Differentially Private Top-k Selection via Stability on Unknown Domain

Ricardo Silva Carvalho, Ke Wang, Lovedeep Gondara
Department of Computing Science
Simon Fraser University

Chunyan Miao
School of Computer Science and Engineering
Nanyang Technological University

Abstract

We propose a new method that satisfies approximate differential privacy for top-k selection with unordered output in the unknown data domain setting, not relying on the full knowledge of the domain universe. Our algorithm only requires looking at the top-$\bar{k}$ elements for any given $\bar{k} \geq k$, thus, enforcing the principle of minimal privilege. Unlike previous methods, our privacy parameter ε does not scale with k, giving improved applicability for scenarios of very large k. Moreover, our novel construction, which combines the sparse vector technique and stability efficiently, can be applied as a general framework to any type of query, thus being of independent interest. We extensively compare our algorithm to previous work of top-k selection on the unknown domain, and show, both analytically and on experiments, settings where we outperform the current state-of-the-art.

1 INTRODUCTION

Many exploratory analyses involve selecting the top-k most frequent elements in a dataset. For example, when analyzing a dataset with users' purchases, a company may need to know the top-100 most popular products. Similarly, in a given medical study, a researcher may need to know the top-5 most common diseases from a dataset of patients. One important aspect of these analyses is that the top-k selection needs to be performed without compromising the privacy of any participating user. It is known that data mining may violate privacy [1], thus for a given analysis that does not state privacy guarantees, users may either not opt-in to participate or opt-out. Additionally, if one can infer users' interests from the analysis' results, auxiliary information may be used to link more data to these users [2], making the seemingly harmless result possibly dangerous.

Currently, one of the most widely adopted formal definitions for preserving privacy is Differential Privacy (DP) [3]. It ensures that the result of an algorithm satisfying DP will have very little impact if we add or remove any user from the dataset used to generate the result. Many companies have adopted differential privacy as a standard, such as Google [4], Microsoft [5] and LinkedIn [6]. Our goal is to design algorithms for differentially private top-k selection that are practical enough to be easily coupled with existing systems.

In security, there is a principle applicable to many different problems called *principle of minimal privilege*, which requires that a system only grants the access or permissions that are strictly necessary to perform a certain task. With this in mind, we design mechanisms in the unknown domain setting, that is, we do not rely on knowing the domain universe and do not assume any structure of the data[1]. In this context, we only look at a small number of elements to perform differentially private top-k selection, while guaranteeing the privacy of the entire data. The unknown domain setting was only recently explored by [8], thus we differ substantially from most of the previous works, such as Exponential Mechanism [9] and Report Noisy Max [10], as they need some or complete knowledge of the data domain.

In general, our algorithms consider elements in a dataset with "importance" defined by their counts, and privately asks for k elements with top/largest counts, looking only at the top-$\bar{k}$ for any given $\bar{k} \geq k$. In this sense, we specifically work on the scenario where the output consists of a *set* of elements, i.e. we consider selecting elements without giving any specific order of importance between them. Additionally, we only consider the restricted subset of elements with the largest counts ($\bar{k}$), irrespective of how

[1] In contrast, itemsets have interesting properties that can be used to reduce the domain analyzed, e.g. see work from [7].

many elements exist in the domain and their counts. This is a huge advantage from the privacy point of view considering that elements with small counts are often the targets of a privacy breach. However, only looking at a subset of elements that depends on the data being analyzed may break privacy. For example, if the sorting that defines the $\bar{k}$ elements considered is highly dependent on a given element that is strongly related to a user, then applying a classical method like Exponential Mechanism may reveal some information about this user, as the presence of such user can considerably influence the results.

With this in mind, we propose a top-k selection method with *unordered* output subject to approximate differential privacy for unknown data domain, only needing access to the true top-$\bar{k}$ elements from the data for $\bar{k} \geq k$. Having observed that previous methods in the unknown domain [8] typically work well on skewed datasets, we work over properties of such input data by designing a mechanism that leverages stability based on the difference of counts of elements and is able to directly output a *set* of elements. Our method combines traditional differential privacy techniques in a novel way and compared to the most similar previous work that is suitable for releasing general online queries our construction gives more efficient results. For this reason our overall framework and privacy analysis are of independent interest.

More specifically, while previous approaches in the unknown domain [8] worked in a peeling manner, selecting elements one at a time, leading to composition of k iterations, we directly design our output as a *set* and privately test many possible outcomes using a classical technique that does not compose ε with the number of tests, which forces our privacy budget ε to not degrade as k increases. This property gives strong applicability to scenarios of large domains and top-k selections with very large k. Also, it is important to note that this fact does not contradict previous work on bounds for top-k selection, such as [11], as they work under weaker assumptions, and we apply our methods over stronger data-dependent properties to have successful private tests.

1.1 CONTRIBUTIONS

We propose a new differentially private top-k selection algorithm for the unknown domain setting, that only looks at the $\bar{k}$ elements with largest counts for any given $\bar{k} \geq k$. The approach combines and improves on important techniques from the literature, obtaining a privacy budget ε independent of k, enforced by rigorous formal guarantees.

An improved novel combination of SVT [12] and Stability [13] was designed to be integrated in our method, with a detailed privacy analysis deriving the parameters needed for optimal results. Our construction gives considerably better parameters than the most similar previous work from [14], being of independent interest, as it can be used for general queries, not just for top-k selection.

We compare our main algorithm to the current state-of-the-art on the unknown domain of private selection, the Limited Domain (LD) procedure [8], which is typically useful for selection on skewed datasets. Although LD gives an approximately ordered output, we emphasize that it is still the current state-of-the-art also for our *unordered* output scenario in the unknown domain. Thus we include a formal utility analysis with a discussion of favorable settings of our work in comparison to LD. Moreover, we perform an empirical evaluation on three real-world datasets and various settings. Since most works on DP top-k selection are solely theoretical – only a minority having experiments – we consider this a relevant contribution. The code for our methods, datasets used, and all experimental trials are made publicly available.

1.2 RELATED WORK

Our work focus is on the trusted curator model of differential privacy, where users trust a central data center, responsible for executing private algorithms for any untrusted analyst. Our private computation does not require looking at all of the existing elements in a dataset to privately select the top-k, therefore we differ from most of the previous works on top-k selection in this model. Many are specific to certain problems, like frequent itemset mining [7], or require more domain knowledge, such as the Exponential Mechanism [9] and Report Noisy Max [10]. Moreover, despite looking at only $\bar{k}$ elements, we still guarantee the privacy of the entire dataset.

Among the closely related works, [15] introduced the Large Margin Mechanism (LMM), which does not depend on the output domain range and seeks to return the element with a maximum count under approximate differential privacy. Nonetheless, LMM tests margin incrementally to choose a "safe" threshold and frequently this process ends up running over the entire domain universe. This reduces LMM to the exponential mechanism plus wasted additional computation efforts.

Recently, [8] proposed the Limited Domain (LD) procedure for differentially private top-k selection over a large domain universe. LD restricts the domain and only needs access to the true top-$\bar{k}$ elements from the data for any chosen $\bar{k} \geq k$. To avoid breaking privacy by only looking at a subset of elements, LD introduces a $\perp$ element that, whenever gets picked, stops the algorithm from running further. For top-k selection, LD will keep picking elements for at most k iterations or until $\perp$ is picked. To the best of our knowledge, LD is the only top-k differentially private algorithm for the unknown domain setting that never requires iterating over the entire domain.

