arXiv:2603.09166v1 [cs.DS] 10 Mar 2026

快速且最优的差分隐私频繁子串挖掘

PEAKER GUO,东京科学大学，日本R
AYNE HOLLAND,CSIRO 的 Data61，澳大利
亚HAO WU,滑铁卢大学，加拿大

给定一个包含 n 个用户贡献字符串的数据集，每个字符串的长度至多为 ℓ，一个关键问题是如何在保护每位用户隐私的同时识别所有频繁子串。Bernardini 等人近期的工作（PODS'25）提出了一种 ε-差分隐私算法，可实现近乎最优误差，但其空间和处理时间开销高达 $O(n^2\ell^4)$，令人难以承受。本文提出一种新的 ε-差分隐私算法，在保留相同近乎最优误差保证的同时，将空间复杂度降至 $O(n\ell + |\Sigma|)$，时间复杂度降至 $O(n\ell \log |\Sigma| + |\Sigma|)$，其中输入字母表为 Σ。我们的方法基于对候选子串的自顶向下探索，同时引入了两项创新：(i) 改进候选生成策略，利用频繁前缀和后缀的结构性质；(ii) 根据频率关系剪枝搜索空间。这些技术消除了先前工作中固有的二次增长，使差分隐私下的频繁子串挖掘能够实现可扩展处理。

CCS 概念：• **安全与隐私** → **隐私保护协议**；• **计算理论** → **算法设计与分析**。

附加关键词与短语：差分隐私、频繁子串挖掘、二叉树机制、后缀树

1 引言

现代数据驱动系统以语料库规模运行。基于用户生成文本训练的语言模型会学习常见且结构良好的表达，并抑制低概率组合。许多自然语言处理流水线会先从私有语料库中挖掘频繁子串，再将其用于下一词预测、自动补全或回复建议 [4, 7, 12, 13]。尽管这种做法在实践中取得了成功，却带来了显著的隐私风险：单个贡献者就可能注入罕见短语，泄露医疗状况、位置或其他个人信息，而模型可能会记住这些内容，并在之后将其呈现出来。

类似的隐忧也存在于自然语言之外，例如公共交通序列 [6] 和基因组字符串 [15], ，其中频繁子串分别对应路线或基序。由于这些语料库可与个人关联，朴素的子串挖掘可能会暴露敏感属性。因此，我们希望发现用于规划和分析的全局模式，同时不泄露任何特定用户是否走过某条特定路线，或是否携带某种特定遗传变异。

差分隐私提供了一种有原则的机制来控制这种风险。它保证可以学习并发布在语料库中出现频率足够高的子串，同时，任何单个用户的存在与否对输出的影响都微乎其微。这自然引出了*差分隐私频繁子串挖掘*问题：输入是一个包含敏感文档的数据库，目标是在保护每位用户的强隐私的同时，发布一组频繁子串。

由于该问题的重要性，相关研究由来已久 [1, 2, 5, 6, 15, 16, 18, 23]。在2025年，Bernardini 等人 [1] 首次对该问题进行了系统性理论研究。他们提出了一种具有严格加性误差保证的算法，并证明了匹配的下界，表明其结果除对数因子外达到了渐近最优。他们工作中的一项关键创新是采用自顶向下的方法，在按几何级数递增的

作者联系信息：Peaker Guo，东京科学大学，日本，peakerguo@gmail.com；Rayne Holland，CSIRO 的 Data61，澳大利亚，rayne.holland@data61.csiro.au；Hao Wu，滑铁卢大学，加拿大，hao.wu1@uwaterloo.ca。

参考文献	频率阈值	Time	空间
Bernardini 等人 [1]	$\Theta\left(\frac{\ell \ln \ell}{\varepsilon}\left(\ln^2 \frac{n\ell}{\beta} + \ln\lvert\Sigma\rvert\right)\right)$	$O(n^2\ell^4 + \lvert\Sigma\rvert)$	$O(n^2\ell^4)$
定理 4.1	$\Theta\left(\frac{\ell \ln^2 \ell_{\text{bit}}}{\varepsilon}\left(\ln^2 \frac{n\ell_{\text{bit}}}{\beta}\right)\right)$	$O(n\ell_{\text{bit}} + \lvert\Sigma\rvert)$	$O(n\ell + \lvert\Sigma\rvert)$

表1. 我们的结果与 Bernardini 等人 [1] 的先前工作比较。 $\ell_{\text{bit}} = \ell \cdot (\lceil \log \lvert\Sigma\rvert \rceil + 1)$。

子串长度上识别候选频繁子串。这既降低了该问题在相似数据集上的敏感度，也减少了因组合而产生的隐私成本。

然而，尽管理论意义重大，他们的方法在计算上代价高昂。对于一个包含 n 个长度为 ℓ 的字符串的数据集，他们的方法需要 $O(n^2\ell^4)$ 的处理时间和空间，在实际规模下难以实现。例如，Kim 等人 [16] 使用的 Reddit数据集[1] 有 $n \approx 10^6$ 个用户，每位用户贡献的字符串总长度为 $\ell \geq 3000$。这引出了以下研究问题：

研究问题：在保留渐近最优误差保证的同时，我们能否将处理时间和空间降低至近线性？

我们的贡献。 我们给出了肯定的回答，概述如下。

定理 *1.1*（定理 *4.1* 的非正式表述）。设 D 是一个由 n 个用户贡献的字符串组成的数据集，这些字符串取自字母表 Σ，且长度均不超过 $\ell \in \mathbb{N}_+$。设 $\varepsilon > 0$ 和 $\beta \in (0, 1)$。存在一种ε-差分隐私算法，以至少 $1 - \beta$ 的概率输出所有在 D 中频数至少为 τ_{T} 的子串，其中某个 $\tau_{\mathsf{T}} \in \tilde{O}\left(\frac{\ell}{\varepsilon}\right)$，同时使用 $O(n\ell \log \lvert\Sigma\rvert + \lvert\Sigma\rvert)$ 时间和 $O(n\ell + \lvert\Sigma\rvert)$ 空间。

与先前工作的比较 [1] 见表1。我们的方法之间存在权衡。我们的方案在频率估计中产生较大的加性误差，因此频繁子串的阈值也较高。主要区别在于，我们按照二进制字母表处理字符串，必要时会从输入字母表 Σ 转换而来。这会产生长度为 $\ell_{\text{bit}} \doteq \ell(\lceil \log \lvert\Sigma\rvert \rceil + 1)$ 的“更长”字符串，$\log \lvert\Sigma\rvert$ 因子因此会出现在加性误差中。

这种对字母表大小 $\lvert\Sigma\rvert$ 的依赖并非实际应用中的限制。在许多应用中，字母表本身就很小：例如，对于基因组数据，通常有 $\Sigma = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$。因此，与对 n 和 ℓ的依赖相比，$\lvert\Sigma\rvert$ 中的对数因子可以忽略，在许多现实场景中不会影响我们算法的可扩展性。

尽管误差界中存在这些对数因子，但我们的是最优的。此外，我们在性能成本上的降低是显著的，并将该问题推进到现实可行的规模。例如，我们的性能成本主要由二进制字母表“转换后”数据库的稀疏后缀树构建成本主导，这一成本在实践中是可接受的。值得注意的是，假设字母表大小为 $O(n\ell)$，我们的结果所需内存与数据集大小成正比。

对未对该问题进行理论处理的相关工作的补充讨论见附录B。

算法概述。 先前工作的主要瓶颈在于如何生成候选子串。例如，给定长度为 k的频繁子串集合 C_k，Bernardini 等人 [1]会测试每个拼接对 $s_1 \cdot s_2$ 的频率，其中 $s_1, s_2 \in \mathsf{C}_k$，以确定长度为 $2k$ 的频繁子串。这会导致每个阶段产生 $\lvert\mathsf{C}_k\rvert^2$ 个候选项，并使运行时间和内存开销都呈二次增长。

[1]频繁子串挖掘中的一个标准基准。

为缓解组合爆炸，我们提出了一种新颖的自顶向下算法，能够有策略地剪枝搜索空间。我们的出发点是这一观察：在已正确识别长度为 k的频繁子串集合 C_k 的前提下，任何长度为 $k+t$ 的更长频繁子串，其中 $t \in \{1,\dots,k\}$，都必须沿着一个出现在 C_k 中的后缀扩展 C 的某个元素。

利用这一结构，我们从 C_k 的后缀构建一棵紧凑字典树 T_k。随后，对于 $s \in \mathsf{C}_k$，我们遍历串联树 $s \circ T_k$，探索由此产生的候选项，其中 s 被视为一棵单路径树，其末节点连接到 T_k 的根节点。在这一自顶向下探索过程中，我们会剪去噪声频率估计低于预定义阈值的整个子树。由于真正频繁的子串必然由已有的频繁前缀延伸而来，这种剪枝在不影响正确性的情况下缩小了大部分搜索空间。由于我们在每次遍历中复用 T_k，每个阶段所耗费的时间与属于该阶段的频繁子串的总长度成正比。因此，总搜索成本与所有频繁子串的总长度成正比。

完整算法是 ε-差分隐私的，并依赖带加性误差 $\tilde{O}(\frac{\ell}{\varepsilon})$ 的噪声频率估计；该误差在多对数因子范围内是最优的。

组织结构。 第2 节介绍必要的定义，并形式化差分隐私频繁子串挖掘问题。第3 节回顾差分隐私以及本算法所用字符串数据结构的技术预备知识。在第4 节与第5节中，我们给出算法，并分析其隐私、效用与渐近代价。最后，第6节给出一些结语。

2 问题设定

在本　　节中，我们形式化定义差分隐私频繁子串挖掘问题　　.

2.1 记号

我们用 $\mathbb{N}$、 $\mathbb{Z}$和 $\mathbb{R}$ 分别表示自然数、整数和实数的集合，并用 $\mathbb{N}_+$ 表示正自然数，用 $\mathbb{R}_{\geq 0}$ 表示非负实数，用 $\mathbb{R}_+$ 表示正实数。对每个 $n \in \mathbb{N}$，我们定义 $[n] \doteq \{1,\dots,n\}$。

令 $\mathsf{D} \doteq \{\mathsf{S}_1, \dots\dots, \mathsf{S}_n\}$ 为一个由 n 个字符串组成的数据集，这些字符串取自 Σ，且每个字符串的长度至多为 $\ell \in \mathbb{N}_+$（即 $\mathsf{S}_i \in \Sigma^{1:\ell}$）。对于每个字符串 $P \in \Sigma^{1:\ell}$，我们将其在数据集 D 中的频率定义为$f_{\mathcal{D}}(P) \doteq \sum_{i=1}^{n} f_{\mathcal{S}_i}(P)$，其中 $f_{\mathcal{S}_i}(P)$ 表示 P在 S_i中的出现次数。我们将 $\Sigma^{1:\ell}$中所有字符串的频率视为向量 $f_{\mathcal{D}} \doteq (f_{\mathcal{D}}(P))_{P \in \Sigma^{1:\ell}}$.

