Free Gap Information from the Differentially Private Sparse Vector and Noisy Max Mechanisms

Zeyu Ding, Yuxin Wang, Danfeng Zhang, Daniel Kifer
Pennsylvania State University, University Park, PA 16802
{zyding, yxwang}@psu.edu, {zhang, dkifer}@cse.psu.edu

ABSTRACT

Noisy Max and Sparse Vector are selection algorithms for differential privacy and serve as building blocks for more complex algorithms. In this paper we show that both algorithms can release additional information for free (i.e., at no additional privacy cost). Noisy Max is used to return the approximate maximizer among a set of queries. We show that it can also release for free the noisy gap between the approximate maximizer and runner-up. This free information can improve the accuracy of certain subsequent counting queries by up to 50%. Sparse Vector is used to return a set of queries that are approximately larger than a fixed threshold. We show that it can adaptively control its privacy budget (use less budget for queries that are likely to be much larger than the threshold) in order to increase the amount of queries it can process. These results follow from a careful privacy analysis.

PVLDB Reference Format:
Zeyu Ding, Yuxin Wang, Danfeng Zhang and Daniel Kifer. Free Gap Information from the Differentially Private Sparse Vector and Noisy Max Mechanisms. *PVLDB*, 13(3): 293-306, 2019.
DOI: https://doi.org/10.14778/3368289.3368295

1. INTRODUCTION

Industry and government agencies are increasingly adopting differential privacy [18] to protect the confidentiality of users who provide data. Current and planned major applications include data gathering by Google [21, 7], Apple [40], and Microsoft [13]; database querying by Uber [27]; and publication of population statistics at the U.S. Census Bureau [33, 9, 25, 2].

The accuracy of differentially private data releases is very important in these applications. One way to improve accuracy is to increase the value of the privacy parameter ϵ, known as the privacy loss budget, as it provides a tradeoff between an algorithm's utility and its privacy protections. However, values of ϵ that are deemed too high can subject a company to criticisms of not providing enough privacy [39]. For this reason, researchers invest significant effort in tuning algorithms [11, 44, 28, 1, 37, 22] and privacy analyses [8, 36, 37, 20] to provide better utility while using smaller privacy budgets.

Differentially private algorithms are built out of smaller components called *mechanisms* [35]. Popular mechanisms include the Laplace Mechanism [18], Geometric Mechanism [24], Noisy Max [19] and Sparse Vector [19, 32]. As we will explain in this paper, the latter two mechanisms, Noisy Max and Sparse Vector, inadvertently throw away information that is useful for designing accurate algorithms. Our contribution is to present novel variants of these mechanisms that provide more functionality at the same privacy cost (under pure differential privacy).

Given a set of queries, Noisy Max returns the identity (not value) of the query that is likely to have the largest value – it adds noise to each query answer and returns the index of the query with the largest noisy value. Meanwhile, Sparse Vector takes a stream of queries and a predefined public threshold T. It tries to return the identities (not values) of the first k queries that are likely larger than the threshold. To do so, it adds noise to the threshold. Then, as it sequentially processes each query, it outputs "$\top$" or "$\bot$", depending on whether the noisy value of the current query is larger or smaller than the noisy threshold. The mechanism terminates after k "$\top$" outputs.

In recent work [42], using program verification tools, Wang et al. showed that Sparse Vector can provide additional information *at no additional cost to privacy*. That is, when Sparse Vector returns "$\top$" for a query, it can also return the gap between its noisy value and the noisy threshold.[1] We refer to their algorithm as Sparse-Vector-with-Gap.

Inspired by this program verification work, we propose novel variations of Sparse Vector and Noisy Max that add new functionality. For Sparse Vector, we show that in addition to releasing this gap information, even stronger improvements are possible – we present an adaptive version that can answer more queries than before by controlling how much privacy budget it uses to answer each query. The intuition is that we would like to spend less of our privacy budget for queries that are probably much larger than the threshold (compared to queries that are probably closer to the threshold). A careful accounting of the privacy impact shows that this is possible. Our experiments confirm that Adaptive-

[1]This was a surprising result given the number of incorrect attempts at improving Sparse Vector based on flawed manual proofs [32] and shows the power of automated program verification techniques.

Proceedings of the VLDB Endowment, Vol. 13, No. 3
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3368289.3368295

Sparse-Vector-with-Gap can answer many more queries than the prior versions [32, 19, 42] at the same privacy cost.

For Noisy Max, we show that it too inadvertently throws away information. Specifically, *at no additional cost to privacy*, it can release an estimate of the gap between the largest and second largest queries (we call the resulting mechanism Noisy-Max-with-Gap). We generalize this result to Noisy Top-K – showing that one can release an estimate of the *identities* of the k largest queries and, at no extra privacy cost, release noisy estimates of the pairwise *gaps* (differences) among the top $k+1$ queries.

The extra noisy gap information opens up new directions in the construction of differentially private algorithms and can be used to improve accuracy of certain subsequent queries. For instance, one common task is to select the approximate top k queries and then use additional privacy loss budget to obtain noisy answers to these queries. We show that a postprocessing step can combine these noisy answers with gap information to improve accuracy by up to 50% for counting queries. We provide similar applications for the free gap information in Sparse Vector.

We prove our results using the alignment of random variables technique [32, 11, 42, 43], which is based on the following question: if we change the input to a program, how must we change its random variables so that output remains the same? This technique is used to prove the correctness of almost all pure differential privacy mechanisms [19] but needs to be used in sophisticated ways to prove the correctness of the more advanced algorithms [32, 11, 19, 42, 43]. Nevertheless, alignment of random variables is often used incorrectly (as discussed by Lyu et al. [32]). Thus a secondary contribution of our work is to lay out the precise steps and conditions that must be checked and to provide helpful lemmas that ensure these conditions are met.

To summarize, our contributions are as follows:

- We provide a simplified template for writing correctness proofs for intricate differentially private algorithms.
- Using this technique, we propose and prove the correctness of two new mechanisms: Noisy-Top-K-with-Gap and Adaptive-Sparse-Vector-with-Gap. These mechanisms improve on the original versions of Noisy Max and Sparse Vector by taking advantage of *free* information (i.e., information that can be released at no additional privacy cost) that those algorithms inadvertently throw away.
- We demonstrate some of the uses of the gap information that is provided by these new mechanisms. When an algorithm needs to use Noisy Max or Sparse Vector to select some queries and then measure them (i.e., obtain their noisy answers), we show how the gap information from our new mechanisms can be used to improve the accuracy of the noisy measurements. We also show how the gap information in Sparse Vector can be used to estimate the confidence that a query's true answer really is larger than the threshold.
- We empirically evaluate these new mechanisms on a variety of datasets to demonstrate their improved utility.

In Section 2, we discuss related work. We present background and notation in Section 3. We present simplified proof templates for randomness alignment in Section 4. We present Noisy-Top-K-with-Gap in Section 5 and Adaptive-Sparse-Vector-with-Gap in Section 6. We present experiments in Section 7, proofs underlying the alignment of randomness framework in Section 8 and conclusions in Section 9. The rest of our proofs can be found in the full version of this paper [14].

2. RELATED WORK

Selection algorithms, such as Exponential Mechanism [34, 38], Sparse Vector [19, 32], and Noisy Max [19] are used to select a set of items (typically queries) from a much larger set. They have applications in hyperparameter tuning [11, 31], iterative construction of microdata [26], feature selection [41], frequent itemset mining [6], exploring a privacy/accuracy tradeoff [30], data pre-processing [12], etc.

Various generalizations have been proposed [30, 5, 41, 38, 10, 31]. Liu and Talwar [31] and Raskhodnikova and Smith [38] extend the exponential mechanism for arbitrary sensitivity queries. Beimel et al. [5] and Thakurta and Smith [41] use the propose-test-release framework [17] to find a gap between the best and second best queries and, if the gap is large enough, release the identity of the best query. These two algorithms rely on a relaxation of differential privacy called approximate (ϵ, δ)-differential privacy [16] and can fail to return an answer (in which case they return $\perp$). Our algorithms work with pure ϵ-differential privacy. Chaudhuri et al. [10] also proposed a large margin mechanism (with approximate differential privacy) which finds a large gap separating top queries from the rest and returns one of them.

There have also been unsuccessful attempts to generalize selection algorithms such as Sparse Vector (incorrect versions are catalogued by Lyu et al. [32]), which has sparked innovations in program verification for differential privacy (e.g., [4, 3, 43, 42]) with techniques such as probabilistic coupling [4] and a simplification based on randomness alignment [43]. These are similar to ideas behind handwritten proofs [11, 19, 32] – they consider what changes need to be made to random variables in order to make two executions of a program, with different inputs, produce the same output. It is a powerful technique that is behind almost all proofs of differential privacy, but is very easy to apply incorrectly [32]. In this paper, we state and prove a more general version of this technique in order to prove correctness of our algorithms and also provide additional results that simplify the application of this technique.

3. NOTATION AND BACKGROUND

In this paper, we use the following notation. D and D' refer to databases. We use the notation $D \sim D'$ to represent adjacent databases.[2] M denotes a randomized algorithm whose input is a database. Ω denotes the range of M and $\omega \in \Omega$ denotes a specific output of M. We use $E \subseteq \Omega$ to denote a set of possible outputs. Because M is randomized, it also relies on a random noise vector $H \in \mathbb{R}^\infty$ (which usually consists of independent zero-mean Laplace random variables). This noise sequence is infinite, but of course M will only use a finite-length prefix of H. When we need to draw attention to the noise, we use the notation $M(D, H)$ to

[2]The notion of adjacency depends on the application. Some papers define it as D can be obtained from D' by modifying one record [18] or by adding/deleting one record [15].

indicate the execution of M with database D and randomness coming from H. Otherwise we use the notation $M(D)$. Define $\mathrm{S}_{\mathrm{M,D:E}} = \{H \mid M(D,H) \in E\}$ to be the set of noise vectors that allow M, on input D, to produce an output in the set $E \subseteq \Omega$. To avoid overburdening the notation, we write $\mathrm{S}_{\mathrm{D:E}}$ for $\mathrm{S}_{\mathrm{M,D:E}}$ and $\mathrm{S}_{\mathrm{D':E}}$ for $\mathrm{S}_{\mathrm{M,D':E}}$ when M is clear from the context. When E consists of a single point ω, we write these sets as $\mathrm{S}_{\mathrm{D:}\omega}$ and $\mathrm{S}_{\mathrm{D':}\omega}$. This notation is summarized in the table below.

Table 1: Notation

Symbol	Meaning
M	randomized algorithm
D, D'	database
$D \sim D'$	D is adjacent to D'
$H = (\eta_1, \eta_2, \dots)$	input noise vector
Ω	the space of all output of M
ω	a possible output; $\omega \in \Omega$
E	a set of possible outputs; $E \subseteq \Omega$
$\mathrm{S}_{\mathrm{D:E}} = \mathrm{S}_{\mathrm{M,D:E}}$	$\{H \mid M(D,H) \in E\}$
$\mathrm{S}_{\mathrm{D:}\omega} = \mathrm{S}_{\mathrm{M,D:}\omega}$	$\{H \mid M(D,H) = \omega\}$

3.1 Formal Privacy

Differential privacy [18, 15, 19] is currently the gold standard for releasing privacy-preserving information about a database. It has a parameter $\epsilon > 0$ known as the privacy loss budget. The smaller it is, the more privacy is provided. Differential privacy bounds the effect of one record on the output of the algorithm (for small ϵ, the probability of any output is barely affected by any person's record).