However, LD uses a fixed probability formulation of $\perp$ that is $O(\log(\bar{k}/\delta))$, repeatedly considering worst-case scenarios over the iterations, which decreases the chance of selecting elements, in consequence diminishing output's utility. Moreover, the total privacy budget ε of LD depends on $O(\sqrt{k})$, which also considerably increases the probability of selecting $\perp$, consequently decreasing the utility of results.

Our work is based on the notion of distance to instability [13] combined with the Sparse Vector Technique (SVT) [12] for identifying the queries that lie above a certain threshold. In our top-k selection, we care about privately finding the indices of the top-i elements for the largest $i \leq k$. Since we do not know this maximum i, this problem can be cast as multiple queries to search for the desired i, to find the indices of top-i elements privately based on the stability of query results. However, with [13] only dealing with a single query and [12] returning only the indications of above/below the threshold for the queries (not the actual query results), it is non-trivial to combine these techniques due to non-straightforward interactions and new issues about privacy guarantees.

Previous work from [14] combined stability and SVT to release at most m query answers in an online fashion. Their algorithm $\mathcal{A}_{OQR}$ uses SVT to only spend privacy budget for the queries that are unstable, stopping after a number of unstable queries is reached, and answering potentially a large number of stable queries. Using such setting for our selection problem would be wasteful, as testing stability for several individual elements can lead to many unstable results, easily reaching the upper limit. Even using our insight that the stability can treat an entire set of elements as one input, their algorithm $\mathcal{A}_{OQR}$ is still not useful. That is because we only need one stable query result for an entire top-k selection and, more importantly, as we discuss in Section 5, $\mathcal{A}_{OQR}$ uses a simple noise construction with suboptimal results. Thus, we design a different improved method, better adjusting the noise distributions, and unbounding the number of unstable queries, to allow a limit of a single stable query with an entire selection. Different from $\mathcal{A}_{OQR}$, this leads to independence of k in the ε parameter of DP.

2 PRELIMINARIES

First we introduce essential concepts in differential privacy. After that we specifically highlight the two techniques, Stability and Sparse Vector techniques, that will be used as the main components in our solutions.

2.1 DIFFERENTIAL PRIVACY

Assume a dataset D with M elements and n users is defined as $D = \{x_1, ..., x_n\}$, with x_{ji} being the ith element of user j. Before defining differential privacy, we need to define the concept of neighboring datasets.

Definition 2.1. *Datasets D and D' are neighbors if they differ in the addition or removal of one user's data.*

Now we are ready to define differential privacy.

Definition 2.2 (Differential privacy, [3])**.** *A randomized mechanism $\mathcal{M}$ is (ε, δ)-differentially private if for neighbors D and D' and all outcome sets $\mathcal{O} \subseteq Range(\mathcal{M})$:*

$$\Pr[\mathcal{M}(D) \in \mathcal{O}] \leq exp(\varepsilon) \Pr[\mathcal{M}(D') \in \mathcal{O}] + \delta \quad (1)$$

Specifically for selection problems, we now show one of the most used algorithms available in the literature: the Exponential Mechanism.

Definition 2.3 (Exponential Mechanism, [9])**.** *On the Exponential Mechanism $EM(D, q, \mathcal{I})$, for all outputs $i \in \mathcal{I}$ and a given quality score function $q : \mathcal{U} \times \mathcal{I} \to \mathbb{R}$ we have:*

$$\Pr[EM(D, q, \mathcal{I}) = i] \propto exp(\frac{\varepsilon q(D, i)}{\Delta q}) \quad (2)$$

where $\Delta q = \sup_{i \in \mathcal{I}} |q(D, i) - q(D', i)|$ for all neighboring datasets $D, D' \in \mathcal{U}$.

Now we state the privacy guarantee obtained from using the Exponential Mechanism.

Lemma 2.4 (Privacy of Exponential Mechanism, [9])**.** *The exponential mechanism is $(2\varepsilon, 0)$-differentially private. Moreover, if the score function q is monotonic in the dataset D, $EM(D, q, \mathcal{I})$ is $(\varepsilon, 0)$-differentially private.*

Finally, we note that using the exponential mechanism k times, each time removing from the domain the previously selected elements, we get a result that is $(k\varepsilon, 0)$-differentially private, obtained by using simple privacy composition from [3].

2.2 STABILITY IN DIFFERENTIAL PRIVACY

Here we introduce the concept of stability in differential privacy, and give an algorithm from [13] that uses it for general stable queries. For this purpose we will consider a function $f : \mathcal{U} \to \mathcal{R}$ defined over the data universe $\mathcal{U}$ to a given finite range $\mathcal{R}$.

Definition 2.5 (Stability, [13])**.** *A function $f : \mathcal{U} \to \mathcal{R}$ is k-stable on input $D \in \mathcal{U}$ if adding or removing any k users' data from D does not change the value of f, that is, $f(D) = f(D')$ for all D' such that $|D \Delta D'| \leq k$. We* *say f is stable on D if it is (at least) 1-stable on D, and unstable otherwise. Also note that if f is k-stable, it is i-stable for any $1 \leq i \leq k$.*

In order to practically use stability in differentially private mechanisms we now define distance to instability.

Definition 2.6 (Distance to instability, [13]). *The distance to instability of a dataset $D \in \mathcal{U}$ with respect to a function f, denoted by $dist_f(D) : \mathcal{U} \to \mathbb{R}$, is the number of users that must be added to or removed from D to reach a dataset that is not stable, i.e. to reach a dataset D' that $f(D) \neq f(D')$. In other words, $dist_f(D) = \underset{k}{\operatorname{argmax}}[f(D)$ is k-stable$]$.*

Note that f is k-stable on D if and only if $dist_f(D) \geq k$.

Using the definitions above, for any function f, $\mathcal{A}_{stab}$ (Algorithm 1) is a differentially private mechanism that outputs $f(D)$ whenever D is sufficiently stable.

Algorithm 1 $\mathcal{A}_{stab}$ [13]: Private estimator for f via distance to instability

```
$\mathcal{A}_{stab}(D, f, dist_f, T, \varepsilon)$
    $\widehat{dist} \longleftarrow dist_f(D) + Lap(1/\varepsilon)$
    **If** $\widehat{dist} > T$, **then** Output $f(D)$  **Else** Output $\perp$
```

Finally we show the privacy guarantee of $\mathcal{A}_{stab}$ obtained using a specific threshold for the distance to stability.

Theorem 2.7 (Theorem 3.2 from [14]). *For threshold $T = \ln(1/\delta)/\varepsilon$, $\mathcal{A}_{stab}$ is (ε, δ)-differentially private.*

Note that if f is unstable on D, i.e., $dist_f(D) = 0$, Algorithm 1 will output $f(D)$ with probability $Pr[Lap(1/\varepsilon) > \ln(1/\delta)/\varepsilon]$, which is at most δ by the tail property of Laplace distribution. This is the key result related to the δ parameter of Theorem 2.7.

2.3 THE SPARSE VECTOR TECHNIQUE

Now we introduce another DP algorithm that we will later use: the Sparse Vector Technique (SVT), described on Algorithm 2 with privacy stated on Theorem 2.8. For SVT we consider a sequence of queries evaluated on a database D in comparison to a given threshold. The goal is to release a bit vector indicating, for each query, whether or not it lies above the threshold.

Theorem 2.8 (Theorem 2 from [12]). *SVT is ($\varepsilon_1 + \varepsilon_2$, 0)-differentially private.*

From Theorem 2.8 and algorithm's description, we see that the privacy of SVT degrades only with the number of queries which actually lie above the threshold, represented by the input c, rather than the total number of queries.