2.2 隐私保证

在我们的设定中，数据集 D 由 n 个用户提供，其中字符串 S_i 属于第 i 个用户。为保护用户隐私，发布的频繁字符串必须不受任何单个字符串内容的影响，从而确保修改某位用户的字符串不会显著影响输出。

这一要求通过差分隐私（DP）的标准概念形式化。我们先回顾适用于任意随机算法的 DP 一般定义，随后再将其具体应用于本文研究的问题。

定义 2.1（ε-不可区分性）。设 $\varepsilon \in \mathbb{R}_{\geq 0}$，并设 $(\mathcal{Z}, \mathrm{F})$ 为一个可测空间。该空间上的两个概率测度 μ 和 ν 若满足以下条件，则称它们是 ε-不可区分的：

$$e^{-\varepsilon} \cdot \mu(E) \leq \nu(E) \leq e^{\varepsilon} \cdot \mu(E), \qquad \forall E \in \mathcal{F}. \tag{1}$$

同理，如果两个随机变量所诱导的分布 ε-不可区分，则称这两个随机变量　　*re.*

定义 2.2（ε-差分隐私 [9]）。设$\mathcal{M}: \mathcal{Y} \to \mathcal{Z}$ 为一个随机算法，且设 $\sim$ 是$\mathcal{Y}$上的对称邻接关系。对于参数 $\varepsilon \in \mathbb{R}_{\geq 0}$，我们称$\mathcal{M}$ 是

ε-差分隐私，如果对于每一对邻接元素 $y \sim y'$，M（y）和M（y'）的分布都是ε-不可区分的。

2.3 差分隐私频繁子串挖掘。

W我们首先正式定义子串被视为频繁的含义 .

定义2.3。 给定阈值 $\tau_\top \in \mathbb{R}_+$，字符串 $P \in \Sigma^{1:\ell}$ 若满足 $f_{\mathcal{D}}(P) \geq \tau_\top$，则称其为 $\tau_\top$-频繁 。类似地，给定阈值 $\tau_\perp \in \mathbb{R}_+$，若 P满足 $f_{\mathcal{D}}(P) \leq \tau_\perp$，则称其为 $\tau_\perp$-不频繁 。

定义2.4（容斥准则）。 给定数据集 $\mathcal{D}$，字符串集合 $C \subseteq \Sigma^{1:\ell}$在阈值 $\tau_\top \geq \tau_\perp \geq 0$ 下满足容斥准则 ，当且仅当：

(i) 每个 $\tau_\top$-频繁字符串 $P \in \Sigma^{1:\ell}$ 都包含在C中。*(ii)* 每个 $\tau_\perp$-不频繁字符串 $P \in \Sigma^{1:\ell}$ 都不包含在C中。我们称这样的集合C 为$(\tau_\top, \tau_\perp)$-IE 集。

该差分隐私频繁子串挖掘问题旨在识别一个$(\tau_\top, \tau_\perp)$-*IE* 集C，同时满足：

(1) 最小化处理时间和空间；(2) 最小化 $\tau_\top$，同时最大化 $\tau_\perp$；以及
(3) 确保算法执行过程中的 ε-差分隐私。

˜有时，我们还希望为每个 $P \in \mathsf{C}$ 输出其真实频率˜ ˜ $f_{\mathcal{D}}(P)$ 的近似值 $f_{\mathcal{D}}(P)$。通过为所有 $P \notin \mathsf{C}$ 设置 $f_{\mathcal{D}}(P) = 0$，向量 $f_{\mathcal{D}}$ 将成为完整频率向量 $f_{\mathcal{D}}$ 的稀疏近似，其中非零项至多有 $O(\sum_{i\in[n]}|\mathsf{S}_i|^2/\tau_\perp) \subseteq O(n\ell^2/\tau_\perp)$ 个。因此，该问题与发布差分隐私稀疏直方图这一经典任务密切相关。

此外，Bernardini 等人 [1] 提出了一种算法，其达到 $\tau_\top$，$\tau_\perp \in \Theta\big(\ell \frac{\ell}{\varepsilon} (\ln^2 \frac{n\ell}{\beta} + \ln|\Sigma|)\big)$，对于任意给定的 $\beta \in (0, 1)$，其成立概率至少为 $1-\beta$，并证明了一个下界，表明 $\tau_\top - \tau_\perp \in \Omega\big(\min\{n, \varepsilon^{-1}\ell\ln|\Sigma|\}\big)$.。由于 $\tau_\perp \geq 0$，这意味着 $\tau_\top \in \Omega\big(\min\{n, \varepsilon^{-1}\ell\ln|\Sigma|\}\big)$.。因此， $\tau_\top$ 和 $\tau_\perp$ 在其算法中均在对数因子范围内渐近紧致。然而，他们的方法需要 $O(n^2\ell^4)$ 的空间和处理时间。因此，主要目标是在保留相同渐近误差保证的同时，开发速度显著更快且空间效率更高的算法。

频繁子串挖掘的差分隐私。 为了将此定义应用于我们的问题，我们将 $\mathrm{M} : \mathrm{Y} \to \mathrm{Z}$ 解释为一种随机算法，它发布（可能带有噪声的）频繁子串。这里，Y 是所有可能数据集的空间，即 $\mathrm{Y} = (\Sigma^{1:\ell})^n$，而 Z 是 $\Sigma^{1:\ell}$ 中所有子集的集合。如果两个数据集 D，$\mathrm{D}' \in \mathrm{Y}$ 中恰有一个用户的字符串不同，我们就将它们视为相邻，也就是说，

$$\mathcal{D} \sim \mathcal{D}' \quad \text{iff} \quad \exists\, i \in [n] \text{ such that } \mathcal{S}_i \neq \mathcal{S}'_i \text{ and } \mathcal{S}_j = \mathcal{S}'_j \text{ for all } j \neq i.$$

频繁子串挖掘算法 ε-差分隐私，当且仅当它在此邻接关系下满足定义2.2 。

3 预备知识

本节将介绍差分隐私方面的技术预备知识，以及算法所依赖的字符串数据结构。

3.1 隐私

引理 3.1（组合 [8]）。设 $\mathcal{M}_1, \mathcal{M}_2$，……，$\mathcal{M}_k$为一列随机化算法，其中 $\mathcal{M}_1 : \mathcal{Y} \to \mathcal{Z}_1$,，且 $\mathcal{M}_i : \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_{i-1} \times \mathcal{Y} \to \mathcal{Z}_i$，对于 $i = 2$，……，k，即 $\mathcal{M}_i$ 可以接受

$\mathcal{M}_1$，……，$\mathcal{M}_{i-1}$ 的输出以及 $\mathcal{Y}$ 中的数据集作为输入。假设对于每个 $i \in [k]$ 以及每一个固定的 $a_1 \in \mathcal{Z}_1$，……，$a_{i-1} \in \mathcal{Z}_{i-1}$，随机映射 $\mathcal{M}_i(a_1, \ldots\ldots, a_{i-1}, \cdot)$ 是 ε_i-差分隐私的。则算法 $\mathcal{M}: \mathcal{Y} \to \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_k$（由依次运行这些 $\mathcal{M}_i$ 得到）对于 $\varepsilon = \sum_{i=1}^{k} \varepsilon_i$. 是 ε-差分隐私的。

引理 3.2（拉普拉斯噪声尾界）。令 $Z \sim$ 拉普拉斯分布(σ) 为尺度参数为 $\sigma > 0$ 的拉普拉斯随机变量。那么，对于任意 $\beta \in (0,1)$,

$$\Pr\left[|Z| > \sigma \ln \tfrac{1}{\beta}\right] \le \beta.$$

引理 3.3（拉普拉斯机制 [9]）。设 $\mathcal{Y}$ 是一个配备邻接关系的定义域 $\sim$，且令 $F: \mathcal{Y} \to \mathbb{R}^d$ 满足

$$\|F(y) - F(y')\|_1 \le \Delta \quad \textit{for all } y \sim y', \textit{ for some } \Delta > 0.$$

对于每个隐私参数 $\varepsilon > 0$，以下机制是 ε-差分隐私的：

$$\mathcal{M}(y) = F(y) + (Z_1, \ldots, Z_d), \quad \textit{where } Z_i \overset{iid}{\sim} \mathrm{Laplace}(\Delta/\varepsilon).$$

二叉树机制通过对 $\lceil \log T \rceil + 1$ 个二进区间的加噪计数求和，表示长度为 T 的序列上的每个前缀和。我们在每个二叉树节点上独立添加拉普拉斯噪声。下面总结了二叉树机制的性质。

引理 3.4（二叉树机制 [3, 10]）。固定 $\varepsilon_0 \in \mathbb{R}_+$，并令 $\sigma \doteq \varepsilon_0^{-1} \cdot \log d$。那么存在一个机制 $\mathcal{B}^\sigma$: $\mathbb{R}^d \to \mathbb{R}^d$，其参数为 σ，并具有以下保证：
▹隐私保证。对于所有 $x, x' \in \mathbb{R}^d$，输出 $\mathcal{B}^\sigma(x)$和 $\mathcal{B}^\sigma(x')$是 $\varepsilon_0 \cdot \|x - x'\|_1$- 不可区分的。▹效用保证。设 $x \in \mathbb{R}^d$ 和 $\beta \in (0,1)$。定义前缀和向量 $y \doteq \left(x_1, x_1 + x_2, \ldots, \sum_{i \in [d]} x_i\right)$. 那么

$$\Pr\left[\|\mathcal{B}^\sigma(x) - y\|_\infty > 2\sqrt{2} \cdot \sigma \cdot \ln \frac{2d}{\beta}\right] \le \beta.$$

▹流式实现。机制 $\mathcal{B}^\sigma$ 支持在线访问：对于每个 $t \in [d]$，收到 x的前 t个坐标后，它会输出 y的前 t个坐标。它使用 $O(\log d)$ 额外空间，并在 $O(\log d)$ 时间内计算 y的每个坐标。

3.2 后缀树

字符串。设 Σ 为一个字母表。字符串是 Σ^*的一个元素。对于字符串 S，其长度为 $|S|$，而$S[i]$表示其第 i^{th}个字符。对于整数 $l \le r$，令 $\Sigma^{l:r} \doteq \{S \in \Sigma^* \mid l \le |S| \le r\}$。从位置 i开始、到位置 j结束的 S 的子串记作 $S[i \ldots j]$。若子串 $S[1 \ldots j]$是 S的前缀，则称其为前缀；若 $S[i \ldots |S|]$ 是 S的后缀，则称其为后缀。令 $\mathrm{suffix}(S)$ 表示 S的所有后缀构成的集合。对于字符串多重集 D，令 $\mathrm{suffix}(\mathrm{D}) \doteq \cup_{S \in \mathrm{D}}$ 中各字符串的后缀集合 $\mathrm{suffix}(S)$。字符串 S和 T的连接为 $S \circ T$。对于两个字符串 P和 S，P在 S 中的一个出现位置是满足$S[i \ldots i + |P| - 1] = P$的位置 i。P在 S中的频率记作 $f_S(P)$，即P 在 S中出现的次数。

字典树。给定一个非空字符串多重集 $\mathrm{D} = \{S_1, \ldots S_k\} \subseteq \Sigma^*$，D 上的字典树 $T_{\mathcal{D}}$是一棵有根树，其中每条边都标记有一个来自 Σ 的单个字符；任一节点的出边标签互不相同；并且对于 D 中的每个字符串 S_i，都存在一条唯一的根到叶路径，其边标签的连接恰好拼出 S_i。对于 S_i的任意前缀 P，令 $\mathrm{node}(P)$ 表示从根出发的路径拼出 P的唯一节点。