Definition 1 (Pure Differential Privacy [15]). *Given an $\epsilon > 0$, a randomized algorithm M with output space Ω satisfies (pure) ϵ-differential privacy if for all $E \subseteq \Omega$ and all pairs of adjacent databases $D \sim D'$, the following holds:*

$$\mathbb{P}(M(D,H) \in E) \leq e^{\epsilon}\mathbb{P}(M(D',H) \in E) \qquad (1)$$

where the probability is only over the randomness of H.

With the notation in Table 1, the differential privacy condition from Equation (1) is $\mathbb{P}(\mathrm{S}_{\mathrm{D:E}}) \leq e^{\epsilon}\mathbb{P}(\mathrm{S}_{\mathrm{D':E}})$.

Differential privacy enjoys the following nice properties:

- Resilience to Post-Processing. If we apply an algorithm A to the output of an ϵ-differentially private algorithm M, then the composite algorithm $A \circ M$ still satisfies ϵ-differential privacy. In other words, privacy is not reduced by post-processing.
- Composition. If $M_1, M_2, \dots, M_k$ satisfy differential privacy with privacy loss budgets $\epsilon_1, \dots, \epsilon_k$, the algorithm that runs all of them and releases their outputs satisfies $(\sum_i \epsilon_i)$-differential privacy.

Many differentially private algorithms take advantage of the Laplace mechanism [34], which provides a noisy answer to a vector-valued query $\boldsymbol{q}$ based on its L_1 *global sensitivity* $\Delta_{\boldsymbol{q}}$, defined as follows:

Definition 2 (L_1 Global Sensitivity [19]). *The global sensitivity of a query $\boldsymbol{q}$ is $\Delta_{\boldsymbol{q}} = \sup_{D \sim D'} \|\boldsymbol{q}(D) - \boldsymbol{q}(D')\|_1$.*

Theorem 1 (Laplace Mechanism [18]). *Given a privacy loss budget ϵ, consider the mechanism that returns $\boldsymbol{q}(D) + H$, where H is a vector of independent random samples from the Laplace($\Delta_{\boldsymbol{q}}/\epsilon$) distribution with mean 0 and scale parameter $\Delta_{\boldsymbol{q}}/\epsilon$. This Laplace mechanism satisfies ϵ-differential privacy.*

Other kinds of additive noise distributions that can be used in place of Laplace include Discrete Laplace [24] (when all query answers are integers or multiples of a common base) and Staircase [23].

4. RANDOMNESS ALIGNMENT

To establish that the algorithms we propose are differentially private, we use an idea called *randomness alignment* that previously had been used to prove the privacy of a variety of sophisticated algorithms [19, 32, 11] and incorporated into verification/synthesis tools [43, 42, 3]. While powerful, this technique is also easy to use incorrectly [32], as there are many technical conditions that need to be checked. In this section, we present results (namely Lemma 1) that significantly simplify this process and make it easy to prove the correctness of our proposed algorithms.

In general, to prove ϵ-differential privacy for an algorithm M, one needs to show $P(M(D) \in E) \leq e^{\epsilon}P(M(D') \in E)$ for all pairs of adjacent databases $D \sim D'$ and sets of possible outputs $E \subseteq \Omega$. In our notation, this inequality is represented as $\mathbb{P}(\mathrm{S}_{\mathrm{D:E}}) \leq e^{\epsilon}\mathbb{P}(\mathrm{S}_{\mathrm{D':E}})$. Establishing such inequalities is often done with the help of a function $\phi_{\mathrm{D,D'}}$, called a *randomness alignment* (there is a function $\phi_{\mathrm{D,D'}}$ for every pair $D \sim D'$), that maps noise vectors H into noise vectors H' so that $M(D', H')$ produces the same output as $M(D,H)$. Formally,

Definition 3 (Randomness Alignment). *Let M be a randomized algorithm. Let $D \sim D'$ be two adjacent databases. A* randomness alignment *is a function $\phi_{\mathrm{D,D'}} : \mathbb{R}^{\infty} \to \mathbb{R}^{\infty}$ such that for all H on which $M(D,H)$ terminates, we have $M(D,H) = M(D', \phi_{\mathrm{D,D'}}(H))$.*

Example 1. Let D be a database that records the salary of every person, which is guaranteed to be between 0 and 100. Let $q(D)$ be the sum of the salaries in D. The sensitivity of q is thus 100. Let $H = (\eta_1, \eta_2, \dots)$ be a vector of independent Laplace($100/\epsilon$) random variables. The Laplace mechanism outputs $q(D) + \eta_1$ (and ignores the remaining variables in H). For every pair of adjacent databases $D \sim D'$, one can define the corresponding randomness alignment $\phi_{\mathrm{D,D'}}(H) = H' = (\eta_1', \eta_2', \dots)$, where $\eta_1' = \eta_1 + q(D) - q(D')$ and $\eta_i' = \eta_i$ for $i > 1$. Note that $q(D) + \eta_1 = q(D') + \eta_1'$, so the output of M remains the same.

In practice, $\phi_{\mathrm{D,D'}}$ is constructed locally (piece by piece) as follows. For each possible output $\omega \in \Omega$, one defines a function $\phi_{\mathrm{D,D',}\omega}$ that maps noise vectors H into noise vectors H' with the following properties: if $M(D,H) = \omega$ then $M(D',H') = \omega$ (that is, $\phi_{\mathrm{D,D',}\omega}$ only cares about what it takes to produce the specific output ω). We obtain our randomness alignment $\phi_{\mathrm{D,D'}}$ in the obvious way by piecing together the $\phi_{\mathrm{D,D',}\omega}$ as follows: $\phi_{\mathrm{D,D'}}(H) = \phi_{\mathrm{D,D',}\omega^*}(H)$, where ω^* is the output of $M(D,H)$. Formally,

Definition 4 (Local Alignment). *Let M be a randomized algorithm. Let $D \sim D'$ be a pair of adjacent databases and ω a possible output of M. A* local alignment *for M is a function $\phi_{\mathrm{D,D',}\omega} : \mathrm{S}_{\mathrm{D:}\omega} \to \mathrm{S}_{\mathrm{D':}\omega}$ (see notation in Table 1) such that for all $H \in \mathrm{S}_{\mathrm{D:}\omega}$, we have $M(D,H) = M(D', \phi_{\mathrm{D,D',}\omega}(H))$.*

Example 2. Continuing the setup from Example 1, consider the mechanism M_1 that, on input D, outputs $\top$ if $q(D) + \eta_1 \geq 10,000$ (i.e. if the noisy total salary is at least $10,000$) and $\perp$ if $q(D) + \eta_1 < 10,000$. Let D' be a database that differs from D in the presence/absence of one record. Consider the local alignments $\phi_{D,D',\top}$ and $\phi_{D,D',\perp}$ defined as follows. $\phi_{D,D',\top}(H) = H' = (\eta_1', \eta_2', \dots)$ where $\eta_1' = \eta_1 + 100$ and $\eta_i' = \eta_i$ for $i > 1$; and $\phi_{D,D',\perp}(H) = H'' = (\eta_1'', \eta_2'', \dots)$ where $\eta_1'' = \eta_1 - 100$ and $\eta_i'' = \eta_i$ for $i > 1$. Clearly, if $M_1(D, H) = \top$ then $M_1(D', H') = \top$ and if $M_1(D, H) = \perp$ then $M_1(D', H'') = \perp$. We piece these two local alignments together to create a randomness alignment $\phi_{\mathrm{D},\mathrm{D}'}(H) = H^* = (\eta_1^*, \eta_2^*, \dots)$ where:

$$\eta_1^* = \begin{cases} \eta_1 + 100 & \text{if } M(D,H) = \top \\ & \text{(i.e. } q(D) + \eta_1 \geq 10,000) \\ \eta_1 - 100 & \text{if } M(D,H) = \perp \\ & \text{(i.e. } q(D) + \eta_1 < 10,000) \end{cases}$$

$$\eta_i^* = \eta_i \text{ for } i > 1$$

Special properties of alignments. Not all alignments can be used to prove differential privacy. In this section we discuss some additional properties that help prove differential privacy. We first make two mild assumptions about the mechanism M: (1) it terminates with probability[3] one and (2) based on the output of M, we can determine how many random variables it used. The vast majority of differentially private algorithms in the literature satisfy these properties.

We next define two properties of a local alignment: whether it is *acyclic* and what its *cost* is.

Definition 5 (Acyclic). *Let M be a randomized algorithm. Let $\phi_{\mathrm{D},\mathrm{D}',\omega}$ be a local alignment for M. For any $H = (\eta_1, \eta_2, \dots)$, let $H' = (\eta_1', \eta_2', \dots)$ denote $\phi_{\mathrm{D},\mathrm{D}',\omega}(H)$. We say that $\phi_{\mathrm{D},\mathrm{D}',\omega}$ is acyclic if there exists a permutation π and piecewise differentiable functions $\psi^{(j)}_{D,D',\omega}$ such that:*

$$\eta'_{\pi(1)} = \eta_{\pi(1)} + \textit{number that only depends on } D,\ D',\ \omega$$
$$\eta'_{\pi(j)} = \eta_{\pi(j)} + \psi^{(j)}_{D,D',\omega}(\eta_{\pi(1)}, \dots, \eta_{\pi(j-1)}) \textit{ for } j \geq 2$$

Essentially, a local alignment $\phi_{\mathrm{D},\mathrm{D}',\omega}$ is acyclic if there is some ordering of the variables so that η_j' is the sum of η_j and a function of the variables that came earlier in the ordering. The local alignments $\phi_{D,D',\top}$ and $\phi_{D,D',\perp}$ from Example 2 are both acyclic (in general, each local alignment function is allowed to have its own specific ordering and differentiable functions $\psi^{(j)}_{D,D',\omega}$). The pieced-together randomness alignment $\phi_{\mathrm{D},\mathrm{D}'}$ itself need not be acyclic.

Definition 6 (Alignment Cost). *Let M be a randomized algorithm that uses H as its source of randomness. Let $\phi_{\mathrm{D},\mathrm{D}',\omega}$ be a local alignment for M. For any $H = (\eta_1, \eta_2, \dots)$, let $H' = (\eta_1', \eta_2', \dots)$ denote $\phi_{\mathrm{D},\mathrm{D}',\omega}(H)$. Suppose each η_i is generated independently from a distribution f_i with the property that $\log(f_i(x)/f_i(y)) \leq |x-y|/\alpha_i$ for all x, y in the domain of f_i – this includes the Laplace(α_i) distribution along with Discrete Laplace [24] and Staircase [23]. Then the cost of $\phi_{\mathrm{D},\mathrm{D}',\omega}$ is defined as:*

$$\textit{cost}(\phi_{\mathrm{D},\mathrm{D}',\omega}) = \sum_i |\eta_i - \eta_i'|/\alpha_i$$

The following lemma uses those properties to establish that M satisfies ϵ-differential privacy.

Lemma 1. *Let M be a randomized algorithm with input randomness $H = (\eta_1, \eta_2, \dots)$. If the following conditions are satisfied, then M satisfies ϵ-differential privacy.*

1. *M terminates with probability 1.*
2. *The number of random variables used by M can be determined from its output.*
3. *Each η_i is generated independently from a distribution f_i with the property that $\log(f_i(x)/f_i(y)) \leq |x-y|/\alpha_i$ for all x, y in the domain of f_i (such as Laplace(α_i)).*
4. *For every $D \sim D'$ and ω there exists a local alignment $\phi_{\mathrm{D},\mathrm{D}',\omega}$ that is acyclic with $\mathrm{cost}(\phi_{\mathrm{D},\mathrm{D}',\omega}) \leq \epsilon$.*
5. *For each $D \sim D'$ the number of distinct local alignments is countable. That is, the set $\{\phi_{\mathrm{D},\mathrm{D}',\omega} \mid \omega \in \Omega\}$ is countable (i.e., for many choices of ω we get the same exact alignment function).*

We defer the proof to Section 8.

