Differentially Private Sequential Data Publication via Variable-Length N-Grams

Rui Chen[*]
Concordia University
Montreal, Canada
ru_che@encs.concordia.ca

Gergely Acs
INRIA
France
gergely.acs@inria.fr

Claude Castelluccia
INRIA
France
claude.castelluccia@inria.fr

ABSTRACT

Sequential data is being increasingly used in a variety of applications. Publishing sequential data is of vital importance to the advancement of these applications. However, as shown by the re-identification attacks on the AOL and Netflix datasets, releasing sequential data may pose considerable threats to individual privacy. Recent research has indicated the failure of existing sanitization techniques to provide claimed privacy guarantees. It is therefore urgent to respond to this failure by developing new schemes with provable privacy guarantees. *Differential privacy* is one of the only models that can be used to provide such guarantees. Due to the inherent sequentiality and high-dimensionality, it is challenging to apply differential privacy to sequential data. In this paper, we address this challenge by employing a *variable-length n-gram model*, which extracts the essential information of a sequential database in terms of a set of variable-length n-grams. Our approach makes use of a carefully designed exploration tree structure and a set of novel techniques based on the Markov assumption in order to lower the magnitude of added noise. The published n-grams are useful for many purposes. Furthermore, we develop a solution for generating a synthetic database, which enables a wider spectrum of data analysis tasks. Extensive experiments on real-life datasets demonstrate that our approach substantially outperforms the state-of-the-art techniques.

Categories and Subject Descriptors

H.2.7 [**Database Administration**]: [Security, integrity, and protection]; H.2.8 [**Database Applications**]: [Data mining]

General Terms

Algorithms, Performance, Security

Keywords

Differential privacy, sequential data, n-gram model, Markov assumption

1. INTRODUCTION

Sequential data (see a formal definition in Section 3.1), such as DNA sequences, web browsing histories and mobility traces, is being increasingly used in a variety of real-life applications, spanning from genome and web usage analysis to location-based recommendation systems. Publishing sequential data is important, since they enable researchers to analyze and understand interesting patterns. For example, mobility traces have become widely collected in recent years and have opened the possibility to improve our understanding of large-scale social networks by investigating how people exchange information, interact and develop social relationships. With billions of handsets in use worldwide, the amount of sequential data has been gigantic. When aggregated, it can help understand complex processes (e.g., the spread of viruses), build better transportation systems, and prevent traffic congestion [20]. While the benefits provided by sequential data are indisputable, it, unfortunately, poses considerable threats to individual privacy [19]. In fact, sequential data might be used by a malicious adversary to discover potential sensitive information about a record owner, such as his habits, religion or relationships. Because privacy is so important to people, companies and researchers are reluctant to publish datasets by fear of being held responsible for potential privacy breaches. As a result, only very few of them are actually released and available. This limits our ability to analyze such data to derive information that could benefit the general public.

Several sequential data sanitization algorithms have been presented recently [19], [3]. However, their privacy properties are still dubious, since they rely on privacy models that are either ad-hoc or considered weak. It is therefore urgent to respond to the failure of existing sanitization techniques by developing new schemes with provable privacy guarantees. *Differential privacy* [9] is one of the only models that can be used to provide such guarantees. The main idea of differential privacy is to add noise to a dataset so that an adversary cannot decide whether a particular record (e.g., a sequence in our case) is included in the dataset or not. Due to the inherent *sequentiality* and *high-dimensionality* of sequential data, it is challenging to apply differential privacy to sequential data. In particular, naively adding noise to the occurrence count of each distinct sequence in the dataset negatively impacts both privacy and utility. First, unless

[*] This work was done when the first author was on an internship at INRIA.

CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA.
Copyright 2012 ACM 978-1-4503-1651-4/12/10 ...$15.00.

we always release the noisy counts of all possible sequences (whether in or not in the dataset), which is computationally infeasible in practice, adding a new record will most likely entail the inclusion of a new sequence that has not been present in the dataset. This clearly breaks ε-differential privacy and requires to relax the privacy model to the weaker (ε, δ)-differential privacy [8]. Second, since most sequences are unique, the added noise will most likely be much larger than the actual occurrence counts, which reduces data utility considerably.

In this paper, we demonstrate that there is no need to release the noisy counts of all possible sequences in order to retain sequentiality information of a sequential dataset. Instead, we leverage on the well-established n-gram model, which has been heavily used in natural language processing [14]. With this model, it is often sufficient to publish the most common n-grams (contiguous subsequences of size n, where n is typically smaller than 5) for accurately "reconstructing" the original dataset. This fact positively impacts both privacy and utility. First, the universe of all grams with a small n value is relatively small (note that our approach does not even require to explore the entire universe of all n-grams), and thus we can employ the stronger ε-differential privacy model. Second, the counts of shorter grams are often large enough to resist noise. Finally, the inherent *Markov assumption* in the n-gram model allows to reduce the injected noise and therefore further improves utility.

Contributions. The major contributions of this paper are three-fold:

- For the first time, we introduce the n-gram model as an effective means of achieving differential privacy in the context of sequential data. To better suit differential privacy, we propose the use of a novel *variable-length* n-gram model, which balances the trade-off between information of the underlying database retained and the magnitude of Laplace noise added. The variable-length n-gram model intrinsically fits differential privacy in the sense that it retains the essential information of a sequential dataset in terms of a set of high-quality n-grams whose counts are large enough to resist Laplace noise.

- We develop a series of techniques to guarantee good utility under the variable-length n-gram model, including an adaptive privacy budget allocation scheme, a formal choice of a threshold value, and the enforcement of consistency constraints. These techniques make use of the inherent Markov assumption in an n-gram model. In addition, we develop an efficient method to generate a synthetic dataset from released n-grams, enabling a wider spectrum of data analysis tasks.

- We conduct an extensive experimental study on the variable-length n-gram model over real-life datasets, which provides important insights for future work. In particular, we demonstrate that our solution substantially outperforms the state-of-the-art techniques [16], [4] in terms of count query and frequent sequential pattern mining.

The rest of the paper is organized as follows. We provide a literature review in Section 2. Section 3 presents the preliminaries of our solution. Section 4 discusses our sanitization solution in detail. Section 5 gives the formal privacy guarantee of our solution. Comprehensive experimental results are reported in Section 6. Finally, we conclude the paper in Section 7.

2. RELATED WORK

Sequential data could be considered as a special type of trajectory data. Due to the ubiquitousness of trajectory (sequential) data, some recent works [1], [21], [24], [12], [5], [18] have been done on *privacy-preserving trajectory data publishing.* Abul et al. [1] propose the (k, δ)-anonymity model based on the inherent imprecision of sampling and positioning systems, where δ represents the possible location imprecision. They propose to modify trajectories by *space translation* so that k different trajectories co-exist in a cylinder of the radius δ. Terrovitis and Mamoulis [21] model an adversary's background knowledge as a set of projections of sequences in a sequential database and propose a data suppression technique to limit the confidence of inferring the presence of a location. Yarovoy et al. [24] propose to k-anonymize a moving object database (MOD) by considering timestamps as the quasi-identifiers. Adversaries are assumed to launch privacy attacks based on *attack graphs.* Monreale et al. [18] present an approach based on spatial generalization in order to achieve k-anonymity. They develop a generalization scheme that depends on the underlying trajectory dataset rather than a fixed grid hierarchy.

Hu et al. [12] present the problem of k-anonymizing a trajectory database with respect to a sensitive event database with the goal of ensuring that every event is shared by at least k users. They propose a new generalization mechanism known as *local enlargement*, which achieves better utility. Chen et al. [5] consider the emerging trajectory data publishing scenario, in which users' sensitive attributes are published with trajectory data, and consequently propose the $(K, C)_L$-privacy model that thwarts both identity linkages on trajectory data and attribute linkages via trajectory data. They develop a generic solution for various data utility metrics by use of *local suppression.* All these approaches [1], [21], [24], [12], [5], [18] are built based on partition-based privacy models, and therefore are not able to provide sufficient privacy protection for sequential data. Compared with the above works, the major contribution of our paper is the use of differential privacy, which provides significantly stronger privacy guarantees.

With the recent emergence of differential privacy, there has been extensive research on applying it to *non-interactive* privacy-preserving data publishing. Blum et al. [2] demonstrate that it is possible to release synthetic private databases that are useful for all queries over a discretized domain from a concept class with polynomial *Vapnik-Chervonenkis* dimension. However, their mechanism is not efficient, taking runtime complexity of $superpoly(|\mathcal{C}|, |\mathcal{I}|)$, where $|\mathcal{C}|$ is the size of a concept class and $|\mathcal{I}|$ the size of the universe. Dwork et al. [10] propose a recursive algorithm for generating a synthetic database with runtime complexity of $poly(|\mathcal{C}|, |\mathcal{I}|)$. This improvement, however, is still insufficient to handle real-life sequential datasets due to the exponential size of $|\mathcal{C}|$. Xiao et al. [23] propose a wavelet-transformation based approach for *relational data* to lower the magnitude of noise, rather than adding independent Laplace noise. Two recent papers [17], [6] point out that *data-dependent* approaches are more efficient and more effective for generating a dif-

Table 1: Dataset

Rec. #	Sequence
1	$I_2 \to I_3 \to I_1$
2	$I_2 \to I_3$
3	$I_3 \to I_2$
4	$I_2 \to I_3 \to I_1$
5	$I_3 \to I_2 \to I_1$
6	$I_2 \to I_3 \to I_1 \to I_2 \to I_3$
7	$I_3 \to I_2$
8	$I_3 \to I_1 \to I_2 \to I_3$

Table 2: 1-grams

Gram	#	Pr
I_1	5	0.21
I_2	9	0.38
I_3	10	0.41

Table 3: 2-grams

Gram	#	Pr	Gram	#	Pr	Gram	#	Pr
$I_1 \to I_1$	0	0	$I_2 \to I_1$	1	0.11	$I_3 \to I_1$	4	0.4
$I_1 \to I_2$	2	0.4	$I_2 \to I_2$	0	0	$I_3 \to I_2$	3	0.3
$I_1 \to I_3$	0	0	$I_2 \to I_3$	6	0.67	$I_3 \to I_3$	0	0
$I_1 \to$ &	3	0.6	$I_2 \to$ &	2	0.22	$I_3 \to$ &	3	0.3

ferentially private release. Mohammed et al. [17] propose a generalization-based sanitization algorithm for *relational data* with the goal of classification analysis. Chen et al. [6] propose a probabilistic top-down partitioning algorithm for *set-valued data*. Both approaches [17], [6] make use of taxonomy trees to adaptively narrow down the output domain. However, these approaches cannot be directly applied to sequential data due to its inherent sequentiality.

