Constructing Comprehensive Summaries of Large Event Sequences

建立大的事件序列的综合概括

摘要

在一段时间内发生的系统和用户的活动为事件序列。之前关于序列挖掘的研究主要集中在探索本地模式，尽管这些模式揭露了本地的关联，但是无法给出整个事件序列的综合概括。另外，我们发现的序列模式的规模很大。在这篇论文中，我们采取另一种方式，就是实现建立简短的摘要，在揭示事件之间的本地联系的同时，能够描述整个序列。

形式上，我们将此问题称为维持概要长度和保证数据准确的平衡的最优化问题。通过两种动态规划算法的结合，我们可以在多项式时间内解决此问题。我们还研究了更加有效的贪心算法，并且证明了它适用于大型的数据集。人造数据集和真实数据集的测试表明，我们的算法是有效的，并且有着高质量的结果，同时揭露了数据中的有趣的本地结构。

Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications—

Data mining; I.5.3 [Pattern Recognition]: Clustering—

Algorithms; E.4 [Coding and Information Theory]: [Data

Compaction and compression]

General Terms

Algorithms, Experimentation, Theory

关键字

时间序列，概括，日志挖掘

1. 介绍

监控系统和用户的活动产生了大量的事件序列，即记录其中每个事件发生的时间关联。网络流量数据、电信网络的报警、记录系统，都是产生了大量事件序列的例子。现有的关于事件序列的数据挖掘方法，虽然在寻找当地的复合结构是成功的，例如片段，但是可以证明，并不能提供数据的全局模型。另外，数据挖掘的算法通常输出太多的模型，超过数据分析的能力。在这篇文章里，我们将对事件序列的分析提供了一个新的视角，这称为如何简明地概括事件序列。

批注 [vx1]: 原文是 episodes
不造怎么翻额。。。

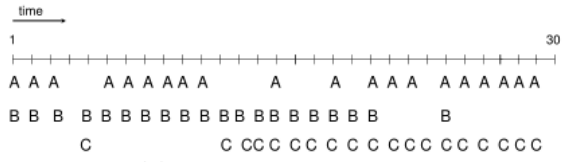
从一个事件分析师的角度来看，一个事件序列概括系统应该有以下几个属性：

- 简短并精确：概括系统应该有短小的概要，并且能精确地描述输入数据。
- 全局的数据描述：得到的概要应该在整个时间段内，事件序列全局结构的表述和它的变化。
- 鉴别本地模型：得到的概要应该揭示关于本地模型的信息；不论是正常的还是可疑的事件，或者是发生在某一个时间点的组合事件，在概要里应该要表示出来，能容易的辨别。
- 没有参数：概要是要给出信息和有用的结果的，基于这个目的的分析不需要多余的调整。

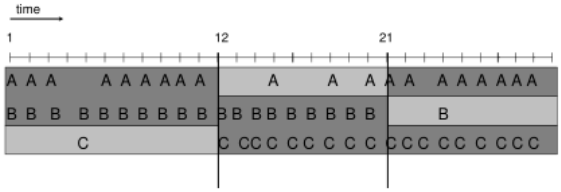
批注 [vx2]: The summaries should give an indication of the global structure of the event sequence and its evolution through time.

据我们所知，对于以上讨论的要求，从在事件序列的分析上做的大部分的工作来看，并没有能够满足的技术。在这篇论文中，我们呈现出一个概括技术，可以展示以上所有的特征。具体而言，

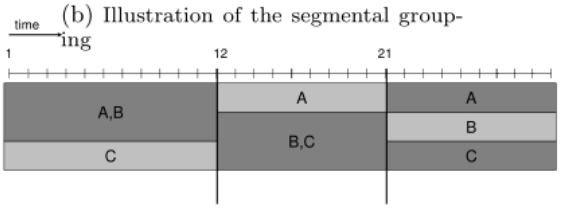
- 为了找到概要长度和描述准确性之间的平衡，我们使用最小描述长度原则（Minimum Description Length（MDL））。
- 我们采用分割模型（segmentation model），通过辨别时间轴上整体的时间间隔，提供一个高度可视化的序列。每个区间出现的事件表现出本区间的规律。
- 在分割后的时间轴上，每一个区间上的事件，类似于群，被一个本地模型描述。我们的本地模型把在时间段内出现频率类似的事件类型聚集起来，这样就能捕获到事件类型之间的本地联系。
- MDL 的使用惩罚过拟合高估数据的复杂模型和一概而论的简单模型。



(a) Input event sequence



(b) Illustration of the segmental grouping



(c) **Output summary**

图 1：一个事件序列的可视化表示，包含三种事件类型 {A, B, C}，时间轴[1, 30]。 图 1（a）表示输入序列；图 1（b）表示区间分组；图 1（c）表示概要的高度可视化。同样色调的灰色对应一组。

例 1. 图 1 给出了一个输入事件序列的例子和这个序列在我们的方法下的输出。输入序列如图 1（a）所示，包含三种事件类型{A, B, C}，时间轴[1, 30]，有 30 个离散的时间戳。图 1（b）给出了我们的方法找出来的实际的分割小组。三个区间被辨别出来：[1, 11]，[12, 20]，[21, 30]。在每个区间内，这些事件被分成两组，有类似发生频率的事件类型被聚在一块：第一个区间，两个组分别有时间类型{A, B}和{C}—A 和 B 在区间[1, 11]发生的频率比 C 高得多。类似地第二个区间的分组是{A}和{B, C}，第三个区间是{A, C}和{B}.