Example 3. Consider the alignment $\phi_{\mathrm{D},\mathrm{D}'}$ from Example 1. We can define all of the local alignments $\phi_{\mathrm{D},\mathrm{D}',\omega}$ to be the same function: $\phi_{\mathrm{D},\mathrm{D}',\omega}(H) = \phi_{\mathrm{D},\mathrm{D}'}(H)$. Clearly $\mathrm{cost}(\phi_{\mathrm{D},\mathrm{D}',\omega}) = \sum_{i=0}^{\infty} \frac{\epsilon}{100}|\eta_i' - \eta_i| = \frac{\epsilon}{100}|q(D') - q(D)| \leq \epsilon$. For Example 2, there are two acyclic local alignments $\phi_{D,D'\top}$ and $\phi_{D,D'\perp}$, both have cost $= 100 \cdot \frac{\epsilon}{100} = \epsilon$. The other conditions in Lemma 1 are trivial to check. Thus both mechanisms satisfy ϵ-differential privacy by Lemma 1.

5. IMPROVING NOISY MAX

In this section, we present novel variations of the Noisy Max mechanism [19]. Given a list of queries with sensitivity 1, the purpose of Noisy Max is to estimate the identity (i.e., index) of the largest query. We show that, in addition to releasing this index, it is possible to release a numerical estimate of the gap between the values of the largest and second largest queries. This extra information comes at no additional cost to privacy, meaning that the original Noisy Max mechanism threw away useful information. This result can be generalized to the setting in which one wants to estimate the identities of the top k queries - we can release (for free) all of the gaps between each top k query and the next best query (i.e., the gap between the best and second best queries, the gap between the second and third best queries, etc). When a user subsequently asks for a noisy answer to each of the returned queries, we show how the gap information can be used to reduce squared error by up to 50% (for counting queries).

5.1 Noisy-Top-K-with-Gap

Our proposed Noisy-Top-K-with-Gap mechanism is shown in Algorithm 1 (the function $\arg\max_c$ returns the top c items). We can obtain the classical Noisy Max algorithm [19] from it by setting $k = 1$ and throwing away the gap information (the boxed items on Lines 7 and 8). The Noisy-Top-K-with-Gap algorithm takes as input a sequence of n

[3]That is, for each input D, there might be some random vectors H for which M does not terminate, but the total probability of these vectors is 0, so we can ignore them.

queries $q_1, \ldots, q_n$, each having sensitivity 1. It adds Laplace noise to each query. It returns the indexes $j_1, \ldots, j_k$ of the k queries with the largest noisy values in descending order. Furthermore, for each of these top k queries q_{j_i}, it releases the noisy gap between the value of q_{j_i} and the value of the next best query. Our key contribution in this section is the observation that these gaps can be released for free. That is, the classical Top-K algorithm, which does not release the gaps, satisfies ϵ-differential privacy. But, our improved version has exactly the same privacy cost yet is strictly better because of the extra information it can release.

Algorithm 1: Noisy-Top-K-with-Gap

input: $\boldsymbol{q}$: a list of n queries of global sensitivity 1
D: database, k: # of indexes, ϵ: privacy budget

function `NoisyTopK (`$\boldsymbol{q}$`, D, k, ε)`**:**
  foreach `i` $\in \{1, \cdots, n\}$ **do**
    $\eta_i \leftarrow \text{Lap}(2k/\epsilon)$
    $\widetilde{q}_i \leftarrow q_i(D) + \eta_i$
  $(j_1, \ldots, j_{k+1}) \leftarrow \arg\max_{k+1}(\widetilde{q}_1, \ldots, \widetilde{q}_n)$
  foreach `i` $\in \{1, \cdots, k\}$ **do**
    $g_i \leftarrow \widetilde{q}_{j_i} - \widetilde{q}_{j_{i+1}}$ `// ` i^{th} `gap`
  return $((j_1, g_1), \ldots, (j_k, g_k))$ `// k queries`

We emphasize that keeping the noisy gaps hidden does not decrease the privacy cost. Furthermore, this algorithm gives estimates of the pairwise gaps between any pair of the k queries it selects. For example, suppose we are interested in estimating the gap between the a^{th} largest and b^{th} largest queries (where $a < b \leq k$). This is equal to $\sum_{i=a}^{b-1} g_i$ because: $\sum_{i=a}^{b-1} g_i = \sum_{i=a}^{b-1} (\widetilde{q}_{j_i} - \widetilde{q}_{j_{i+1}}) = \widetilde{q}_{j_a} - \widetilde{q}_{j_b}$ and hence its variance is $\text{Var}(\widetilde{q}_{j_a} - \widetilde{q}_{j_b}) = 16k^2/\epsilon^2$.

The original Noisy Top-K mechanism satisfies ϵ-differential privacy. In the special case that all the q_i are counting queries[4] then it satisfies $\epsilon/2$-differential privacy [19]. We will show the same properties for Noisy-Top-K-with-Gap. We prove the privacy property in this section and then in Section 5.2 we show how to use this gap information. However, first it is important to discuss the difference between the theoretical analysis of Noisy Top-K [19] and its practical implementation on finite-precision computers.

Implementation issues. The analysis of the original Noisy Max mechanism assumed the use of *true* Laplace noise (a continuous distribution) so that ties are impossible between the largest and second largest noisy queries [19]. On finite precision computers, ties are possible (breaking the privacy proof [19]) with some probability δ. Thus, one would settle for a slightly weaker guarantee called (ϵ, δ)-differential privacy [16]. It roughly states that the privacy conditions of pure ϵ-differential privacy fail with probability at most δ. In practice, one would discretize the Staircase [23] or Laplace distribution so that it outputs a multiple of some base γ. In the full version of this paper [14], we show that if there are n queries with sensitivity 1 and discretized Lapalce($1/\epsilon$) noise with base γ is added to each of them, the probability of a tie is upper bounded by $\delta = \epsilon\gamma n^2$ (similar calculations can be performed with the Staircase distribution). Thus this is an upper bound on the probability that the differential privacy guarantees will fail. Typically, one would expect γ to be close to machine epsilon (e.g., $\approx 2^{-52}$) so the probability of a tie is negligible. In this section we will also analyze our algorithms under the assumption of continuous noise. Thus the privacy guarantees can fail with this negligible probability δ (hence satisfying (ϵ, δ)-differential privacy [16]).

[4]i.e., when a person is added to a database, the value of each query either stays the same or increases by 1.

Local alignment. To prove the privacy of Algorithm 1, we need to create a local alignment function for each possible pair $D \sim D'$ and output ω. Note that our mechanism uses precisely n random variables. Let $H = (\eta_1, \eta_2, \ldots)$ where η_i is the noise that should be added to the i^{th} query. We view the output $\omega = ((j_1, g_1), \ldots, (j_k, g_k))$ as k pairs where in the i^{th} pair (j_i, g_i), the first component j_i is the index of i^{th} largest noisy query and the second component g_i is the gap in noisy value between the i^{th} and $(i+1)^{\text{th}}$ largest noisy queries. As discussed in the implementation issues, we will base our analysis on continuous noise so that the probability of ties among the top $k+1$ noisy queries is 0. Thus each gap is positive: $g_i > 0$.

Let $\mathcal{I}_\omega = \{j_1, \ldots, j_k\}$ and $\mathcal{I}_\omega^c = \{1, \ldots, n\} \setminus \mathcal{I}_\omega$. I.e., $\mathcal{I}_\omega$ is the index set of the k largest noisy queries selected by the algorithm and $\mathcal{I}_\omega^c$ is the index set of all unselected queries. For $H \in \text{S}_{\text{D}:\omega}$ define $\phi_{\text{D},\text{D}',\omega}(H) = H' = (\eta'_1, \eta'_2, \ldots)$ as

$$\eta'_i = \begin{cases} \eta_i & i \in \mathcal{I}_\omega^c \\ \eta_i + q_i - q'_i + \max\limits_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max\limits_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l) & i \in \mathcal{I}_\omega \end{cases} \tag{2}$$

The idea behind this local alignment is simple: we want to keep the noise of the losing queries the same (when the input is D or its neighbor D'). But, for each of the k selected queries, we want to align its noise to make sure it wins by the same amount when the input is D or its neighbor D'.

LEMMA 2. *Let M be the Noisy-Top-K-with-Gap algorithm. For all $D \sim D'$ and ω, the functions $\phi_{\text{D},\text{D}',\omega}$ defined above are acyclic local alignments for M. Furthermore, for every pair $D \sim D'$, there are countably many distinct $\phi_{\text{D},\text{D}',\omega}$.*

Proof. Given $D \sim D'$ and $\omega = ((j_1, g_1), \ldots, (j_k, g_k))$, for any $H = (\eta_1, \eta_2, \ldots)$ such that $M(D, H) = \omega$, let $H' = (\eta'_1, \eta'_2, \ldots) = \phi_{\text{D},\text{D}',\omega}(H)$. We show that $M(D', H') = \omega$. Since $\phi_{\text{D},\text{D}',\omega}$ is identity on components $i \in \mathcal{I}_\omega^c$, we have $\max\limits_{l \in \mathcal{I}_\omega^c}(q'_l + \eta'_l) = \max\limits_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l)$. So, for the k^{th} selected query:

$$\begin{aligned}(q'_{j_k} + \eta'_{j_k}) - \max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta'_l) &= (q'_{j_k} + \eta'_{j_k}) - \max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) \\ &= (q_{j_k} + \eta_{j_k}) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l) = g_k > 0\end{aligned}$$

where the last line follows from Equation 2. This means on D' the noisy query with index j_k is larger than the best of the unselected noisy queries by the same margin as it is on D. Furthermore, for all $1 \leq i < k$, we have

$$\begin{aligned}&(q'_{j_i} + \eta'_{j_i}) - (q'_{j_{i+1}} + \eta'_{j_{i+1}}) \\ =&(q_{j_i} + \eta_{j_i} + \max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l)) \\ &\quad - (q_{j_{i+1}} + \eta_{j_{i+1}} + \max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l)) \\ =&(q_{j_i} + \eta_{j_i}) - (q_{j_{i+1}} + \eta_{j_{i+1}}) = g_i > 0.\end{aligned}$$

In other words, the query with index j_i is still the i^{th} largest query on D' by the same margin. Thus $M(D', H') = \omega$.

The local alignments are clearly acyclic (any permutation that puts $\mathcal{I}_\omega^c$ before $\mathcal{I}_\omega$ does the trick). Also, note that $\phi_{\mathrm{D},\mathrm{D}',\omega}$ only depends on ω through $\mathcal{I}_\omega$ (the indexes of the k largest queries). There are n queries and therefore $\binom{n}{k} = \frac{n!}{(n-k)!k!}$ distinct $\phi_{\mathrm{D},\mathrm{D}',\omega}$. □

Alignment cost and privacy. To establish the alignment cost, we need the following lemma and definition.