后缀树。 对于 $S \in \Sigma^*$，后缀树 是对 S的所有后缀构成的字典树进行路径压缩后得到的结果，具体做法是将每条极大非分支路径收缩为一条边，其标签为该路径上各边标签的连接。它包含 $O(|S|)$ 个节点，可在 $O(|S|)$时间内构建，假设Σ 是一个多项式规模整数字母表 [11, 17, 20, 21]。对于后缀树中的节点 u，

▹ $\mathrm{str}(u)$ 表示其路径标签。▹ $\mathrm{parent}(u)$ 表示其父节点。▹对于符号 $c \in \Sigma$，$\mathrm{child}(u, c)$ 表示满足边 uv的标签以 c开头的子节点 v。如果不存在这样的 v，则 $\mathrm{child}(u, c) = \emptyset$。▹ $\mathrm{children}(u) \doteq \{(\mathrm{child}(u, c)\ c) : c \in \Sigma$且 $\mathrm{child}(u, c) \neq \emptyset\}$。

每个子串 P都可以通过从根节点出发，沿着 P中的字符遍历后缀树来定位。这样的遍历需要 $O(|P|)$ 时间，并会结束于唯一的位点（节点或边上的某个位置）。如果遍历在通向 u的边内结束，则 P等于 $\mathrm{str}(u)$ 中适当长度的前缀。 P 的频率等于以 u为根节点的子树中的叶节点数。由于位点位于同一条压缩边上的所有子串具有相同的频率，因此我们可以在分支节点处存储精确频率。这需要额外的 $O(|S|)$ 空间。

我们采用如下构造来存储一个字符串的等间距后缀子集。

定义 3.5（r-间隔稀疏后缀树）。 设 S 为一个字符串，并设 $r \geq 1$。 S的 r-间隔稀疏后缀树 是由所有起始下标 j 满足 $j \equiv 1$ （模 r）的后缀 $S[j \ldots |S|]$构建而成的路径压缩字典树。

以下结果给出了在字符串多重集的 r-间隔后缀上构建的 r-间隔稀疏后缀树的性质。

引理 3.6（[14]）。 对于 $r \geq 1$以及一个多重集$\{S_1, \ldots, S_n\}$，其中 $|S_i| \equiv 0$（模 r），令 $T = S_1\#_1^r S_2\#_2^r \cdots S_n\#_n^r$。则 T的 r-间隔稀疏后缀树可在 $O(|T|)$ 时间和$O(|T|/r)$ 空间内构造。

该结果要求字符串长度是 r 的倍数，从而将所有字符串中正确的（间隔为 r 的）后缀存储在树中。这也是字符串分隔符重复 r 次的原因。

对于 $\mathcal{D} = \{S_1, \ldots, S_n\}$，在构建 r 间隔后缀树时，对于 $T = S_1\#_1^r S_2\#_2^r \cdots S_n\#_n^r$，我们可以为每个显式节点 v添加一个频率计数器，用于存储 $f_{\mathcal{D}}(\mathtt{str}(v))$个以 v为根节点的子树中的叶节点数。这样做不会影响引理3.6中的渐近界。

3.3 重轻分解

有根树的重轻分解 将 T划分为边不相交的路径。具体而言，对于 T 中的非叶节点 u，设 v是 u 的子节点，且其以 v 为根的子树包含的节点数最多；我们称边 uv 为重 边，而对于 u 的其他任意子节点 w，称边 uw 为轻 边。分解定义如下。首先，从根节点到某个叶节点、且其中每条边都是重边的路径称为重路径。然后，对于每条满足 u 位于重路径上而 v 不在重路径上的边 uv，递归分解以 v 为根节点的子树。

引理 3.7（[19]）。若树 T有 m个节点，则可以在 $O(m)$ 时间内构造出 T的重轻分解，并且 T的每条根到叶路径至多包含 $\lceil \log m \rceil$ 条轻边。

直观来说，该引理成立，是因为遍历一条轻边会使当前子树的大小至少缩小一半。因此，根到叶路径在到达叶节点之前，最多会经历 $\lceil \log m \rceil$ 次这样的缩减。

4 算法

下面总结了我们算法的主要性质。

定理 4.1。 设 $\mathcal{D} = \{S_1, \ldots, S_n\}$ 是由 n 个用户贡献的字符串组成的数据集，这些字符串来自字母表 Σ，且每个字符串的长度至多为 $\ell \in \mathbb{N}_+$。设 $\varepsilon > 0, \beta \in (0, 1)$，且 $\ell_{\text{bit}} = \ell(\lceil \log |\Sigma| \rceil + 1)$，并令

$$\varepsilon_0 = \frac{\varepsilon / \log \ell}{4\ell \log(n\ell_{\text{bit}})} \quad \textit{and} \quad \tau^* = \varepsilon_0^{-1} \cdot (\log \ell_{\text{bit}}) \cdot \ln(n\ell_{\text{bit}}/\beta).$$

那么，对于阈值

$$\tau_\top = 9 \cdot \tau^* \quad \textit{and} \quad \tau_\perp \geq \ell(\log \ell_{\text{bit}})$$

存在一个 ε-差分隐私算法，它以至少 $1 - \beta$ 的概率输出集合 C，该集合满足 $(\tau_\top, \tau_\perp)$-容斥准则，同时使用 $O(n\ell_{\text{bit}} + |\Sigma|)$ 时间和 $O(n\ell + |\Sigma|)$空间。

在本节余下部分，我们将概述实现定理 4.1的算法。形式化分析将在第5节进行。

4.1 先前技术

在深入探讨算法的细节之前，我们先考察早期方法，以说明为何在大规模数据中以隐私保护方式挖掘频繁子串仍是一项具有挑战性的问题。

一种简单的方法是直接为 $\Sigma^{1:\ell}$ 中的所有字符串构建一个加噪的（ε-差分隐私）直方图。随后，我们仅输出其加噪频数超过预设阈值的字符串。然而，这种方法存在两项局限。首先，频率向量 $(f_{\mathcal{D}}(p))_{p \in \Sigma^{1:\ell}}$ 的敏感度为 $\Omega(\ell^2)$，因此要实现有效的隐私保护，就需要添加大量噪声。其次，由于字符串数量随字符串长度呈指数增长，这种方法在计算上不可行。

为缓解全集爆炸和问题敏感度，Bernardini、Bille、Gørtz 和 Steiner[1] 采用了一种自顶向下策略。从单个字符开始，如果某个候选子串的前缀的含噪计数超过阈值，他们就将搜索扩展到更长的候选子串。这样便能大幅剪枝指数级的搜索空间。然而，他们的方法仍受所考虑的候选子串数量限制。例如，给定由所有长度为 k 的 $\tau_\top$-频繁子串构成的集合 C_k，为了生成 C_{2k}，他们会穷举考虑 C_k 中子串的所有 $|\mathsf{C}_k|^2$ 种连接方式。这会导致运行时间和空间出现（难以承受的）二次增长。

此外，他们只考虑对数数量的子串长度对应的候选项，随后在其余字符串长度上插值，得到所有可能的频繁子串。这样做的好处是降低了搜索算法的敏感度，从而减少隐私噪声。然而，这一插值步骤也会导致运行时间和空间至少出现二次增长。

4.2 概述

为克服这些挑战，我们采用自顶向下搜索过程，只深入子空间中子串可能频繁的区域。然而，与 Bernardini 等人 [1], 相比，我们生成候选项时更加谨慎，只在真正需要的地方投入精力。

4.2.1 预处理。 我们使用二进制字母表处理 D 中的字符串。Σ 中的每个符号都会被转换为一个二进制码字，后接终止符 $\$ \notin \Sigma$。对于 $r = \lceil \log |\Sigma| \rceil + 1$，字符串长度会增加到 $\ell_{\text{bit}} = \ell r$。编码使用码本 E 完成：$\Sigma \to \{x\$ \mid x \in \{0,1\}^{r-1}\}$。

假设 Σ 中的每个字符可以存储在 $O(1)$ 个字中，则码本可以存储在直接访问数组中，占用 $O(|\Sigma|)$ 的空间。通过连接，码本可扩展到字符串。也就是说，对于字符串 $S = x_1 \cdots x_n$，$E(S) \doteq E(x_1) \cdots E(x_n)$。

每个编码后的字符末尾的终止符是必要的，它可以防止提取出未按字符对齐的频繁子串。例如，子串‘AC’（对应于 A,C $\in \Sigma$）可能会在 D 中频繁出现。然而，如果符号被编码为 E(A) = 0000 和 E(C) = 0101，那么如果未识别出 A 和 C 之间的边界，我们可能会错误地输出‘0001’。

此外，在二进制字母表上操作时，我们需要确保输出的子串定义明确。也就是说，我们希望避免输出无法正确解码的频繁子串。继续上面的例子，子串‘00$0101$’可能在编码数据集中频繁出现，但它无法解码为 Σ^* 中的子串。为了确保输出正确，我们只生成字符对齐 子串，其定义如下。

定义 4.2（字符对齐字符串）。设 $b \doteq \lceil \log |\Sigma| \rceil$ 且 $r \doteq b + 1$。固定一个单射块编码 $E\colon \Sigma \to \{x\$ \mid x \in \{0,1\}^b\}$，并通过连接将其扩展到字符串。长度为 k 的字符串 $P \in \{0, 1, \$\}^k$ 是字符对齐 的，当且仅当存在整数 $a \geq 0$ 和 $0 \leq c \leq r$，使得 $k = ar + c$ 和 $P[1 \ldots ar]$ 可以写成 $B_1 \cdots B_a$，其中每个 $B_i \in E(\Sigma)$。

注意，字符对齐子串中只有最后一个块可能是不完整的。例如，对于 $b = 2$，$r = 3$和 E，它们在图 2的说明文字中定义，$P = 01\$10\$1$ 是字符对齐的，因为存在 $a = 2$ 和 $c = 1$，使得 $7 = 2 \cdot 3 + 1$ 和 $P[1 \ldots 6] = 01\$ \cdot 10\$$。数据集的字符对齐子串具有如下敏感度。

引理 4.3。设 $r \doteq \lceil \log |\Sigma| \rceil + 1$。对于每个 $0 \leq k \leq \ell_{\text{bit}} = \ell r$，令 $\mathcal{A}_k$表示长度为-k字符对齐编码子串 $P \in${0、1、^{k}构成的族。那么，向量 $(f_{\mathcal{D}}(P))_{P \in \mathcal{A}_k}$的 L_1-敏感度有上界 2ℓ。

证明。更改 D 中的一条记录 S，只会影响那些源自 S 本身中出现的子串 P的计数 $f_{\mathcal{D}}(P)$。在二进制编码下，编码字符串 E(S) 的长度为ℓr。字符对齐子串只能从对应于原始字符边界的位置开始，即索引 1，$r+1$，$2r+1$，……，$(\ell-1)r-1$。因此，长度为 k的任意字符对齐字符串，在 S 中至多有 $\ell_{\text{bit}}/r - k + 1 \leq \ell$个可能的起始位置。由此，替换 S 至多会使 $f_{\mathcal{D}}(P)$ 中的 ℓ 个计数改变 +1，另有至多 ℓ 个计数改变 −1，总变化至多为 2ℓ L_1。□