最后，图 1（c）显示，从概念上看这种概括方式的输出结果是什么样的。一个区间内小组的颜色表示了这个组内时间的发生率：颜色深的对应发生频率高。

1.1 问题表述和方法

我们解决以下问题：假定一个事件序列 S 是在一时间区间[1, n]记录事件的发生的。另外，用$\mathcal{E}$表示序列内的不同事件类型。我们的目标是把观察区间分成几个区间，区间内的本地活动横跨[1, n]，然后在每个区间内，根据在这个区间内的发生频率进行辨别。我们将这种数据描述模型成为段分组（segmental grouping）。目前为止，针对这篇论文的目标，我们只考虑了时间轴的离散化。我们另外假设不同类型的事件在每一个不同的时间戳上发生是独立的，与基于事件类型的平稳概率和区间本身是独立的。

我们形式上将找到最有段分组的问题称为最优化问题。通过惩罚复杂和简单的模型，我们研究出一种没有参数的方法，用多项式时间算法最佳地解决了上面的概括问题。动态规划是这些最优算法的核心。算法的计算复杂度仅与实践发生的时间戳的数量有关，而不是时间戳的总数量。

尽管我们的研究的主要目的是辩证地分析大型的审查日志，我们还推测此中展现的技术还能够用于其它不同的领域。比如说，用我们的方法，可以为生物数据、数据流和大型文档建立有用的摘要。

1.2 路线图

余下的论文是这么组织的：在第 2 部分，给出一些基本的符号规定。在第 3 部分，在形式上描述了概括的流程和对应的寻找最好概要的最优化问题。解决问题的算法将在第 4 部分提到，实验将在第 5 部分给出。我们在第 6 部分回顾相关工作并在第 7 部分作总结。

2. 准备工作

事件序列包含发生在具体某个时间点的事件。序列里的每个事件都有一个相关的出现时间。我们假设一个集合 $\mathcal{E}$ 有m 个不同的事件类型。一个事件这样表示，(E, t)，其中，$E \in \mathcal{E}$，E是一种事件类型，t是该事件在时间轴上的发生时间。考虑到事件发生的离散的时间轴是区间$[1, n]$内的正数，那么时间轴就有n个不同相空的时间戳，每个时间戳都有可能发生不同类型的事件。

我们用一个$m \times n$ 的矩阵 S 表示一个事件序列，$S(i,t) = 1$ 表示事件E_i发生在时间点t. 在一个特定的时间t，不同类型的事件可以同时发生。Name 矩阵 S 的每一列可能不止有一个 1 输入。但是，在任一时间t，同一个事件类型下仅有一个事件能够发生（如果多个同类型时间发生在一个时间t，会被视为副本二忽略）。

图 1（a）显示了一个序列S，有$m = 3$ 个事件类型，$\mathcal{E} = \{A, B, C\}$. 这些事件发生在时间轴[1, 30]. 那么就有 30 个时间戳，在时间戳上三个时间里任意一个可能发生。

给定一个区间$I \subseteq [1, n]$，用$S[I]$表示 S 在区间 I 的 $m \times |I|$ 的规划。最后，对于事件类型$E \in \mathcal{E}$ 和区间$I \subseteq [1, n]$，表示成事件E 的在区间 I 的出现次数，用$n(E, I)$ 表示。

核心思想是找到一种分割能把输入的时间轴$[1, n]$ 的分割成连续的、没有重叠的、概括$[1, n]$的区间。我们把这些区间叫做段。再者，我们想要把S进行分割表示成$S = (S_1, \ldots, S_k)$. 这样一个分割就被定义为$k + 1$个边界$\{b_1, b_2, \ldots, b_k, b_{k+1}\}$，其中$b_1 = 1$，$b_{k+1} = n + 1$，每一个$b_j, 2 \leq j \leq k$ 在 $[2, n]$ 内取整数。所以，第 j 个段对应子区间$S[b_j, b_{j+1} - 1]$. 图 1(a) 的一种对输入事件序列的分割在图 1(b)中表现出来。输入序列被分割成三部分，用边界定义则为$\{1, 12, 21, 31\}$.

现在看定义在时间区间I 的具体的段S_i ，那么$S_i = S[I]$. 我们用模型M_i 描述对应到S_i的部分数据。将模型M_i 设想为把事件类型 $\mathcal{E}$ 分成组$\{X_{i1}, \ldots, X_{il}\}$，其中$X_{ij} \subseteq \mathcal{E}$ ，$X_{ij} \cap X_{ij'} = \emptyset\ (j \neq j', 1 \leq j, j' \leq l)$. 每一组$X_{ij}$用一个单独的参数$p(X_{ij})$描述，表示在数据段$S_i$能够看到$X_{ij}$任何一个类型的事件发生的概率。

考虑图 1(b)（或者图 1(c)）的输出分割的第一段，定义了段$I_1 = [1,11]$，长度为$|I_1| = 11$. 在这个例子里，$S_1 = S[I_1]$里的数据划分成几个组$X_{11} = \{A, B\}, X_{12} = \{C\}$. 本地模型$M_i$是这样描述的

$$p(X_{11}) = \frac{1}{2}\frac{n(A, I_1) + n(B, I_1)}{|I_1|} = \frac{19}{22}$$
$$p(X_{12}) = \frac{n(C, I_1)}{|I_1|} = \frac{1}{11}$$

概括问题。 我们的总体目标是要辨别出集合在时间轴上的的边界，将S 划分成段$(S_1, \dots, S_k)$，在每个段 S_i 识别出一个本地模型M_i来最好地描述S_i中的数据。

划分S成段$(S_1, \dots, S_k)$和对应的模型$M_1, \dots, M_k$，构成了段分组，或者说S的概要。以下的讨论，会交换地使用属于概要和段分组。