LEMMA 3. *Let $(x_1, \ldots, x_m), (x'_1, \ldots, x'_m) \in \mathbb{R}^m$ be such that $\forall i, |x_i - x'_i| \leq 1$. Then $|\max_i\{x_i\} - \max_i\{x'_i\}| \leq 1$.*

Proof. Let s be an index that maximizes x_i and let t be an index that maximizes x'_i. Without loss of generality, assume $x_s \geq x'_t$. Then $x_s \geq x'_t \geq x'_s \geq x_s - 1$. Hence $|x_s - x'_t| = x_s - x'_t \leq x_s - (x_s - 1) = 1$. □

DEFINITION 7 (Monotonicity). *A list of numerical queries $\boldsymbol{q} = (q_1, q_2, \ldots)$ is monotonic if for all pair of adjacent databases $D \sim D'$ we have either $\forall i : q_i(D) \leq q_i(D')$, or $\forall i : q_i(D) \geq q_i(D')$.*

Counting queries are clearly monotonic. Now we can establish the privacy property of our algorithm.

THEOREM 2. *The Noisy-Top-K-with-Gap mechanism satisfies ϵ-differential privacy. If all of the queries are counting queries, then it satisfies $\epsilon/2$-differential privacy.*

Proof. First we bound the cost of the alignment function defined in (2). Recall that the η_i's are independent $\mathrm{Lap}(2k/\epsilon)$ random variables. By Definition 6

$$\begin{aligned}\mathrm{cost}(\phi_{\mathrm{D},\mathrm{D}',\omega}) =& \sum_{i=1}^{\infty} |\eta'_i - \eta_i| \frac{\epsilon}{2k} \\ =& \frac{\epsilon}{2k} \sum_{i \in \mathcal{I}_\omega} |q_i - q'_i + \max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l)|.\end{aligned}$$

By the global sensitivity assumption we have $|q_i - q'_i| \leq 1$. Apply Lemma 3 to the vectors $(q_l + \eta_l)_{l \in \mathcal{I}_\omega^c}$ and $(q'_l + \eta_l)_{l \in \mathcal{I}_\omega^c}$, we have $|\max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l)| \leq 1$. Therefore,

$$\begin{aligned}&|q_i - q'_i + \max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l)| \\ \leq& |q_i - q'_i| + |\max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l)| \leq 1 + 1 = 2.\end{aligned}$$

Furthermore, if $\boldsymbol{q}$ is monotonic, then

- either $\forall i : q_i \leq q'_i$ in which case $q_i - q'_i \in [-1, 0]$ and $\max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l) \in [0, 1]$,
- or $\forall i : q_i \geq q'_i$ in which case $q_i - q'_i \in [0, 1]$ and $\max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l) \in [-1, 0]$.

In both cases we have $q_i - q'_i + \max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l) \in [-1, 1]$ so $|q_i - q'_i + \max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l)| \leq 1$. Therefore,

$$\begin{aligned}\mathrm{cost}(\phi_{\mathrm{D},\mathrm{D}',\omega}) =& \frac{\epsilon}{2k} \sum_{i \in \mathcal{I}_\omega} |q_i - q'_i + \max_{l \in \mathcal{I}_\omega^c}(q'_l + \eta_l) - \max_{l \in \mathcal{I}_\omega^c}(q_l + \eta_l)| \\ \leq& \frac{\epsilon}{2k} \sum_{i \in \mathcal{I}_\omega} 2 \quad (\text{or } \frac{\epsilon}{2k} \sum_{i \in \mathcal{I}_\omega} 1 \text{ if } \boldsymbol{q} \text{ is monotonic}) \\ =& \frac{\epsilon}{2k} \cdot 2|\mathcal{I}_\omega| \quad (\text{or } \frac{\epsilon}{2k} \cdot |\mathcal{I}_\omega| \text{ if } \boldsymbol{q} \text{ is monotonic}) \\ =& \epsilon \quad (\text{or } \epsilon/2 \text{ if } \boldsymbol{q} \text{ is monotonic}).\end{aligned}$$

Conditions 1 through 3 of Lemma 1 are trivial to check, 4 and 5 follow from Lemma 2 and the above bound on cost. Therefore, Theorem 2 follows from Lemma 1. □

5.2 Utilizing Gap Information

Let us consider one scenario that takes advantage of the gap information. Suppose a data analyst is interested in the identities and values of the top k queries. A typical approach would be to split the privacy budget ϵ in half – use $\epsilon/2$ of the budget to identify the top k queries using Noisy-Top-K-with-Gap. The remaining $\epsilon/2$ budget is evenly divided between the selected queries and is used to obtain noisy measurements (i.e. add $\mathrm{Laplace}(2k/\epsilon)$ noise to each query answer). These measurements will have variance $\sigma^2 = 8k^2/\epsilon^2$. In this section we show how to use the gap information from Noisy-Top-K-with-Gap and postprocessing to improve the accuracy of these measurements.

Problem statement. Let $q_1, \ldots, q_k$ be the true answers of the top k queries that are selected by Algorithm 1. Let $\alpha_1, \ldots, \alpha_k$ be their noisy measurements. Let $g_1, \ldots, g_{k-1}$ be the noisy gaps between $q_1, \ldots, q_k$ that are obtained from Algorithm 1 for free. Then $\alpha_i = q_i + \xi_i$ where each ξ_i is a $\mathrm{Laplace}(2k/\epsilon)$ random variable and $g_i = q_i + \eta_i - q_{i+1} - \eta_{i+1}$ where each η_i is a $\mathrm{Laplace}(4k/\epsilon)$ random variable, or a $\mathrm{Laplace}(2k/\epsilon)$ random variable if the query list is monotonic (recall the mechanism was run with a privacy budget of $\epsilon/2$). Our goal is then to find the *best linear unbiased estimate* (BLUE) [29] β_i of q_i in terms of the measurements α_i and gap information g_i.

THEOREM 3. *With notations as above let $\boldsymbol{q} = [q_1, \ldots, q_k]^T$, $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_k]^T$ and $\boldsymbol{g} = [g_1, \ldots, g_{k-1}]^T$. Suppose the ratio $\mathrm{Var}(\xi_i) : \mathrm{Var}(\eta_i)$ is equal to $1 : \lambda$. Then the BLUE of $\boldsymbol{q}$ is $\boldsymbol{\beta} = \frac{1}{(1+\lambda)k}(X\boldsymbol{\alpha} + Y\boldsymbol{g})$ where*

$$X = \begin{bmatrix} 1+\lambda k & 1 & \cdots & 1 \\ 1 & 1+\lambda k & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1+\lambda k \end{bmatrix}_{k \times k}$$

$$Y = \left(\begin{bmatrix} k-1 & k-2 & \cdots & 1 \\ k-1 & k-2 & \cdots & 1 \\ k-1 & k-2 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ k-1 & k-2 & \cdots & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ k & 0 & \cdots & 0 \\ k & k & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ k & k & \cdots & k \end{bmatrix} \right)_{k \times (k-1)}$$

For proof, see the full version of this paper [14]. Even though this is a matrix multiplication, it is easy to see that it translates into the following algorithm that is linear in k:

1. Compute $\alpha = \sum_{i=1}^{k} \alpha_i$ and $p = \sum_{i=1}^{k-1}(k-i)g_i$.
2. Set $p_0 = 0$. For $i = 1, \ldots, k-1$ compute the prefix sum $p_i = \sum_{j=1}^{i} g_j = p_{i-1} + g_i$.
3. For $i = 1, \ldots, k$, set $\beta_i = (\alpha + \lambda k \alpha_i + p - k p_{i-1})/(1+\lambda)k$.

Now, each β_i is an estimate of the value of q_i. How does it compare to the direct measurement α_i (which has variance $\sigma^2 = 8k^2/\epsilon^2$)? The following result compares the expected error of β_i (which used the direct measurements and the gap information) with the expected error of using only the direct measurements (i.e., α_i only).

COROLLARY 1. *For all $i = 1, \ldots, k$, we have*

$$\frac{E(|\beta_i - q_i|^2)}{E(|\alpha_i - q_i|^2)} = \frac{1+\lambda k}{k+\lambda k} = \frac{\mathrm{Var}(\xi_i) + k\,\mathrm{Var}(\eta_i)}{k(\mathrm{Var}(\xi_i) + \mathrm{Var}(\eta_i))}.$$

For proof, see the full version of this paper [14]. In the case of counting queries, we have $\mathrm{Var}(\xi_i) = \mathrm{Var}(\eta_i) = 8k^2/\epsilon^2$ and thus $\lambda = 1$. The error reduction rate is $\frac{k-1}{2k}$ which is close to 50% when k is large. Our experiments in Section 7 confirm this theoretical result.

6. IMPROVING SPARSE VECTOR

In this section we propose a novel variant that can answer more queries than both the original Sparse Vector [19, 32] and the Sparse-Vector-with-Gap of Wang et al. [42]. We also discuss how the free gap information can be used.

6.1 Adaptive-Sparse-Vector-with-Gap

The Sparse Vector techniques are designed to solve the following problem in a privacy-preserving way: given a stream of queries (with sensitivity 1), find the first k queries whose answers are larger than a public threshold T. This is done by adding noise to the queries and threshold and finding the first k queries whose noisy answers exceed the noisy threshold. Sometimes this procedure creates a feeling of regret – if these k queries are much larger than the threshold, we could have used more noise (hence consumed less privacy budget) to achieve the same result. In this section, we show that Sparse Vector can be made adaptive – so that it will probably use more noise (less privacy budget) for the larger queries. This means if the first k queries are very large, it will still have privacy budget left over to find additional queries that are likely to be over the threshold. Our Adaptive Sparse Vector is shown in Algorithm 2.

The main idea behind this algorithm is that, given a target privacy budget ϵ and an integer k, the algorithm will create three budget parameters: ϵ_0 (budget for the threshold), ϵ_1 (baseline budget for each query) and ϵ_2 (smaller alternative budget for each query, $\epsilon_2 < \epsilon_1$). The privacy budget allocation between threshold and queries is controlled by a hyperparameter $\theta \in (0,1)$ on Line 2. These budget parameters are used as follows. First, the algorithm adds Laplace($1/\epsilon_0$) noise to the threshold and consumes ϵ_0 of the privacy budget. Then, when a query comes in, the algorithm first adds a lot of noise (i.e., Laplace($2/\epsilon_2$)) to the query. The first "if" branch checks if this value is much larger than the noisy threshold (i.e. checks if the gap is $\geq \sigma$ for some[5] σ). If so, then it outputs the following three items: (1) $\top$, (2) the noisy gap, and (3) the amount of privacy budget used for this query (which is ϵ_2). The use of alignments will show that failing this "if" branch consumes no privacy budget. If the first "if" branch fails, then the algorithm adds more moderate noise to the query answer (i.e., Laplace($2/\epsilon_1$)). If this noisy value is larger than the noisy threshold, the algorithm outputs: ($1'$) $\top$, ($2'$) the noisy gap, and ($3'$) the amount of privacy budget consumed (i.e., ϵ_1). If this "if" condition also fails, then the algorithm outputs: ($1''$) $\perp$ and ($2''$) the privacy budget consumed (0 in this case).

[5]In our algorithm, we set σ to be 2 standard deviations of the Laplace($2/\epsilon_2$) distribution.