4.2.2 频繁子串挖掘。该过程分为 $\lceil \log \ell \rceil$ 个阶段，并以阈值参数 $\tau_\perp$ 作为输入（参见定义 2.3）。设 $\Sigma_E \doteq \{x\$ \mid x \in \{0,1\}^{r-1}\}$。在阶段 i中，当 $k = r \cdot 2^{i-1}$时，算法会以隐私方式输出 $\Sigma_E^{(k+1):2k} \equiv \Sigma^{(k+1):2k}$中字符串对应的频繁子串集。该过程采用归纳方式，将已有的长度为 k 的频繁子串扩展，以生成更多候选项。由于扩展是逐位进行的，算法还必须确定中间长度上的频繁子串（即长度为 $ar + c$的子串，其中 $c < r$）。这种按位分解加快了剪枝过程。如果仅使用一般字母表 Σ，每次子串扩展都需要额外考虑 $O(|\Sigma|)$个候选项。使用二进制字母表时，每次扩展至多只需考虑两个额外候选项，代价是增加 $O(\log |\Sigma|)$轮。因此，在位级别探测频数可以提高扩展过程的效率。

从基础阶段 $i = 1$开始，我们构造频率向量$(f_{\mathcal{D}}(\gamma))_{\gamma \in \{0,1\}^{r-1}\$}$。根据引理 4.3，该向量的敏感度以 2ℓ为界。因此，我们可以构造加噪计数$\tilde{}$ $\tilde{}$($f_{\mathcal{D}}(\gamma))_{\gamma \in \{0,1\}^{r-1}\$}$，使其以高概率满足 ε_0-差分隐私，并且根据引理 3.2 和引理 3.3，加性误差为 $O(\ell/\varepsilon_0)$。长度为 r的字符对齐子串集合 C$_r$随后

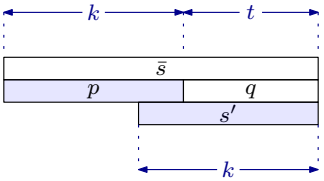


图 1. 引理 4.4 的示意图，其中字符串 C_k 已突出显示。

˜通过选取噪声频率高于阈值 $\tau \in O(\ell/\varepsilon)$的子串来构造，该阈值将在后文定义。

在每个后续阶段 $i > 1$ 中，当 $k = r \cdot 2^{i-1}$ 时，我们根据 C_k 中的子串计算 C_{k+1}、……、C_{2k}。这一过程基于以下关键观察。

引理 4.4. 给定一个关于 Σ 的数据集 $\mathcal{D}$，令 C_k 表示包含所有长度为 k 且频率至少为 τ_{T} 的子串的集合。令 $C_{k,t}$ 表示由所有长度为 $k+t$ 的字符串 $\bar{s}$ 构成的集合，其中某个 $t \in [k]$ 满足

$$(i)\ \bar{s}[1 \dots k] \in C_k,$$
$$(ii)\ \exists s' \in C_k \text{ such that } \bar{s}[k+1 \dots k+t] \text{ is a suffix of } s'.$$

那么，对每个 $t \in [k]$，每个长度为 $k+t$ 且频率至少为 τ_{T} 的子串都包含在 $C_{k,t}$ 中。

证明。假设子串 $\hat{s}$ 满足 $|\hat{s}| = k+t$，且频率至少为 τ_{T}。只需证明 $\hat{s} \in \mathsf{C}_{k,t}$。根据定义2.3，$f_{\mathcal{D}}(\hat{s}) \geq \tau_{\mathsf{T}}$。对于 (i)，即前缀条件，考虑前缀 $p = \hat{s}[1 \dots k]$。对于子串 $\hat{s} = \mathsf{S}[i \dots i+k+t-1]$ 的任意出现位置，其中 $\mathsf{S} \in \mathsf{D}$，$p = \mathsf{S}[i \dots i+k-1]$ 会出现在相同位置。因此，$f_{\mathcal{D}}(p) \geq f_{\mathcal{D}}(\hat{s})$。根据假设，所有频率至少为 τ_{T} 的子串都属于 C_k。因此，$p \in \mathsf{C}_k$。对于 (ii)，即后缀条件，令 $q = \hat{s}[k+1 \dots k+t]$ 表示 $\hat{s}$ 的长度为 t 的后缀，并令 $s' = \hat{s}[t+1 \dots k+t]$ 表示 $\hat{s}$ 的长度为 k 的后缀。由于 $\hat{s}$ 的频率至少为 τ_{T}，$\hat{s}$ 的每个长度为 k 的子串也都具有至少为 τ_{T} 的频率；因此 $s' \in \overline{\mathsf{C}_k}$。此外，$q$ 是 s' 的后缀。因此，$\hat{s} \in \mathsf{C}_{k,t}$，证毕。由此，$\hat{s} \in \mathsf{C}_{k,t}$。□

引理 4.4 在图1中进行了可视化。

候选扩展。 为便于候选生成，我们构建字符对齐后缀的树 T_k，这些后缀来自 C_k。由于 T_k是通过后处理计算得到的，因此不会产生隐私成本 C_k。根据引理 4.4，在 C_k 满足容斥准则的条件下，任何 τ_{T}频次的字符对齐子串，长度为 $k+t$，都必须：

(1) 以某个 $s \in \mathsf{C}_k$ 开头，(2) 后接一个子串 q，该子串构成从 T_k 的根节点出发、长度为 t 的路径。

换言之，存在 $u \in T_k$，其中 $\mathsf{str}(u) = q$ 且 $|q| = t$。因此，要枚举长度为 $k+1$、$\dots 2k$ 的所有τ_{T}频繁子串，只需搜索以下形式的路径串联

$$s \circ \mathsf{str}(u), \quad u \in T_k,\ s \in C_k.$$

候选搜索。我们搜索每棵串联树 $s \circ T_k, \forall s \in \mathsf{C}_k$，其中将 s视为一棵单路径树，其末节点表示 s的最终字符，并连接到 T_k 的根节点。对 $s \circ T_k$的搜索从节点 r 开始，其中 $\mathsf{str}(r) = s$（子树 T_k 的根节点）。对于深度为 $k+t$ 的每个˜ ˜节点 $u \in s \circ T_k$，我们计算该节点所表示子串的含噪频数 $f = f_{\mathcal{D}}(\mathsf{str}(u))$。如果

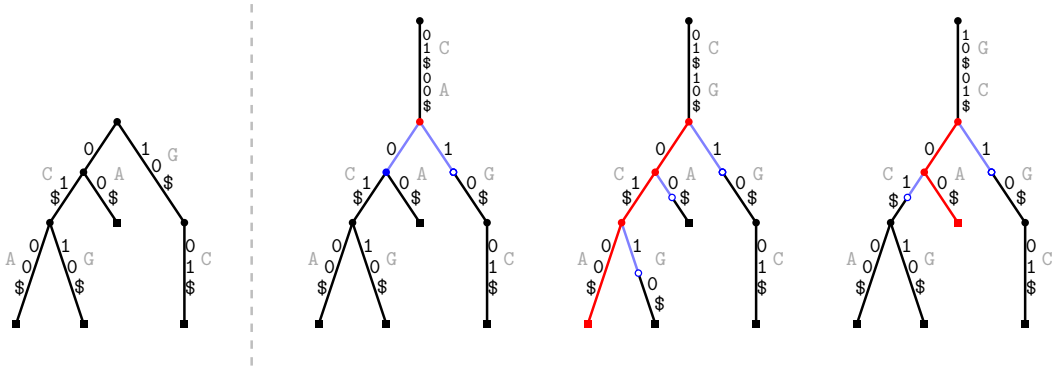


图 2. 算法 1 中的一个阶段 $i = 2$，其中 D $= \{\mathrm{CGCA}, \mathrm{CGCA}, \mathrm{CATA}\}$，$\tau_\perp = 2$，以及由 A $\mapsto$ 00$、C $\mapsto$ 01$、G $\mapsto$ 10$ 和 T $\mapsto$ 11$ 指定的分块编码E。此处，$r = 3$ 和 $k = 6$。左图：T_6，$\mathsf{C}_6 = \{\mathrm{01\$00\$}, \mathrm{01\$10\$}, \mathrm{10\$01\$}\}$ 的 r间隔稀疏后缀树 $\equiv \{\mathrm{CG}, \mathrm{GC}, \mathrm{CA}\}$，偏移量为 1、轮次为 3（即从位置 1 和 4 开始的后缀）。右图：对每个 $s \in \mathsf{C}_6$，在 $s \circ T_6$ 上的遍历。每次搜索都从子树 T_6 的根节点开始。在遍历过程中遇到每个节点 $u \in s \circ T_6$ 时，都会计算含噪频数 $\tilde{f} = f_{\mathcal{D}}(\mathrm{str}(u))$（详情见第 4.3 节）。若 $\tilde{f} \geq \tau_\perp$，则将 $\mathrm{str}(u)$ 加入 C_{6+t}，其中$6 + t = |$本例中 $\mathrm{str}(u)| \in \{9, 12\}$。红色节点表示噪声频率为 $\geq \tau_\perp$ 的子串，即 $\mathsf{C}_9 = \{\mathrm{01\$10\$01\$}, \mathrm{10\$01\$00\$}\}$ 中的子串 $\equiv \{\mathrm{CGC}, \mathrm{GCA}\}$，以及 $\mathsf{C}_{12} = \{\mathrm{01\$10\$01\$00\$}\}$ 中的子串 $\equiv \{\mathrm{CGCA}\}$。蓝色节点（实心圆点）和边内的位置（空心圆点）表示搜索被剪枝的位置。

含噪计数 $\tilde{f}$超过阈值 τ，我们将 $\mathrm{str}(u)$ 加入 C_{k+t}。否则，如果含噪计数未超过该阈值，我们通过返回到节点 $v = \mathrm{parent}(u)$ 来剪枝搜索。阈值 τ的选取使得：若 $\mathrm{str}(u)$ 是 $\tau_\top$-频繁的，则 $\tilde{f} \geq \tau$以高概率成立；反之，若 $\mathrm{str}(u)$ 是$\tau_\perp$-非频繁的，则 $\tilde{f} \geq \tau$以可忽略的概率发生。

效率由剪枝步骤保证。直观而言，对于每个 $t \in [k]$，只有长度为 $k+t$ 的 $\tau_\perp$-频繁字符串才能加入 C_{k+t}，而此类（字符对齐）字符串最多有 $O(n\ell/\tau_\perp) \subseteq O(n)$ 个，适用于 $\tau_\perp \geq \ell$。因此，以高概率，在所有树 $\{s \circ T_k \mid s \in \mathsf{C}_k\}$ 中，我们只遍历 $O(n)$ 条路径，且每条路径的长度都不超过 k。此外，我们只构造一次 T_k，并在该阶段后续搜索中重复使用它。

这就完成了归纳步骤。该过程继续进行，最终完成所有阶段$i \in \{1, \ldots, \lceil \log \ell \rceil\}$。针对 $i = 2$ 的一个阶段示例见图 2，完整过程的伪代码见算法1。

算法的输出是所有定义良好的 子串组成的集合

$$\bigcup_{j \leq \ell_{\mathrm{bit}} : j \equiv 0 \pmod{r}} C_j.$$

我们可以在进行候选生成和测试的同时，构建 $\bigcup_{j \leq \ell_{\mathrm{bit}}} \mathsf{C}_j$ 的字典树，从而构造这个集合。也就是说，每当挖掘出一个新的子串，我们就将对应的位添加到输出字典树中。在算法结束时，可以通过一次深度优先搜索，剪除表示未定义子串的叶节点。