To our best knowledge, the paper [4] is the only work that applies differential privacy to sequential data release. They make use of the prefix tree structure to group sequences with the identical prefix into the same branch. However, with the growth of the prefix tree, the number of sequences falling into a branch decreases quickly, resulting in poor utility. In addition, though not dedicated to sequential data, McSherry and Mahajan [16] develop a method for finding frequent (sub)strings. This method also makes use of a prefix structure. In contrast, we make use of the variable-length n-gram model, which achieves significantly improved utility.

3. PRELIMINARIES

3.1 Sequential Database

Let $\mathcal{I} = \{I_1, I_2, \cdots, I_{|\mathcal{I}|}\}$ be the universe of items, where $|\mathcal{I}|$ is the size of the universe. The semantic meaning of an item could be different from application to application. For example, an item could be a station in a transportation system, a word in a natural language processing application, or a nucleobase in a DNA sequence. Each record in a sequential database is a sequence of (time-)ordered items drawn from the universe. For instance, a sequence can represent the movement history of a record owner (i.e., his trajectory), where each item corresponds to a location visited, or a user's password, where each item corresponds to a character.

Formally, a *sequence* S of length $|S|$ is an ordered list of items $S = L_1 \to L_2 \to \cdots \to L_{|S|}$, where $\forall 1 \le i \le |S|$, $L_i \in \mathcal{I}$. An item may occur multiple times in S, and may occur consecutively in S. Therefore, $S = I_1 \to I_2 \to I_2$ is a valid sequence. A *sequential database* $\mathcal{D}$ of size $|\mathcal{D}|$ is composed of a multiset of sequences $\mathcal{D} = \{S_1, S_2, \cdots, S_{|\mathcal{D}|}\}$. Table 1 presents a sample sequential database with $\mathcal{I} = \{I_1, I_2, I_3\}$.

3.2 N-Gram Model

An n-gram model is a type of probabilistic prediction model based on an $(n-1)$-order *Markov* model. It can compactly model large-scale sequential data and provide scalable trade-off between storage and accuracy. N-gram models have been proven to be very robust in modeling sequential data and have been widely used in probability, communication theory, computational linguistics (e.g., statistical natural language processing), computational biology (e.g., biological sequence analysis), and data compression.

N-gram models estimate the probability of the next item for a given sequence by making use of the *Markov independence assumption* (of order $n-1$) that the occurrence of each item in a sequence depends only on the previous $n-1$ items (instead of *all* previous items), where n is typically a small value (e.g., 3-5). Let the probability that a sequence $L_1 \to L_2 \to \ldots \to L_i$, where $L_j \in \mathcal{I}$ ($\forall 1 \le j \le i$) and $i \ge n$, is followed by $L_{i+1} \in \mathcal{I}$ be denoted by $P(L_{i+1}|L_1 \to L_2 \to \ldots \to L_i)$. Then, under the n-gram model, $P(L_{i+1}|L_1 \to L_2 \to \ldots \to L_i) :\approx P(L_{i+1}|L_{i-n+2} \to L_{i-n+3} \to \ldots \to L_i)$.

N-gram models provide a trade-off between storage and accuracy: a larger n value retains more information of the dataset, but it requires more storage and time to process. For example, Tables 2 and 3 show the set of all unigrams and 2-grams, respectively, along with their counts and probabilities for the sample dataset in Table 1, where & is a special symbol representing the termination of a sequence. Consider the calculation of the (approximate) number of occurrences of $I_3 \to I_1 \to I_2 \to I_3$, whose true number is 2. Using 2-grams, one possible approximation is $\#(I_3 \to I_1) \cdot P(I_2|I_1) \cdot P(I_3|I_2) = 4 \cdot 0.4 \cdot 0.67 = 1.07$. In contrast, using 3-grams, a better approximation could be $\#(I_3 \to I_1 \to I_2) \cdot P(I_3|I_1 \to I_2) = 2 \cdot 1.0 = 2.0$. However, this better scheme requires to process all 3-grams at the cost of storage and time.

3.3 Differential Privacy

Dwork proves that *absolute* privacy protection is impossible in the presence of background knowledge [7], resulting in the notion of differential privacy based on *indistinguishability*. Differential privacy [9] requires that the outcome of any computation be insensitive to the change of a single record. It follows that any information that can be learned from the database with a record can also be learned from the one without this record. Consequently, for a record owner, it means that any privacy breach will not be a result of participating in the database.

Definition 3.1 (Differential privacy). A privacy mechanism $\mathcal{A}$ gives ε-differential privacy if for any database $\mathcal{D}_1$ and $\mathcal{D}_2$ differing on at most one record, and for any possible output $O \in Range(\mathcal{A})$,

$$Pr[\mathcal{A}(\mathcal{D}_1) = O] \le e^{\varepsilon} \times Pr[\mathcal{A}(\mathcal{D}_2) = O] \quad (1)$$

where the probability is taken over the randomness of $\mathcal{A}$.

A fundamental concept for achieving differential privacy is the *global sensitivity* of a function [9] that maps an underlying database to (vectors of) reals.

Definition 3.2 (Global Sensitivity). For any function $f : \mathcal{D} \to \mathbb{R}^d$, the sensitivity of f is

$$\Delta f = \max_{\mathcal{D}_1, \mathcal{D}_2} ||f(\mathcal{D}_1) - f(\mathcal{D}_2)||_1 \quad (2)$$

for all $\mathcal{D}_1, \mathcal{D}_2$ differing in at most one record.

Laplace Mechanism. A standard mechanism to achieve differential privacy is to add properly calibrated Laplace noise to the true output of a function, which is known as *Laplace mechanism* [9]. It takes as inputs a database $\mathcal{D}$, a function f, and the privacy parameter ε. The noise is generated according to a Laplace distribution with the probability density function (pdf) $p(x|\lambda) = \frac{1}{2\lambda}e^{-|x|/\lambda}$, where λ is determined by both Δf and the desired privacy parameter ε. Let $\mathcal{L}(\lambda)$ denote a Laplace random variable with a probability density function defined as above.

THEOREM 3.1. *For any function $f : \mathcal{D} \to \mathbb{R}^d$, the mechanism*

$$Laplace(\mathcal{D}, f, \varepsilon) = f(\mathcal{D}) + [\mathcal{L}_1(\lambda), \mathcal{L}_2(\lambda), \ldots, \mathcal{L}_d(\lambda)] \quad (3)$$

gives ε-differential privacy if $\lambda = \Delta f/\varepsilon$ and $\mathcal{L}_i(\lambda)$ are i.i.d Laplace random variables.

4. SANITIZATION ALGORITHM

4.1 Overview

The main idea of our scheme is simple: we add properly calibrated Laplace noise to the counts of high-quality grams and release them. Our goal is two-fold: (1) to release grams whose real counts are large enough to increase utility[1], and (2) to maximize the sizes of released grams (i.e., the n value) to preserve as much sequentiality information as possible. There is a fundamental trade-off between the utility of noisy n-grams and their sizes: shorter grams enjoy smaller relative error due to Laplace noise but carry less sequentiality information; longer grams contain more sequentiality information but have smaller counts (and thus larger relative error). In this paper, we address this trade-off by releasing *variable-length* n-grams with counts larger than a threshold [2] and of sizes less than a maximal size n_{max} [3]. For most practical datasets, setting n_{max} to a small value (e.g., 3-5) has been sufficient to capture most of the sequentiality information. Since short grams are typically of large real counts, this property, which is also experimentally justified in Appendix A, explains why the n-gram model is so powerful and why it provides an excellent basis for differentially private sequential data publishing.

To identify the set of high-quality (i.e., having low relative error) n-grams with possibly varying n values ($1 \leq n \leq n_{max}$) from an input sequential dataset, we propose a well-designed tree structure, called *exploration tree*. It groups grams with the same prefix into the same branch so that all possible n-grams with size $1 \leq n \leq n_{max}$ can be explored efficiently. The exploration starts with unigrams and then proceeds to longer grams until n_{max} is reached. Intuitively, if the noisy count of a gram g is small (i.e., close to the standard deviation of the added noise), its real count also tends to be small and thus the relative error is large. Since all grams having the prefix g (i.e., all nodes in the subtree rooted at g) have smaller real counts than g's real count, they can be omitted from further computation. This observation makes our approach significantly faster than naively processing every single gram regardless of its size. It also explains why we do not adopt the approach that generates all possible n-grams and then prunes the tree.