Algorithm 2 SVT [12]: Privately indicate if sensitivity-1 queries are above threshold

```
**SVT** ($D$, $\{q_1, ..., q_m\}$, $T$, $\varepsilon_1$, $\varepsilon_2$, c)
    Let $\hat{T} = T + \text{Lap}(\frac{1}{\varepsilon_1/c})$, ctr $= 0$
    **for** each query $q_i$ **do**
        Let $\nu_i = \text{Lap}(\frac{2}{\varepsilon_2/c})$
        **If** $q_i(D) + \nu_i \geq \hat{T}$, **then** Output $\top$; ctr = ctr + 1
        **Else** Output $\perp$
        **Abort** if $ctr \geq c$
    **end for**
```

3 PRIVATE TOP-K SELECTION

Assume a dataset D with M elements and n users is defined as $D = \{x_1, ..., x_n\}$ with neighbors differing in any one user's data. For $i \in \{1, 2, ..., M\}$, x_{ji} denotes the ith element of user x_j, which can be either 0 or 1. Let the function $c_i(D) = \sum_{j=1}^{n} x_{ji}$ be the sum of the ith element of all the users on dataset D and $c(D)$ the corresponding vector defined by $c_i(D)$.

Top-k selection. In general, to select the top-k elements in a dataset D means that we seek to privately select and output k elements $i_1, ..., i_k$, with $i_j \in \{1, 2, ..., M\}$, such that $c_{i_1}(D), ..., c_{i_k}(D)$ are as large as possible. Generally our mechanisms will receive as input only a histogram $h(D)$ with the $\bar{k}$ elements with largest counts, i.e. $h(D) = \{c_{i_1}(D), c_{i_2}(D), ..., c_{i_{\bar{k}}}(D)\} \in \mathbb{N}^{\bar{k}}$, such that $c_{i_1}(D) \geq c_{i_2}(D) \geq ... \geq c_{i_{\bar{k}}}(D)$ for $i_j \in \{1, 2, ..., M\}$. In this sense, we consider $\mathbf{I} : \mathbb{Z} \to \mathbb{Z}$ to be a mapping function, such that $\mathbf{I}(i)$ returns the original index of the element with the ith largest count $c_{\mathbf{I}(i)} = h_i$ on D. For example, $\mathbf{I}(1)$ is the index of the element with the largest count, with value $c_{\mathbf{I}(1)}(D)$, also represented as $h_1(D)$ on the sorted histogram.

To describe our mechanism, we start by connecting its output, which will be a set with indices representing a selection of elements, to the notion of stability on Definition 2.5. The idea is to privately use distance to instability, as in $\mathcal{A}_{stab}$, to return a set of stable elements. To be able to run $\mathcal{A}_{stab}$ multiple times until we find one stable result, we integrate multiple testings of distance to instability into the SVT, to only pay ε privacy budget once for the stable result, and not pay for the unstable testings. We will see that only the δ part, inside a log term, will compose when combining $\mathcal{A}_{stab}$ and SVT.

First we relate our output functions with the notion of stability from Definition 2.5. We define $f_j : \mathcal{U} \to \mathcal{R}$ as a function that returns the set of *indices* of the j elements with largest counts on a given dataset $D \in \mathcal{U}$. In this sense, the result is unordered, with $\mathcal{R}$ being the set of all possible sets of elements containing zero or more elements among the j elements with largest counts.

Now, please note that for a given $j < M$ if we have $h_j(D) - h_{j+1}(D) - 1 > 0$ then the set of elements on position $1 \leq i \leq j$ of the sorted histogram is the same on D and any neighbour D' of D differing in one user's data. The reason being that any user can change any given element's count by only a value of one.

Therefore, if $f_j(D)$ returns a set $S \in \mathcal{R}$ of elements on positions 1 to j of the sorted histogram such that $h_j(D) - h_{j+1}(D) - 1 > 0$, then this means $f_j(D)$ is 1-stable. Following this idea, we get that such $f_j(D)$ has the value $h_j(D) - h_{j+1}(D) - 1$ as the distance to instability, according to Definition 2.6. Therefore, we can define a query $q_j : \mathcal{U} \to \mathbb{R}$ on a dataset $D \in \mathcal{U}$ to be the distance to instability of the function $f_j(D)$, calculated by $h_j(D) - h_{j+1}(D) - 1$.

With this in mind, for a given index j and histogram $h(D)$, we can test the distance to instability of $f_j(D)$, represented as the query $q_j(D)$, using Algorithm 1. Since this is a private test, we do not know the position j with a large $h_j(D) - h_{j+1}(D)$ that would lead to increased chance of success. So we want to do multiple tests on multiple positions, and once we find a stable result we can stop. This scenario fits perfectly the use of the SVT on Algorithm 2 with $c = 1$. Therefore, using the combined algorithms mentioned above, we start testing the distance to instability on position $\bar{k} \geq k$ and keep decreasing/testing $\bar{k}$ until we have one success.

Moreover if we have a stable result on position $j > k$ we can either use Top-k EM to get k elements out of the j (paying ε_{EM}) or return a random selection of k elements without paying additional budget. Otherwise if the stable result is on position $j \leq k$, we just return the set of elements from position 1 to j on the sorted histogram, without any particular ordering.

Our method for privately computing the top-k elements, **TS**, is fully described in Algorithm 3. Although **TS** works for any $p_1 \neq 1/3$, we will usually set it to 0.37, giving 37% of ε to ε_1 and 63% to ε_2. We give more budget to ε_2 because increasing it also increases the chances of a success by decreasing T. The consequence in turn is increasing the noise added to T and decreasing the noise added to each query q_i. Since noise varies and a large threshold would be very hard to be above, focusing on improving the fixed T instead of the noise tends to improve results. Moreover the exact percentages we use come from the guidelines of [12], which gives an optimal privacy budget allocation between ε_1 and ε_2.

Also note **TS** was designed to be as flexible as possible, dealing with $\bar{k} \geq k$ and adding a call to EM if desired. Still we emphasize that the main ε is independent of k and highlight that the most common use of **TS** may be setting $\varepsilon_{EM} = 0$ and possibly $\bar{k} = k$. Finally, note we operate on the **unknown** domain, like [8], as we only look at the $\bar{k} + 1$ elements with largest counts. Before we show the privacy guarantee of **TS**, we give a supporting lemma.