注 4.5。 引理4.4允许采用一种更强但也更昂贵的变体：可以用看似更强的条件 *(iii)* 替代条件 *(ii)* $\bar{s}[t+1 \ldots k+t] \in C_k$. 这是 *Bernardini* 等人利用的观察结果。 [1]然而，同时满足 *(i)* 和 *(iii)* 需要针对不同的前缀选择，找出长度为 t 的后缀本身属于 C_k；这会导致成对检查的数量激增，并大幅增加运行时间。

算法 1： 差分隐私频繁子串挖掘

要求： 使用字母表 {0， 1， $} 存储和处理字符串

定义 DPFreqSub(D ε， β， $\tau_\perp$)

  $r = \lceil$对数 $|\Sigma|\rceil + 1$； $\ell_{\text{bit}} = \ell r$；

  $\varepsilon_0 = \frac{\varepsilon/\text{对数 } \ell}{4\ell \log(n\ell_{\text{bit}})}$；

  $\tau \leftarrow 4\cdot \varepsilon_0^{-1}\cdot \log(\ell\text{位}) \ln(n\ell_{\text{bit}}/\beta) + \tau_\perp$

  对于 $\gamma \in \{0, 1\}^{r-1}\$$ **执行**

    如果 $f_{\mathcal{D}}(\gamma)$ + 拉普($2\ell\varepsilon^{-1}$ 对数 ℓ) $> \tau$ **则**

      $\mathsf{C}_r \leftarrow \mathsf{C}_r \cup \gamma$

  对于 $i \in \{1, \ldots, \lceil$对数 $\ell\rceil - 1\}$ **执行**

    $k \leftarrow r\cdot 2^{i-1}$

    $T_k \leftarrow r$-C_k 的稀疏后缀树，轮次为 r

    如果 $|C_k| > \frac{n}{\text{对数 } \ell_{\text{bit}}}$ **then**

      终止

    对于 $s \in C_k$ **执行**

      从节点(s)开始，对 $s \circ T_k$ 执行深度优先搜索

      对于 在深度 $k + t$ 遇到的节点 u **执行**

        $\tilde{f} \leftarrow \text{NoisyCount}(s \circ T_k, u, \varepsilon_0)$

        if $\tilde{f} > \tau \wedge t < k + 1$ **则**

          $\mathsf{C}_{k+t} \leftarrow \mathsf{C}_{k+t} \cup \{\text{str}(u)\}$

          继续对 u的子节点进行深度优先搜索

        else

          在节点 u处剪枝搜索

  返回 $\bigcup_{j\le \ell_{\text{bit}}:\ j\equiv 0 (\text{模 } r)} C_j$

相比之下，引理 *4.4* 允许优雅地构造稀疏树 T_k，并在搜索扩展时重复用于每个 $s \in C_k$。这种复用显著缩短了运行时间，也是我们的方法与 *Bernardini* 等人的方法之间的关键区别。 *[1]*。

4.3 计算噪声频数

使用二进制编码字符串 $E(\mathsf{S}_1)\#_1^r E(\mathsf{S}_2)\#_2^r \ldots E(\mathsf{S}_n)\#_1^r$ 上间隔为 r 的稀疏后缀树，构造 D 中子串的精确频率预言机。禁止根据真实频率搜索频率达到 τ_{T} 的项。然而，仅对每个精确计数添加拉普拉斯噪声，会导致非最优的加性误差。例如，在给定搜索深度 $k + t$ 处，根据引理 4.3，精确频率计数的敏感度为 $\mathsf{O}(\ell)$。根据引理 3.3，我们可以在保持 ε^*-差分隐私的同时评估这些候选项，并达到 $\mathsf{O}(\ell/\varepsilon^*)$ 的加性误差。然而，根据引理 3.1，即使在经过剪枝的搜索空间中，要在所有阶段实现 ε-差分隐私，我们仍需要 $\varepsilon^* = \varepsilon/\ell$，并会得到 $\mathsf{O}((\ell)^2/\varepsilon)$ 的加性误差。

相反，我们采用一种估算子串频数的方法，该方法最初由 Bernardini 等人提出[1], ，巧妙地运用了二叉树机制。下面概述这种方法，以及如何在每次遍历期间实时计算这些估算值。

遍历候选树之前

$$\mathcal{T}_k = \{s \circ T_k \mid s \in C_k\},$$

算法 2： 在遍历候选树期间计算含噪计数 T

要求： T的重轻分解

定义 NoisyCount(T, u, ε_0)
  $\sigma = \varepsilon_0^{-1} \cdot$ 对数 ℓ_{bit}
  $r \leftarrow$ 包含 u的重路径的根节点
  如果 $u = r$ **则**
    初始化参数为 σ的二叉树机制 B_r^{σ} // 引理 3.4
    $f \leftarrow f_{\mathcal{D}}(\mathrm{str}(u))$
  else
    $f \leftarrow f_{\mathcal{D}}(\mathrm{str}(u)) - f_{\mathcal{D}}(\mathrm{str}(\mathrm{parent}(u)))$
  B_r^{σ}.递增(f)
  返回$\mathcal{B}_r^{\sigma}$.查询

我们对基树应用重轻分解 T_k。该分解仅取决于 T_k的拓扑结构。每条重边均指向其子树包含节点数最多的子节点；若存在并列，则任意选择。该分解在每次对$\mathcal{T}_k$进行遍历时 $s \circ T_k$均会被复用，并在每棵候选树上诱导出相应的分解，将其划分为顶点不相交的重路径。根据引理3.7，任意根到叶路径至多与 $O(\log |T_k|) = O(\log(n\ell_{\text{bit}}))$条重路径相交。

考虑任意节点 $u^* \in s \circ T_k$。令包含 u^*的重路径为

$$P_{u^*} = (u_0, u_1, \ldots, u_j = u^*, \ldots, u_d).$$

$\mathrm{str}(u^*)$ 的频率可表示为

$$f_{\mathcal{D}}(\mathtt{str}(u^*)) = f_{\mathcal{D}}(\mathtt{str}(u_0)) + \sum_{z=1}^{j} \big(f_{\mathcal{D}}(\mathtt{str}(u_z)) - f_{\mathcal{D}}(\mathtt{str}(u_{z-1}))\big).$$

因此，要估计 $f_{\mathcal{D}}(\mathrm{str}(u^*))$，只需发布以下对象的差分隐私前缀和

$$(f_{\mathcal{D}}(\mathtt{str}(u_0)),\ f_{\mathcal{D}}(\mathtt{str}(u_1)) - f_{\mathcal{D}}(\mathtt{str}(u_0)),\ \ldots,\ f_{\mathcal{D}}(\mathtt{str}(u_j)) - f_{\mathcal{D}}(\mathtt{str}(u_{j-1}))).$$

这正是二叉树机制所提供的功能：用于计算前缀和的差分隐私计数装置。

分解中的每条重路径都维护自己的二叉树机制，并以该路径的根节点命名。借助这一结构，我们在每次遍历期间动态计算含噪计数。当遇到节点 u 时，如果它是某条重路径的根节点，我们便使用适当校准的噪声[2] 初始化一个新的二叉树机制，并将值 $f_{\mathcal{D}}(\mathrm{str}(u))$ 加入其中。否则，如果 v是包含 u的重路径的根节点，我们就将值加入以 v 初始化的二叉树机制

$$f_{\mathcal{D}}(\mathtt{str}(u)) - f_{\mathcal{D}}(\mathtt{str}(\mathtt{parent}(u))).$$

该过程形式化地描述于算法 2中。后续章节将详细介绍所提供的加性误差和隐私保证。

5 分析

在本节中，我们始终假设所有输入字符串均来自二进制字母表。若字母表不是二进制字母表，则可将所有字符串转换为二进制形式，使字符串长度增加至$\ell_{\text{bit}} = \ell r$，其中 $r = \lceil \log |\Sigma| \rceil + 1$。

[2]Laplace standard deviation $\sigma = \varepsilon_0^{-1} \log \ell_{\text{bit}}$, for $\varepsilon_0 = \frac{\varepsilon/\log \ell}{4\ell \log(n\ell_{\text{bit}})}$, suffices.

5.1 正确性

引理 5.1. 对于任意参数 $\beta \in (0,1)$ 和 $\varepsilon \in \mathbb{R}_{\geq 0}$，令

$$\varepsilon_0 = \frac{\varepsilon/\log \ell}{4\ell \log(n\ell_{\text{bit}})} \quad and \quad \tau^* = \varepsilon_0^{-1} \cdot (\log \ell_{\text{bit}}) \cdot \ln(n\ell_{\text{bit}}/\beta)$$

然后，在算法 *1*中，对于阈值

$$\tau_\top = 9 \cdot \tau^* \quad and \quad \tau_\perp \geq \ell \log \ell_{\text{bit}}$$

集合 $C = \bigcup_{k=r}^{r\ell} C_k$ 以至少 $1-\beta$ 的概率满足$(\tau_\top, \tau_\perp)$-容斥准则。

证明。 设 $\beta_0 = \beta/\log \ell$。证明采用归纳法进行。算法 1 中的测试阈值设为

$$\tau = 4 \cdot \varepsilon_0^{-1} (\log \ell_{\text{bit}}) \cdot \ln(n\ell_{\text{bit}}/\beta) + \tau_\perp = 4 \cdot \tau^* + \tau_\perp.$$

在阶段 $i = 1$中，发现长度为 r的频繁子串。对于每个子串 $\gamma \in \{0,1\}^{r-1}\$$，$\text{Lap}(2\varepsilon^{-1}\ell \log \ell)$ 噪声被加到真实频率上。因此，根据引理 3.2，以概率$1-\beta_0$，$\forall \gamma \in\in \{0,1\}^{r-1}\$$，

$$|\tilde{f}_{\mathcal{D}}(\gamma) - f_{\mathcal{D}}(\gamma)| < 2\varepsilon^{-1} \cdot \ell(\log \ell) \ln(2/\beta_0) < \varepsilon_0^{-1} \ln(2/\beta).$$

这意味着：(i) 分配给 C_r 的任意长度为 r 的子串的最小频率为

$$\tau - \varepsilon_0^{-1} \ln(2/\beta) > (4-1) \cdot \tau^* + \tau_\perp > \tau_\perp,$$

以及 (ii) 频率大于 $\tau_\top$ 的任何子串的噪声估计至少为

$$\tau_\top - \varepsilon_0^{-1} \ln(2/\beta) > (9-1) \cdot \tau^* > \tau,$$

因此，对于所有长度为 r 的子串，$(\tau_\top, \tau_\perp)$-容斥准则适用于 C_r。根据构造，C_r 中的所有子串均为字符对齐子串。基本情形得证。