为了能够设计出针对概括问题的算法，对于没有正式的问题定义的目标，我们首先需要定义它的最优化方程。我们的最优化方程是根据最小描述长度（MDL）原理产生的。

3. 时间序列的分割模型

在正式建立模型之前，先回顾下最小描述长度（MDL）原理。下面展示如何把 MDL 原理应用于使概括问题形式化。

3.1 最小描述长度原理

MDL 原理[15, 16]让我们能够把过度概括和过拟合之间的平衡要求转化为一个计算要求。

简而言之，MDL 原则有如下表述：假设P'和P两方想要进行交流，假设P 想要用尽可能少的位把事件序列S 发送给P'。为了让P实现最少的交流开销，她想从一系列的模型$\mathcal{M}$中选择模型M，用模型M描述她的数据。然后她就能把模型M和在给定传输模型下用来描述数据所需的其它信息发给P'。

因此，P需要对模型M进行编码，并在此前提下对数据编码。我们根据模型和相应数据的总体编码所需的位数来评估所选模型的质量。

MDL 避免了用小数据开销的复杂模型和大数据开销的简单模型。它试着在两个极端之间寻找平衡。很明显 MDL 是一种一般化的原理，可以由一系列的模型假设实例化。之前已经有过将 MDL 成功用于各种设置，有决策树分类器[11]，遗传序列建模[12]，几套字符串模式，等等。我们用剩下的部分描述 MDL 原理的实现。

3.2 编码规划

回想我们用分割模型（把输入的观察区间$[1, n]$ 划分为连续的、不重叠的区间$I_1, \ldots, I_k$）对事件序列建模。所以，S 被分割成$(S_1, \ldots, S_k)$，其中$S_i = S[I_i]$. 每一段S_i 里的数据用模型M_i描述；本地模型实际上是一群事件类型，基于他们在S_i 的发生频率。

本地编码规划：开始，我们要描述一个评估流程：在一个单独的段S_i 内对数据进行编码所需要的位数 。

用模型M_i把S_i（相当于所有正在或不在S_i里的事件类型）的列划分成l组 $X_1, \dots, X_l$. 每一组X_j用一个参数$p(X_j)$描述，表示小组X_j里包含在S_i里的任一事件类型的发生率。给定X_j和 $p(X_j)$ $(1 \le j \le l)$的位数，假设事件和事件类型发生的独立性，数据S_i的可能性用模型 M_i给出：

$$\Pr(S_i|M_i) = \prod_{j=1}^{l} \prod_{E \in X_j} p(X_j)^{n(E,I)} (1 - p(X_j))^{|I| - n(E,I)}.$$

在模型M_i条件下描述数据 S_i 的所需位数为 $-\log(\Pr(S_i|M_i))$[1]. 所以由M_i描述的S_i的本地数据开销为

$$L_D(S_i|M_i) = -\log(\Pr(S_i|M_i)) \qquad (1)$$

$$= -\sum_{j=1}^{l} \sum_{E \in X_j} (n(E,I) \log p(X_j) + (|I| - n(E,I)) \log(1 - p(X_j)))$$

等式（1）给出了在模型M_i条件下描述数据 S_i 的所需位数。对于S_i的编码，我们也要计算用来编码模型M_i需要的位数，称为本地模型开销$L_M(M_i)$. 为了对M_i进行编码，我们需要描述关联于每个小组$X_j (1 \le j \le l)$的事件类型，以及对于每一个X_j我们需要详细说明参数 $p(X_j)$. 根据标准观点[15]，需要$log\ m$位数描述每一个$p(X_j)$. 由于我们有l组，需要$l\ log\ m$位数来对l个不一样的$p(X_j)$的位数进行编码。划分的编码有一点狡猾。首先注意到如果我们固定事件类型的顺序$X_1, \dots, X_l$[2]，那么我们需要$m \log m$个位数来详细描述这个顺序，也需要$l\ log\ m$ 个位数来区分在固定序列上的l个划分点。这是因为对于这个固定的顺序，它的划分点是$[1, m]$内的整数，所以就需要$\log m$个位数描述每一个划分点。综上，得到M_i的本地模型开销

$$L_M(M_i) = 2l \log m + m \log m . \qquad (2)$$

所以，描述一个段S_i的总的本地花费（单位为位）就是在给定模型M_i 的条件下描述S_i 的位数和描述模型M_i 自己的位数。把等式（1）和（2）加起来，得到本地花费大约为

$$L_L(S_i, M_i) = L_D(S_i|M_i) + L_M(M_i). \quad (3)$$

生产模型：上述的编程流程呈现出了如下的数据生成过程：在段S_i里，不同类型的事件是独立产生的。对于任意一种事件类型$E \in X_j\ (1 \le j \le l)$，一个类型$E$的事件在每个事件点$t \in I$ 是等概率发生的。

批注 [vx3]: Generative model

全局编码流程：全局编码模型指的是部分模型M（把S 分割成段$(S_1, \dots, S_k)$）；每一段用边界标明，并且对应本地模型M_i . 如果对于每一段i，S_i里的数据用上述的编码流程描述，其他的信息就只需对在时间轴$[1, n]$定义了段的起点的边界进行编码。由于由n 个可能得边界位置，k个段边界的编码需要 $k \log n$ 个位数。所以，描述的总长度（单位：位（bits））是

$$T_L(S, M) = k \log n + \sum_{i=1}^{k} L_L(S_i, M_i)$$

其中，$L_L(S_i, M_i)$用等式（3）估算。

3.3 重申问题定义

对于概述问题，我们已经准备好给出正式的定义。

问题 1. （概述）给定观察时间段内$[1, n]$的事件序列S，其中发生的事件类型是集合$\mathcal{E}$里的， 找出整数k和一个把S 划分成$(S_1, \ldots, S_k)$的划分群M，并且对于每一段S_i辨别出最好的本地模型 M_i，那么最小的描述总长度是

$$\mathrm{T}_L(S, M) = k \log n + \sum_{i=1}^{k} \mathrm{L}_L(\mathrm{S}_i, M_i) \text{ ，} \qquad (4)$$