Algorithm 2: Adaptive-Sparse-Vector-with-Gap. The hyperparameter $\theta \in (0,1)$ controls the budget allocation between threshold and queries.

input : $\boldsymbol{q}$: a list of queries of global sensitivity 1
D: database, ϵ: privacy budget, T: threshold
k: minimum number of above-threshold queries algorithm is able to output

1 **function** AdaptiveSparseVector ($\boldsymbol{q}$, D, T, k, ϵ):
2 $\epsilon_0 \leftarrow \theta\epsilon$; $\epsilon_1 \leftarrow (1-\theta)\epsilon/k$; $\epsilon_2 \leftarrow \epsilon_1/2$; $\sigma \leftarrow 4\sqrt{2}/\epsilon_2$
3 $\eta \leftarrow \mathrm{Lap}(1/\epsilon_0)$
4 $\widetilde{T} \leftarrow T + \eta$
5 cost $\leftarrow \epsilon_0$
6 **foreach** i $\in \{1, \cdots, \mathrm{len}(\boldsymbol{q})\}$ **do**
7 $\xi_i \leftarrow \mathrm{Lap}(2/\epsilon_2)$; $\eta_i \leftarrow \mathrm{Lap}(2/\epsilon_1)$
8 **if** $q_i(D) + \xi_i - \widetilde{T} \geq \sigma$ **then**
9 **output:** ($\top$, $q_i(D)+\xi_i-\widetilde{T}$, bud_used $= \epsilon_2$)
10 cost $\leftarrow$ cost $+ \epsilon_2$
11 **else if** $q_i(D) + \eta_i - \widetilde{T} \geq 0$ **then**
12 **output:** ($\top$, $q_i(D)+\eta_i-\widetilde{T}$, bud_used $= \epsilon_1$)
13 cost $\leftarrow$ cost $+ \epsilon_1$
14 **else**
15 **output:** ($\perp$, bud_used $= 0$)
16 **if** cost $> \epsilon - \epsilon_1$ **then break**

To summarize, for each query, if the top branch succeeds then the privacy budget consumed is ϵ_2, if the middle branch succeeds, the privacy cost is ϵ_1, and if the bottom branch succeeds, there is no additional privacy cost. These properties can be easily seen by focusing on the local alignment – if $M(D,H)$ produces a certain output, how much does H need to change to get a noise vector H' so that $M(D',H')$ returns the same exact output.

Local alignment. To create a local alignment for each pair $D \sim D'$, let $H = (\eta, \xi_1, \eta_1, \xi_2, \eta_2, \ldots)$ where η is the noise added to the threshold T, and ξ_i (resp. η_i) is the noise that should be added to the i^{th} query q_i in Line 8 (resp. Line 11), if execution ever reaches that point. We view the output $\omega = (w_1, \ldots, w_s)$ as a variable-length sequence where each w_i is either $\perp$ or a nonnegative gap (we omit the $\top$ as it is redundant), together with a tag $\in \{0, \epsilon_1, \epsilon_2\}$ indicating which branch w_i is from (and the privacy budget consumed to output w_i). Let $\mathcal{I}_\omega = \{i \mid \mathrm{tag}(w_i) = \epsilon_2\}$ and $\mathcal{J}_\omega = \{i \mid \mathrm{tag}(w_i) = \epsilon_1\}$. That is, $\mathcal{I}_\omega$ is the set of indexes where the output is a gap from the top branch, and $\mathcal{J}_\omega$ is the set of indexes where the output is a gap from the middle branch. For $H \in \mathrm{S}_{\mathrm{D}:\omega}$ define $\phi_{\mathrm{D},\mathrm{D}',\omega}(H) = H' = (\eta', \xi_1', \eta_1', \xi_2', \eta_2', \ldots)$ where

$$\eta' = \eta + 1,$$
$$(\xi_i', \quad \eta_i') = \begin{cases} (\xi_i + 1 + q_i - q_i', \quad \eta_i), & i \in \mathcal{I}_\omega \\ (\xi_i, \quad \eta_i + 1 + q_i - q_i'), & i \in \mathcal{J}_\omega \\ (\xi_i, \quad \eta_i), & \text{otherwise} \end{cases} \tag{3}$$

In other words, we add 1 to the noise that was added to the threshold (thus if the noisy $q(D)$ failed a specific branch, the

noisy $q(D')$ will continue to fail it because of the higher noisy threshold). If a noisy $q(D)$ succeeded in a specific branch, we adjust the query's noise so that the noisy version of $q(D')$ will succeed in that same branch.

LEMMA 4. *Let M be the Adaptive-Sparse-Vector-with-Gap algorithm. For all $D \sim D'$ and ω, the functions $\phi_{\mathrm{D},\mathrm{D}',\omega}$ defined above are acyclic local alignments for M. Furthermore, for every pair $D \sim D'$, there are countably many distinct $\phi_{\mathrm{D},\mathrm{D}',\omega}$.*

Proof. Pick an adjacent pair $D \sim D'$ and an $\omega = (w_1, \dots, w_s)$. For a given $H = (\eta, \xi_1, \eta_1, \dots)$ such that $M(D, H) = \omega$, let $H' = (\eta', \xi_1', \eta_1', \dots) = \phi_{\mathrm{D},\mathrm{D}',\omega}(H)$. Suppose $M(D', H') = \omega' = (w_1', \dots, w_t')$. Our goal is to show $\omega' = \omega$. Choose an $i \le \min(s, t)$.

- If $i \in \mathcal{I}_\omega$, then by (3) we have
$$q_i' + \xi_i' - (T + \eta') = q_i' + \xi_i + 1 + q_i - q_i' - (T + \eta + 1)$$
$$= q_i + \xi_i - (T + \eta) \ge \sigma.$$
This means the first "if" branch succeeds in both executions and the gaps are the same . Therefore, $w_i' = w_i$.
- If $i \in \mathcal{J}_\omega$, then by (3) we have
$$q_i' + \xi_i' - (T + \eta') = q_i' + \xi_i - (T + \eta + 1)$$
$$= q_i' - 1 + \xi_i - (T + \eta) \le q_i + \xi_i - (T + \eta) < \sigma,$$
$$q_i' + \eta_i' - (T + \eta') = q_i' + \eta_i + 1 + q_i - q_i' - (T + \eta + 1)$$
$$= q_i + \eta_i - (T + \eta) \ge 0.$$
The first inequality is due to the sensitivity restriction: $|q_i - q_i'| \le 1 \implies q_i' - 1 \le q_i$. These two equations mean that the first "if" branch fails and the second "if" branch succeeds in both executions, and the gaps are the same. Hence $w_i' = w_i$.
- If $i \notin \mathcal{I}_\omega \cup \mathcal{J}_\omega$, then by a similar argument we have
$$q_i' + \xi_i' - (T + \eta') \le q_i + \xi_i - (T + \eta) < \sigma,$$
$$q_i' + \eta_i' - (T + \eta') \le q_i + \eta_i - (T + \eta) < 0.$$
Hence both executions go to the last "else" branch and $w_i' = (\perp, 0) = w_i$.

Therefore for all $1 \le i \le \min(s, t)$, we have $w_i' = w_i$. That is, either ω' is a prefix of ω, or vice versa. Let $\boldsymbol{q}$ be the vector of queries passed to the algorithm and let $\mathrm{len}(\boldsymbol{q})$ be the number of queries it contains (which can be finite or infinity). By the termination condition of Algorithm 2 we have two possibilities.

- $s = \mathrm{len}(\boldsymbol{q})$: in this case there is still enough privacy budget left after answering $s-1$ above-threshold queries, and we must have $t = \mathrm{len}(\boldsymbol{q})$ too because $M(D', H')$ will also run through all the queries (it cannot stop until it has exhausted the privacy budget or hits the end of the query sequence).
- $s < \mathrm{len}(\boldsymbol{q})$: in this case the privacy budget is exhausted after outputting w_s and we must also have $t = s$.

Thus $t = s$ and hence $\omega' = \omega$. The local alignments are clearly acyclic (e.g., use the identity permutation). Note that $\phi_{\mathrm{D},\mathrm{D}',\omega}$ only depends on ω through $\mathcal{I}_\omega$ and $\mathcal{J}_\omega$ (the sets of queries whose noisy values were larger than the noisy threshold). There are only countably many possibilities for $\mathcal{I}_\omega$ and $\mathcal{J}_\omega$ and thus countably many distinct $\phi_{\mathrm{D},\mathrm{D}',\omega}$. □

Alignment cost and privacy. Now we establish the alignment cost and the privacy property of Algorithm 2.

THEOREM 4. *The Adaptive-Sparse-Vector-with-Gap satisfies ϵ-differential privacy.*

Proof. Again, the only thing nontrivial is to bound the alignment cost. We use the $\epsilon_0, \epsilon_1, \epsilon_2$ and ϵ defined in Algorithm 2. From (3) we have

$$\mathrm{cost}(\phi_{\mathrm{D},\mathrm{D}',\omega}) = \epsilon_0|\eta' - \eta| + \sum_{i=1}^{\infty}\left(\frac{\epsilon_2}{2}|\xi_i' - \xi_i| + \frac{\epsilon_1}{2}|\eta_i' - \eta_i|\right)$$
$$= \epsilon_0 + \sum_{i\in\mathcal{I}_\omega}\frac{\epsilon_2}{2}|1 + q_i - q_i'| + \sum_{i\in\mathcal{J}_\omega}\frac{\epsilon_1}{2}|1 + q_i - q_i'|$$
$$\le \epsilon_0 + \epsilon_2|\mathcal{I}_\omega| + \epsilon_1|\mathcal{J}_\omega| \le \epsilon.$$

The first inequality is from sensitivity assumption: $|1 + q_i - q_i'| \le 1 + |q_i - q_i'| \le 2$. The second inequality is from loop invariant on Line 16: $\epsilon_0 + \epsilon_2|\mathcal{I}_\omega| + \epsilon_1|\mathcal{J}_\omega| = \texttt{cost} \le \epsilon - \epsilon_1 + \max(\epsilon_1, \epsilon_2) = \epsilon$. □

We note that if we remove the first branch of Algorithm 2 (Line 8 through 10) or set $\sigma = \infty$, we recover the Sparse-Vector-with-Gap algorithm of Wang. et al. [42]. Also, Algorithm 2 can be easily extended with multiple additional "if" branches. For simplicity we do not include such variations. In our setting, $\epsilon_2 = \epsilon_1/2$ so, theoretically, if queries are very far from the threshold, our adaptive version of Sparse Vector will be able to find twice as many of them as the non-adaptive version. Lastly, if all queries are monotonic queries, then Algorithm 2 can be further improved: we can use $\mathrm{Lap}(1/\epsilon_2)$ and $\mathrm{Lap}(1/\epsilon_1)$ noises instead in Line 7.[6]

6.2 Utilizing Gap Information

When Sparse-Vector-with-Gap or Adaptive-Sparse-Vector-with-Gap returns a gap γ_i for a query q_i, we can add to it the public threshold T. This means $\gamma_i + T$ is an estimate of the value of $q_i(D)$. We can ask two questions: how can we improve the accuracy of this estimate and how can we be confident that the true answer $q_i(D)$ is really larger than the threshold T?

Lower confidence interval. Recall that the randomness in the gap in Adaptive-Sparse-Vector-with-Gap (Algorithm 2) is of the form $\eta_i - \eta$ where η and η_i are independent zero mean Laplace variables with scale $1/\epsilon_0$ and $1/\epsilon_*$, where ϵ_* is either ϵ_1 or ϵ_2, depending on the branch. The random variable $\eta_i - \eta$ has the following lower tail bound:

LEMMA 5. *For any $t \ge 0$ we have*

$$\mathbb{P}(\eta_i - \eta \ge -t) = \begin{cases} 1 - \frac{\epsilon_0^2 e^{-\epsilon_* t} - \epsilon_*^2 e^{-\epsilon_0 t}}{2(\epsilon_0^2 - \epsilon_*^2)} & \epsilon_0 \ne \epsilon_* \\ 1 - (\frac{2+\epsilon_0 t}{4})e^{-\epsilon_0 t} & \epsilon_0 = \epsilon_* \end{cases}$$

For proof see the full version of this paper [14]. For any confidence level, say 95%, we can use this result to find a

[6] In the case of monotonic queries, if $\forall i : q_i \ge q_i'$, then the alignment changes slightly: we set $\eta' = \eta$ (the random variable added to the threshold) and set the adjustment to noise in winning "if" branches to $q_i - q_i'$ instead of $1 + q_i - q_i'$ (hence cost terms become $|q_i - q_i'|$ instead of $|1 + q_i - q_i'|$). If $\forall i : q_i \le q_i'$ then we keep the original alignment but in the cost calculation we note that $|1 + q_i - q_i'| \le 1$ (due to the monotonicity and sensitivity).

number $t_{.95}$ such that $\mathbb{P}((\eta_i - \eta) \geq -t_{.95}) = .95$. This is a lower confidence bound, so that the true value $q_i(D)$ is $\geq$ our estimated value $\gamma_i + T$ minus $t_{.95}$ with probability 0.95.

