [5, 9]}| = 2 ≥ *min_sup*，在 SPT 中生成结点 $N_{<A>}$的孩子结点 $N_{<AA>}$；接着，进行第二次情节增长，将已发现的频繁情节<AA>与频繁 1-情节<A>串接后形成新情节<AAA>，得<AAA>.sup = |{[1, 5]}| = 1 < *min_sup*，因此，无须以<AAA>为前缀进行情节增长，算法返回至上一层，将已发现的频繁情节<AA>与频繁 1-情节<B>串接后形成新情节<AAB>，…；最后，在 S 上发现了如表 2 所示的 18 个频繁情节，挖掘过程中产生的共享前缀树如图 1 所示。

3.5. 算法 MANEPI 正确性证明

引理 2. 给定最小发生集的情节α和 e，过程 *ComputeMO* 能够正确计算新情节 *concat*(α,e)的最小发生集。

证明：对于 *e.mo* 中的每一个发生$[t'_s,t'_s]$ (第 3 行)，*ComputeMO* 通过单遍扫描 *e.mo* 查找一个最大的 j(第 4 行)满足$[t_{js}, t_{je}] \in \alpha.mo$ 且 $t_{je} < t'_s$ 且 $t_{(j+1)e} > t'_s$，若不存在这样的 j，则停止扫描 *e.mo*。因$[t_{js},t_{je}]$是α出现在 e 的发生$[t'_s,t'_s]$之前的最后一次发生，故$[t_{js},t'_s]$是 *concat*(α,e)的一次最小发生，*concat*(α,e).*mo* 是 *concat*(α,e)的所有最小发生组成的集合。引理 2 证毕。■

引理 3. 给定最小发生集的情节β，过程 *ComputeMANO* 能够正确计算β最早出现的所有最小且非重叠发生组成的最大集合。

证明：

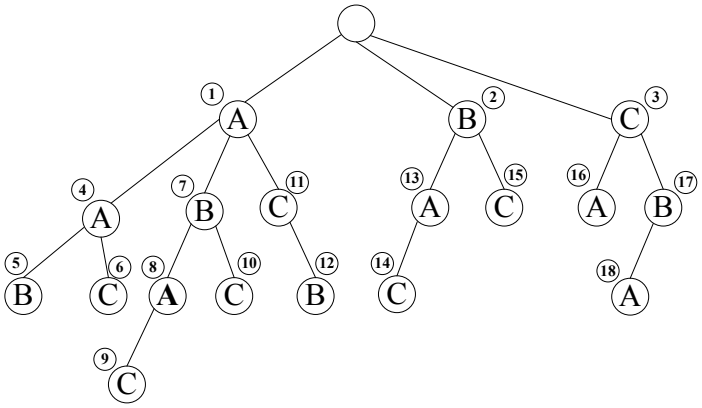

Figure 1. The shared prefix tree for frequent episodes within S
图 1. S 上的频繁情节共享前缀树

Table 2. All frequent episodes with S
表 2. S 上的频繁情节

number	episode	mo	mano
1	<A>	{[1,1], [2,2], [5,5], [9,9], [12,12]}	{1,2,3,4,5}
2	<B>	{[3,3], [7,7], [11,11]}	{1,2,3}
3	<C>	{[6,6], [10,10], [13,13]}	{1,2,3}
4	<AA>	{[1,2], [2,5], [5,9], [9,12]}	{1,3}
5	<AAB>	{[1,3], [2,7], [5,11]}	{1,3}
6	<AAC>	{[2,6], [5,10], [9,13]}	{1,3}
7	<AB>	{[2,3], [5,7], [9,11]}	{1,2,3}
8	<ABA>	{[2,5], [5,9], [9,12]}	{1,3}
9	<ABAC>	{[2,6], [5,10], [9,13]}	{1,3}
10	<ABC>	{[2,6], [5,10], [9,13]}	{1,3}
11	<AC>	{[5,6], [9,10], [12,13]}	{1,2,3}
12	<ACB>	{[5,7], [9,11]}	{1,2}
13	<BA>	{[3,5], [7,9], [11,12]}	{1,2,3}
14	<BAC>	{[3,6], [7,10], [11,13]}	{1,2,3}
15	<BC>	{[3,6], [7,10], [11,13]}	{1,2,3}
16	<CA>	{[6,9], [10,12]}	{1,2}
17	<CB>	{[6,7], [10,11]}	{1,2}
18	<CBA>	{[6,9], [10,12]}	{1,2}