问题 1 给出了一个最优化方程，其中包含数据编码和模型编码所需的位数。注意，总的模型的花费可以分解为全局分割模型编码的花费$k \log n$ 加上用等式（2）估算的不同本地模型的编码花费。给定模型下数据编码的花费就是每一段的本地数据花费的总和。

我们用$\mathrm{L}_L{}^*(\mathrm{S}_i)$ 表示所有可用的本地模型M_i 中 $\mathrm{L}_L(\mathrm{S}_i, M_i)$的最小值。类似地，我们用$\mathrm{T}_L{}^*(S)$表示所有可能的概括$M$下$\mathrm{T}_L(S, M)$.

由于最优化方程的定义是在 MDL 原理的条件下建立的，这个方程满足：（a）复杂的概述会被惩罚，因为他们过度匹配数据了；（b）简单的概述会被惩罚，因为他们过度概括而无法在预计的准确性下描述数据。另外，用 MDL 原理可以实现无参数的问题公式化。分析师在尝试从输入序列S 中提取资料，就需要没有参数的设置。

4. 算法

尽管本地模型选取和时间轴上段的边界位置有明显的相互影响，事实上，问题 1 可以在多项式时间内最优化解决。

给定数据段S_i，问题辨别本地模型使得$L_L(S_i, M_i)$最小，我们可以称之为本地分组问题，在形式上我们将进行如下的定义：

问题 2. （本地分类）给定序列S 和区间$I \subseteq [1, n]$，找到最优本地模型M_i 使得在模型M_i下$S_i = S[I]$的本地描述长度最小。也就是找到M_i 使得

$$M_i = \arg\min_{M_i'} L_L(S_i, M_i') = \arg\min_{M_i'}(L_D(S_i|M_i') + L_M(M_i'))$$

我们的算法的解决方法采用如下表述。

表述 1. 如果本地分组模型（问题 2）能够最优在多项式时间内解决，那么概述问题（问题 1）也就能最优在多项式时间内解决。

这部分的余下内容里，我们针对概述问题给出最优多项式时间的算法。算法采用如上的表述。此外，我们对于概述问题提供了一个次佳的但是实用、高效的算法。

4.1 寻找最优全局模型

首先，对于概述问题，我们展示一个最优动态规划算法。我们也要表明不是区间$[1, n]$所有的可能的分段都有可能作为概述问题的解决方案。

理论 1. 对于$I \subseteq [1, n]$，令$L_L^*(S[I]) = \min_{M_i} L_L(S[I], M_i)$. 那么通过用以下的动态规划的递归方程估算，问题 1 就能最优地解决了。对于$1 \le i \le n$

$$T_L^*(S[1, i]) = \min_{1 \le j \le i}\{T_L^*(S[1, j]) + L_L^*(S[j + 1, i])\}. \quad (5)$$

由于空间限制，省略了最优化的证明。但是一个类似的证明在[2]中可以找到。我们称实现递归方程（5）的动态规划算法为区间-DP 算法。 如果需要时间T_L来估算$\mathrm{L_L}^*(\mathrm{S}[I])$，那么区间-DP 的运行时间为$O(n^2T_L)$.

不是区间$[1,n]$上所有的点都满足最优分割的段边界。事实上，只有一个事件发生的事件戳才有可能成为段的边界。下面的命题总结了这个事实。

命题 1. 考虑跨越区间$[1,n]$事件序列S，令$T \subseteq \{1,2,\ldots,n\}$ 为有事件发生的时间戳，那么最优分割模型的段边界是集合T的子集。

命题 1 提供了一个优化了区间-DP 算法，时间从$O(n^2T_L)$ 变为 $O(|T|^2T_L)$，其中$|T| \leq n$. 那么，递归方程（5）的估算不需要遍历$\{1,\ldots,n\}$, 而是遍T（事件发生的时间点的集合）里的点。尽管就渐进运行时间而言，命题 1 没有提供优化，但是实际上，有很多真实数据里，$|\mathrm{T}| \ll n$. 所以命题 1 还是非常有用的。在我们应用于真实数据集的实验揭示了这个事实。

4.2 贪心算法

这里的贪心算法是一中可以替代段 DP，并能自底向上求解 S 中 M 的和。本算法从 M1 的和开始，所有的数据点在我们的段中。在算法的第 t 步，我们定义 b 为M_t的边界，b 的移除会使$T_L(S,M^t)$最大。通过移除边界$T_L(S,M^t)$. 同时移除边界 b 我们获得M^{t+1}的和。如果没有边界使得最终结果最小，我们 的算法就输出M^t。

因为在算法中最多有 n−1 个迭代器可以移除，所以这里最多有 n−1 个候选边界。每个迭代器的边界总是可以发现最大的花费。通过堆这一数据结构我们可以在 O(1)的时间内解决。