对于归纳情形，任取 $j < k+t$，其中 $k = r \cdot 2^i$ 且 $t \in \{1, \ldots, k\}$，假设 $(\tau_\top, \tau_\perp)$-容斥准则适用于 C_j 中长度为 j 的字符对齐子串。根据引理 4.4，任何长度为 $k+t$ 且频率至少为 $\tau_\top$ 的子串 $\bar{s}$，都对应于某个节点$\bar{v} \in s \circ T_k$，其中存在某个 $s \in \mathsf{C}_k$，使得 $\text{str}(\bar{v}) = \bar{s}$。此外，由于 $\bar{s}$ 的任意前缀 p 的频率为$f_{\mathcal{D}}(p) \geq f_{\mathcal{D}}(\bar{s})$，因此根据归纳假设，$s^* \in \mathsf{C}_{k+t-1}$，其中 $s^* = \text{str}(\text{parent}(\bar{v}))$。因此，$\bar{s}$ 在算法 1 的执行过程中会作为候选项被观测到。

深度$k+t$处所有候选项的噪声估计均由二叉树机制生成，其拉普拉斯标准差为 $\sigma = \varepsilon_0^{-1} \log \ell_{\text{bit}}$，作用于长度至多为 ℓ比特的前缀和。此外，根据归纳假设，由于D中长度为 $k+t$的字符对齐子串至多有 $n\ell$ 个，因此可得 $|\mathsf{C}_{k+t-1}| < n\ell/\tau_\perp < n$。长度为 $k+t$的候选子串数量至多为 $2|\mathsf{C}_{k+t-1}|$，因为对应于C_{k+t-1}中某个字符串的每个节点至多有两个子节点。令$\mathsf{F} = \{s \in \text{children}(u) \mid \text{str}(u) \in \mathsf{C}_{k+t-1}\}$。根据引理 3.4与并集界，$\forall s \in \mathsf{F}$，其概率为 $1-\beta_0$，

$$|\tilde{f}_{\mathcal{D}}(s) - f_{\mathcal{D}}(s)| \leq \sigma \cdot \ln \frac{|\mathcal{F}|}{\beta_0} = \sigma \cdot \ln \frac{2|C_{k+t-1}|}{\beta_0} \leq 4 \cdot \varepsilon_0^{-1} (\log \ell_{\text{bit}}) \cdot \ln(n\ell_{\text{bit}}/\beta) = 4\tau^*$$

这意味着：(i) 分配给C_{k+t}的任意字符串的最小频率至少为

$$\tau - 4 \cdot \tau^* > (4-4) \cdot \tau^* + \tau_\perp = \tau_\perp$$

以及（ii）任何频率大于 $\tau_\top$ 的字符，其噪声估计至少为

$$\tau_\top - 4 \cdot \tau^* = (9-4)\tau^* \geq 4 \cdot \tau^* + \tau_\perp = \tau.$$

因此，$(\tau_\top, \tau_\perp)$-容斥准则适用于 C_{k+t} 中所有长度为 $k+t$ 的子串。至此，归纳情形和证明均告完成。□

由于有 最多有 $n\ell$ 个长度为 k 的字符对齐子串，以下结果显然成立 .

推论 5.2。 对于参数 $\beta \in (0, 1)$和 $\tau_\perp = \ell \cdot \log \ell_{\text{bit}}$，对于所有 $k \in \{0, \ldots\ldots, \ell_{\text{bit}}\}$，且概率至少为 $1 - \beta$,

$$|C_k| \le \frac{n}{\log \ell_{\text{bit}}}.$$

5.2 隐私

引理 5.3。 对于 $\varepsilon \in \mathbb{R}_{\ge 0}$，算法 *1* 是 ε-差分隐私。

引理 5.3 的证明。 我们证明算法 1 的每个阶段都是 $\varepsilon/\log \ell$-差分隐私。通过对 $\log \ell$ 个阶段应用组合引理（引理 3.1），整体算法是 ε-差分隐私。

阶段 1*(*发布 C_r）。隐私保证直接由 $(f_{\mathcal{D}}(\gamma))_{\gamma \in \{0,1\}^{r-1}\$}$ 的敏感度得出，该敏感度至多为 2ℓ (引理 4.3)，并且由拉普拉斯机制的隐私保证得出（引理 3.3）。

阶段 i*(*发布 C_{k+1}, ……, C_{2k}，对于 $k = r \cdot 2^{i-1}$）。考虑所有 $s \in \mathsf{C}_k$ 对应的 $s \circ T_k$ 中全部重路径的集合 $\mathrm{HP}(\mathsf{C}_k)$。对于每条重路径 $\vec{u} = (u_0, u_1, \ldots\ldots, u_d) \in \mathrm{HP}(\mathsf{C}_k)$，定义频率差向量

$$\Delta f_{\mathcal{D}}(\vec{u}) = \Big(f_{\mathcal{D}}(\mathtt{str}(u_0)), f_{\mathcal{D}}(\mathtt{str}(u_1)) - f_{\mathcal{D}}(\mathtt{str}(u_0)), \ldots, f_{\mathcal{D}}(\mathtt{str}(u_d)) - f_{\mathcal{D}}(\mathtt{str}(u_{d-1}))\Big).$$

记 B 为二叉树机制（引理 3.4），其参数为 ε_0。设 D′ 是一个相邻数据集，与另一个数据集恰好相差一位用户的字符串。于是，根据引理 3.4，$\mathrm{B}(\Delta f_{\mathcal{D}}(\vec{u}))$ 和 $\mathrm{B}(\Delta f_{\mathcal{D}'}(\vec{u}))$ 是 $(\varepsilon_0 \cdot \|\Delta f_{\mathcal{D}}(\vec{u}) - \Delta f_{\mathcal{D}'}(\vec{u})\|_1)$-不可区分的（定义 2.1）。根据组合引理（引理 3.1），该集合

$$\{\Delta f_{\mathcal{D}}(\vec{u}) : \vec{u} \in \mathrm{HP}(C_k)\} \quad \text{and} \quad \{\Delta f_{\mathcal{D}'}(\vec{u}) : \vec{u} \in \mathrm{HP}(C_k)\}$$

是 $(\varepsilon_0 \cdot \sum_{\vec{u} \in \mathrm{HP}(C_k)} \|\Delta f_{\mathcal{D}}(\vec{u}) - \Delta f_{\mathcal{D}'}(\vec{u})\|_1)$-不可区分的。若我们证明

$$\sum_{\vec{u} \in \mathrm{HP}(C_k)} \|\Delta f_{\mathcal{D}}(\vec{u}) - \Delta f_{\mathcal{D}'}(\vec{u})\|_1 \le 4\ell \log(n\ell_{\text{bit}}), \tag{2}$$

那么，令 $\varepsilon_0 = \frac{\varepsilon}{\log 4\ell} \cdot \log(n\ell_{\text{bit}})$(算法 1 第 3 行）可确保阶段 i满足 $\varepsilon/\log \ell$-差分隐私，因为其输出可视为对 $\{\Delta f_{\mathcal{D}}(\vec{u})$ 的后处理：$\vec{u} \in \mathrm{HP}(C_k)\}$。接下来需要证明等式（2）。为此，我们使用以下两个引理，其证明见附录 A。

引理 5.4（重路径频率差的敏感度）。设 $\vec{u} = (u_0, \ldots, u_d)$是$\mathrm{HP}(C_k)$中的一条重路径，并且$\mathcal{D}'$ 是一个相邻数据集，两者恰好相差一个用户的字符串：$\mathcal{D}' = \mathcal{D} \setminus \{\mathcal{S}\} \cup \{\mathcal{S}'\}$，其中$\mathcal{S}$、$\mathcal{S}' \in \{0,1\}^{1:\ell_{\text{bit}}}$。那么

$$\|\Delta f_{\mathcal{D}}(\vec{u}) - \Delta f_{\mathcal{D}'}(\vec{u})\|_1 \le 2 \cdot \max\{f_{\mathcal{S}}(\mathtt{str}(u_0)), f_{\mathcal{S}'}(\mathtt{str}(u_0))\}.$$

引理 5.5（受影响的重路径质量）。设 $k = r \cdot 2^{i-1}$ 处于阶段 $i > 1$，并设 $\mathrm{HP}(C_k)$ 为通过对 T_k 进行重轻分解而在候选树 $\{s \circ T_k : s \in C_k\}$ 上诱导出的重路径族。对于相邻数据集 $D' = D \setminus \{S\} \cup \{S'\}$，其中 $S, S' \in \{0,1\}^{1:\ell_{\text{bit}}}$,

$$\sum_{\vec{u} \in \mathrm{HP}(C_k)} \max\Big\{f_{\mathcal{S}}(\mathrm{str}(\mathrm{head}(\vec{u}))), f_{\mathcal{S}'}(\mathrm{str}(\mathrm{head}(\vec{u})))\Big\} \le 2\ell \log(n\ell_{\text{bit}}),$$

其中 头部$(\vec{u})$ 是 $\vec{u}$ 的头部。

结合引理 5.4 & 5.5，我们观察到

$$\sum_{\vec{u}\in \mathrm{HP}(C_k)} \left\|\Delta f_{\mathcal{D}}(\vec{u}) - \Delta f_{\mathcal{D}'}(\vec{u})\right\|_1 \le \sum_{\vec{u}\in \mathrm{HP}(C_k)} 2\cdot \max\left\{f_{\mathcal{S}}(\mathrm{str}(\mathrm{head}(\vec{u}))), f_{\mathcal{S}'}(\mathrm{str}(\mathrm{head}(\vec{u})))\right\} \le 4\ell \log(n\ell_{\mathrm{bit}}).$$

至此，证明完成。 □

5.3 性能

引理 5.6。 算法 *1* 的运行时间为 $O(n\ell_{\mathrm{bit}} + |\Sigma|)$，空间复杂度为 $O(n\ell + |\Sigma|)$。

证明。我们将总成本分解为三个部分：

(i) 构造 D 上步长为 r、轮次为 $r = \lceil \log |\Sigma| \rceil + 1$ 的稀疏后缀树，以获得精确的频率计数，记为 $ST_{\mathcal{D}}$，并为每个分支节点 u 标注以该节点为根的子树中的叶节点数量 u（在所有阶段开始前构造并标注一次）；(ii) 构造 C 中字符对齐后缀的树$_k$（每个阶段构造一次），记为 T_k，并计算其重路径分解；(iii) 在每个阶段 i，遍历拼接树 $s \circ T_k$（针对 $s \in T_k$），同时维护其在 $ST_{\mathcal{D}}$ 中对应的位置。

运行时间。 令 $E\colon \Sigma \to \{x\$ \mid x \in \{0,1\}^{r-1}\}$ 表示用于将字符转换为二进制字符串的码本。码本可存储在 $O(|\Sigma|)$ 内存中。对于组件 (i)，我们在字符串 $T = E(\mathsf{S}_1)\#_1^r \dots E(\mathsf{S}_n)\#_n^r$ 上构造一个间隔为 r 的稀疏后缀树 $ST_{\mathcal{D}}$。通过将每个编码后的字符存储在 $O(1)$ 个字中，我们可以在 $O(n\ell)$ 时间和空间内构造 T。根据引理 3.6，构造 $ST_{\mathcal{D}}$ 需要

$$O\Big(\sum_{\mathcal{S}\in\mathcal{D}} |\mathcal{S}|\Big) = O(n\ell_{\mathrm{bit}})$$

时间，并生成一棵包含

$$O\Big(\sum_{\mathcal{S}\in\mathcal{D}} |\mathcal{S}|/r\Big) = O(n\ell_{\mathrm{bit}}/r) = O(n\ell) \tag{3}$$