1) 过程 *ComputeMANO* 首先将$\beta.mo$ 中第一个发生$[t_{1s}, t_{1e}]$的索引号 $i(i = 1)$添加至$\beta.mano$ 中(第 1 行)，然后从索引号 $j = i + 1$ 开始(第 4 行)单遍扫描$\beta.mo$，以查找最小的 k(第 5 行)满足$[t_{is}, t_{ie}] \in \beta.mo$ 且 $t_{ie} < t_{ks}$，即查找β的第一个与$[t_{is}, t_{ie}]$非重叠的发生$[t_{ks}, t_{ke}]$，若不存在这样的 k，则停止扫描$\beta.mo$。由于每次添加至$\beta.mano$ 中的发生(第 6 行)均与当前$\beta.mano$ 中的最后一次发生非重叠，所以最终的$\beta.mano$ 应是β最早出现的所有非重叠发生组成的集合。

2) 由引理 2 知，$\beta.mano$ 是β的最小发生集。

综合 1)和 2)，引理 3 证毕。■

引理 4：对于已发现的频繁情节α，过程 *MineGrow* 能够正确发现所有以α为前缀的频繁情节，并在 *SPT* 中生成相应的结点。

证明对于频繁情节α，过程 *MineGrow* 根据引理 2 和引理 3，进行以α为前缀的一次情节增长(第 2-4 行)，若 $concat(\alpha,e)$是非频繁的，则不在 SPT 中生成相应的结点，根据定理 1，$concat(\alpha,e)$的所有超情节也是非频繁的，过程 *MineGrow* 将以α为前缀进行下一次情节增长(第 1 行)；若 $concat(\alpha,e)$是频繁的，则在 *SPT* 中生成相应的结点(第 6 行)，并以 $concat(\alpha,e)$为前缀进行一次情节增长(第 7 行)，如此迭代结束后，能够发现所有以α为前缀的频繁情节，并在 *SPT* 中生成相应的结点。引理 4 证毕。■

引理 5：给定事件流 *S* 和支持度阈值 *min_sup*，算法 *MANEPI* 能够正确发现 *S* 上的所有频繁情节，并在 *SPT* 中生成相应结点。

证明 *MANEPI* 首先单遍扫描 *S* 发现所有频繁 1-情节(第 1 行)，然后对每个频繁 1-情节进行如下处理：在 *SPT* 中生成相应结点，并以该情节为前缀进行情节增长(第 2~6 行)。由引理 4 知，*MANEPI* 最终能够发现所有频繁情节，并在 *SPT* 中生成相应结点。引理 2.5 得证。■

3.6. 算法 MANEPI 复杂度分析

设 *S* 为 *L* 个事件组成的事件流，ε为 *S* 上的事件类型集，*FE* 为 *S* 上的所有频繁情节组成的集合，*sup_max* 为所有频繁 1-情节的最大支持度，则算法 *MANEPI* 的复杂度分析如下：

定理 2：*MANEPI* 的时间复杂度为$O(|FE| \cdot |\varepsilon| \cdot L)$。

证明：对于 *FE* 中的每个频繁情节α和ε中的每个频繁 1-情节 *e*，旨在找到最早出现的 $concat(\alpha,e).mano$ 的情节增长操作是 *MANEPI* 的主要时间代价。完成一次情节增长只需单遍扫描$\alpha.mo$ 和 *e.mo*，其时间复杂度为$O(L)$。由定理 1 可知，*MANEPI* 只需以 *FE* 中的每个情节为前缀进行情节增长，增长的次数为频繁 1-情节的个数，其上界为$|\varepsilon|$，因此，算法 *MANEPI* 完成全部的情节增长操作所需的时间复杂度为$O(|FE| \cdot |\varepsilon| \cdot L)$。定理 2 证毕。■

定理 3：*MANEPI* 的空间复杂度为$O(sup_max \cdot |FE|)$。

证明算法 *MANEPI* 共发现了$|FE|$个频繁情节，而每个频繁情节至多有 *sup_max* 个最小且非重叠的发生，故算法所需的空间复杂度为$O(sup_max \cdot |FE|)$。定理 3 证毕。■