Algorithm 1 Sequential Database Sanitization

```
Input: Raw sequential database $\mathcal{D}$
Input: Privacy budget $\varepsilon$
Input: Maximal sequence length $\ell_{max}$
Input: Maximal $n$-gram size $n_{max}$
Output: Private sequential database $\widetilde{\mathcal{D}}$
Truncate each $S \in \mathcal{D}$ by keeping the first $\ell_{max}$ items;
Create an exploration tree $\mathcal{T}$ with a virtual root;
$i = 0$;
while $i < n_{max}$ do
    for each non-leaf node $v_{ij} \in$ levelSet$(i, \mathcal{T})$
    and $lb(v_{ij}) \neq$ & do
        Calculate $\varepsilon_{v_{ij}}$;                //see Section 4.3.2
        $U_c \leftarrow$ all possible children of $v_{ij}$ with labels $\mathcal{I} \cup \{\&\}$;
        //Compute the noisy count of each $u_k \in U_c$
        $\mathbf{Q} = \{|g(u_1)|, |g(u_2)|, \cdots, |g(u_{|\mathcal{I}|+1})|\}$;
        $\widetilde{\mathbf{Q}} = Laplace(\mathcal{D}, \mathbf{Q}, \varepsilon_{v_{ij}})$;     //$\Delta\mathtt{Q} = \ell_{max}$
        for each node $u_k \in U_c$ do
            Add $u_k$ to $\mathcal{T}$;
            if $c(u_k) < \theta$ then            //see Section 4.3.3
                Mark $u_k$ as leaf;
    $i$++;
Enforce consistency on $\mathcal{T}$;           //see Section 4.3.4
Generate $\widetilde{\mathcal{D}}$ from $\mathcal{T}$;              //see Section 4.3.5
return $\widetilde{\mathcal{D}}$;
```

4.2 Terminology

We first give some notations used in our solution. The exploration tree is denoted by $\mathcal{T}$. Each node $v \in \mathcal{T}$ is labeled by an item $I \in \mathcal{I} \cup \{\&\}$, where & is a special symbol representing the termination of a sequence. The function $lb(v)$ returns v's item label. Each node v is associated with an n-gram defined by the sequence of items from the root of $\mathcal{T}$ to v, denoted by $g(v)$. We slightly abuse the term *count* to mean the number of occurrences of $g(v)$ in the input dataset, which is denoted by $|g(v)|$. Note that an n-gram may occur multiple times in a sequence. For example, the count of $I_2 \to I_3$ in the sample dataset in Table 1 is 6, instead of 5. Each node v also keeps a noisy version of $|g(v)|$, denoted by $c(v)$. In addition, each node v conveys a conditional probability, denoted by $P(v)$, which predicts the probability of the transition from v's parent to v. $P(v)$ can be obtained by normalizing the noisy counts of v's siblings and v. For example, in Figure 1, $P(v_5) = P(I_1|I_2 \to I_3) = 4/(4+0+1+2) = 4/7$. The set of all nodes in level i of $\mathcal{T}$ is denoted by levelSet$(i, \mathcal{T})$ and these nodes represent all i-grams in the dataset. The level number of node v in $\mathcal{T}$ is denoted by level$(v, \mathcal{T})$. The root of $\mathcal{T}$ is in level zero. In the sequel, the probability $P(L_{i+1}|L_j \to L_{j+1} \to \ldots \to L_i)$ is shortly denoted by $P(L_{i+1}|L_i^j)$.

4.3 Detailed Descriptions

4.3.1 Private Sequential Database Release

Algorithm 1 provides an overview of our approach. It takes as inputs a sequential database $\mathcal{D}$, the total privacy budget ε, the maximal sequence length ℓ_{max} and the maximal n-gram size n_{max}, and returns a sanitized sequential

[1] The added noise is calibrated to the global sensitivity and is independent of the count values. Thus, larger counts provide better utility in terms of relative error.

[2] This threshold is set to limit the magnitude of noise in released data.

[3] Beyond n_{max}, the utility gain of longer grams is usually smaller than the utility loss due to noise.

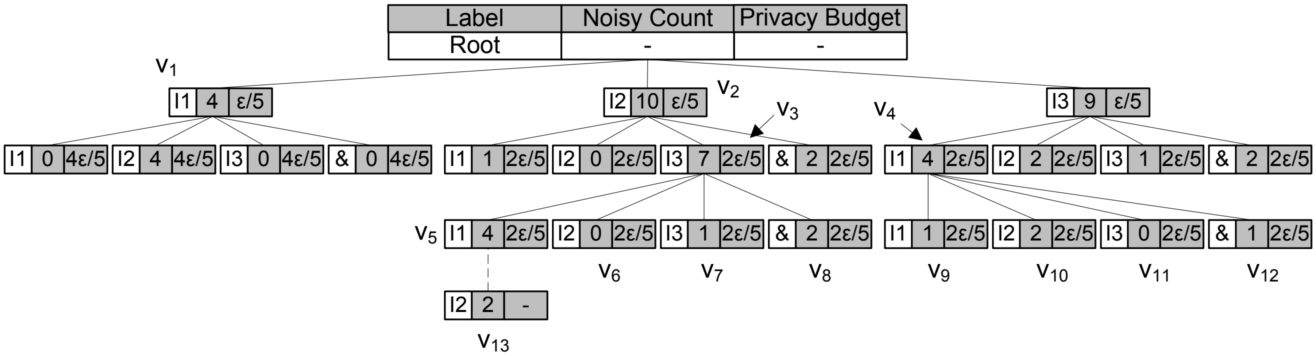


Figure 1: The exploration tree of the sample data

database $\widetilde{\mathcal{D}}$ satisfying ε-differential privacy. ℓ_{max} is a parameter specified by the data holder to limit the influence of a single sequence in computation. The algorithm considers only the first ℓ_{max} items in each input sequence. A larger ℓ_{max} allows more information to be retained from $\mathcal{D}$, but requires more noise to be injected in later computation; a smaller ℓ_{max} does the opposite. We discuss and report the effect of different ℓ_{max} values in Section 6, and provide insights for a data holder to select a good ℓ_{max} value in practice. n_{max} bounds the height of the exploration tree $\mathcal{T}$ and thus the maximal size of released grams. The choice of n_{max} affects the privacy parameter assigned to each level of $\mathcal{T}$, and, therefore, is also related to the magnitude of noise. In practice, n_{max} could be set to 5, which is the maximal n value popularly used in the literature. Similarly, we present more details on the selection of a reasonable n_{max} in Section 6. We emphasize that this does not mean that all released grams have a size of n_{max} but rather their sizes can vary between 1 and n_{max}.

In Algorithm 1, we first preprocess $\mathcal{D}$ by keeping only the first ℓ_{max} items of each sequence in order to bound the influence of a single sequence by ℓ_{max} (Line 1). The construction of $\mathcal{T}$ starts by creating an empty tree with a virtual root (Line 2). In Lines 4-15, the algorithm iteratively constructs each level of $\mathcal{T}$. For level i of $\mathcal{T}$, we decide whether to expand a node $v_{ij} \in \mathsf{levelSet}(i, \mathcal{T})$ by comparing its noisy count $c(v_{ij})$ with a threshold θ. If $c(v_{ij}) \geq \theta$, we expand v_{ij} by explicitly considering every possible item in $\mathcal{I} \cup \{\&\}$ as a child of v_{ij} in order to satisfy differential privacy. By definition, nodes labeled by & cannot be expanded because it means the termination of a sequence. The entire exploration process ends when either the depth of the tree reaches n_{max} or no node can be further expanded (since their noisy counts do not pass θ or their privacy budgets have run out). Example 4.1 illustrates the construction of a possible exploration tree on the sample dataset in Table 1.

EXAMPLE 4.1. *Given $n_{max} = 5$, $\ell_{max} = 5$ and $\theta = 3$, the construction of a possible exploration tree over the sample dataset in Table 1 is illustrated in Figure 1 (ignore the privacy budget information and node v_{13} for now).*

In the following, we detail the key components of Algorithm 1: how to compute the privacy budget ε_{ij} for each node in $\mathcal{T}$ (Section 4.3.2), how to compute the threshold θ for each node (Section 4.3.3), how to make $\mathcal{T}$ consistent (Section 4.3.4), and how to generate a synthetic version of the input database $\mathcal{D}$ from $\mathcal{T}$ (Section 4.3.5). Finally, in Section 5, we prove the privacy guarantee of our scheme.

4.3.2 Adaptive Privacy Budget Allocation

Given the maximal gram size n_{max}, a simple privacy budget allocation scheme is to expect the height of $\mathcal{T}$ to be n_{max} and uniformly assign $\frac{\varepsilon}{n_{max}}$ to each level of $\mathcal{T}$ to calculate the noisy counts of the nodes in each level. However, in reality, many (or even all) root-to-leaf paths have a length much shorter than n_{max} for the reason of their counts not being able to pass θ. Hence assigning privacy parameters solely based on n_{max} is clearly not optimal. For example, in Example 4.1, since the height of the exploration tree is 3 and $n_{max} = 5$, at least $\frac{2\varepsilon}{5}$ privacy budget would be wasted in all paths.