Algorithm 3 TS: Top Stable algorithm for top-k selection via distance to instability

```
TS($D, k, \bar{k}, \varepsilon, p_1, \varepsilon_{EM}, \delta$)
 1: Let $\varepsilon_1 = p_1\varepsilon$, $\varepsilon_2 = (1-p_1)\varepsilon$ and $c = 2\varepsilon_1/\varepsilon_2$
 2: For $\delta_{max} = \left(\frac{2\delta_q^c + \delta_q - c(\delta_q^c + 2\delta_q)}{4(1-c)}\right)$, let $\delta_q$ be the
    $\underset{\delta_q}{\text{argmax}}\left[\delta_{max}\right]$ subject to $\delta_{max} \leq \delta/\bar{k}$
 3: Let $T = \ln(1/\delta_q)/(\varepsilon_2/2)$ and $\hat{T} = T + \text{Lap}(1/\varepsilon_1)$
 4: for $i$ in [$\bar{k}$, $\bar{k} - 1$, ..., 2, 1] do
 5:     Let $f_i(D)$ be the set of original indices of the
        elements from position 1 to $i$ on the sorted histogram
 6:     Let $q_i(D) = h_i(D) - h_{i+1}(D) - 1$
 7:     out = $\mathcal{A}_{stab}(D, f_i, q_i, \hat{T}, \frac{\varepsilon_2}{2})$
 8:     If out = $\perp$, then Output $\perp$
 9:     Else
10:         If $i > k$, then Output result of Top-$k$ EM
            on $f_i(D)$ with privacy budget $\varepsilon_{EM}$
11:         Else Output $f_i(D)$ in a random order
12:         Halt # Ends execution after first success
13: end for
```

Lemma 3.1. *For given parameters, ε_a, ε_b, δ, and $c \neq 1$ where $c = \varepsilon_b/\varepsilon_a$, $\Pr[Lap(\frac{1}{\varepsilon_a}) > log(1/\delta)/\varepsilon_a + Lap(\frac{1}{\varepsilon_b})]$ is at most $\frac{2\delta^c + \delta - c(\delta^c + 2\delta)}{4(1-c)}$.*

Proof. First, for simplicity we denote $\delta_T = (\delta)^{\varepsilon_b/\varepsilon_a} = \delta^c$ and $T = log(1/\delta)/\varepsilon_a = log(1/\delta) \cdot \frac{\varepsilon_b}{\varepsilon_a}/\varepsilon_b = log(1/(\delta)^{\varepsilon_b/\varepsilon_a})/\varepsilon_b = log(1/\delta_T)/\varepsilon_b$. This way, using simple tail properties of the Laplace distribution, where $X \simeq Lap(b)$ and $t > 0$ gives $\Pr[X < -tb] = \exp(-t)/2$, we have:

$$\Pr[Lap(1/\varepsilon_b) < -T] \leq \frac{\delta_T}{2} \tag{3}$$

Now the probability we want to calculate becomes:

$\Pr[Lap(1/\varepsilon_a) > log(1/\delta)/\varepsilon_a + Lap(1/\varepsilon_b)] =$
$\Pr[Lap(1/\varepsilon_a) > T + Lap(1/\varepsilon_b)] =$
$\Pr[Lap(1/\varepsilon_b) < Lap(1/\varepsilon_a) - T]$

Now we rewrite the above as the convolution of two Laplace random variables. We will denote the density of a $Lap(1/\varepsilon_a)$ random variable as $p_a(\cdot)$. We also separate the same integral expression into three different intervals to simplify the proofs.

$$\int_{-\infty}^{\infty} p_a(x) \cdot \Pr[Lap(1/\varepsilon_b) < x - T]dx =$$

$$\int_{-\infty}^{0} p_a(x) \cdot \Pr[Lap(1/\varepsilon_b) < x - T]dx + \quad (4)$$

$$\int_{0}^{T} p_a(x) \cdot \Pr[Lap(1/\varepsilon_b) < x - T]dx + \quad (5)$$

$$\int_{T}^{\infty} p_a(x) \cdot \Pr[Lap(1/\varepsilon_b) < x - T]dx \quad (6)$$

Now we bound each of the terms 4, 5, and 6 separately.

For the term 4, we apply Equation 3 above and then use $\int_{-\infty}^{0} p_a(x) = 1/2$ for the Laplace distribution:

$\int_{-\infty}^{0} p_a(x) \cdot \Pr[Lap(1/\varepsilon_b) < x - T]dx \leq$
$\frac{\delta_T}{2} \int_{-\infty}^{0} p_a(x)dx = \frac{\delta_T}{4}$

Similarly we obtain the bound for the term 6 as:

$\int_{T}^{\infty} p_a(x) \cdot \Pr[Lap(1/\varepsilon_b) < x - T]dx \leq$
$\int_{T}^{\infty} p_a(x)dx = \exp(-\varepsilon_a T)/2 = \frac{\delta}{2}$

Finally for the term 5, we start by changing variable to $y = x - T$, having $y \leq 0$ for $x \in [0, T]$. Looking only at the probability part of the second term we then get:

$\Pr[Lap(1/\varepsilon_b) < x - T] = \int_{-\infty}^{x-T} p(y)dy = \frac{1}{2}e^{(x-T)\varepsilon_b}$

Substituting this result into the term 5 gives:

$\int_{0}^{T} p_a(x) \cdot \Pr[Lap(1/\varepsilon_b) < x - T]dx =$
$\int_{0}^{T} p_a(x)\frac{1}{2}e^{(x-T)\varepsilon_b}dx$

Now we write p_a as the PDF of the Laplace distribution without the absolute value since $x \in [0, T]$, for $\frac{\varepsilon_b}{\varepsilon_a} \neq 1$:

$\int_{0}^{T} p_a(x) \cdot \Pr[Lap(1/\varepsilon_b) < x - T]dx =$
$\int_{0}^{T} \frac{\varepsilon_a}{2}e^{-x\varepsilon_a}\frac{1}{2}e^{(x-T)\varepsilon_b}dx = \frac{\varepsilon_a}{4}\left(\frac{e^{-\varepsilon_b T} - e^{-\varepsilon_a T}}{\varepsilon_a - \varepsilon_b}\right)$

Noting that $e^{-\varepsilon_b T} = \delta_T$ and $e^{-\varepsilon_a T} = \delta$, above becomes:

$\int_{0}^{T} p_a(x) \cdot \Pr[Lap(1/\varepsilon_b) < x - T]dx = \frac{\varepsilon_a}{4}\left(\frac{\delta_T - \delta}{\varepsilon_a - \varepsilon_b}\right)$

Putting together all three terms with $\delta_T = \delta^c$ gives:

$\int_{-\infty}^{\infty} p_a(x) \cdot \Pr[Lap(1/\varepsilon_b) < x - T]dx \leq$
$\frac{\delta_T}{4} + \frac{\delta}{2} + \frac{\varepsilon_a}{4}\left(\frac{\delta_T - \delta}{\varepsilon_a - \varepsilon_b}\right) = \frac{\delta^c}{4} \cdot \left(\frac{2\varepsilon_a - \varepsilon_b}{\varepsilon_a - \varepsilon_b}\right) + \frac{\delta}{4} \cdot \left(\frac{\varepsilon_a - 2\varepsilon_b}{\varepsilon_a - \varepsilon_b}\right)$

Using $\varepsilon_b = c \cdot \varepsilon_a$, with $c \neq 1$ as we used $\frac{\varepsilon_b}{\varepsilon_a} \neq 1$, and rearranging terms gives the result of this lemma. □

Theorem 3.2. **TS** *on Algorithm 3 is* ($\varepsilon + \varepsilon_{EM}$, δ)*-DP.*

Proof. First, consider the group of queries is defined as $\mathcal{Q}$, then each $q_i \in \mathcal{Q}$ on Line 6 calculates the distance to instability of the corresponding function f_i. Now, for simplicity, we split our algorithm into three phases: First, for every $q_i \in \mathcal{Q}$, **TS** either commits to $\top$ or outputs $\bot$ based on the input dataset D. Next if it commits to $\top$, then it can use $f_i(D)$. Finally, with $f_i(D)$ we can either apply Top-k EM or get a random sample of elements.

Thus we can consider three algorithms $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$, where $\mathcal{A}_1$ outputs a sequence of $\bot$ and $\top$, corresponding to the first phase above, $\mathcal{A}_2$ is invoked to output $f_i(D)$ only for the queries $q_i(D)$ that $\mathcal{A}_1$ outputs $\top$, and $\mathcal{A}_3$ is only invoked to output a selection of elements from $f_i(D)$ also for the queries $q_i(D)$ that $\mathcal{A}_1$ outputs $\top$. Notice that the combination of $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ is equivalent to **TS**.