个节点的树。标注分支节点需要进行一次 $O(n\ell)$ 遍历。

对于组件 (ii)，固定阶段 i，并令 $k = r\cdot 2^{i-1}$。令 $\mathsf{C}_k = \{s_1, \dots, s_{|\mathsf{C}_k|}\}$。与频率预言机 T_k 类似，C_k 中字符对齐后缀的树可以通过在字符串 $s_1\#_1^r s_2\#_2^r \cdots s_{|\mathsf{C}_k|}\#_{|\mathsf{C}_k|}^r$ 上构造间隔为 r 的稀疏后缀树来实现。由于轮次的选择，该结构支持对字符对齐后缀进行深度优先搜索，前提是在子串分隔符 $\#_i$ 处进行剪枝。因此，根据引理 3.6，构造 T_k 的代价有界于

$$O\left(\sum_{s\in C_k} |s|\right) = O(|C_k|k) = O\left(\frac{n\cdot k}{\log \ell_{\mathrm{bit}}}.\right)$$

由推论 5.2。对所有阶段求和得到时间复杂度

$$O\left(\frac{n}{(\log \ell_{\mathrm{bit}})} \sum_{i=0}^{\lceil \log \ell \rceil} r\cdot 2^i\right) = O\left(\frac{n\ell_{\mathrm{bit}}}{\log \ell_{\mathrm{bit}}}\right).$$

T_k 的重路径分解通过一次深度优先搜索计算，每个节点的工作量为常数。注意，T_k 用压缩（非分支）路径实现。根据定义，该路径上的每条（虚拟）边都是重边，因为从该路径分支出去的节点是空子树。因此，计算重路径分解的代价与 C_k 上 r-间隔稀疏后缀树中的节点数成正比。对所有阶段求和后，这需要 $O(n\ell)$ 时间。

对于分量 (iii)，在阶段 1 我们执行 $|\Sigma|$ 常数时间带噪频率查询。在阶段 $i > 1$(使用 $k = r\cdot 2^{i-1}$) 我们对以下形式的串联树执行深度优先搜索：$s \circ T_k$，

其中 $s \in T_k$。一旦子串长度超过 $2k$，或噪声频率低于测试阈值 τ，搜索便会立即剪枝。遍历在 C_k 的间隔为 r 的稀疏后缀树中进行。每一步中，我们要么移动到树中的实际节点，要么移动到虚拟节点，其位点位于压缩分支内部。值得注意的是，压缩分支上的节点可以逐个揭示，而无需展开整个分支（该分支可能表示长度为 $O(k)$ 的子串）。因此，在 T_k 中遍历一个分支时，每揭示一个字符都只需常数时间。

在每个访问到的位点，我们都会计算一个含噪频数，用于阈值检验。含噪计数通过在重路径上在线使用二叉树机制（算法 2）生成，并且需要精确的子串频率来计算前缀和。为了获得精确频率，$\circ$ 中的深度优先搜索会在 $ST_{\mathcal{D}}$ 中镜像执行。对于每个 C，我们存储一个指向节点 $\mathsf{ST}_{\mathcal{D}}$ 的指针，其中 $\mathrm{str}(u_s) = s$，然后从 u_s继续遍历。在 $ST_{\mathcal{D}}$ 中，每一步都耗费常数时间。当遍历 $ST_{\mathcal{D}}$ 中的压缩边时（该压缩边可能表示长度为 $O(\ell_{\mathrm{bit}})$ 的子串），字符及其频率可以按每步 $O(1)$ 的时间增量式地揭示。因此，生成每个精确频率的成本为 $O(1)$。

在遍历 $s \circ T_k$ 期间触及的每个节点都会在一个二叉树机制中形成一个节点。由于可以在常数时间内获得精确频率信息，因此，根据引理 3.4，更新和查询每个二叉树机制中的每个节点需要 $O(\log d)$时间，其中 $d \le k$是相应重路径的长度。由于每个 $\bar{s} \in C_{k+t}$ 最多产生两个额外的候选节点和频率估计，因此阶段 i 的所有遍历总成本为

$$\sum_{t=1}^{k} 2 \cdot |C_{k+t}| \cdot O(\log k) = O\left(\frac{n \cdot k \cdot \log k}{\log \ell_{\mathrm{bit}}}\right) = O(n \cdot k)$$

时间。对 $\log \ell$轮次的所有阶段求和，累计运行时间为 $O(n\ell_{\mathrm{bit}})$，这与构建$ST_{\mathcal{D}}$的时间成本成正比。加上阶段 1 的 $O(|\Sigma|)$成本，即得所述界限。

内存。 对于内存成本，我们在整个算法执行过程中存储 $ST_{\mathcal{D}}$，并且在阶段 i期间，对于 $k = r \cdot 2^i$，我们为 T_k中的每个字符串存储 C_k。根据公式 3，存储 $ST_{\mathcal{D}}$所需的内存为 $O(n\ell)$。根据引理 T_k、引理 3.6和推论 5.2，构建并存储 T_k所需的内存有界为

$$O\left(\sum_{s \in C_k} |s|/r\right) = O(|C_k|k/r) = O\left(\frac{n \cdot k}{r \cdot \log \ell_{\mathrm{bit}}}\right) = O\left(\frac{n \cdot \ell_{\mathrm{bit}}}{r \cdot \log \ell_{\mathrm{bit}}}\right) = O\left(\frac{n \cdot \ell}{\log \ell_{\mathrm{bit}}}\right).$$

T_k 可在阶段结束时删除。

在遍历串联树 $s \circ T_k$的过程中，一旦离开二叉树机制所属的路径，就可以将其删除。因此，包括存储码本的成本在内，执行算法所需的空间为 $O(n\ell + |\Sigma|)$。□

6 结论

在本工作中，我们研究了差分隐私频繁子串挖掘问题，并提出了一种新的 ε-差分隐私算法，该算法在误差保证上与 Bernardini 等人 [1] 的近乎最优误差保证相匹配，同时将空间复杂度从 $O(n^2\ell^4)$ 降至 $O(n\ell + |\Sigma|)$，并将时间复杂度从 $O(n^2\ell^4 + |\Sigma|)$ 降至 $O(n\ell_{\mathrm{bit}} + |\Sigma|)$。我们改进后的候选生成策略与频率引导的剪枝规则消除了先前工作中的二次增长，从而能够在差分隐私条件下进行可扩展的频繁子串挖掘。未来工作包括将这些技术扩展到更丰富的模式挖掘任务，并在大规模数据集上评估其经验性能。

参考文献

[1] G. Bernardini、P. Bille、I. L. Gørtz 与 T. A. Steiner，“差分隐私子串与文档计数”，《ACM 数据管理汇刊》，第 3 卷，第 2 期，第 1–27 页，2025年。[2] L. Bonomi 与 L. Xiong，“一种用于挖掘具有差分隐私的序列模式的两阶段算法”，载于第 *22* 2013, 269 278 [ ] 10 1145 2505515 2505553 届 CIKM 会议论文集，第 – 页。在线。可获取：https://doi.org/ . / . T. H. Chan、E. Shi 与 D. Song，“统计数据的隐私与持续发布”，《ACM 信息系统安全汇刊》，第 14 卷，第 3 期，第 26:1–26:24 页，2011年。[4] M. X. Chen、B. N. Lee、G. Bansal、Y. Cao、S. Zhang、J. Lu、J. Tsay、Y. Wang、A. M. Dai、Z. Chen、T. Sohn 与 Y. Wu，“Gmail 智能撰写：实时辅助写作”，载于第 25 届 ACM SIGKDD 知识发现与数据挖掘国际会议论文集，KDD 2019，美国安克雷奇，2019年8月4–8日，A. Teredesai、V. Kumar、Y. Li、R. Rosales、E. Terzi 与 G. Karypis 编。ACM，2019年，第 2287–2295 页。 [在线]。可获取：https://doi.org/ 10 1145 3292500 3330723 . / . R. Chen、G. Ács 与 C. Castelluccia，“通过可变长度 n 元语法实现差分隐私序列数据发布”，载于第 *19* 2012, 638 649 [ ] 10 1145 2382196 2382263 届 CCS 会议论文集，第 – 页。在线。可获取：https://doi.org/ . / . R. Chen、B. C. M. Fung、B. C. Desai 与 N. M. Sossou，“差分隐私交通数据发布：蒙特利尔交通系统案例研究”，载于第 18 届 KDD 会议论文集，2012年，第 213–221 页。 [在线]。可获取：https://doi.org/ 10 1145 2339530 2339564 . / . B. Deb、P. Bailey 与 M. Shokouhi，“使用匹配条件变分自编码器多样化回复建议”，载于 2019 年计算语言学协会北美分会会议：人类语言技术，NAACL-HLT 2019，美国明尼阿波利斯，2019年6月2–7日，第 2 卷（行业论文），A. Loukina、M. Morales 与 R. Kumar 编。计算语言学协会，2019年，第 40–47 页。 [ ] 10 18653 1 19 2006 在线。可获取：https://doi.org/ . /v /n - C. Dwork 与 A. Roth，“差分隐私的算法基础”，《理论计算机科学基础趋势》，第 9 卷，第 3-4 期，第 211–407 页，2014年。[9] C. Dwork、F. McSherry、K. Nissim 与 A. D. Smith，“在隐私数据分析中根据敏感度校准噪声”，载于《密码学理论》，第三届密码学理论会议，TCC 2006，美国纽约，2006年3月4–7日，会议论文集，计算机科学讲义丛书，S. Halevi 与 T. Rabin 编，第 3876 卷。Springer，2006年，第 265–284 页。[10] C. Dwork、M. Naor、T. Pitassi 与 G. N. Rothblum，“持续观测下的差分隐私”，载于第 42 届 ACM 计算理论研讨会论文集，STOC 2010，美国马萨诸塞州剑桥，2010年6月5–8日，L. J. Schulman 编。ACM，2010年，第 715–724 页。[11] M. Farach，“大字母表上的最优后缀树构造”，载于第 38 届计算机科学基础年度研讨会（FOCS）。IEEE计算机学会，1997年，第 137–143 页。[12] B. Hu、Z. Lu、H. Li 与 Q. Chen，“用于匹配自然语言句子的卷积神经网络架构”，载于《神经信息处理系统进展 27：2014 年神经信息处理系统年度会议》，2014年12月8–13日，加拿大魁北克蒙特利尔，Z. Ghahramani、M. Welling、C. Cortes、N. D. Lawrence 与 K. Q. Weinberger 编，2014年，第 2042–2050 页。 [在线]。可获取：https://proceedings.neurips.cc/paper/2014/hash/b9d487a30398d42ecff55c228ed5652b-Abstract.html [13] A. Kannan、K. Kurach、S. Ravi、T. Kaufmann、A. Tomkins、B. Miklos、G. Corrado、L. Lukács、M. Ganea、P. Young 与 V. Ramavajjala，“智能回复：面向电子邮件的自动回复建议”，载于第 22 届 ACM SIGKDD 知识发现与数据挖掘国际会议论文集，美国加利福尼亚州旧金山，2016年8月13–17日，B. Krishnapuram、M. Shah、A. J. Smola、C. C. Aggarwal、D. Shen 与 R. Rastogi 编。ACM，2016年，第 955–964 页。 [ ] 10 1145 2939672 2939801 在线。可获取：https://doi.org/ . / . J. Kärkkäinen 与 E. Ukkonen，“稀疏后缀树”，载于国际计算与组合学会议。Springer，1996年，第 219–230 页。[15] T. Khatri、G. G. Dagher 与 Y. Hou，“通过差分隐私后缀树实现隐私保护基因组数据发布”，载于第 *15* 2019, 569 584 [ ] 10 1007 978 3 030 37228 6_28 届 EAI 会议论文集，第 – *34* 页。在线。可获取：https://doi.org/ . / - - - - K. Kim 、 S. Gopi 、 J. Kulkarni 与 S. Yekhanin，“ 差分隐私 n 元语法提取 ”，载于第 届 NeurIPS 会议论文集，2021年，第 5102–5111 页。 [在线]。可获取：https://proceedings.neurips.cc/paper/2021/hash/28ce9bc954876829eeb56ff46da8e1ab- Abstract.html [17] E. M. McCreight，“一种节省空间的后缀树构造算法”，《ACM 学报》，第 23 卷，第 2 期，第 262–272 页，1976年。[18] M. Nunez-del Prado、Y. Maehara-Aliaga、J. Salas、H. Alatrista-Salas 与 D. Megías，“一种用于挖掘频繁序列模式的基于图的差分隐私算法”，《应用科学》，第 12 卷，第 4 期，2022年。 [在线]。可获取：https://www.mdpi.com/ 2076 3417 12 4 2131 - / / / D. D. Sleator 与 R. E. Tarjan，“一种用于动态树的数据结构”，《计算机与系统科学学报》，第 26 卷，391，1983年 3，362第 期 第 – 页。[20] E. Ukkonen，“在线后缀树构造”，《Algorithmica》，第 14 卷，第 3 期，第 249–260 页，1995年。