To address this drawback, we propose an adaptive privacy budget allocation scheme that allows private operations to make better use of the total privacy budget ε. Intuitively, a desirable privacy budget allocation scheme should take into consideration the length of a root-to-leaf path: for a shorter path, each node in the path should receive more privacy budget; for a longer path, each node should use less privacy budget. Hence we adaptively estimate the length of a path based on known *noisy* counts and then distribute the remaining privacy budget as per the estimated length.

At the beginning of the construction of $\mathcal{T}$, in the absence of information from the underlying dataset, we can only assume that each root-to-leaf path is of the same length n_{max} so that our algorithm would not exceptionally halt due to running out of privacy budget. Therefore, $\frac{\varepsilon}{n_{max}}$ is used to calculate the noisy counts of nodes in level 1. Once we obtain some information from the underlying dataset (e.g., nodes' noisy counts), we can make more accurate predictions on the length of a path.

For a node v in level $i \geq 2$ with noisy count $c(v)$, we predict the height h_v of the subtree rooted at v, denoted by $\mathcal{T}_v$, as follows. Let P_{max} be the estimation of the probability of transiting from v to the *mode* of its children (i.e., v's child with the largest noisy count). Assume that the probability of the mode at *each* level of $\mathcal{T}_v$ is also P_{max} [4]. Under this assumption, we can estimate the largest noisy count of the nodes in level h_v of $\mathcal{T}_v$ by $c(v) \cdot (P_{max})^{h_v}$. Recall the fact that $\mathcal{T}_v$ will not be further expanded if none of the nodes in level h_v can pass the threshold θ. We get $c(v) \cdot (P_{max})^{h_v} = \theta$,

[4] A more precise estimation could be obtained by applying the Markov assumption to each level of $\mathcal{T}_v$ at the cost of efficiency.

that is, $h_v = \log_{P_{max}} \frac{\theta}{c(v)}$. Since the height of $\mathcal{T}_v$ is bounded by $n_{max} - i$, we have

$$h_v = \min(\log_{P_{max}} \frac{\theta}{c(v)}, n_{max} - i).$$

Next we discuss how to calculate P_{max} for v. Let the i-gram associated with v be $L_1 \to L_2 \to \cdots \to L_i$ ($\forall 1 \leq j \leq i$, $L_j \in \mathcal{I} \cup \{\&\}$). Then we need to estimate the probability distribution of v's children from the noisy counts known by far. We resort to the Markov assumption for this task. Recall that the order $i-1$ Markov assumption states $P(L_{i+1}|L_i^1) :\approx P(L_{i+1}|L_i^2)$. Since $P(L_{i+1}|L_i^2)$ may *not* be known in $\mathcal{T}$ (because we expand a node only when it passes the threshold θ), we consider a chain of Markov assumptions (of different orders)

$$P(L_{i+1}|L_i^1) :\approx P(L_{i+1}|L_i^2) :\approx P(L_{i+1}|L_i^3) :\approx \cdots :\approx P(L_{i+1})$$

to find the best estimation of $P(L_{i+1}|L_i^1)$, which is the conditional probability with the longest condition (i.e., the leftmost conditional probability in the chain) that is known in $\mathcal{T}$. Since $\mathcal{T}$ contains all unigrams, there is always an estimation of $P(L_{i+1}|L_i^1)$, denoted by $\widetilde{P}(L_{i+1}|L_i^1)$. P_{max} is then defined to be

$$\max_{L_{i+1} \in \mathcal{I} \cup \{\&\}} \widetilde{P}(L_{i+1}|L_i^1).$$

If $P(L_{i+1}|L_i^1)$ and $\widetilde{P}(L_{i+1}|L_i^1)$ are represented by nodes v and v' in $\mathcal{T}$, respectively, then v' is the *Markov parent* of v in $\mathcal{T}$, and any pair of corresponding nodes in the above chain are *Markov neighbors*. For example, in Figure 1, v_4 is the Markov parent of v_5; v_5 and v_1 are Markov neighbors.

Once P_{max} is calculated, we can calculate the privacy parameter ε_v that is used for calculating the noisy counts of v's children as follow:

$$\varepsilon_v = \frac{\bar{\varepsilon}}{\min(\log_{P_{max}} \frac{\theta}{c(v)}, n_{max} - i)},$$

where $\bar{\varepsilon}$ is the remaining privacy budget (i.e., the total privacy budget ε minus the sum of privacy parameters consumed by v and v's ancestors). It can be observed that this scheme ensures that the privacy budget used in a root-to-leaf path is always $\leq \varepsilon$.

EXAMPLE 4.2. *Continue from Example 4.1. For all nodes in level 1, $\frac{\varepsilon}{5}$ is used to calculate their noisy counts. For the expansion of the node labeled by v_1 in Figure 1, we have $P_{max} = \frac{10}{4+10+9} = 0.43$ and $h_{v_1} = 1$. Therefore, the noisy counts of v_1's children are calculated with privacy parameter $\varepsilon - \frac{\varepsilon}{5} = \frac{4\varepsilon}{5}$. For the expansion of node v_2, we get $P_{max} = \frac{10}{4+10+9} = 0.43$ and $h_{v_2} = 2$. Hence its children's noisy counts are calculated with privacy parameter $\frac{\varepsilon - \frac{\varepsilon}{5}}{2} = \frac{2\varepsilon}{5}$. For the expansion of node v_3, we have $P_{max} = \frac{4}{9}$ and $h_{v_3} = 1$. Thus, $\frac{2\varepsilon}{5}$ is used to compute the noisy counts of v_3's children.*

The sensitivities of $\mathbf{Q}$ (Line 9) in different levels are different. For $\mathbf{Q}$ in level i, a single record of length $\leq \ell_{max}$ can change $\mathbf{Q}$ by at most $\ell_{max} - i + 1$. However, under the adaptive privacy budget allocation scheme, we have to use the largest sensitivity among all levels, that is ℓ_{max}, in all Laplace mechanisms; otherwise, ε-differential privacy may be violated.

4.3.3 Computing Threshold θ

A node in $\mathcal{T}$ is not further expanded if its noisy count is less than the threshold θ. The main source of error in $\mathcal{T}$ comes from the nodes that are of a true count of zero but of a noisy count greater than θ (referred to as *false nodes*). For this reason, we design a threshold to limit the total number of false nodes in $\mathcal{T}$ with the goal of lowering the magnitude of noise in $\mathcal{T}$.

For each expansion, a false node v will generate, on average, $|\mathcal{I}|P_\theta$ false children, where P_θ is the probability of Laplace noise passing θ. This is because a descendant of v must have a true count of zero. With the expansion of $\mathcal{T}$, the number of false nodes accumulates *exponentially* with the factor of $|\mathcal{I}|P_\theta$, resulting in excessive noise. To limit the exponential growth of false nodes, we require $|\mathcal{I}|P_\theta \leq 1$, that is, $P_\theta \leq \frac{1}{|\mathcal{I}|}$. Since, under Laplace mechanism, given the threshold θ and the privacy parameter ε',

$$P_\theta = \int_\theta^\infty \frac{\varepsilon'}{2\ell_{max}} \exp\left(-\frac{x\varepsilon'}{\ell_{max}}\right) dx = \frac{1}{2} \exp\left(-\frac{\varepsilon'\theta}{\ell_{max}}\right),$$

we get the threshold $\theta = \frac{\ell_{max} \cdot \ln \frac{|\mathcal{I}|}{2}}{\varepsilon'}$. We show in Section 6 that this threshold is effective in eliminating false nodes while having limited influence on nodes with large counts.

4.3.4 Enforcing Consistency Constraints

The generated exploration tree $\mathcal{T}$ may contain some inconsistencies for the reason that: (1) the sum of children's noisy counts is very unlikely to equal their parent's noisy count, and (2) there are some leaf nodes whose noisy counts are missing (since their counts cannot pass the threshold θ). In this section, we propose a method to resolve such inconsistencies with the goal of improving data utility. In Appendix A, we experimentally show that this method helps achieve better performance.

The general idea is to approximate the missing counts by making use of the Markov assumption and then normalize children's counts based on their parent's count. More specifically, our method works as follows. If none of the children of a node v in $\mathcal{T}$ exceed the threshold θ, it is strong evidence that v should not be further expanded, and therefore all children of v (leaf nodes in $\mathcal{T}$) are assigned noisy counts 0. If all children pass θ, we first calculate the conditional probability of each child based on the sum of all children's noisy counts, and then obtain a consistent approximation by multiplying this probability with their parent's noisy count. If some children (but *not* all) of v pass θ, we approximate the noisy counts of the other children by the Markov assumption. Let v_c and $\mathcal{C}(v_c)$ denote a child of v whose noisy count cannot pass θ (called a *missing node*) and its Markov parent in $\mathcal{T}$, respectively. Let V denote the set of v's children. We partition V into V^+ and V^-, where V^+ contains all nodes passing the threshold, whereas V^- contains the rest.

1. Define the following ratio for each $v_i \in V^-$:

$$r_{v_i} = \frac{P(\mathcal{C}(v_i))}{\sum_{v_j \in V^+} P(\mathcal{C}(v_j))}$$

For each $v_j \in V^+$, let $A(v_j)$ denote the noisy count resulted by the Laplace mechanism in Line 10 of Algorithm 1.