First, since $\mathcal{A}_1$ is just executing the SVT from Algorithm 2, especially with $c = 1$, and we use $\varepsilon_1 + \varepsilon_2 = \varepsilon$, by Theorem 2.8, it satisfies (ε, 0)-differential privacy. Next we analyze the privacy of $\mathcal{A}_2$ based on the output of $\mathcal{A}_1$ with the use of $\mathcal{A}_{stab}$ (Algorithm 1).

Consider any particular query $q_i \in \mathcal{Q}$. For any dataset D' s.t. $|D \Delta D'| = 1$, there are two possibilities: either $f_i(D) = f_i(D')$, or $f_i(D) \neq f_i(D')$.

When $f_i(D) = f_i(D')$, if $\mathcal{A}_1$ outputs $\bot$, algorithm $\mathcal{A}_2$ is not invoked and hence the privacy guarantee is not affected. Moreover, if $\mathcal{A}_1$ outputs $\top$, then result is already differentially private, meaning $\Pr[\mathcal{A}_2(D, f_i) = s] = \Pr[\mathcal{A}_2(D', f_i) = s]$ for any output set $s \in \mathcal{R}$, i.e. privacy budget is not affected.

Now when $f_i(D) \neq f_i(D')$, it follows that $f_i(D)$ and $f_i(D')$ are unstable, i.e. $dist_{f_i}(D) = dist_{f_i}(D') = 0$, implying $\Pr[\mathcal{A}_{stab}(D, f_i) = \bot] = \Pr[\mathcal{A}_{stab}(D', f_i) = \bot]$ which only outputs $\top$ with $\Pr[Lap(1/(\varepsilon_2/2)) > log(1/\delta_q)/(\varepsilon_2/2) + Lap(1/\varepsilon_1)]$. Using Lemma 3.1 with $\varepsilon_a = \varepsilon_2/2$, $\varepsilon_b = \varepsilon_1$ and $\delta = \delta_q$, we have that the probability above is at most $\delta_{max} = \frac{2\delta_q^c + \delta_q - c(\delta_q^c + 2\delta_q)}{4(1-c)}$ for $c = 2\varepsilon_1/\varepsilon_2$.

Therefore, for our parameters, we conclude that $\mathcal{A}_2$ is only invoked to output $f_i(D)$ with probability at most δ_i. Since we do not know how many queries correspond to $f_i(D) \neq f_i(D')$, we bound this quantity by their total, which is $\bar{k}$ since we have $\bar{k}$ queries. Therefore, by simple composition, $\mathcal{A}_2$ is only invoked to output $f_i(D)$ with probability at most $\delta = \delta_{max} \cdot \bar{k}$. Since on Line 2 of Algorithm 3 we get the maximum δ_q that satisfies $\delta_{max} \cdot \bar{k} \leq \delta$, this implies it satisfies (0, δ)-differential privacy.

Finally, $\mathcal{A}_3$ is just simple use of Top-k EM, which adds up (ε_{EM}, 0)-differential privacy to the overall mechanism. This results in **TS** being ($\varepsilon + \varepsilon_{EM}$, δ)-DP. □

4 UTILITY ANALYSIS

In this section, we give a general utility bound on outputting k elements, similar to previous work on the unknown domain. Additionally, we compare our utility with the current state-of-the-art on a specific setting where we obtain guarantees of improved results.

Theorem 4.1. *With $3/2 < c < 2$ and assuming a $0 < \delta_q < 1$ that gives $\delta_{max} = \delta/\bar{k}$, $\mathbf{TS}$ outputs k elements with probability at least $1-\beta$, if for any given i such that $k \leq i \leq \bar{k}$ we have $h_i - h_{i+1} \geq 1 + (log(\bar{k}/\delta) + log(1/\beta))/(\varepsilon/4)$.*

Proof. Refer to $\mathbf{TS}$ for notations. We first claim $\delta/\bar{k} \leq \delta_q$. For $\delta_{max} = \delta/\bar{k}$, from Line 2 of Algorithm 3, we have $\delta/\bar{k} = \frac{2\delta_q^c+\delta_q-c(\delta_q^c+2\delta_q)}{4(1-c)} = \frac{c(\delta_q^c+2\delta_q)-2\delta_q^c-\delta_q}{4(c-1)} = \frac{(c-2)\delta_q^c+(2c-1)\delta_q}{4(c-1)}$. With $1.5 < c < 2$ we have $(c-2) < 0$ and $\frac{2c-1}{4(c-1)} < 1$, therefore: $\delta/\bar{k} = \frac{(c-2)\delta_q^c+(2c-1)\delta_q}{4(c-1)} < \frac{(2c-1)\delta_q}{4(c-1)} < \delta_q$. This proves the claim.

Given that we have $\delta_q > \delta/\bar{k}$ or $1/\delta_q < \bar{k}/\delta$, and $c < 2$ implies $\varepsilon_2/2 > \varepsilon/4$, during the proof we will use $h_i - h_{i+1} \geq 1 + (log(1/\delta_q) + log(1/\beta)/(\varepsilon_2/2)$, which is smaller than $1 + (log(\bar{k}/\delta) + log(1/\beta)/(\varepsilon/4)$. Therefore if $\mathbf{TS}$ outputs k elements for the former requirement it also outputs for the latter. Also for simplicity we write $q_i = h_i - h_{i+1} - 1$.

The probability that $\mathbf{TS}$ outputs k elements comes from the stability test, i.e. $\Pr[q_i + Lap(1/(\varepsilon_2/2)) > log(1/\delta_q)/(\varepsilon_2/2) + Lap(1/\varepsilon_1)]$. Then here with $q_i \geq (log(1/\delta_q) + log(1/\beta)/(\varepsilon_2/2)$, the probability can be rewritten as:

$\Pr[q_i + Lap(\frac{1}{\varepsilon_2/2}) > log(1/\delta_q)/(\varepsilon_2/2) + Lap(\frac{1}{\varepsilon_1})] \geq$
$\Pr[(log(1/\delta_q) + log(1/\beta)/(\varepsilon_2/2) + Lap(\frac{1}{\varepsilon_2/2}) > log(1/\delta_q)/(\varepsilon_2/2) + Lap(\frac{1}{\varepsilon_1})] =$
$\Pr[(log(1/\beta)/(\varepsilon_2/2) + Lap(\frac{1}{\varepsilon_2/2}) > Lap(\frac{1}{\varepsilon_1})] =$
$1 - \Pr[log(1/\beta)/(\varepsilon_2/2) + Lap(\frac{1}{\varepsilon_2/2}) < Lap(\frac{1}{\varepsilon_1})] =$
$1 - \Pr[Lap(\frac{1}{\varepsilon_1}) > log(1/\beta)/(\varepsilon_2/2) + Lap(\frac{1}{\varepsilon_2/2})] =$
$1 - \Pr[Lap(\frac{1}{\varepsilon_2/2}) > log(1/\beta)/(\varepsilon_2/2) + Lap(\frac{1}{\varepsilon_1})]$

where the last step is due to the symmetry of the Laplace distribution. Using Lemma 3.1 with $\varepsilon_a = \varepsilon_2/2$, $\varepsilon_b = \varepsilon_1$ and $\delta = \beta$, we have that the probability above $\Pr[Lap(\frac{1}{\varepsilon_2/2}) > log(1/\beta)/(\varepsilon_2/2) + Lap(\frac{1}{\varepsilon_1})]$ is at most $\frac{2(\beta)^c+\beta-c((\beta)^c+2(\beta))}{4(1-c)}$ for $c = 2\varepsilon_1/\varepsilon_2$. With $1.5 < c < 2$ we have $(c-2) < 0$ and $\frac{2c-1}{4(c-1)} < 1$, thus we simplify:

$\Pr[Lap(\frac{1}{\varepsilon_2/2}) > log(1/\beta)/(\varepsilon_2/2) + Lap(\frac{1}{\varepsilon_1})] \leq \frac{2(\beta)^c+\beta-c((\beta)^c+2(\beta))}{4(1-c)} = \frac{(c-2)(\beta)^c+(2c-1)(\beta)}{4(c-1)} \leq \frac{(2c-1)(\beta)}{4(c-1)} \leq \beta.$

Thus, probability of satisfying this lemma is $1-\beta$. $\square$

A similar utility guarantee was given by [8] for their Limited Domain (LD) algorithm also on the unknown domain, which we compare to now.