迭代器 t 在堆中的记录是M^t的边界。边界的集合是 { b_1, b_2... b_l } 。每个记录b_j与G(b_j)相关，并来自M^t。这个记录也许是正面的在$T_L(S, M^t)$增长或者描述的长度减小。对于每个迭代器 t 的点b_j有

$$\begin{aligned} G(b_j) \quad = \quad & \mathrm{LL}^*\left(\mathbf{S}\left[b_{j-1}, b_{j+1}-1\right]\right)+\log n \\ & -\mathrm{LL}^*\left(\mathbf{S}\left[b_{j-1}, b_j-1\right]\right)-\log n \\ & -\mathrm{LL}^*\left(\mathbf{S}\left[b_j, b_{j+1}-1\right]\right)-\log n . \end{aligned}$$

上述等式的第一行中左面的等于数据 S 在减掉b_j和合并$[b_{j-1}, b_j]$在单一的段中。负的成本对应于相同的两段b_j和 $[b_{j-1}, b_j]$。迭代器b_j被更新时，b_{j+1}, b_{j-1}同时要被更新。在右边的簿记中 L_L*是对于两次更新间隔的差异，因此是 O(2T_L)。此外每次迭代都需要 O(logn)的时间，因此总的运行时间是 O($T_L$nlogn)，命题 1 加上后的运行时间是 O($T_l|T|\log|T|$)。

4.3 寻找最优本地模型

在本节中我们讲使用本地群(LocalGrouping) 同样用本地 DP 算法能多项式时间内很好的解决。我们成这个算法为本地 DP 算法。下面的命题讲关心本地 DP 算法。

命题 2.考虑到区间 I 并使 Si=S[i]。不失一般性假设在 Ɛ 中的事件被排序为 n(E1, I) ≥ n(E2, I) ≥ . . . ≥ n(Em, I) ，假设局部最优的模型 Mi 的结构是 X1, . . . , Xl. 然后我们有：如果 Ej1 ∈ Xl 和 Ej2 ∈ Xl 在 j2 >j1 的情况下，然后我们有对于所有的E_j,有 j0 ∈ {j1 + 1, . . . , j2 − 1} 然后我们有 Ej0 ∈ Xl。

命题 2. 关于群在事件类型在区间 I 重视事件类型并重视他们的频率出现在 S[i]。

下面一段命题是寻找最优的 p（Xj)使得最小化 L_D对于模型M_i在 ε 分割 $x_1 \ \dots \ x_n$ 尽可能的简单。更具体的是 p（Xj)的值是在区间 I 内 E ∈ Xj，出现概率的平均值。

命题 3，假设区间 I ⊆ [1, n], 并且对于数据$S_i = S[I]$有本地模型 M_i。使得M_i有 ε 分割 $x_1 \ \dots \ x_n$。然后，对于每一个M_j 在 $1 \le j \le l$ 的时候 $\mathrm{p}(X_j)$ 有最小的 $L_D(S_i|M_i)$

$\mathrm{P}(X_j) = \frac{1}{|X_j|}\sum_{E\in X_j}\frac{n(E,I)}{|I|}$

在本节的其余命题中，我们讲假定时间在 ε 区间是按照命题 2 中描述进行排列的。鉴于此排列用 ε(j) 来表示在事件排列顺序中第 j 个事件，同时 ε(j , l)表示 j , j+1 ... j + l 的集合。更进一步的，对于事件段 S_i 我们用 $S_i(j,l)$ 来表示 S_i的子集与 ε(j , l) 一致的地方。

$$\begin{aligned} &\mathrm{LL}^*\left(\mathbf{S}_i[1,j]\right) = m\log m + \qquad (6)\\ &\min_{1\le l\le j}\ \left\{\mathrm{LL}^*\left(\mathbf{S}_i[1,l]\right) + \mathrm{U}\left(\mathbf{S}_i[l+1,j]\right) + 2\log m\right\}, \end{aligned}$$

这里的

$$\begin{aligned} &\mathrm{U}\left(\mathbf{S}_i[l+1,j]\right) = \\ &= -\sum_{E\in\mathcal{E}(l+1,j)} n\left(E,I_i\right)\log p^* \\ &\quad -\sum_{E\in\mathcal{E}(l+1,j)} \left(|I| - n\left(E,I\right)\right)\log(1-p^*), \end{aligned}$$

同时在命题 3 中 p^* 如下式子

$p^* = \sum_{E\in E(l+1,j)}\frac{n(E,I)}{|n|}$

对于 mlogm 条款的递归相当于花费了编码排序事件类型 Si，而条款 2logm 编码的数字的比特数就是发生的概率对任一时间在群 E (l + 1, j)和他本身中。需要注意的是事件类型的顺序每段需要被发送一次，而每个组和群体出现的概率信息需要每组发送一次

定理 2．本地 DP 算法对数据段 Si 的递归估值发现好的模型是在多项式时间。

本地 DP 算法的时间复杂度是 O(m^2) 每一个下标 j 递归的值都属于 l 属于 1 ≤ l ≤ j ，由于 j 最大值是 m，所以时间复杂度最大是 O(m^2) 。在这个二次运行时间的假设下，

在预处理的步骤后我们能计算出 U()的值对于所有的 j , l 组合。事实上渐进的期限 O(m^2)同样包括隐藏排序事件类型的花费在基于频率出现在S_i的 E，这样复杂度为 O(m$\log m$)。

需要注意类似于 4.1 解的命题 1 和 4.1 节同样可以被用在这里。这意味着，事件类型在递归是不发生在S_i可以被忽略。

4.4 本地贪心算法

类似于在$[1, n]$寻找最优段边界的贪心算法（详见 4.2），在这里我们给出一个代替 Local-DP 算法的贪心算法，称为本地贪心（LocalGreedy）算法。通过使用和 4.2 节里同样的数据结构，LocalGreedy 算法的运行时间是$O(m\ \log m)$.