2. If $\mathsf{level}(\mathcal{C}(v_c), \mathcal{T}) \geq 2$, $\forall v_i \in V^-$,
$$A(v_i) = r_{v_i} \cdot \sum_{v_j \in V^+} A(v_j)$$
3. Otherwise,
   (a) If $\sum_{v_j \in V^+} A(v_j) \leq c(v)$, $\forall v_i \in V^-$,
   $$A(v_i) = \frac{c(v) - \sum_{v_j \in V^+} A(v_j)}{|V^-|}$$
   (b) Otherwise, $\forall v_i \in V^-$, $A(v_i) = 0$
4. Renormalize: $\forall v_i \in V$, $c(v_i) = c(v) \cdot \frac{A(v_i)}{\sum_{v_j \in V} A(v_j)}$

If v_c can find a high-quality Markov parent in $\mathcal{T}$ (i.e., one representing an n-gram with $n \geq 2$ [5]), we estimate its counts from its high-quality siblings based on the ratio defined in Step 1. The idea behind this definition is that the *ratio* of any node insignificantly changes between Markov neighbors, and hence, it can be well approximated from the Markov parents. Otherwise, we approximate the noisy counts by assuming a uniform distribution, that is, equally distribute the count left among the missing nodes (Step 3). In Step 4, these estimated counts are normalized by the parent's count in order to obtain consistent approximations.

EXAMPLE 4.3. *Continue from Example 4.1. Suppose that* $A(v_9) = 2.1$*,* $A(v_{10}) = 4$*,* $A(v_{11})$ *does not pass* θ*, and* $A(v_{12}) = 1.9$*. Since* $r_{v_{11}} = 0/(4+0+0) = 0$*,* $A(v_{11}) :\approx (1.9 + 4 + 2.1) \cdot 0 = 0$*. Finally, renormalizing the result, we obtain* $c(v_9) = 4 \cdot 2.1/(2.1 + 4 + 1.9 + 0) \approx 1, c(v_{10}) = 4 \cdot 4/(2.1 + 4 + 1.9 + 0) = 2, c(v_{11}) = 0, c(v_{12}) = 4 \cdot 1.9/(2.1 + 4 + 1.9 + 0) \approx 1$*.*

4.3.5 Synthetic Sequential Database Construction

The released n-grams are useful for many data analysis tasks. However, it is often necessary to generate a synthetic database for different types of queries and tasks. In this section, we propose an efficient solution to construct a synthetic sequential database from the exploration tree $\mathcal{T}$ (Line 17). The general idea is to iteratively generate longer grams (up to size ℓ_{max}) based on the Markov assumption and make use of the theorem below for synthetic sequential database construction.

THEOREM 4.1. *Given the set of* n*-grams with size* $1 \leq n \leq \ell_{max}$*, the (truncated) input database (with maximal sequence length* ℓ_{max}*) can be uniquely reconstructed.*

PROOF. (Sketch) Since ℓ_{max}-grams can only be supported by sequences of length ℓ_{max}, all sequences of length ℓ_{max} can be uniquely reconstructed by ℓ_{max}-grams. Once all ℓ_{max} sequences have been identified, we can update the n-gram counts by decreasing the numbers of occurrences of the n-grams in ℓ_{max} sequences. The resulting set of n-grams ($1 \leq n \leq \ell_{max} - 1$) can then be considered as if they were generated from an input database with maximal sequence length $\ell_{max} - 1$. Therefore, sequences of length $\ell_{max} - 1$ can be uniquely reconstructed as well. Following this iterative process, all sequences can be uniquely identified. This proof explains the way we generate the synthetic database based on noisy n-grams. □

Intuitively, longer grams can be generated by "joining" shorter grams. Formally, we define a *join* operation over two n-grams. Let $g_1 = L_{11} \to L_{12} \to \cdots \to L_{1n}$ and $g_2 = L_{21} \to L_{22} \to \cdots \to L_{2n}$. Then g_1 can join with g_2 if $\forall 2 \leq i \leq n$, $L_{1i} = L_{2(i-1)}$, denoted by $g_1 \bowtie g_2$, and $g_1 \bowtie g_2 = L_{11} \to L_{12} \to \cdots \to L_{1n} \to L_{2n}$. Note that the join operation is *not* symmetric: it is possible that g_1 can join with g_2, but not vice versa.

Let the height of $\mathcal{T}$ be h. We iteratively extend $\mathcal{T}$ by generating n-grams with $h < n \leq \ell_{max}$, starting from level h of $\mathcal{T}$. We extend $\mathcal{T}$ level by level, where level $n+1$, representing all $(n+1)$-grams, can be generated by joining all possible n-grams in level n. Let g_1 and g_2 be two n-grams that can be joined. Then we can estimate the count of the joined $(n+1)$-gram as follows:

$$\begin{aligned}
|g_1 \bowtie g_2| &= c(g_1) \times P(L_{2n}|g_1) \\
&= c(g_1) \times P(L_{2n}|L_{11} \to L_{12} \to \cdots \to L_{1n}) \\
&\approx c(g_1) \times P(L_{2n}|L_{12} \to L_{13} \to \cdots \to L_{1n}) \\
&= c(g_1) \times P(L_{2n}|L_{21} \to L_{22} \cdots \to L_{2(n-1)}) \\
&\approx c(g_1) \times \frac{c(L_{2n}^{21})}{c(L_{2(n-1)}^{21})}
\end{aligned}$$

Note that all counts in the above equation are noisy ones for the reason of privacy (see more details in Section 5). Since $c(L_{2n}^{21})$ and $c(L_{2(n-1)}^{21})$ must have been known in the extended $\mathcal{T}$, $|g_1 \bowtie g_2|$ can be computed. We keep extending $\mathcal{T}$ until: 1) ℓ_{max} has been reached, or; 2) no grams in a level can be joined.

EXAMPLE 4.4. *Continue from Example 4.1.* $I_2 \to I_3 \to I_1$ *and* $I_3 \to I_1 \to I_2$ *can be joined to generate* $I_2 \to I_3 \to I_1 \to I_2$*. Its count can be estimated by* $c(I_2 \to I_3 \to I_1) \times \frac{c(I_3 \to I_1 \to I_2)}{c(I_3 \to I_1)} = 4 \times \frac{2}{4} = 2$*. This 4-gram is represented as a new node in* $\mathcal{T}$*, as illustrated by* v_{13} *in Figure 1. Similarly,* $I_2 \to I_3 \to I_1$ *can join with* $I_3 \to I_1 \to I_1$*, resulting in* $I_2 \to I_3 \to I_1 \to I_1$*. Since these two 4-grams cannot be joined, the extension of* $\mathcal{T}$ *ends at level 4.*

After extending $\mathcal{T}$, we can generate the synthetic database in the following way. Let the height of the extended $\mathcal{T}$ be h_e. We start from level h_e. For each $v \in \mathsf{levelSet}(h_e, \mathcal{T})$, we publish $c(v)$ copies of $g(v)$, and update the counts of all nodes supported by $g(v)$ (i.e., all nodes representing a gram that can be generated from $g(v)$). An n-gram supports *at most* $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ nodes and therefore requires *at most* $\frac{n(n+1)}{2}$ updates. With a hash map structure, each update can be done in constant time. Moreover, this reconstruction process can further be improved by publishing $c(v)$ only if updating the count of each supported node does not result in negative values. This optimization is also applied in our performance evaluations in Section 6.

EXAMPLE 4.5. *Continue from Example 4.4. For node* v_{13} *in level 4 of* $\mathcal{T}$*, we publish 2 copies of* $I_2 \to I_3 \to I_1 \to I_2$ *and update the counts of all nodes supported by* $g(v_{13})$*, that is, the nodes representing* I_1*,* I_2*,* I_3*,* $I_2 \to I_3$*,* $I_3 \to I_1$*,* $I_1 \to I_2$*,* $I_2 \to I_3 \to I_1$ *and* $I_3 \to I_1 \to I_2$*.*

5. PRIVACY ANALYSIS

We give the privacy guarantee of our approach below.

THEOREM 5.1. *Algorithm 1 satisfies* ε*-differential privacy.*

[5] A unigram conveys an unconditional probability and therefore cannot provide a very accurate estimation.

Proof. (Sketch) Due to the correlation of the counts in the same level of $\mathcal{T}$ (i.e., a single sequence can affect multiple counts in a level), the *sequential composition* and *parallel composition* properties [15] must be applied with caution. Hence, we prove the theorem by the definition of ε-differential privacy. Consider two neighboring databases $\mathcal{D}$ and $\mathcal{D}'$. We first consider Lines $1-15$ of Algorithm 1, that is, the construction of $\mathcal{T}$. Let this part be denoted by $\mathcal{A}$. Then we need to prove $\frac{Pr[\mathcal{A}(\mathcal{D})=\mathcal{T}]}{Pr[\mathcal{A}(\mathcal{D}')=\mathcal{T}]} \leq e^{\varepsilon}$. In essence, $\mathcal{T}$ is built on the noisy answers to a set of count queries (via Laplace mechanism). Let V denote the set of all possible nodes that can appear in $\mathcal{T}$ (i.e., $|V| = |\mathcal{I}|^{n_{max}}$). A node v's privacy parameter is denoted by ε_v, and its count in $\mathcal{D}$ and $\mathcal{D}'$ by $Q(\mathcal{D}, v)$ and $Q(\mathcal{D}', v)$, respectively. Then we have

$$\frac{Pr[\mathcal{A}(\mathcal{D}) = \mathcal{T}]}{Pr[\mathcal{A}(\mathcal{D}') = \mathcal{T}]} = \prod_{v \in V} \frac{\exp(-\varepsilon_v \frac{|c(v)-Q(\mathcal{D},v)|}{\ell_{max}})}{\exp(-\varepsilon_v \frac{|c(v)-Q(\mathcal{D}',v)|}{\ell_{max}})} \qquad (4)$$

Note that a single record can only affect *at most* ℓ_{max} root-to-leaf paths. Let these affected paths be indexed by i, and let $\pi(i)$ denote the set of all nodes along such a path i.