Improving accuracy. To improve accuracy, one can split the privacy budget ϵ in half. The first half $\epsilon' \equiv \epsilon/2$ can be used to run Sparse-Vector-with-Gap (or Adaptive-Sparse-Vector-with-Gap) and the second half $\epsilon'' \equiv \epsilon/2$ can be used to provide an independent noisy measurement of the selected queries (i.e. if we selected k queries, we add Laplace(k/ϵ'') noise to each one). Denote the selected queries by $q_1, \ldots, q_k$, the noisy gaps by $\gamma_1, \ldots, \gamma_k$ and the independent noisy measurements by $\alpha_1, \ldots, \alpha_k$. The noisy estimates can be combined together with the gaps to get improved estimates β_i of $q_i(D)$ in the standard way (inverse-weighting by variance):

$$\beta_i = \left(\frac{\alpha_i}{\text{Var}(\alpha_i)} + \frac{\gamma_i + T}{\text{Var}(\gamma_i)}\right) \Big/ \left(\frac{1}{\text{Var}(\alpha_i)} + \frac{1}{\text{Var}(\gamma_i)}\right).$$

As shown in [32], the optimal budget allocation between threshold noise and query noises *within* SVT (and therefore also Sparse-Vector-with-Gap) is the ratio $1 : (2k)^{\frac{2}{3}}$. Under this setting, we have $\text{Var}(\gamma_i) = 8(1 + (2k)^{\frac{2}{3}})^3/\epsilon^2$. Also, we know $\text{Var}(\alpha_i) = 8k^2/\epsilon^2$. Therefore,

$$\frac{E(|\beta_i - q_i|^2)}{E(|\alpha_i - q_i|^2)} = \frac{\text{Var}(\beta_i)}{\text{Var}(\alpha_i)} = \frac{(1 + \sqrt[3]{4k^2})^3}{(1 + \sqrt[3]{4k^2})^3 + k^2} < 1.$$

Since $\lim_{k\to\infty} \frac{(1+\sqrt[3]{4k^2})^3}{(1+\sqrt[3]{4k^2})^3+k^2} = \frac{4}{5}$, the improvement in accuracy approaches 20% as k increases. For monotonic queries, the optimal budget allocation *within* SVT is $1 : k^{\frac{2}{3}}$. Then we have $\text{Var}(\gamma_i) = 8(1+k^{\frac{2}{3}})^3/\epsilon^2$ and the error reduction rate is $1 - \frac{(1+\sqrt[3]{k^2})^3}{(1+\sqrt[3]{k^2})^3+k^2}$ which is close to 50% when k is large. Our experiments in Section 7 confirm this improvement.

7. EXPERIMENTS

We now evaluate the algorithms proposed in this paper.

7.1 Datasets

For evaluation, we used the two real datasets from [32]: BMP-POS, Kosarak and a synthetic dataset T40I10D100K created by the generator from the IBM Almaden Quest research group. These datasets are collections of transactions (each transaction is a set of items). In our experiments, the queries correspond to the counts of each item (i.e. how many transactions contained item #23?) The statistics of the datasets are listed below.

Table 2: Statistics of Datasets

Dataset	# of Records	# of Unique Items
BMS-POS	515,597	1,657
Kosarak	990,002	41,270
T40I10D100K	100,000	942

7.2 Gap Information + Postprocessing

The first set of experiments is to measure how gap information can help us improve estimates in selected queries. We use the setup of Sections 5.2 and 6.2. That is, a data analyst splits the privacy budget ϵ in half. She uses the first half to select k queries using Noisy-Top-K-with-Gap or Sparse-Vector-with-Gap (or Adaptive-Sparse-Vector-with-Gap) and then uses the second half of the privacy budget to obtain independent noisy measurements of each selected query.

If one were unaware that gap information came for free, one would just use those noisy measurements as estimates for the query answers. The error of this approach is the gap-free baseline. However, since the gap information does come for free, we can use the postprocessing described in Sections 5.2 and 6.2 to improve accuracy (we call this latter approach Sparse-Vector-with-Gap with Measures and Noisy-Top-K-with-Gap with Measures).

We first evaluate the percent improvement in mean squared error (MSE) of the postprocessing approach compared to the gap-free baseline and compare this improvement to our theoretical analysis. As discussed in Section 6.2, we set the budget allocation ratio *within* the Sparse-Vector-with-Gap algorithm (i.e., the budget allocation between the threshold and queries) to be $1 : k^{\frac{2}{3}}$ for monotonic queries and $1 : (2k)^{\frac{2}{3}}$ otherwise – such a ratio is recommended in [32] for the original Sparse Vector. The threshold used for Sparse-Vector-with-Gap is randomly picked from the top $2k$ to top $8k$ in each dataset for each run.[7] All numbers plotted are averaged over 10,000 runs. Due to space constraints, we only show experiments for counting queries (which are monotonic).

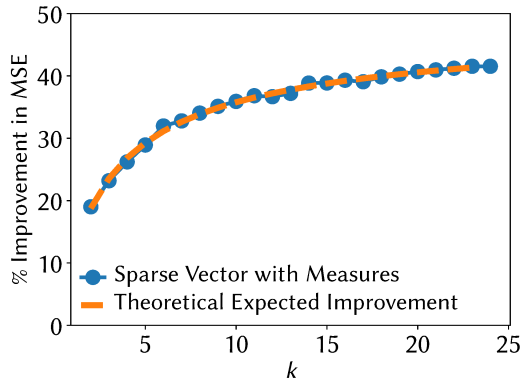
Our theoretical analysis in Sections 5.2 and 6.2 suggested that the improvements can reach up to 50% in case of monotonic queries (and 20% for non-monotonic queries) as k increases. This is confirmed in Figures 1a, for Sparse-Vector-with-Gap and Figures 1b, for our Top-K algorithm using the BMS-POS dataset (results for the other datasets are nearly identical). These figures plot the theoretical and empirical percent improvement in MSE as a function of k and show the power of the free gap information.

We also generated corresponding plots where k is held fixed and the total privacy budget ϵ is varied. We only present the result for the kosarak dataset as results for the other datasets are nearly identical. For Sparse-Vector-with-Gap, Figures 2a confirms that this improvement is stable for different ϵ values. For our Top-K algorithm, Figures 2b confirms that this improvement is also stable for different values of ϵ.

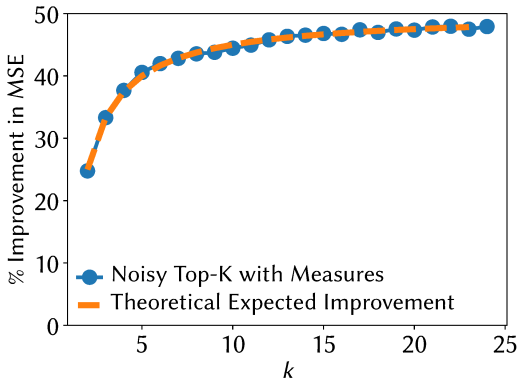
7.3 Benefits of Adaptivity

In this subsection we present an evaluation of the budget-saving properties of our novel Adaptive-Sparse-Vector-with-Gap algorithm to show that it can answer more above-threshold queries than Sparse Vector and Sparse-Vector-with-Gap at the same privacy cost (or, conversely, answer the same number of queries but with leftover budget that can be used for other purposes). First note that Sparse Vector and Sparse-Vector-with-Gap both answer exactly the same amount of queries, so we only need to compare Adaptive-Sparse-Vector-with-Gap to the original Sparse Vector [19, 32]. In both algorithms, the budget allocation between the threshold noise and query noise is set according to the ratio $1 : k^{\frac{2}{3}}$ (i.e., the hyperparameter θ in Adaptive-Sparse-Vector-with-Gap is set to $1/(1 + k^{\frac{2}{3}})$), following recommendations for SVT by Lyu et. al. [32]. The threshold is randomly picked from the top $2k$ to top $8k$ in each dataset and all reported numbers are averaged over 10,000 runs.

[7] Selecting thresholds for SVT in experiments is difficult, but we feel this may be fairer than averaging the answer to the top k^{th} and $k + 1^{\text{th}}$ queries as was done in prior work [32].

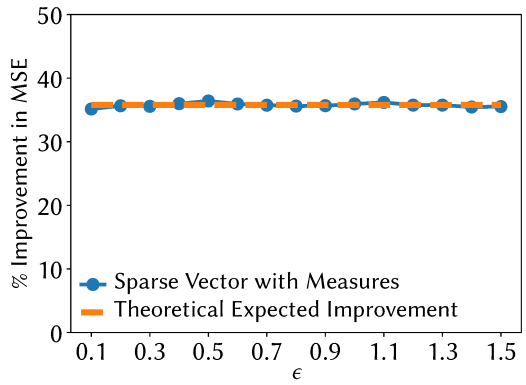


(a) Sparse-Vector-with-Gap with Measures, BMS-POS.

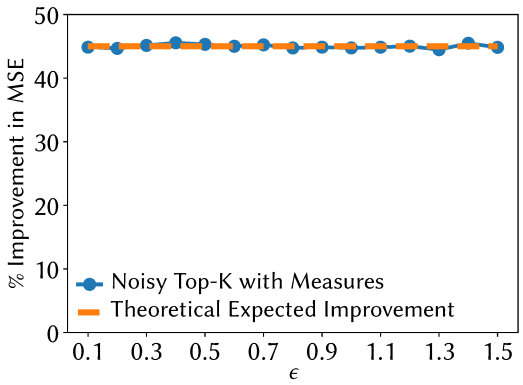


(b) Noisy-Top-K-with-Gap with Measures, BMS-POS.

Figure 1: Improvement percentage of Mean Squared Error on monotonic queries, for different k, for Sparse Vector with Gap and Noisy Top-K with Gap when half the privacy budget is used for query selection and the other half is used for measurement of their answers. Privacy budget $\epsilon = 0.7$.



(a) Sparse-Vector-with-Gap with Measures, kosarak.