作为贪心算法，LocalGreedy 以一种由底向上的方式，计算了S_i 的全局划分X. 它始于分组 X^1，其中每个事件类型被分配到自己的组。在算法的第 t 步，会考虑到分组X^t，并且这个算法把两个小组合并，这两个小组引起了$L_L(S_i, M_i)$最大幅度的减小。这个合并引起了划分X^{t+1}. 如果没有合并，导致成本降低，这个算法就会停止并且输出划分X^t.

4.5 合并算法

区间 DP 算法和贪心算法都需要一个估算不同数据区间的L_L^*的方程。L_L^*的值可以用 Local-DP 算法或者 LocalGreedy 算法估算。这个设定为解决概述问题创建了四种不同的算法。DP-DP 算法包含区间 DP 和 Local-DP，DP-Greedy 算法包含区间 DP 和 LocalGreedy，Greedy-DP 算法包含了 Greedy 和 Local-DP，Greedy-Greedy 算法包含了 Greedy 和 LocalGreedy。DP-DP 给出了概述问题的最优解决方案。但是所有其他的算法也给出了高质量的结果，同时也给出了大量的计算上的优化。

就渐进运行时间而言，DP-DP 算法需要时间$O(n^2m^2)$，DP-Greedy 需要$O(n^2m\ \log m)$，Greedy-DP 需要$O(m^2n\ \log n)$，Greedy-Greedy 需要$O(nm\ \log n \log m)$.

5. 实验评估

在这部分，我们展示了在一些基于合成数据和真实数据的实验。这些实验估算的主要目标是表明我们针对概述问题探究出的四种算法（详见第 4 部分）能够提供高质量的结果。也就是说，我们还表明，没有实现最优的基于贪心的算法（DP-Greedy, Greedy-DP, Greedy- Greedy）用了接近于最优的位数对输入序列编码，同时产生有意义的概述。另外，相比于 DP-DP 算法，基于贪心的方法提供了大量的计算上的优化。

批注 [vx4]: 这个要翻？

我们的算法在版本 1.4.2 的 Java 上实现。实验环境是建立在 Windows XP SP 2 的工作平台，带有 3GHz 奔腾 4 处理器和 1GB 的内存。

我们用一个算法A计算了解决方法的质量，通过报告压缩比例$\mathrm{CR}(A)$，其中A 是这些算法的任意一个：DP-DP, DP-Greedy, Greedy-DP, Greedy- Greedy. 如果M_A是由算法A 对于输入S概述问题选取的概括，我们定义算法A 的压缩率为

批注 [vx5]: 这个要翻？

$$\mathrm{CR}(A) = \frac{T_L(\mathrm{S},\ M_A)}{T_L(\mathrm{S},\ M_{unit})} \qquad (7)$$

概括M_{unit} 是分别描述S 上每一件事的模型，这样一个模型有n 个段边界（每个时间戳为一个段）和每一段 m 个组，且对应于一个没有概述的模型。 根据定义，压缩率在[0, 1]之间，$\mathrm{CR}(A)$ 越小，通过A 实现的压缩程度越大。

5.1 合成数据实验

在这部分我们给出了一些基于合成实验的实验。这些实验的目标有三层。首先，是为了演示我们的算法能找到正确的模型用于数据生成；第二，为了表明它们压缩输入数据；第三，是为了展示另一种算法：贪心，尽管经证明它的最优表现在实际运用中不如 DP-DP 算法。

数据集：我们生成了如下的合成数据： 我们先固定了n ，观察区间的长度，m ，发生在序列内的不同事件类型数，k，我们在生成的事件序列里人工设置的段数。除了$\{0\}$ 和 $\{n+1\}$，我们从$\{2,\dots,n\}$中随机选择了其它 $k-1$ 个独立的段边界。这些边界定义了k 个段。在每个段 $I_i=[b_i,b_{i+1}]$，我们随机地选择要形成的组的数量。每个这样的组X_{ij}用参数$p(X_{ij})$描述，即为组X_{ij}每个事件在段I_i 中发生概率。$p(X_{ij})$的值通常在[0, 1]之间。

参数V被用于控制生成事件序列的干扰程度。当$V=0$时，对于每一个段I_i 和段内每一个X_{ij}，任何事件类型$E\in X_{ij}$ 在任一时间戳$t\in I_i$上是以概率$p(X_{ij})$独立生成的。当$V>0$，任何事件类型$E\in X_{ij}$在任一时间戳$t\in I_i$上生成的概率是由正态分布 $\mathcal{N}(p(X_{ij}),V)$取样而来的。

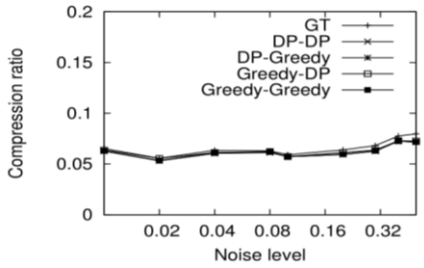


Figure 2: Synthetic datasets: $n = 1000$, $m = 20$, $k = 10$; **x-axis: noise level** $V \in \{0.01, 0.02, 0.04, 0.08, 0.1, 0.2, 0.3, 0.4, 0.5\}$, **y-axis: compression ratio for algorithms** `DP-DP`, `DP-Greedy`, `Greedy-DP` **and** `Greedy-Greedy`.