Equation 4 could be rewritten as

$$\begin{aligned} \frac{Pr[\mathcal{A}(\mathcal{D}) = \mathcal{T}]}{Pr[\mathcal{A}(\mathcal{D}') = \mathcal{T}]} &= \prod_{v \in V} \frac{\exp(-\varepsilon_v \frac{|c(v)-Q(\mathcal{D},v)|}{\ell_{max}})}{\exp(-\varepsilon_v \frac{|c(v)-Q(\mathcal{D}',v)|}{\ell_{max}})} \\ &\leq \exp\left(\frac{\sum_{v \in V} \varepsilon_v |Q(\mathcal{D}, v) - Q(\mathcal{D}', v)|}{\ell_{max}}\right) \\ &= \exp\left(\frac{\sum_{i=1}^{\ell_{max}} \sum_{v \in \pi(i)} \varepsilon_v}{\ell_{max}}\right) \end{aligned}$$

where the last equality is due to the fact that $|Q(\mathcal{D}, v) - Q(\mathcal{D}', v)|$ is the number of *affected* root-to-leaf paths that traverse v. Hence, each affected path increases the nominator with the total privacy budget of nodes that it traverses. Since $\sum_{v \in \pi(i)} \varepsilon_v = \varepsilon$ for all $1 \leq i \leq \ell_{max}$, we have

$$\begin{aligned} \frac{Pr[\mathcal{A}(\mathcal{D}) = \mathcal{T}]}{Pr[\mathcal{A}(\mathcal{D}') = \mathcal{T}]} &\leq \exp\left(\frac{1}{\ell_{max}} \sum_{j=1}^{\ell_{max}} \varepsilon\right) \\ &\leq e^{\varepsilon} \end{aligned}$$

Note that ε_v is calculated based on *noisy* counts. Hence, the construction of $\mathcal{T}$ satisfies ε-differential privacy. In addition to the construction of $\mathcal{T}$, we enforce consistency constraints on $\mathcal{T}$ and generate the synthetic sequential database in Lines 16 and 17. Since these two steps are conducted on noisy data and do not require access to the original database, they satisfy 0-differential privacy. Therefore, our solution as a whole gives ε-differential privacy. □

6. PERFORMANCE ANALYSIS

6.1 Error Analysis

The error of the sanitized data comes from three major sources: first, using n-grams with $1 \leq n \leq h$ to estimate longer n-grams with $h < n \leq \ell_{max}$ (recall that h is the height of the unextended tree, see Section 4.3.5); second, the truncation conducted to limit the effect of a single sequence; third, the noise added to the n-grams with $1 \leq n \leq h$ to satisfy differential privacy. We call the first two types of error *approximation error* and the last type of error *Laplace error*. Given a specific ε value, the total error of our approach is determined by ℓ_{max} and n_{max}. Intuitively, a smaller ℓ_{max} value incurs larger approximation error, but meanwhile it introduces less Laplace error because of a smaller sensitivity. Analogously, a smaller n_{max} value causes larger approximation error, but results in more accurate counts. Therefore our goal is to identify good values for ℓ_{max} and n_{max} that minimize the sum of approximation error and Laplace error. Due to the space limit, we experimentally study the effect of varying ℓ_{max} and n_{max} values on the performance of our solution and provide our insights in Appendix A. In general, our solution is designed to perform stably well under a relatively wide range of ℓ_{max} and n_{max} values. In the rest of this section, we only report the major results of our solution in terms of count query and frequent sequential pattern mining.

Table 4: Experimental dataset characteristics.

Datasets	$\|\mathcal{D}\|$	$\|\mathcal{I}\|$	$max\|S\|$	$avg\|S\|$
MSNBC	989,818	17	14,795	5.7
STM	1,210,096	342	121	6.7

6.2 Experimental Evaluation

We experimentally evaluate the performance of our solution (referred to as *N-gram*) in terms of two data analysis tasks, namely *count query* and *frequent sequential pattern mining*. As a reference point, for count query, we compare the utility of our solution with the approach proposed in [4], which relies on a prefix tree structure (referred to as *Prefix*); for frequent sequential pattern mining, we compare our approach with both *Prefix* and the method designed in [16] for finding frequent (sub)strings (referred to as *FFS*). Two real-life sequential datasets are used in our experiments. *MSNBC* describes sequences of URL categories browsed by users in time order on *msnbc.com*. It is publicly available at the UCI machine learning repository[6]. *STM* records sequences of stations visited by passengers in time order in the Montreal transportation system. It is provided by the *Société de transport de Montréal*[7]. The detailed characteristics of the datasets are summarized in Table 4, where $|\mathcal{D}|$ is the number of records (sequences) in $\mathcal{D}$, $|\mathcal{I}|$ is the universe size, $max|S|$ is the maximum length of sequences in $\mathcal{D}$, and $avg|S|$ is the average length of sequences.

6.2.1 Count Query

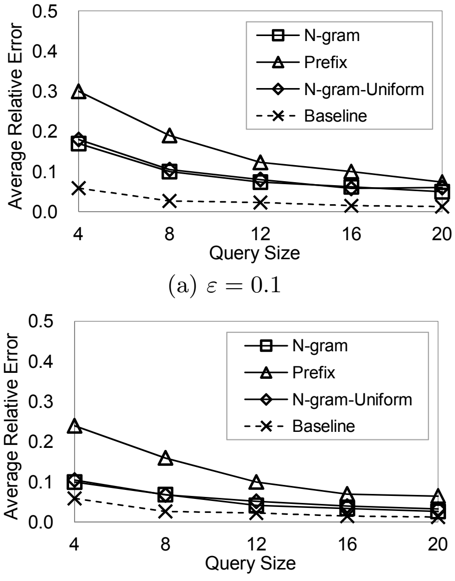
To evaluate the performance of our approach for count queries, we follow the evaluation scheme that has been widely used in previous works [23], [22], [6]. The utility of a count query Q is measured by the *relative error* of its answer on the sanitized sequential database $Q(\widetilde{\mathcal{D}})$ with respect to the true answer on the original database $Q(\mathcal{D})$, which is formalized as follows:

$$error(Q(\widetilde{\mathcal{D}})) = \frac{|Q(\widetilde{\mathcal{D}}) - Q(\mathcal{D})|}{\max\{Q(\mathcal{D}), s\}},$$

where s is a *sanity bound* that mitigates the effect of the queries with extremely small *selectivities* [23], [22]. Following the convention, s is set to 0.1% of $|\mathcal{D}|$, the same setting

[6] http://archive.ics.uci.edu/ml/

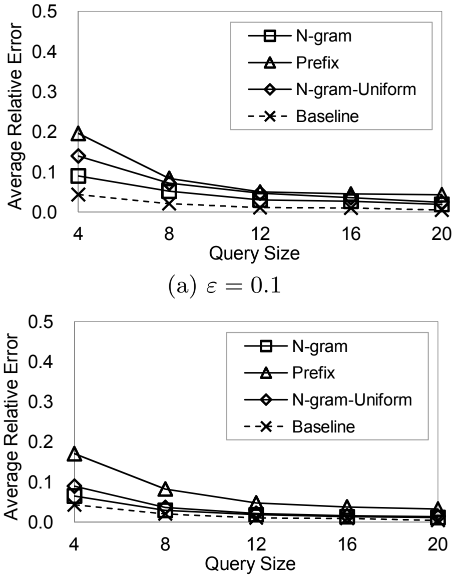
[7] http://www.stm.info



(a) $\varepsilon = 0.1$

(b) $\varepsilon = 1.0$

Figure 2: Average relative error vs. ε on *MSNBC*



(a) $\varepsilon = 0.1$

(b) $\varepsilon = 1.0$

Figure 3: Average relative error vs. ε on *STM*

as [23], [22], [6]. In particular, the answer to Q is defined to be the number of occurrences of Q in a database. For example, given $Q = I_2 \rightarrow I_3$, its answer over the dataset in Table 1 is 6. This type of count queries plays an important role in many applications, for example, calculating riderships in a transportation system.

In the first set of experiments, we examine the average relative errors of count queries under different query sizes (i.e., the number of items in a query) and different privacy budgets. We divide all queries into five subsets with different maximal query sizes (4, 8, 12, 16 and 20). For each subset, we generate 10,000 random queries of sizes that are uniformly distributed at random between 1 and its maximal size. Each item in a query is uniformly selected at random from the item universe.

Figures 2 and 3 report the average relative errors of different schemes under different query sizes over two typical ε values [8] while fixing $\ell_{max} = 20$ and $n_{max} = 5$. It can be observed that the average relative errors of *N-gram* are consistently lower than those of *Prefix* under all settings. The improvements are substantial, ranging from 32% to 63%. The relative errors of *N-gram* are relatively small even under a strong privacy requirement (i.e., $\varepsilon = 0.1$).