Theorem 4.2. *For $\bar{k} = k \geq 2$, the minimum gap $h_k - h_{k+1}$ needed to output k elements with probability at least $1-\beta$ is smaller for $\mathbf{TS}$ when compared to LD if $4 \cdot log(k/(\delta \cdot \beta)) < \sqrt{k/2} \cdot log(k^2/(\delta \cdot \beta))$.*

Proof. Lemma 8.1 on [8] states that LD returns k elements with probability $1-\beta$ if $h_k - h_{\bar{k}+1} \geq 1 + (log(\bar{k}/\delta) + log(k/\beta))/\varepsilon_{iter}$, where ε_{iter} is the budget per iteration. Using $\bar{k} = k$ for LD and $i = k$ for $\mathbf{TS}$ according to Theorem 4.1, gives a distance requirement for $\mathbf{TS}$: $h_k - h_{k+1} \geq 1 + 4 \cdot log(k/(\delta \cdot \beta))/\varepsilon$ and for LD: $h_k - h_{k+1} \geq 1 + log(k^2/(\delta \cdot \beta))/\varepsilon_{iter}$.

Since the ε_{iter} for LD is the budget per iteration, we need to consider the composition of k iterations. From LD's privacy guarantee, we have that $\varepsilon \geq \varepsilon_{iter} \cdot \sqrt{k/2}$, which gives for LD: $h_k - h_{k+1} \geq 1 + \sqrt{k/2} \cdot log(k^2/(\delta \cdot \beta))/\varepsilon$. This shows better results for $\mathbf{TS}$ whenever
$4 \cdot log(k/(\delta \cdot \beta)) < \sqrt{k/2} \cdot log(k^2/(\delta \cdot \beta))$. $\square$

Thus, from Theorem 4.2 it is easy to see, for example, that $\mathbf{TS}$ outperforms LD for $k > 32$. In theory there is always a constant k_c that for any $k > k_c$ we have $\mathbf{TS}$ outperforming LD, but in practice that depends heavily on the dataset and can be a lot larger than 32. One of the reasons is that on Theorem 4.2 we used $\bar{k} = k$, even though the utility statement on LD is for the distance between the kth largest elements and the $(\bar{k}+1)$th largest element. This forces algorithms to only return elements among the true top-k but is a requirement more related to $\mathbf{TS}$ than LD. Additionally, the utility bound for LD can be considered very loose, and it can give better results than $\mathbf{TS}$ for some values of $k > 32$. Additionally, Theorem 4.2 is not restrictive for all values of k, which means that for small values of k $\mathbf{TS}$ can still outperform LD depending on the value of the gap $h_k - h_{k+1}$.

5 PRIVATE GENERAL QUERIES

As stated before, $\mathcal{A}_{OQR}$ of [14] also uses a combination of stability and SVT. However we emphasize that our construction is essentially very different and with better use of privacy budget. For this reason here we discuss the main differences and give a general version of both $\mathcal{A}_{OQR}$ and $\mathbf{TS}$, to be used with any kind of query, even on online setting, not restricted to top-k selection.

First, see that $\mathcal{A}_{OQR}$ was designed to release at most m query answers, bounding the number of unstable queries and releasing up to m stable. On the other hand $\mathbf{TS}$ bounds the stable queries to release one stable and discard all the unstable it finds. As explained before, already by design $\mathcal{A}_{OQR}$ is improper for our top-k selection.

The most essential difference appears when we see that $\mathcal{A}_{OQR}$ has to consider the absolute value on the noise added to the threshold in order to guarantee differential privacy, which considerably decreases its utility by only increasing the noisy threshold. Keeping the noise added to the threshold without the absolute value does not guarantee the δ part of DP, as their Lemma 3.4 does not hold anymore. More discussion on the Supplementary File.

To improve the noise construction, we added a novel privacy analysis on our Theorem 3.2 using Lemma 3.1. The result tailors our parameters to guarantee DP, without needing to use the absolute value on the noise of the threshold. Such privacy analysis is of independent interest, and we can also apply it to $\mathcal{A}_{OQR}$ for general online queries in their application setting. Therefore, for completion we give a general improved version of $\mathcal{A}_{OQR}$ and also a generic version of $\mathbf{TS}$, to be used with any kind of query. Due to space constraints we defer the pseudocode to the Supplementary Material on Appendix A.

6 EXPERIMENTS

For the experimental evaluation, we directly compare our proposed method, $\mathbf{TS}$, with the Limited Domain (LD) procedure from [8] on three different public datasets. Complete code is publicly available[2].

Datasets Our datasets are location-based check-ins, Gowalla[16], BrightKite[16] and Foursquare[17]. We process the data such that each user gives at most one single count value of 1 when he has visited a certain location, even if the user visits the location many times. Our goal is to select top-k most visited locations. Table 1 shows dataset details.

Table 1: Overview of datasets. Number of users, elements, elements of count 1 and 50/99/100 (max) perc. of counts.

Dataset	Users	Elements	Perc. of 1	Percentile 50th	99th	100th
Gowalla	107,092	1,280,968	49%	2	24	2931
BrightKite	51,406	772,966	87%	1	7	3204
Foursquare	2,293	100,191	60%	1	28	1274

Comparison Metrics For evaluation, we consider two comparison metrics. The first is $\mathcal{P}$, for *Proportion of true top-k*, defined as the number of true top-k elements returned divided by k. The other is $\mathcal{S}$, for *Relative Sum*, as the sum of the counts of the elements selected divided by the sum of the counts of the true top-k elements. Both $\mathcal{P}$ and $\mathcal{S}$ are in the range of $[0, 1]$ with the value 1 representing the true top-k elements being returned. A larger value means a closer approximation to the true top-k elements.

[2]https://github.com/ricardocarvalhods/diff-priv-top-k-stability

Settings For each given number of users n, showed on Table 1, we use the privacy parameters $\delta = 1/n$ and try three different values of ε: 0.4 (small), 0.8 (medium) and 1.0 (large). A value of $\varepsilon < 0.5$ may be considered too small, but we note that this is the most common setting in a private top-k selection, as this task is often one step of a bigger analysis, thus needing to work with reduced ε.