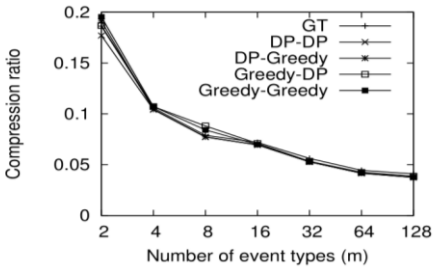


Figure 3: Synthetic datasets: $n = 1000$, $V = 0.04$, $k = 10$; **x-axis: number of event types** $m \in \{2, 4, 8, 16, 32, 64, 128\}$, **y-axis: compression ratio for algorithms** `DP-DP`, `DP-Greedy`, `Greedy-DP` **and** `Greedy-Greedy`.

算法的准确性： 图 2 展示了 4 种不同算法（DP-DP, DP-Greedy, Greedy-DP, Greedy-Greedy）的压缩率，作为提高干扰程度V（取值范围[0.01, 0.5]）的方程。 这次实验，我们固定$n = 1000, k = 10, m = |\mathcal{E}| = 20$. 除了我们的四个算法，我们还展示了地面真值模型（GT）的压缩率。这个模型在数据生成过程中使用过。从图 2 我们可以看出四种算法用很小的CR值，接近7%[3]，给出了概述。除此之外，这个压缩率和地面真值模型给出的压缩率和接近。事实上，对于干扰程度高（$V = 0.3, 0.4, 0.5$），相对于地面真值模型，我们的算法得到了较好的CR。这是因为对于高干扰程度，地面真值模型成为描述数据最佳模型的概

批注 [vx6]: 这个要翻？

率比较低。总体而言，就算是基于贪心的算法，就数据编码的位数而言，呈现出来的结果大多和 DP-DP 算法的一致。

图 3，展示了我们的算法的压缩率，作为序列里事件类型数m的函数。在这个实验里，我们对m取了不同的值$\{2, 4, 8, 16, 32, 128\}$，并且固定了数据生成过程中余下的参数，$n = 1000, k = 10, V = 0.04$. 因为在之前的实验里，经观察发现，我们的算法得到的压缩率和对应的地面真值模型得到的是一致的。此外，我们发现我们的算法都能得到同样的压缩率，所以可交替使用。注意，由于事件类型数的增加，不论是地面真值模型得到的压缩率，还是经由我们的算法发现的模型得到的压缩率，都会减小，也就是说会找到相对于原始数据比较的概述。这是因为数据里有越多的事件类型，我们的概述就会发现越多的本地模型，M_{unit} 在另一方面，是忽略了本地小组的存在的。结果，对于大量事件类型，式（7）的分母比分子增长地快得多。

5.2 真实数据实验

在这一部分，我们进一步阐述在真实生活中，我们的算法的实用性。通过使用 Windows XP 下托管的事件日志，我们再次展示了，四个算法为输入序列大大压缩数据和产生等效和直观的模型。

真实的数据集包括应用程序日志，安全日志和我们设备上 Windows XP 事件查看器 [4] 上显示的系统日志。这些应用程序日志在各个应用程序上记录着，安全日志记录事件，例如有效和无效的登录尝试，与资源使用有关的事件。最后，系统日志，包含由 Windows XP 系统组件记录的事件。我们使用存储三个日志文件中的每一个日志记录与以下字段：事件类型（Event_Type），日期（Date），时间（Time），来源（Source），目录

（Category），事件（Event），用户（User），和计算机（Computer）。我们将三个日志文件中的每一个输出为一个单独的文件，并单独处理它们。

我们的应用程序日志跨越期间 2007 年 6 月至 2007 年 11 月 ，安全日志跨越期间 2007 年 5 月至 2007 年 11 月，系统日志跨越期间 2005 年 11 月到 2007 年 11 月。对于所有这些文件，我们认为我们的电脑上发现的所有记录的事件，没有任何修饰。

	application	security	system
Observation Period	06/07-11/07	05/07 - 11/07	11/05-11/07
Observation Period (millisecs)	12,313,576,000	14,559,274,000	61,979,383,000
Number of events (N)	2673	7548	6579
Number of event types (m)	45	11	64
RUNNING TIMES (secs)			
DP-DP	3252	2185	34691
CP-Greedy	976	2373	8310
Greedy-DP	18	1	91
Greedy-Greedy	7	1	24
COMPRESSION RATIO CR(A)			
DP-DP	0.04	0.32	0.03
DP-Greedy	0.04	0.32	0.03
Greedy-DP	0.04	0.34	0.03
Greedy-Greedy	0.04	0.33	0.03

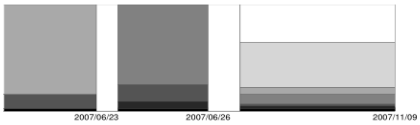
Table 1: Experiments with real datasets

把事件类型考虑成事件类型（Event_Type）、来源（Source）和事件（Event）的唯一的结合，把事件的时间戳视为日期（Data）和时间（Time）的结合，得到特征如表 1 所示的数据集。需要注意的是系统记录中的事件在一个毫秒粒度级别。所以，应用程序、安全、系统日志的时间轴的实际长度(n) 分别为$n = 12{,}313{,}576{,}000, n = 14{,}559{,}274{,}000$ 和 $n = 61{,}979{,}383{,}000$. 然而，这一事实不影响我们的算法的性能，由命题 1，算法性能仅依赖于事件真正发生的时间戳的数量N，三个数据集的N 分别为$N = 2673, N = 7548 \; and \; N = 6579$.