To demonstrate the effectiveness of the n-gram model, we apply the synthetic database generation technique described in Section 4.3.5 on *non-noisy* 5-grams of both *MSNBC* and *STM*, and issue count queries on the two synthetic databases (referred to as *Baseline*). The average relative errors of *Baseline* give the approximation error due to the employment of the n-gram model, while the differences between *Baseline* and *N-gram* ascribe to Laplace error. As one can observe, the approximation errors are relatively small on both datasets, demonstrating that the n-gram model is effective in capturing the essential sequentiality information of a database. For Laplace error, we stress that the n-gram model provides a general and flexible framework that can accommodate other more advanced noise injection mechanisms, such as the *matrix mechanism* [13] and the *MWEM mechanism* [11]. Hence it may require less noise added than Laplace mechanism, resulting in smaller Laplace error. It may even allow a larger n_{max} value to be used and therefore further reduce approximation error. Thus, we deem that the variable-length n-gram model bears great promise for differentially private sequential data release.

To prove the benefit of our adaptive privacy budget allocation scheme, we report the average relative errors of a variant of *N-gram* (referred to as *N-gram-Uniform*), in which the adaptive allocation scheme is replaced by the uniform allocation scheme described in Section 4.3.2. The improvement is less obvious on *MSNBC* because many paths are actually of length n_{max}, whereas the improvement on *STM* is noticeable, especially when $\varepsilon = 0.1$.

Due to the truncation operation conducted in Algorithm 1, any count query with a size greater than ℓ_{max} receives an answer 0 on the sanitized dataset. However, we point out that in reality it is *not* a problem because the true answer of such a query is typically very small (if not 0). For many real-life analyses (e.g., ridership analysis), the difference between such a small value and 0 is negligible. In addition, this limitation also exists in *Prefix* and is inherent in any differentially private mechanism because Laplace mechanism cannot generate reliable answers on extremely small values.

6.2.2 Frequent Sequential Pattern Mining

The second data analysis task we consider is frequent sequential pattern mining, a more specific data mining task. Given a positive integer number K, we are interested in the top K most frequent sequential patterns (i.e., most frequent subsequences) in the dataset. This data analysis task helps, for example, a transportation agency better understand passengers' transit patterns and consequently optimize its network geometry.

We compare the performance of *N-gram* with *Prefix* and *FFS*. All size-1 frequent patterns are excluded from the results since they are of less interest and trivial in frequent sequential pattern mining. We would like to clarify that *FFS*

[8] According to [16], $\varepsilon = 0.1$ and $\varepsilon = 1.0$ correspond to *high* and *medium* privacy guarantees, respectively.

Table 5: True positive ratio vs. K value on $MSNBC$

(a) $\varepsilon = 0.1$

K value	20	40	60	80	100
N-gram	100%	90%	93%	96%	94%
Prefix	85%	78%	80%	84%	86%
FFS	70%	63%	57%	58%	55%

(b) $\varepsilon = 1.0$

K value	20	40	60	80	100
N-gram	100%	93%	97%	99%	97%
Prefix	90%	82.5%	85%	90%	89%
FFS	70%	63%	57%	58%	55%

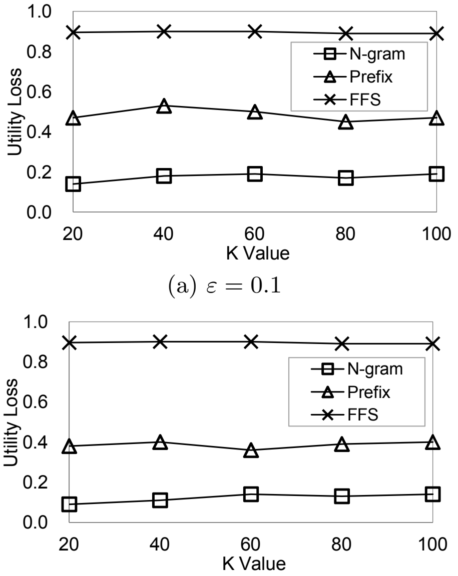
Table 6: True positive ratio vs. K value on STM

(a) $\varepsilon = 0.1$

K value	20	40	60	80	100
N-gram	95%	93%	93%	94%	91%
Prefix	65%	68%	75%	83%	82%
FFS	35%	33%	35%	36%	43%

(b) $\varepsilon = 1.0$

K value	20	40	60	80	100
N-gram	100%	100%	98%	100%	98%
Prefix	70%	68%	80%	86%	85%
FFS	35%	33%	35%	36%	43%



(a) $\varepsilon = 0.1$

(b) $\varepsilon = 1.0$

Figure 4: Utility loss vs. K on *MSNBC*

actually has two assumptions: 1) all frequent patterns are of the same length; 2) the lengths of frequent patterns are identical to the lengths of input sequences. Since generally these two assumptions cannot be satisfied in a frequent sequential pattern mining task, it is not fair to directly compare *FFS* with *N-gram* and *Prefix*. However, there are very few approaches that support frequent sequential pattern mining under differential privacy. Hence we still report the performance of *FFS* and provide insights on the key factor that guarantees high utility on frequent sequential pattern mining. For both *FFS* and *Prefix*, we have tested various parameter settings and report the best results we have obtained.

To give an intuitive impression on the performance of these three approaches, we first report their *true positive ratios* under different K and ε values in Tables 5 and 6. Given K, we generate the top K most frequent sequential patterns, up to a maximum pattern size of 7, on both the original dataset $\mathcal{D}$ and the sanitized dataset $\widetilde{\mathcal{D}}$, which are denoted by $F_K(\mathcal{D})$ and $F_K(\widetilde{\mathcal{D}})$, respectively. The true positive ratio is then defined to be the percentage of frequent patterns that are correctly identified, that is, $\frac{|\mathcal{F}_K(\mathcal{D}) \cap \mathcal{F}_K(\widetilde{\mathcal{D}})|}{K}$. The results indicate that *N-gram* can reliably identify the most frequent patterns in a given database with strong privacy guarantee.

To measure the utility of sanitized data more precisely, we adopt the metric proposed in [6], which further takes into consideration the accuracy of the supports of patterns in $F_K(\widetilde{\mathcal{D}})$ [9]. The utility loss on the sanitized dataset is defined to be the difference between $F_K(\mathcal{D})$ and $F_K(\widetilde{\mathcal{D}})$, that is,

$$\frac{\sum_{F_i \in F_K(\mathcal{D})} \dfrac{|sup(F_i, F_K(\mathcal{D})) - sup(F_i, F_K(\widetilde{\mathcal{D}}))|}{sup(F_i, F_K(\mathcal{D}))}}{K},$$

where $sup(F_i, F_K(\mathcal{D}))$ and $sup(F_i, F_K(\widetilde{\mathcal{D}}))$ denote the supports of F_i in $F_K(\mathcal{D})$ and $F_K(\widetilde{\mathcal{D}})$, respectively. If $F_i \notin F_K(\widetilde{\mathcal{D}})$, $sup(F_i, F_K(\widetilde{\mathcal{D}})) = 0$. Therefore, if the metric equals 0, it means that $F_K(\mathcal{D})$ is identical to $F_K(\widetilde{\mathcal{D}})$ (even the support of every frequent pattern); if the metric equals 1, it implies that $F_K(\mathcal{D})$ and $F_K(\widetilde{\mathcal{D}})$ are totally different.

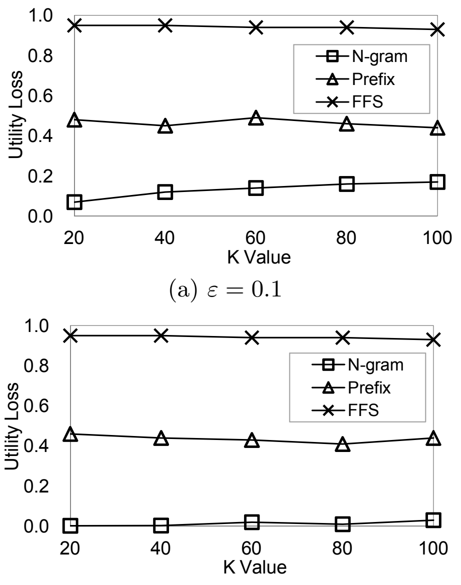
In Figures 4 and 5, where $\ell_{max} = 20$ and $n_{max} = 5$, we can observe that our proposal significantly outperforms the other two approaches. In addition, for the frequent patterns that are correctly identified, the relative errors of their supports are typically very small even when $\varepsilon = 0.1$. The main reason is that *N-gram* extracts the essential information of a database in terms of a set of n-grams, which are actually the most frequent patterns in the database. This fact allows *N-gram* to perform well even under a small ε value. In contrast, in *Prefix*, the noise added to a frequent pattern's count accumulates quickly in proportion to the number of longer sequences that contain this frequent pattern. The major limitation of *FFS* is its prefix data structure, which generates frequent patterns based on very short prefixes.

7. CONCLUSION

In this paper, we proposed a novel approach to differentially private sequential data publication based on a variable-length n-gram model. This model extracts the essential information of a sequential database in terms of a set of variable-length n-grams whose counts are relatively large and therefore subject to lower Laplace error. We developed a set of key techniques that are vital to the success of the n-gram model. Furthermore, we designed a synthetic sequential database construction method, which allows published n-grams to be used for a wider range of data analysis tasks. Extensive experiments on real-life datasets proved that our solution substantially outperforms state-of-the-art techniques [16], [4] in terms of count query and frequent sequential pattern mining.

8. ACKNOWLEDGMENTS

[9] The support of a pattern is the number of its occurrences in a database.