4. 实验评估

本文通过七组实验来评估算法 *MANEPI*。实验对比了 *MANEPI* 和经典算法 *MINEPI* [1]和 *NONEPI* [12]的运行时间、内存开销和挖掘质量。实验的硬件环境为 3.6 GHz Intel(R) Core(TM) i7-4790 CPU，内存 8 GB，操作系统为 Windows 8，程序采用 VC++ 6.0 实现。

4.1. 数据集

我们选用 CNKI 的一个 Web 服务器上从 2017 年 11 月 1 日至 2017 年 11 月 30 日的日志数据，该日志数据包括了相关读者对 156259 种不同文献的 207498 个阅读记录。

4.2. 实验结果

实验 1：运行时间 vs 支持度阈值通过改变支持度阈值，得到如图 2 所示的三个算法在数据集上的运行时间(以毫秒为单位)。

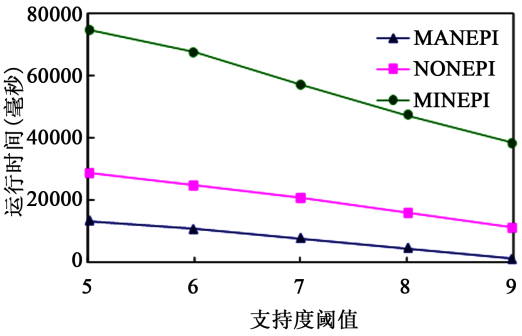

Figure 2. Averageruntime vs. min_sup
图 2. 运行时间 vs 支持度阈值

可以看出，随着支持度阈值的减小，三个算法的运行时间都线性增加，导致上述事实的主要原因是：*MANEPI* 采用了深度优先搜索策略，而 *MINEPI* 和 *NONEPI* 采用了广度优先搜索策略。

实验 2：运行时间 vs 序列长度选择数据集上前 6 天、前 12 天、前 18 天、前 24 天的事件作为四个序列，设定支持度阈值为 7，得到如图 3 所示的序列长度对算法时间性能的影响。

可以看出，三个算法的运行时间都随着序列长度的增加而线性增加，且它们的时间性能对比结果同实验 1，导致这些现象的原因与实验 1 的解释相同。

实验 3：内存开销 vs 支持度阈值通过改变支持度阈值，得到如图 4 所示的三个算法在数据集上的内存开销(以 KB 为单位)。

可以看出，随着支持度阈值的减小，三个算法的内存开销都线性增加，导致这些现象的原因与实验 1 的解释相同。

实验 4：内存开销 vs 序列长度设定支持度阈值为 7，得到如图 5 所示的序列长度对三个算法内存开销的影响。

可以看出，三个算法的内存开销都随着序列长度的增加而线性增加，且 *MANEPI* 的空间性能优于 *MINEPI* 和 *NONEPI*，导致这些现象的原因与实验 1 的解释相同。

实验 5：频繁情节个数 vs 支持度阈值通过改变支持度阈值，得到如图 6 所示的算法发现的频繁情节个数。

可以看出，随着支持度阈值的减小，*MANEPI* 和 *MINEPI* 均发现了更多的频繁情节，这是因为在同一事件流上，支持度阈值越小，满足支持度阈值要求的频繁情节就越多。同时，可以观察到在同一事件流上 *MINEPI* 挖掘的频繁情节要多于 *MANEPI*，这一现象也证实了 *MINEPI* 容易导致情节发生“过计数”问题，原因在于 *MINEPI* 基于最小发生来计算一个情节的支持度，而 *MANEPI* 基于最小且非重叠发生来计算一个情节的支持度。

实验 6：频繁情节个数 vs 序列长度设定支持度阈值为 7，得到如图 7 所示的序列长度对 *MANEPI* 和 *MINEPI* 发现频繁情节个数的影响。

可以看出，随着序列长度的增加，两个算法发现了更多的频繁情节，这是因为在同一支持度阈值下，事件流越长，满足支持度阈值要求的频繁情节就越多。同时，我们也观察到与实验 5 相同的另一现象，解释原因同实验 5。

实验 7：平均区间长度 vs 情节长度我们使用平均区间长度来评估算法的挖掘质量。所谓平均区间长度，是指相同长度频繁情节的所有发生区间的平均值，该值越小，越能体现一个情节中事件类型之间的紧随关系，挖掘的质量也就越高。设定支持度阈值为 7，得到如图 8 所示的情节长度对 MANEPI 和 NONEPI 的平均区间长度的影响。