Initially, we show the results for k equal to 3 (small), 10 (medium) and 50 (large). $\mathbf{TS}$ uses $p_1 = 0.37$, as discussed by [12]. Moreover, we note that the comparison between $\mathbf{TS}$ and LD is multifaceted, without a global best, therefore, even though $\mathbf{TS}$ allows a broad choice of parameters, we opt to simplify and fix $\varepsilon_{EM} = 0$ and $\bar{k} = k$ for a scenario of *unordered* output. Note that although LD gives ordered output, it still is the best option to compare with for unknown domain even without needing order. For LD we compose $k + 1$ iterations, and consider the extra iteration for selecting $\bar{k}$ from $[k, ..., 5k]$ using their optimization method. On each individual setting we run $2,000$ trials. After overall results we also add discussion on settings where $\mathbf{TS}$ outperforms LD and otherwise.

6.1 OVERALL RESULTS

Here we show the results for the settings described, where on Table 2 we compare $\mathbf{TS}$ and LD regarding $\mathcal{P}$, and on Table 3 with respect to $\mathcal{S}$.

Table 2: Comparison of TS and LD using $\mathcal{P}$.

		Gowalla			BrightKite			Foursquare		
ε	Alg.	k: 3	10	50	k: 3	10	50	k: 3	10	50
0.4	TS	**0.98**	**1.00**	0.20	**1.00**	**0.77**	**0.14**	**1.00**	**0.64**	0.11
	LD	0.79	0.76	**0.24**	**1.00**	0.47	0.10	0.67	**0.64**	**0.13**
0.8	TS	**1.00**	**1.00**	**0.40**	**1.00**	**0.80**	0.16	**1.00**	**0.90**	0.18
	LD	**1.00**	0.90	0.38	**1.00**	0.79	**0.25**	0.72	0.85	**0.23**
1	TS	**1.00**	**1.00**	**0.45**	**1.00**	0.80	0.18	**1.00**	0.90	0.18
	LD	**1.00**	0.90	**0.45**	**1.00**	**0.88**	**0.28**	0.86	**0.98**	**0.25**

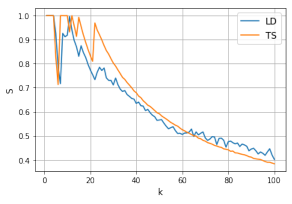
Table 3: Comparison of TS and LD using $\mathcal{S}$.

		Gowalla			BrightKite			Foursquare		
ε	Alg.	k: 3	10	50	k: 3	10	50	k: 3	10	50
0.4	TS	**0.98**	**1.00**	0.39	**1.00**	**0.86**	**0.38**	**1.00**	0.75	0.28
	LD	0.81	0.80	**0.44**	**1.00**	0.67	0.32	0.70	**0.78**	**0.32**
0.8	TS	**1.00**	**1.00**	**0.59**	**1.00**	**0.89**	0.42	**1.00**	**0.94**	0.39
	LD	**1.00**	0.92	0.56	**1.00**	0.88	**0.51**	0.75	0.91	**0.46**
1.0	TS	**1.00**	**1.00**	**0.63**	**1.00**	0.89	0.44	**1.00**	0.94	0.40
	LD	**1.00**	0.92	**0.63**	**1.00**	**0.94**	**0.56**	0.87	**0.97**	**0.50**

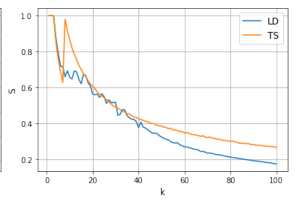
Overall, for the specific settings we used, we see $\mathbf{TS}$ with better utility more times than LD. However, that is no assurance of consistently better results. To further investigate how the two methods compare we dive into some of the settings evaluated, to detail our improvements.

6.2 DISCUSSION AND ADDITIONAL RESULTS

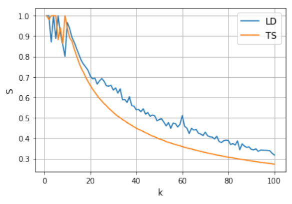
Generally, as the utility of most mechanisms in DP can considerably vary for different ε, k and data distribution, for a thorough evaluation, we detail some of the settings used on this section for $1 \leq k \leq 100$, describing the overall behaviour in cases where our method, compared to LD, is consistently better, consistently worse, or performs better on a given range of k. We also add results and discussion for settings with very large k.



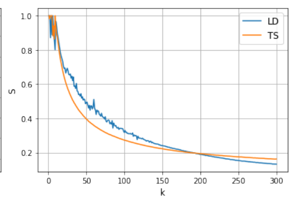
(a) Stable range



(b) Consistently better



(c) Consistently worse



(d) Very large k

Figure 1: Additional in depth results

Stable range Since our method relies on stability, which may only happen in certain regions of a histogram, **TS** often shows behaviour of improved utility when k is below some upper limit. As an example, for the setting where we used $\varepsilon = 0.8$, the Gowalla dataset shows improved results of $\mathcal{S}$ when $k \leq 60$, as we can observe on Figure 1a.

More specifically, if we define the "stable range" interval as $1 \leq k \leq 60$ and "bad range" for $k > 60$, on average we get the metrics $\mathcal{P}$ and $\mathcal{S}$ as showed on Table 4. We see, for example, that on the "stable range" **TS** shows an improvement of 7% on $\mathcal{P}$ compared to LD.

Consistently better Different settings can also result in **TS** giving consistently better results than LD, almost independent of k. For example, on the setting where we used $\varepsilon = 0.4$, BrightKite dataset shows improved results almost always, i.e. for almost all values of k, as we can observe on Figure 1b, which plots $\mathcal{S}$ for $1 \leq k \leq 100$.

More specifically, only on a few cases **TS** is worse than

Table 4: Average $\mathcal{P}$ and $\mathcal{S}$ on stable or bad ranges.

Algorithm	Stable Range		Bad Range	
	$\mathcal{P}$	$\mathcal{S}$	$\mathcal{P}$	$\mathcal{S}$
TS	**0.67**	**0.78**	0.24	0.44
LD	0.60	0.73	**0.27**	**0.47**

LD, e.g. for $k = 30$, **TS** gives $\mathcal{S}$ of 0.49 and LD of 0.52. On the other hand there are cases of great improvements, e.g. for $k = 8$, **TS** gives $\mathcal{S}$ of 0.98 and LD of 0.66.

Consistently worse Finally, we cannot affirm **TS** always outperforms LD, thus we show settings where **TS** gets consistently worse results. For example, on the setting where we used $\varepsilon = 1.0$, the Foursquare dataset shows LD outperforming **TS** for almost all values of k. This can observed on Figure 1c, where we plot $\mathcal{S}$ for $1 \leq k \leq 100$.

Looking in more depth, we have only on a few cases where **TS** becomes better than LD, e.g. for $k = 9$ we have that **TS** gives $\mathcal{S}$ of 1.00 and LD of 0.80. Considerably more frequent are the cases that LD is better, e.g. for $k = 32$, LD gives $\mathcal{S}$ of 0.66 while **TS** shows 0.51.

Very large k It is important to notice that, as stated on the utility analysis, there is always a constant k_c such that for any $k \geq k_c$ we have **TS** outperforming LD. We can intuitively understand this fact from the property that the ε of **TS** does not degrade with k. With this in mind, we show as an example in Figure 1d the result of the case above where **TS** was consistently worse than LD. In fact, when we reach $k \geq 200$ we have **TS** outperforming LD.