(a) $\varepsilon = 0.1$

(b) $\varepsilon = 1.0$

Figure 5: Utility loss vs. K on *STM*

This work was supported in part by a grant from the EIT ICT Labs to INRIA. The authors would like to thank the anonymous reviewers for their constructive comments.

9. REFERENCES

[1] O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, pages 376–385, 2008.

[2] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

[3] F. Bonchi, L. V. Lakshmanan, and H. W. Wang. Trajectory anonymity in publishing personal mobility data. *SIGKDD Explorations Newsletter*, 13(1):30–42, 2011.

[4] R. Chen, B. C. M. Fung, and B. C. Desai. Differentially private trajectory data publication. *CoRR*, abs/1112.2020, 2011.

[5] R. Chen, B. C. M. Fung, N. Mohammed, and B. C. Desai. Privacy-preserving trajectory data publishing by local suppression. *Information Sciences*, in press.

[6] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *PVLDB*, 4(11):1087–1098, 2011.

[7] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[8] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[10] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390, 2009.

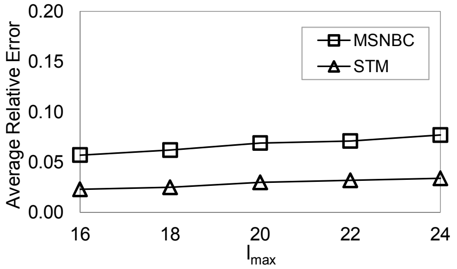


Figure 6: Average relative error vs. ℓ_{max} ($\varepsilon = 1.0$)

[11] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. *CoRR*, abs/1012.4763, 2012.

[12] H. Hu, J. Xu, S. T. On, J. Du, and J. K.-Y. Ng. Privacy-aware location data publishing. *ACM Transactions on Database Systems*, 35(3):17, 2010.

[13] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, pages 123–134, 2010.

[14] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[15] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30, 2009.

[16] F. McSherry and R. Mahajan. Differentially private network trace analysis. In *SIGCOMM*, pages 123–134, 2010.

[17] N. Mohammed, R. Chen, B. C. M. Fung, and P. S. Yu. Differentially private data release for data mining. In *SIGKDD*, pages 493–501, 2011.

[18] A. Monreale, G. Andrienko, N. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, and S. Wrobel. Movement data anonymity through generalization. *Transactions on Data Privacy*, 3(2):91–121, 2010.

[19] P. Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA Law Review*, 2010.

[20] B. Sheridan. A trillion points of data. *Newsweek*, March 2009.

[21] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *MDM*, pages 65–72, 2008.

[22] X. Xiao, G. Bender, M. Hay, and J. Gehrke. iReduct: Differential privacy with reduced relative errors. In *SIGMOD*, pages 229–240, 2011.

[23] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, pages 225–236, 2010.

[24] R. Yarovoy, F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Anonymizing moving objects: How to hide a MOB in a crowd? In *EDBT*, pages 72–83, 2009.

APPENDIX

A. ADDITIONAL EXPERIMENTS

In this section, we present the performance of *N-gram* under different ℓ_{max} and n_{max} values and discuss several hints for selecting reasonable ℓ_{max} and n_{max} values. In addition,

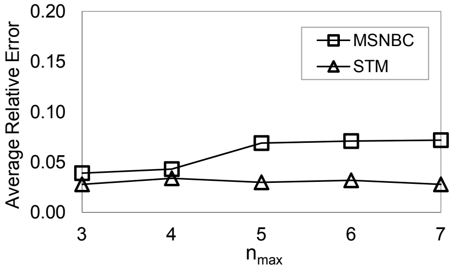
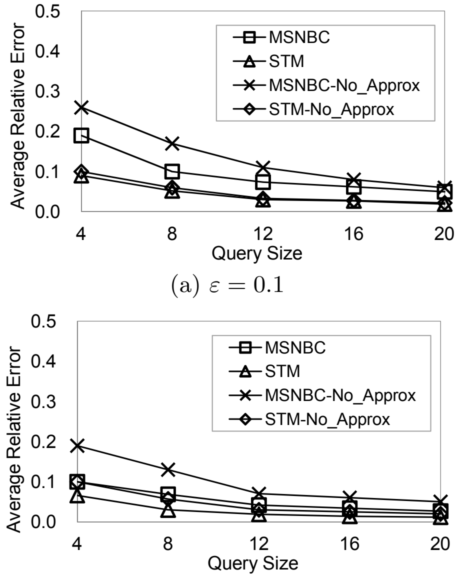


Figure 7: Average relative error vs. n_{max} ($\varepsilon = 1.0$)



(a) $\varepsilon = 0.1$

(b) $\varepsilon = 1.0$

Figure 8: Effect of node count approximation.

we demonstrate the utility improvement due to the approximation technique proposed in Section 4.3.4.

A.1 Count Query

We examine the impact of different parameters (i.e., ℓ_{max} and n_{max}) of the n-gram model on average relative error of count queries. In Figure 6, we study how relative error changes under different ℓ_{max} values with $\varepsilon = 1.0$, $n_{max} = 5$ and query size equal to 8. In theory, a larger ℓ_{max} value allows more information of the underlying database to be retained at the cost of higher sensitivity (and hence larger Laplace noise). Therefore, the selection of ℓ_{max} needs to take into consideration the trade-off between approximation error and Laplace error. However, in reality, a reasonable ℓ_{max} value could be chosen more easily because the average sequence length of many real-life datasets is relatively small. Consequently, Laplace error is the major concern in this case. This is confirmed by Figure 6. Since most sequences in $MSNBC$ and STM are of a small length, when ℓ_{max} is sufficiently large (i.e., 16), increasing ℓ_{max} does not significantly lower approximation error, but simply increases Laplace noise. Moreover, we can observe that our approach performs relatively stable under varying ℓ_{max} values. This can be explained by the large counts of short grams, which are more resistant to Laplace noise.

Figure 7 examines the performance of *N-gram* with respect to varying n_{max} values, where $\varepsilon = 1.0$, $\ell_{max} = 20$ and query size is 8. Similar to the selection of ℓ_{max}, the selection of n_{max} also involves the trade-off between approximation error and Laplace error. A larger n_{max} reduces approximation error while increasing Laplace error. To obtain a reasonable trade-off, we develop the adaptive privacy budget allocation scheme and the formal choice of the threshold value, which automatically select the best gram sizes on the fly. Even a data holder specifies an unreasonably large n_{max}, our approach will end up with shorter grams. Therefore, it can be observed that the performance of our solution is insensitive to varying n_{max} values. From our experimental results, we believe that, in most cases, $n_{max} = 5$ is a good choice. In addition, we point out that a good n_{max} value is related to $|\mathcal{D}|$ and $|\mathcal{I}|$, a larger $|\mathcal{D}|$ or a smaller $|\mathcal{I}|$ suggests a larger n_{max} value.

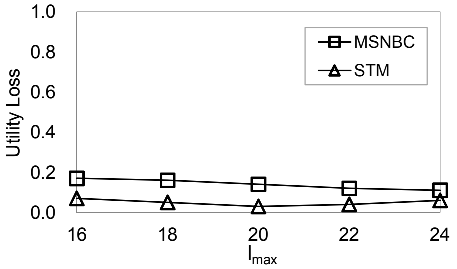


Figure 9: Utility loss vs. ℓ_{max} ($\varepsilon = 1.0$)

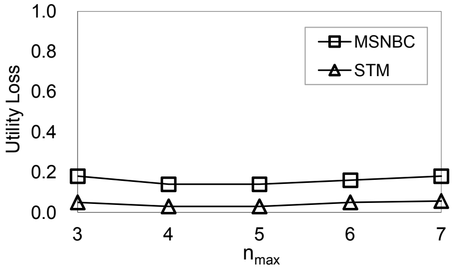


Figure 10: Utility loss vs. n_{max} ($\varepsilon = 1.0$)

One key technique that we develop to improve accuracy of count queries is to enforce consistency constraints by approximating the counts of the nodes that cannot pass the threshold (Section 4.3.4). In the next set of experiments, we demonstrate that this technique indeed improves the accuracy of count queries compared to the case where we naively set the noisy counts of all nodes that cannot pass the threshold to 0. In Figure 8, we set $\ell_{max} = 20$ and $n_{max} = 5$. *MSNBC_No-Approx* and *STM_No-Approx* give the relative errors of *N-gram* without the approximation technique. As we can observe, this technique improves the relative error for all query sizes under different ε values, up to 47%.

A.2 Frequent Sequential Pattern Mining

In the last set of experiments, we study the impact of ℓ_{max} and n_{max} on frequent sequential pattern mining. Figure 9 reports the utility loss of *N-gram* under different ℓ_{max} values with $\varepsilon = 1.0$ and $n_{max} = 5$. The aforementioned trade-off in the selection of ℓ_{max} still applies to frequent sequential pattern mining. This time, we can clearly observe such a trade-off in Figure 9: when ℓ_{max} is small, the approximate error is the main source of error; when ℓ_{max} becomes larger, the total error is dominated by Laplace error. Nevertheless, *N-gram* can provide good utility for a wide range of ℓ_{max} values. This property makes it easier for a data holder to select a good ℓ_{max} value.

Similar trade-off due to n_{max} can be observed in Figure 10, where ℓ_{max} is fixed to 20. There exists a n_{max} value that minimizes the sum of approximation error and Laplace error. Due to the series of techniques we propose, the utility lost under different n_{max} values is comparable.