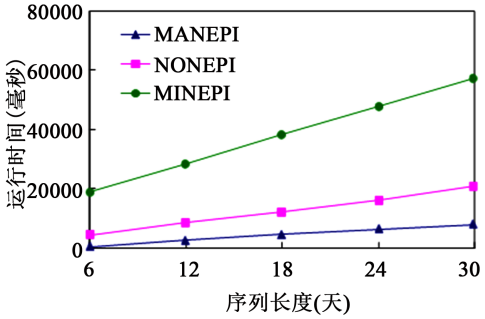

Figure 3. Averageruntime vs.sequence length
图 3. 运行时间 vs 序列长度

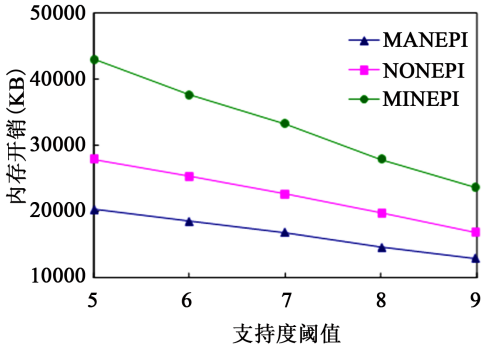

Figure 4. Average memory vs. min_sup
图 4. 内存开销 vs 支持度阈值

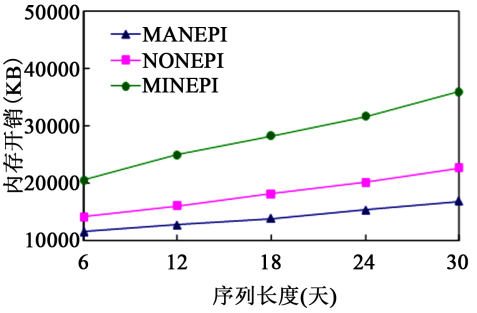

Figure 5. Average memory vs. sequence length
图 5. 内存开销 vs 序列长度

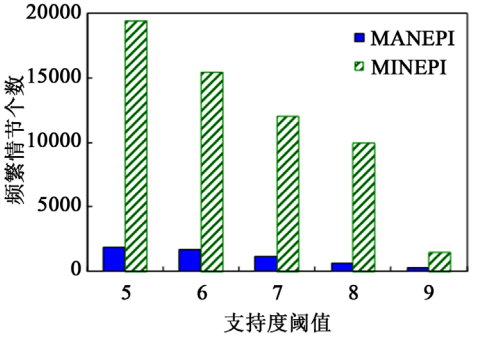

Figure 6. Number of frequent episodes vs. min sup
图 6. 频繁情节个数 vs 支持度阈值

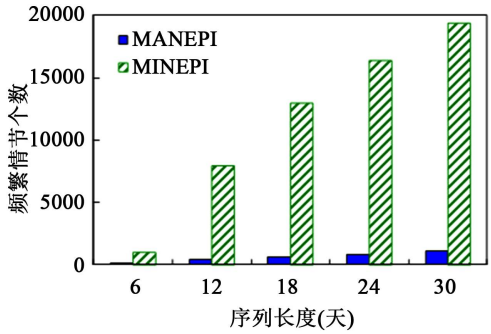

Figure 7. Number of frequent episodes vs. sequence length
图 7. 频繁情节个数 vs 序列长度

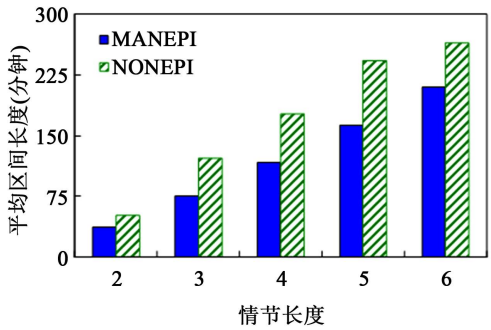

Figure 8. Average interval vs. episode length
图 8. 平均区间长度 vs 情节长度

可以看出，随着情节长度的增加，两个算法的平均区间长度也在增加。同时，*MANEPI* 的挖掘质量高于 *NONEPI*，这是因为 *NONEPI* 所考虑的非重叠发生未必是最小发生，挖掘的频繁情节并不能很好地描述一个情节中事件类型之间的紧随关系，下面举例说明这个问题。