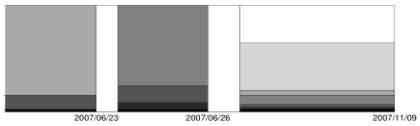
所用时间的计算都以秒为单位记录在表 1 中。例如，方法 DP-DP 在系统数据集上用的时间大约有 10 小时，这样，对于有大量时间类型的大数据集，这种方法是不实用的。我们看到对于同一组数据，Greedy-Greedy 算法跑了 24 秒。

最后，我们把四种不同算法，对于三种数据集，得到的压缩率，呈现在表 1 中。这个结果表明，基于贪心的方法产生的概述和 DP-DP 算法一样好。所以表 1 的结果进一步说明，尽管 DP-DP 是解决方法里最佳的，但是在处理大数据集上不实用。另一方面，基于贪心的算法给出了既准确有扼要的概述，而且在实际使用中更有效。

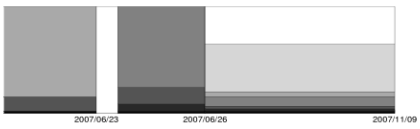
结果结构相似。我们观察发现，对于同一组数据，我们的算法都能够得到一致的压缩率。自然要问的问题是，是否对于实际的模型，他们的输出还是在结构相似。换而言之，是否报告出来的段分组有一样的段边界？是否报告的段内的小组是相似的？



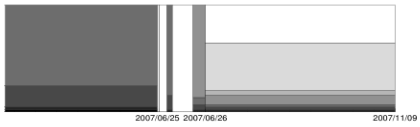
(a) DP-DP algorithm



(b) DP-Greedy algorithm



(c) Greedy-DP algorithm



(d) Greedy-Greedy algorithm

Figure 4: Output segmental groupings of different algorithms for the application log data.

图 4 的就是用一个对这个问题的肯定回答。这个图将报告里算法 DP-DP, DP-Greedy, Greedy-DP, Greedy- Greedy（分别为图 4(a), 4(b), 4(c), 4(d)）用于应用程序日志数据集的段分组的输出形象化。

批注 [vx7]: 这个要翻？

每一个子图对应一种不同算法的输出，并且应该这么解释：x 轴对应划分的时间轴，竖直的线定义了时间轴上段的边界。在每个段内，不同的组也能够不同的颜色显示（颜色深代表这个组的时间类型在这一段内更有可能发生）。每个组竖直方向的长度和它的大小成正比。从图 4，我们得出的主要结论是 DP-DP 和 DP-Greedy 算法输出的段分组时一致

的，四种算法在段内分组的时候输出相互非常近似。明显的相似之处，就是所有的分割在观察期间的初期都有一个很大范围的段，在结尾处有一个更大范围的段。在这些段内，可以看出小组的数量。在两个大范围段之间的区间，DP-DP，DP-Greedy，Greedy-DP 展示出类似的结构，可以看出他们的段边界几乎一致。Greedy-Greedy 算法得到的边界似乎是不一样的。但是，仔细观察发现，这些 Greedy-Greedy 算法得到的边界距离其他算法的边界并不远。Greedy-Greedy 经鉴别实际上和其他三种算法划分的边界非常相近。

6. 相关工作

虽然我们不知道任何工作提出相同事件序列的总结模型，我们的工作显然与序列挖掘和时间序列分析的工作重叠了。

和我们的工作类似的，是挖掘情节和序列模式的工作（[1, 3, 10, 13, 19]）。那种工作大多集中于就研究识别在集中时间内，离散事件的结构。尽管这些算法识别了本地事件模式，被称为频率事件，他们不能提供一个事件序列的全局描述，也不能关心产生模式的简洁性。

批注 [vx8]: episodes

通过一个分割模型表达的事件序列的概述在[9]中提出。但是其中提到的技术只能对单一的事件类型的序列建模。在每一个本地区间，事件的发生用一个恒定强度的模型建模。事实上，可以将我们的模型视为[9]提出来的模型的一般化，因为我们实际上是把事件类型分成几个有恒定强度的小组。

相关的还有[8]提出的分割框架，用于识别遗传序列里的大块的结构。这里也用了最小描述长度方法，来辨别段边界的数量和位置。但是，这些模型建立在每一

块，为所研究的基因序列的特定建模要求服务。例如，以[8]为例，在每个找到本地模型是一个 NP-hard 任务。

在一个高级别，我们的模型和用于时间序列分割标准的分割模型（详见[4, 5, 6, 18]，参考文献尽管不全但是具有象征意义）有一种明显的联系。类似地，有一条同样有趣的工作线，处理时间系列数据上本地模式的探索，例如[12, 17, 20]。但是，这种和我们工作的联系在一个很高的级别，因为我们关注事件序列为不是一系列的时间，同时我们考虑的本地模型与之前考虑的模型很不一样。同样程度的联系出现在了和 HMMs[14]的联系中。但是 HMMs 的假设与我们建立的模型假设是不一样的。

7. 总结

针对概括大量时间序列，并且这些序列连续不断的记录着系统和用户的活动，我们提供了一个框架和算法的解决方法。我们的框架是建立在构建数据的区间分组。区间分组是将时间轴分割成区间，在每个区间里，不同类型的事件基于区间内的事件频率组合在一起。我们的方法是建立在 MDL 原则上的，这样我们建立的概要短小精炼，能够没有赘余地准确地描述数据。

我们的贡献在于把区间分割定义为一个概括时间序列的模型，再结合 MDL 原则，我们就可以把事件序列的概括问题转化为具体的最优化问题。我们证明了，可以利用两种动态规划算法的结合，最佳情况下在多项式时间内解决这个问题。另外我们还针对同一个问题设计实验了贪心算法。虽然经过证明，这些算法并不是最优的，但是还是几只高效地在实际运

用中给出了高质量的结果。我们所有的算法都是没有参数的，在实际使用中会给出有意义的概要。

8. 参考文献