7 CONCLUSION

We have proposed a new method for selecting the top-k elements on unknown data domain with *unordered* output, only looking at the $\bar{k}$ elements with the largest counts. Unlike previous work, our proposed method's privacy budget ε does not depend on k. We have designed a novel construction combining the Sparse Vector Technique and Stability, with rigorous formal guarantees, giving a framework that can be applied to general queries and is of independent interest. We have provided theoretical guarantees and utility analysis, in addition to extensive empirical evidence on multiple real-life datasets, showing the specific settings that our proposed method outperforms the current state-of-the-art on the unknown domain.

ACKNOWLEDGEMENTS

This research is in part supported by a discovery grant and a CGS-D award from Natural Sciences and Engineering Research Council of Canada, and in part supported by NRF Singapore (NRFI05-2019-0002).

References

[1] M. Kantarcioğlu, J. Jin, and C. Clifton, "When do data mining results violate privacy?," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 599–604, ACM, 2004.

[2] C. Dwork and M. Naor, "On the difficulties of disclosure prevention in statistical databases or the case for differential privacy," *Journal of Privacy and Confidentiality*, vol. 2, no. 1, 2010.

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

[4] 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*, pp. 1054–1067, ACM, 2014.

[5] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *Advances in Neural Information Processing Systems*, pp. 3571–3580, 2017.

[6] K. Kenthapadi and T. T. Tran, "Pripearl: A framework for privacy-preserving analytics and reporting at linkedin," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 2183–2191, ACM, 2018.

[7] J. Lee and C. W. Clifton, "Top-k frequent itemsets via differentially private fp-trees," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 931–940, ACM, 2014.

[8] D. Durfee and R. M. Rogers, "Practical differentially private top-k selection with pay-what-you-get composition," in *NeurIPS*, pp. 3527–3537, 2019.

[9] F. McSherry and K. Talwar, "Mechanism design via differential privacy.," in *FOCS*, vol. 7, pp. 94–103, 2007.

[10] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, 2014.

[11] M. Bafna and J. Ullman, "The price of selection in differential privacy," in *Proceedings of the 2017 Conference on Learning Theory* (S. Kale and O. Shamir, eds.), vol. 65 of *Proceedings of Machine Learning Research*, (Amsterdam, Netherlands), pp. 151–168, PMLR, 07–10 Jul 2017.

[12] M. Lyu, D. Su, and N. Li, "Understanding the sparse vector technique for differential privacy," *Proceedings of the VLDB Endowment*, vol. 10, no. 6, pp. 637–648, 2017.

[13] A. G. Thakurta and A. Smith, "Differentially private feature selection via stability arguments, and the robustness of the lasso," in *CLT*, pp. 819–850, 2013.

[14] R. Bassily, O. Thakkar, and A. G. Thakurta, "Model-agnostic private learning," in *Advances in Neural Information Processing Systems*, pp. 7102–7112, 2018.

[15] K. Chaudhuri, D. J. Hsu, and S. Song, "The large margin mechanism for differentially private maximization," in *Advances in Neural Information Processing Systems*, pp. 1287–1295, 2014.

[16] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1082–1090, 2011.

[17] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu, "Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 129–142, 2014.

A Supplementary Material

As discussed on Section 5, here we detail general algorithms for privately releasing queries via stability, i.e., mechanisms that work for any type of function that we can calculate the distance to instability. Although we leave parameters mostly open to set, please note that we suggest the budget allocation according to [12]. Additionally, both methods below can work on the online setting, and due to our analysis on Lemma 3.1, both require $p_1 \neq 1/3$.

First we adapt **TS** for general queries in the setting where we want to ignore potentially a large number of unstable queries out of the m, until we find one stable to return.

Algorithm 4 FS: First Stable algorithm for releasing a single stable query via distance to instability

```
$\mathbf{TS}(D, \{f_1, f_2, ..., f_m\}, \varepsilon, p_1, \delta)$
 1: Let $\varepsilon_1 = p_1\varepsilon$, $\varepsilon_2 = (1-p_1)\varepsilon$ and $c = 2\varepsilon_1/\varepsilon_2$
 2: For $\delta_{max} = \left(\frac{2\delta_q^c + \delta_q - c(\delta_q^c + 2\delta_q)}{4(1-c)}\right)$, let $\delta_q$ be the $\underset{\delta_q}{\text{argmax}}\left[\delta_{max}\right]$ subject to $\delta_{max} \leq \delta/m$
 3: Let $T = \ln(1/\delta_q)/(\varepsilon_2/2)$
 4: Set $\hat{T} = T + \text{Lap}(1/\varepsilon_1)$
 5: for each function $f_i(D)$ do
 6:     Let $q_i(D) = dist_{f_i(D)}$ # Distance-to-Instability
 7:     out $= \mathcal{A}_{stab}(D, f_i, q_i, \hat{T}, \frac{\varepsilon_2}{2})$
 8:     If out $= \perp$, then Output $\perp$
 9:     Else
10:         Output $f_i(D)$
11:         Halt # Ends execution after first success
12: end for
```

Now we adjust $\mathcal{A}_{OQR}$ from [14] for improved noise parameters and flexibility of budget allocation. $\mathcal{A}_{OQR}$ is used the setting where we want to answer as many stable queries as possible out of the m given, while discarding a limited number k of unstable queries.

We emphasize that the change needed is because their Theorem 3.6 uses their Lemma 3.4, stating $\Pr[Lap(1/\varepsilon) > log(2m/\delta)/\varepsilon] \leq \delta/(2m)$, when actually the correct expression to be guaranteed with at most probability $\delta/(2m)$ is $\Pr[Lap(1/\varepsilon) > log(2m/\delta)/\varepsilon + Lap(1/(\varepsilon/2))]$, which is only at most $\delta/(2m)$ if we use $|Lap(1/(\varepsilon/2))|$ (absolute value) instead of $Lap(1/(\varepsilon/2))$. Not including such absolute value requires the rigorous formal analysis we did on our Lemma 3.1, allowing improved results as the threshold will not always be increased by positive noise.

Finally, we also note that, from the analysis on [12], we do not need to draw fresh noise on Line 10 of Algorithm 5, and the utility analysis based on SVT described on [14] can also be adapted to reflect the changes we included.

Algorithm 5 $\mathcal{A}_{OQR-v2}$: Online Query Release via distance to instability (version 2)

```
$\mathbf{TS}(D, \{f_1, f_2, ..., f_m\}, k, \varepsilon, p_1, \delta)$
 1: Let $\varepsilon_1 = p_1\varepsilon/(\sqrt{32k\log(2/\delta)})$
 2: Let $\varepsilon_2 = (1-p_1)\varepsilon/(\sqrt{32k\log(2/\delta)})$
 3: Let $c = 2\varepsilon_1/\varepsilon_2$
 4: For $\delta_{max} = \left(\frac{2\delta_q^c + \delta_q - c(\delta_q^c + 2\delta_q)}{4(1-c)}\right)$, let $\delta_q$ be the $\underset{\delta_q}{\text{argmax}}\left[\delta_{max}\right]$ subject to $\delta_{max} \leq \delta/(2m)$
 5: Let $T = \ln(1/\delta_q)/(\varepsilon_2/2)$
 6: Set $\hat{T} = T + \text{Lap}(1/\varepsilon_1)$ and ctr $= 0$
 7: for each function $f_i(D)$ do
 8:     Let $q_i(D) = dist_{f_i(D)}$ # Distance-to-Instability
 9:     out $= \mathcal{A}_{stab}(D, f_i, q_i, \hat{T}, \frac{\varepsilon_2}{2})$
10:     If out $= \perp$, then ctr $=$ ctr $+ 1$
11:     Output out
12:     Abort if $ctr \geq k$ # Ends after k failures
13: end for
```