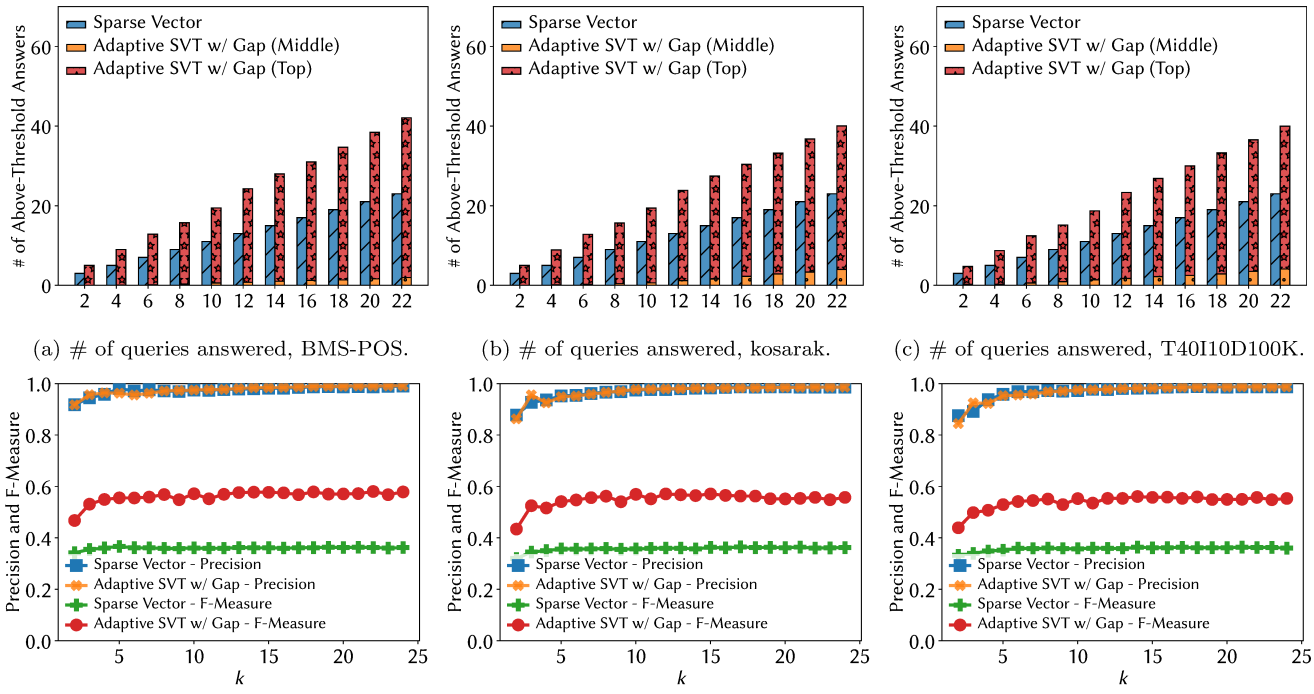


(b) Noisy-Top-K-with-Gap with Measures, kosarak.

Figure 2: Improvement percentage of Mean Squared Error on monotonic queries, for different ϵ, for Sparse Vector with Gap and Noisy Top-K with Gap when half the privacy budget is used for query selection and the other half is used for measurement of their answers. k is set to 10.

Number of queries answered. We first compare the number of queries answered by each algorithm as the parameter k is varied from 2 to 22 with a privacy budget of $\epsilon = 0.7$ (results for other settings of the total privacy budget are similar). The results are shown in Figure 3a, 3b, and 3c. In each of these bar graphs, the left (blue) bar is the number of answers returned by Sparse Vector and the right bar is the number of answers returned by Adaptive-Sparse-Vector-with-Gap. This right bar is broken down into two components: the number of queries returned from the top "if" branch (corresponding to queries that were significantly larger than the threshold even after a lot of noise was added) and the number of queries returned from the middle "if" branch. Queries returned from the top branch of Adaptive-Sparse-Vector-with-Gap have less privacy cost than those returned by Sparse Vector. Queries returned from the middle branch of Adaptive-Sparse-Vector-with-Gap have the same privacy cost as in Sparse Vector. We see that most queries are answered in the top branch of Adaptive-Sparse-Vector-with-Gap, meaning that the above-threshold queries were generally large (much larger than the threshold). Since Adaptive-Sparse-Vector-with-Gap uses more noise in the top branch, it uses less privacy budget to answer those queries and uses the remaining budget to provide additional answers (up to an average of 18 more answers when k was set to 22).

Precision and F-Measure. Although the adaptive algorithm can answer more above-threshold queries than the original, one can still ask the question of whether the returned queries really are above the threshold. Thus we can look at the precision of the returned results (the fraction of returned queries that are actually above the threshold) and the widely used F-Measure (the harmonic mean of precision and recall). One would expect that the precision of Adaptive-Sparse-Vector-with-Gap should be less than that of Sparse Vector, because the adaptive version can use more noise when processing queries. In Figures 3d, 3e, and 3f we compare the precision and F-Measure of the two algorithms. Generally we see very little difference in precision. On the other hand, since Adaptive-Sparse-Vector-with-Gap answers more queries while maintaining high precision, the recall of Adaptive-Sparse-Vector-with-Gap would be much



(a) # of queries answered, BMS-POS. (b) # of queries answered, kosarak. (c) # of queries answered, T40I10D100K.

(d) Precision and F-Measure, BMS-POS. (e) Precision and F-Measure, kosarak. (f) Precision and F-Measure, T40I10D100K.

Figure 3: Results for Adaptive-Sparse-Vector-with-Gap under different k's for monotonic queries. Privacy budget $\epsilon = 0.7$ and x-axis: k.

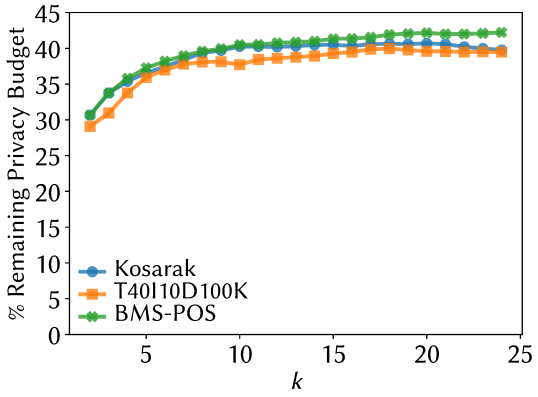


Figure 4: Remaining privacy budget when Adaptive-Sparse-Vector-with-Gap is stopped after answering k queries using different datasets. Privacy budget $\epsilon = 0.7$.

larger than Sparse Vector, thus leading to the F-Measure being roughly 1.5 times that of Sparse Vector.

Remaining Privacy Budget. If a query is large, Adaptive-Sparse-Vector-with-Gap may only need to use a small part of the privacy budget to determine that the query is likely above the noisy threshold. That is, it may produce an output in its top branch, where a lot of noise (hence less privacy budget) is used. If we stop Adaptive-Sparse-Vector-with-Gap after k returned queries, it may still have some privacy budget left over (in contrast to standard versions of Sparse Vector, which use up all of their privacy budget). This remaining privacy budget can then be used for other data analysis tasks. For all three datasets, Figure 4 shows the percentage of privacy budget that is left over when Adaptive-Sparse-Vector-with-Gap is run with parameter k and stopped after k queries are returned. We see that roughly 40% of the privacy budget is left over, confirming that Adaptive-Sparse-Vector-with-Gap is able to save a significant amount of privacy budget.

8. GENERAL RANDOMNESS ALIGNMENT AND PROOF OF LEMMA 1

In this section, we prove Lemma 1, which was used to establish the privacy properties of the algorithms we proposed. The proof of the lemma requires a more general theorem for working with randomness alignment functions. We explicitly list all of the conditions needed for the sake of reference (many prior works had incorrect proofs because they did not have such a list to follow). In the general setting, the method of randomness alignment requires the following steps.

1. For each pair of adjacent databases $D \sim D'$ and $\omega \in \Omega$, define a randomness alignment $\phi_{\mathrm{D},\mathrm{D}'}$ or local alignment functions $\phi_{\mathrm{D},\mathrm{D}',\omega} : \mathrm{S}_{\mathrm{D}:\omega} \to \mathrm{S}_{\mathrm{D}':\omega}$ (see notation in Table 1). In the case of local alignments this involves proving that if $M(D, H) = \omega$ then $M(D', \phi_{\mathrm{D},\mathrm{D}',\omega}(H)) = \omega$.
2. Show that $\phi_{\mathrm{D},\mathrm{D}'}$ (or all the $\phi_{\mathrm{D},\mathrm{D}',\omega}$) is one-to-one (it does not need to be onto). That is, if we know D, D', ω and we are given the value $\phi_{\mathrm{D},\mathrm{D}'}(H)$ (or $\phi_{\mathrm{D},\mathrm{D}',\omega}(H)$), we can obtain the value H.

3. For each pair of adjacent databases $D \sim D'$, bound the *alignment cost* of $\phi_{\mathrm{D},\mathrm{D}'}$ ($\phi_{\mathrm{D},\mathrm{D}'}$ is either given or constructed by piecing together the local alignments). Bounding the alignment cost means the following: If f is the density (or probability mass) function of H, find a constant a such that $f(H)/f(\phi_{\mathrm{D},\mathrm{D}'}(H)) \leq a$ for all H (except a set of measure 0). In the case of local alignments, one can instead show the following. For all ω, and adjacent $D \sim D'$ the ratio $f(H)/f(\phi_{\mathrm{D},\mathrm{D}',\omega}(H)) \leq a$ for all H (except on a set of measure 0).
4. Bound the *change-of-variables cost* of $\phi_{\mathrm{D},\mathrm{D}'}$ (only necessary when H is not discrete). One must show that the Jacobian of $\phi_{\mathrm{D},\mathrm{D}'}$, defined as $J_{\phi_{\mathrm{D},\mathrm{D}'}} = \frac{\partial\, \phi_{\mathrm{D},\mathrm{D}'}}{\partial H}$, exists (i.e. $\phi_{\mathrm{D},\mathrm{D}'}$ is differentiable) and is continuous except on a set of measure 0. Furthermore, for all pairs $D \sim D'$, show the quantity $|\det J_{\phi_{\mathrm{D},\mathrm{D}'}}|$ is lower bounded by some constant $b > 0$. If $\phi_{\mathrm{D},\mathrm{D}'}$ is constructed by piecing together local alignments $\phi_{\mathrm{D},\mathrm{D}',\omega}$ then this is equivalent to showing the following (i) $|\det J_{\phi_{\mathrm{D},\mathrm{D}',\omega}}|$ is lower bounded by some constant $b > 0$ for every $D \sim D'$ and ω; and (ii) for each $D \sim D'$, the set Ω can be partitioned into countably many disjoint measurable sets $\Omega = \bigcup_i \Omega_i$ such that whenever ω and ω^* are in the same partition, then $\phi_{\mathrm{D},\mathrm{D}',\omega}$ and $\phi_{\mathrm{D},\mathrm{D}',\omega^*}$ are the same function. Note that this last condition (ii) is equivalent to requiring that the local alignments must be defined without using the axiom of choice (since non-measurable sets are not constructible otherwise) and for each $D \sim D'$, the number of distinct local alignments is countable. That is, the set $\{\phi_{\mathrm{D},\mathrm{D}',\omega} \mid \omega \in \Omega\}$ is countable (i.e., for many choices of ω we get the same exact alignment function).

THEOREM 5. *Let M be a randomized algorithm that terminates with probability 1 and suppose the number of random variables used by M can be determined from its output. If, for all pairs of adjacent databases $D \sim D'$, there exist randomness alignment functions $\phi_{\mathrm{D},\mathrm{D}'}$ (or local alignment functions $\phi_{\mathrm{D},\mathrm{D}',\omega}$ for all $\omega \in \Omega$ and $D \sim D'$) that satisfy conditions 1 though 4 above, then M satisfies $\ln(a/b)$-differential privacy.*

Proof. We need to show that for all $D \sim D'$ and $E \subseteq \Omega$, $\mathbb{P}(\mathrm{S}_{\mathrm{D}:\mathrm{E}}) \leq (a/b)\mathbb{P}(\mathrm{S}_{\mathrm{D}':\mathrm{E}})$.