<基于隐马尔可夫模型的多步攻击预测研究，面向分布数据安全的误用检测算法和入侵检测系统的研究，隐马尔可夫模型在入侵检测中的应用，基于入侵响应的入侵警报关联性的攻击预测算法，基于因果网络的攻击计划认别与预测，面向网络安全的基于入侵事件的早期预警方法>是算法 *MANEPI* 及 *NONEPI* 均能发现的一个频繁 6-情节，该情节刻画了读者如下的行为模式：

旨在研究一个基于隐马尔可夫模型的攻击预测方法(见第一篇阅读文档)，这些读者首先理解了两种常用的入侵检测模型，即误用检测模型(见第二篇阅读文档)和异常检测模型(见第三篇阅读文档)，然后分析了现有攻击方法的不足(见后三篇阅读文档)。

由算法 *MANEPI* 和 *NONEPI* 得到的该情节的平均区间长度分别是 180 分钟和 300 分钟，即表示大多读者完成上述 6 篇文献的阅读分别需要 3 小时和 5 小时左右。显然，*MANEPI* 的挖掘结果更有助于 CNKI 向读者提供个性化的文献阅读推荐服务。

5. 结束语

本文提出了一个采用深度优先搜索策略和共享前缀树存储结构的频繁情节挖掘算法 *MANEPI*，该算法只需单遍扫描给定的事件流，且挖掘过程不产生候选频繁情节。实验评估表明算法 *MANEPI* 具有较好的时空性能和较高的挖掘质量，挖掘结果能够揭示读者的阅读习惯和预测读者的阅读倾向，从而有助于数字图书馆向读者提供更好的个性化服务。未来我们将研究能够融合多种支持度定义的频繁情节挖掘算法。

基金项目

本文受江苏省自然科学基金(BK20141307)，江苏省“333 工程”基金(BRA2015212)，教育部“云数融合科教创新”基金(2017B06109)，江苏省“大创”基金(201712917007Y)资助。

参考文献

[1] Mannila, H., Toivonen, H. and Verkamo, A.I. (1997) Discovering Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, **1**, 259-289. https://doi.org/10.1023/A:1009748302351

[2] Méger, N. and Rigotti, C. (2004) Constraint-Based Mining of Episode Rules and Optimal Window Sizes. *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Springer, Pisa, 313-324.

[3] Lin, Y.-F., Huang, C.-F. and Tseng, V.S. (2017) A Novel Methodology for Stock Investment Using High Utility Episode Mining and Genetic Algorithm. *Applied Soft Computing*, **59**, 303-315. https://doi.org/10.1016/j.asoc.2017.05.032

[4] Ma, X., Pang, H.H. and Tan, K.L. (2004) Finding Constrained Frequent Episodes Using Minimal Occurrences. *Proceedings of the 4th IEEE International Conference on Data Mining*, Brighton, 1-4 November 2004, 471-474.

[5] Zhou, W.Z., Liu, H.Y. and Cheng, H. (2010) Mining Closed Episodes from Event Sequences Efficiently. *Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, Springer, Hyderabad, 310-318. https://doi.org/10.1007/978-3-642-13657-3_34

[6] Wu, J.J., Wan, L. and Xu, Z.R. (2011) Algorithms to Discover Complete Frequent Episodes in Sequences. *Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, Shenzhen, 24-27 May 2011, 267-278.

[7] Wu, C.W., Lin, Y.F., Yu, P.S., *et al*. (2013) Mining High Utility Episodes in Complex Event Sequences. *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM Press, Chicago, 536-544.

[8] 林树宽, 乔建忠. 一种基于情节矩阵和频繁情节树的情节挖掘方法[J]. 控制与决策, 2013, 28(3): 339-344.

[9] Ao, X., Luo, P., Li, C.K., *et al*. (2014) Discovering and Learning Sensational Episodes of News Events. *Proceedings of the 23rd International World Wide Web Conferences*, ACM Press, Seoul, 217-218.

[10] Laxman, S. (2006) Discovering Frequent Episodes: Fast Algorithms, Connections with HMMs and Generalizations. Indian Institute of Science, Bangalore.

[11] Laxman, S., Sastry, P.S. and Unnikrishnan, K. (2005) Discovering Frequent Episodes and Learning Hidden Markov Models: A Formal Connection. *IEEE Transactions on Knowledge and Data Engineering*, **17**, 1505-1517.

[12] Laxman, S., Sastry, P.S. and Unnikrishnan, K.P. (2007) A Fast Algorithm for Finding Frequent Episodes in Event Streams. *Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM Press, San Jose, 410-419.