[21] P. Weiner，“线性模式匹配算法”，载于 第*14*届年度开关与自动机理论研讨会，美国艾奥瓦州艾奥瓦城，*1973*年*10*月*15*日至*17*日。美国加利福尼亚州洛斯阿拉米托斯：IEEE计算机学会，1973年，第1–11页。[22] S. Xu、X. Cheng、S. Su、K. Xiao 和 L. Xiong，“差分隐私频繁序列挖掘”，*IEEE*知识与数据工程汇刊，第
28, 11, 2910 2926, 2016 [ ] 10 1109 2016 2601106 卷，第。页码–。在线。获取地址：
[23]
https://doi.org/./TKDE..J. Zhang、X. Xiao 和 X. Xie，“Privtree：一种用于层次分解的差分隐私算法”，载于 *2016*年*SIGMOD*会议论文集，2016年，第155–170页。 [在线]。获取地址：https://doi.org/10.1145/2882903.2882928

第 5.2 节的延迟证明

引理5.4的证明。根据定义，对于每个字符串 P, $f_{\mathcal{D}}(P) = \sum_{S^* \in \mathcal{D}} f_{S^*}(P)$。因此，当 $\mathrm{D}' = \mathrm{D} \setminus \{\mathrm{S}\} \cup \{\mathrm{S}'\}$ 时，

$$\begin{aligned}\left\|\Delta f_{\mathcal{D}}(\vec{u}) - \Delta f_{\mathcal{D}'}(\vec{u})\right\|_1 &= \left|f_{\mathcal{D}}(\mathtt{str}(u_0)) - f_{\mathcal{D}'}(\mathtt{str}(u_0))\right| \\ &\quad + \sum_{j\in[d]} \left|\left(f_{\mathcal{D}}(\mathtt{str}(u_j)) - f_{\mathcal{D}}(\mathtt{str}(u_{j-1}))\right) - \left(f_{\mathcal{D}'}(\mathtt{str}(u_j)) - f_{\mathcal{D}'}(\mathtt{str}(u_{j-1}))\right)\right| \\ &= \left|f_{\mathcal{S}}(\mathtt{str}(u_0)) - f_{\mathcal{S}'}(\mathtt{str}(u_0))\right| \\ &\quad + \sum_{j\in[d]} \left|f_{\mathcal{S}}(\mathtt{str}(u_j)) - f_{\mathcal{S}}(\mathtt{str}(u_{j-1})) - \left(f_{\mathcal{S}'}(\mathtt{str}(u_j)) - f_{\mathcal{S}'}(\mathtt{str}(u_{j-1}))\right)\right|.\end{aligned}$$

由于每个向量 $\Delta f_{\mathcal{S}}(\vec{u})$ 和 $\Delta f_{\mathcal{S}'}(\vec{u})$ 的元素均为非正数（频数沿重路径递减），根据三角不等式可得：

$$\begin{aligned}&\sum_{j\in[d]} \left|f_{\mathcal{S}}(\mathtt{str}(u_j)) - f_{\mathcal{S}}(\mathtt{str}(u_{j-1})) - \left(f_{\mathcal{S}'}(\mathtt{str}(u_j)) - f_{\mathcal{S}'}(\mathtt{str}(u_{j-1}))\right)\right| \\ &\le \sum_{j\in[d]} \left(f_{\mathcal{S}}(\mathtt{str}(u_{j-1})) - f_{\mathcal{S}}(\mathtt{str}(u_j))\right) + \sum_{j\in[d]} \left(f_{\mathcal{S}'}(\mathtt{str}(u_{j-1})) - f_{\mathcal{S}'}(\mathtt{str}(u_j))\right) \\ &\le f_{\mathcal{S}}(\mathtt{str}(u_0)) + f_{\mathcal{S}'}(\mathtt{str}(u_0)).\end{aligned}$$

因此，

$$\begin{aligned}\left\|\Delta f_{\mathcal{D}}(\vec{u}) - \Delta f_{\mathcal{D}'}(\vec{u})\right\|_1 &= \left|f_{\mathcal{D}}(\mathtt{str}(u_0)) - f_{\mathcal{D}'}(\mathtt{str}(u_0))\right| + f_{\mathcal{S}}(\mathtt{str}(u_0)) + f_{\mathcal{S}'}(\mathtt{str}(u_0)) \\ &\le 2 \cdot \max\{f_{\mathcal{S}}(\mathtt{str}(u_0)), f_{\mathcal{S}'}(\mathtt{str}(u_0))\}.\end{aligned}$$

□

引理的证明 5.5。 固定 $\mathcal{S}$。在 k 长度的子串在 $\mathcal{S}$ 中（共有$\le \ell$）每个字符对齐起点至多诱导某个 $s \circ T_k$中的一次遍历。根据引理 3.7，重轻分解保证任意根到叶遍历至多经过 $\lceil \log |T_k| \rceil$ 条重路径。对每条经过的路径的头部计入一个单位。根据推论 5.2，对所有路径求和可得

$$\begin{aligned}\sum_{\vec{u}\in \mathsf{HP}(C_k)} f_{\mathcal{S}}(\mathtt{str}(\mathsf{head}(\vec{u}))) &\le \sum_{s\in \mathtt{suffix}(\mathcal{S})} \sum_{\vec{u}\in \mathsf{HP}(C_k)} \mathbb{1}[\mathtt{str}(\mathsf{head}(\vec{u})) \preceq s] \\ &\le \ell \lceil \log |T_k| \rceil \\ &\le 2\ell \log(n\ell_{\mathrm{bit}}).\end{aligned}$$

□

B 相关工作

尽管差分隐私模式挖掘已有大量研究，但以往方法在根本上难以解决因隐私预算拆分导致的效用下降问题，或缺乏正式的渐近效用保证。

连续子串和 q-gram 挖掘。Kim 等人 [16] 通过自顶向下搜索提取频繁 q-gram，搜索长度最多为 q。用本文的记号表示，他们的算法可以解释为对长度最多为 q的频繁子串进行自顶向下搜索。在确定一组C_k 潜在频繁且长度为 k 的子串候选项后，对于某个 $k < q$，他们通过探索并剪枝C_{k+1} 中的子串来构建 $(C_k \times C_1) \cap (C_1 \times C_k)$. 这需要在多个 q搜索阶段之间分配隐私预算，进而降低效用保证。

类似地，Chen 等人 [5] 研究了为 $q = 1$、……、 $q_{\max}$ 查找频繁 q-gram 的问题。他们在定义域为所有可能的 $q_{\max}$-gram 的树上采用自顶向下搜索，并将隐私预算分配到树的各个高度，使用启发式方法在各层之间自适应地分配预算。他们的论文提出了实用的分配策略，但没有对所得方法进行正式的效用分析。

Khatri 等人 [15] 研究了基因组数据中的私有子串计数问题。他们在输入字符串的所有后缀上构建了一棵反向树，其中每条边由字母表中的一个符号标记，每条从根节点到节点的路径对应于数据集中出现的某个后缀的逆序，每个节点存储该后缀的出现次数。随后，他们应用拉普拉斯机制对该反向树中的节点计数进行私有化。利用私有化后的计数，他们在后处理步骤中构建了一棵后缀树。然而，对于在所得后缀树上回答子串计数查询的效用保证，并未得到明确分析。

前缀与轨迹挖掘。 Chen 等人 [6] 研究了蒙特利尔地区公共交通数据的频繁序列模式挖掘，将每位乘客的行程表示为一个按时间排序的车站序列。他们的方法以差分隐私方式在所有公交序列上构建前缀树，从而捕获在数据集中作为前缀频繁出现的子序列。然而，与前缀不对齐的频繁子序列可能无法被该方法准确捕获。

Zhang 等人 [23] 提出了一种用于树构建的自适应自顶向下方法。他们的算法最初是为处理空间数据和构建四叉树而设计的。我们将其框架调整到字符串场景，并在此简要概述该算法。当输入数据集由字符串组成时（如果两个数据集仅相差一个字符串，则称它们是相邻的），该算法通过仅保留频数超过所选阈值的节点，在这些字符串上构建前缀树。一个关键见解是，当目标是以隐私保护方式仅发布前缀树的拓扑结构时，他们的方法无需在树高范围内分配隐私预算。然而，当还必须发布前缀节点计数时，他们的方法并未提供正式的效用分析。

Bonomi 和 Xiong [2] 研究了私有 top-k 频繁模式挖掘问题，旨在同时输出频繁前缀和频繁子串。他们采用两阶段方法：在第一阶段，构建差分隐私前缀树以识别频繁前缀，并将其作为频繁子串的候选项；在第二阶段，运用指纹技术对这些候选项进行转换和细化，得到最终输出。第一阶段需要在不同树层级之间分配隐私预算，他们还引入启发式方法来改进分配。然而，该论文并未对其方法提供正式的效用分析。

非连续序列模式。 Xu 等人 [22] 研究了如何在字符串数据集上以隐私保护方式识别频繁子序列。与子串不同，子序列不

必连续出现。他们的算法也采用自顶向下搜索策略，按长度递增的顺序探索候选频繁子序列。为缩小搜索空间，他们在每一步都使用数据集的一个子采样子集。然而，该论文没有提供渐近效用分析。

Nunez-del Prado 等人 [18] 提出了一种基于图的差分隐私方法，用于发布频繁序列模式。给定在预处理步骤中获得的一组频繁模式，他们构建用户–模式二分图，并应用边差分隐私噪声图机制；随后从受保护的图中重新计算模式支持度，以发布这些模式，并提供差分隐私保证，全程无需进一步访问用户的序列。他们使用若干经验指标评估效用，并报告称该方法优于先前工作；然而，与前述方法一样，他们没有提供正式的渐近效用保证。