First we note that if we have a randomness alignment $\phi_{\mathrm{D},\mathrm{D}'}$, we can define corresponding local alignment functions as follows $\phi_{\mathrm{D},\mathrm{D}',\omega}(H) = \phi_{\mathrm{D},\mathrm{D}'}(H)$ (in other words, they are all the same). The conditions on local alignments are a superset of the conditions on randomness alignments, so for the rest of the proof we work with the $\phi_{\mathrm{D},\mathrm{D}',\omega}$.

Let $\phi_1, \phi_2, \ldots$ be the distinct local alignment functions (there are countably many of them by Condition 4). Let $E_i = \{\omega \in E \mid \phi_{\mathrm{D},\mathrm{D}',\omega} = \phi_i\}$. By Conditions 1 and 2 we have that for each $\omega \in E_i$, ϕ_i is one-to-one on $\mathrm{S}_{\mathrm{D}:\omega}$ and $\phi_i(\mathrm{S}_{\mathrm{D}:\omega}) \subseteq \mathrm{S}_{\mathrm{D}':\omega}$. Note that $\mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}} = \cup_{\omega \in E_i} \mathrm{S}_{\mathrm{D}:\omega}$ and $\mathrm{S}_{\mathrm{D}':\mathrm{E}_\mathrm{i}} = \cup_{\omega \in E_i} \mathrm{S}_{\mathrm{D}':\omega}$. Furthermore, the sets $\mathrm{S}_{\mathrm{D}:\omega}$ are pairwise disjoint for different ω and the sets $\mathrm{S}_{\mathrm{D}':\omega}$ are pairwise disjoint for different ω. It follows that ϕ_i is one-to-one on $\mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}}$ and $\phi_i(\mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}}) \subseteq \mathrm{S}_{\mathrm{D}':\mathrm{E}_\mathrm{i}}$. Thus for any $H' \in \phi_i(\mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}})$ there exists $H \in \mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}}$ such that $H = \phi_i^{-1}(H')$. By Conditions 3 and 4, we have $\frac{f(H)}{f(\phi_i(H))} = \frac{f(\phi_i^{-1}(H'))}{f(H')} \leq a$ for all $H \in \mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}}$, and $|\det J_{\phi_i}| \geq b$ (except on a set of measure 0). Then the following is true:

$$\begin{aligned}
\mathbb{P}(\mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}}) &= \int_{\mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}}} f(H)dH \\
&= \int_{\phi_i(\mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}})} f(\phi_i^{-1}(H'))\ \frac{1}{|\det J_{\phi_i}|}dH' \\
&\leq \int_{\phi_i(\mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}})} af(H')\frac{1}{b}dH' = \frac{a}{b}\int_{\phi_i(\mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}})} f(H')dH' \\
&\leq \frac{a}{b}\int_{\mathrm{S}_{\mathrm{D}':\mathrm{E}_\mathrm{i}}} f(H')dH' = \frac{a}{b}\mathbb{P}(\mathrm{S}_{\mathrm{D}':\mathrm{E}_\mathrm{i}}).
\end{aligned}$$

The second equation is the change of variables formula in calculus. The last inequality follows from the containment $\phi_i(\mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}}) \subseteq \mathrm{S}_{\mathrm{D}':\mathrm{E}_\mathrm{i}}$ and the fact that the density f is non-negative. In the case that H is discrete, simply replace the density f with a probability mass function, change the integral into a summation, ignore the Jacobian term and set $b = 1$. Finally, since $E = \cup_i E_i$ and $E_i \cap E_j = \emptyset$ for $i \neq j$, we conclude that

$$\mathbb{P}(\mathrm{S}_{\mathrm{D}:\mathrm{E}}) = \sum_i \mathbb{P}(\mathrm{S}_{\mathrm{D}:\mathrm{E}_\mathrm{i}}) \leq \frac{a}{b}\sum_i \mathbb{P}(\mathrm{S}_{\mathrm{D}':\mathrm{E}_\mathrm{i}}) = \frac{a}{b}\mathbb{P}(\mathrm{S}_{\mathrm{D}':\mathrm{E}}).$$

□

We now present the proof of **Lemma 1.**

Proof. Let $\phi_{\mathrm{D},\mathrm{D}',\omega}(H) = H' = (\eta'_1, \eta'_2, \ldots)$. By acyclicity there is some permutation π under which $\eta_{\pi(1)} = \eta'_{\pi(1)} - c$ where c is some constant depending on $D \sim D'$ and ω. Thus $\eta_{\pi(1)}$ is uniquely determined by H'. Now (as an induction hypothesis) assume $\eta_{\pi(1)}, \ldots, \eta_{\pi(j-1)}$ are uniquely determined by H' for some $j > 1$, then $\eta_{\pi(j)} = \eta'_{\pi(j)} - \psi^{(j)}_{D,D',\omega}(\eta_{\pi(1)}, \ldots, \eta_{\pi(j-1)})$, so $\eta_{\pi(j)}$ is also uniquely determined by H'. Thus by strong induction H is uniquely determined by H', i.e., $\phi_{\mathrm{D},\mathrm{D}',\omega}$ is one-to-one. It is easy to see that with this ordering, $J_{\phi_{\mathrm{D},\mathrm{D}',\omega}}$ is an upper triangular matrix with 1's on the diagonal. Since permuting variables doesn't change $|\det J_{\phi_{\mathrm{D},\mathrm{D}',\omega}}|$, we have $|\det J_{\phi_{\mathrm{D},\mathrm{D}',\omega}}| = 1$ since that is the determinant of upper triangular matrices. Furthermore, (recalling the definition of the cost of $\phi_{\mathrm{D},\mathrm{D}',\omega}$), clearly

$$\ln \frac{f(H)}{f(\phi_\omega(H))} = \sum_i \ln \frac{f_i(\eta_i)}{f_i(\eta'_i)} \leq \sum_i |\eta_i - \eta'_i|/\alpha_i \leq \epsilon$$

The first inequality follows from Condition 3 of Lemma 1 and the second from Condition 4. □

9. CONCLUSIONS AND FUTURE WORK

In this paper we introduced the Adaptive Sparse Vector with Gap and Noisy Top-K with Gap mechanisms, which were based on the observation that the classical Sparse Vector and Noisy Max mechanisms could release additional information at no cost to privacy. We also provided applications of this free gap information.

Future directions include using this technique to design additional mechanisms as well as finding new applications for these mechanisms in fine-tuning the accuracy of data release algorithms that use differential privacy.

Acknowledgments

This work was supported by NSF Awards CNS-1702760 and CNS-1931686.

10. REFERENCES

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[2] J. M. Abowd. The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2867–2867. ACM, 2018.

[3] A. Albarghouthi and J. Hsu. Synthesizing coupling proofs of differential privacy. *Proceedings of the ACM on Programming Languages*, 2(POPL):58, 2017.

[4] G. Barthe, M. Gaboardi, B. Gregoire, J. Hsu, and P.-Y. Strub. Proving differential privacy via probabilistic couplings. In *IEEE Symposium on Logic in Computer Science (LICS)*, 2016.

[5] A. Beimel, K. Nissim, and U. Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. *Theory of Computing*, 12(1):1–61, 2016.

[6] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010.

[7] A. Bittau, U. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, 2017.

[8] M. Bun and T. Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Proceedings of the 14th International Conference on Theory of Cryptography - Volume 9985*, 2016.

[9] U. S. C. Bureau. On the map: Longitudinal employer-household dynamics. `https://lehd.ces.census.gov/applications/help/onthemap.html#!confidentiality_protection`.

[10] K. Chaudhuri, D. Hsu, and S. Song. The large margin mechanism for differentially private maximization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, 2014.

[11] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

[12] Y. Chen, A. Machanavajjhala, J. P. Reiter, and A. F. Barrientos. Differentially private regression diagnostics. In *IEEE 16th International Conference on Data Mining (ICDM)*, 2016.

[13] B. Ding, J. Kulkarni, and S. Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[14] Z. Ding, Y. Wang, D. Zhang, and D. Kifer. Free gap information from the differentially private sparse vector and noisy max mechanisms. *arXiv preprint arXiv:1904.12773*, 2019.

[15] C. Dwork. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12, Berlin, Heidelberg, 2006. Springer-Verlag.

[16] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

[17] C. Dwork and J. Lei. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 371–380. ACM, 2009.

[18] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[19] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(34):211–407, 2014.

[20] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, 2019.

[21] Ú. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067. ACM, 2014.

[22] M. Fanaeepour and B. I. P. Rubinstein. Histogramming privately ever after: Differentially-private data-dependent error bound optimisation. In *Proceedings of the 34th International Conference on Data Engineering*, ICDE. IEEE, 2018.

[23] Q. Geng and P. Viswanath. The optimal mechanism in differential privacy. In *2014 IEEE International Symposium on Information Theory*, 2014.

[24] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. In *STOC*, pages 351–360, 2009.

[25] S. Haney, A. Machanavajjhala, J. M. Abowd, M. Graham, M. Kutzbach, and L. Vilhuber. Utility cost of formal privacy for releasing national employer-employee statistics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, 2017.

[26] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *NIPS*, 2012.

[27] N. Johnson, J. P. Near, and D. Song. Towards practical differential privacy for sql queries. *PVLDB*, 11(5):526–539, 2018.

[28] I. Kotsogiannis, A. Machanavajjhala, M. Hay, and G. Miklau. Pythia: Data dependent differentially private algorithm selection. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, 2017.

[29] E. Lehmann and G. Casella. *Theory of Point Estimation*. Springer Verlag, 1998.

[30] K. Ligett, S. Neel, A. Roth, B. Waggoner, and S. Z. Wu. Accuracy first: Selecting a differential privacy level for accuracy constrained ERM. In *NIPS*, 2017.

[31] J. Liu and K. Talwar. Private selection from private candidates. *arXiv preprint arXiv:1811.07971*, 2018.

[32] M. Lyu, D. Su, and N. Li. Understanding the sparse vector technique for differential privacy. *PVLDB*, 10(6):637–648, 2017.

[33] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: From theory to practice on the map. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 277–286, 2008.

[34] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103, 2007.

[35] F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, pages 19–30, 2009.

[36] I. Mironov. Rényi differential privacy. In *30th IEEE Computer Security Foundations Symposium, CSF*, 2017.

[37] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and lfar Erlingsson. Scalable private learning with pate. In *International Conference on Learning Representations (ICLR)*, 2018.

[38] S. Raskhodnikova and A. D. Smith. Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism. In *FOCS*, pages 495–504. IEEE Computer Society, 2016.

[39] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang. Privacy loss in apple's implementation of differential privacy. In *3rd Workshop on the Theory and Practice of Differential Privacy at CCS*, 2017.

[40] A. D. P. Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(8), 2017.

[41] A. G. Thakurta and A. Smith. Differentially private feature selection via stability arguments, and the robustness of the lasso. In *Proceedings of the 26th Annual Conference on Learning Theory*, 2013.

[42] Y. Wang, Z. Ding, G. Wang, D. Kifer, and D. Zhang. Proving differential privacy with shadow execution. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, pages 655–669, New York, NY, USA, 2019. ACM.

[43] D. Zhang and D. Kifer. Lightdp: Towards automating differential privacy proofs. In *ACM Symposium on Principles of Programming Languages (POPL)*, pages 888–901, 2017.

[44] D. Zhang, R. McKenna, I. Kotsogiannis, M. Hay, A. Machanavajjhala, and G. Miklau. Ektelo: A framework for defining differentially-private computations. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, 2018.