A Framework for Practical Differentially Private Top-k Selection

Ricardo Silva Carvalho
rsilvaca@sfu.ca
Simon Fraser University
Canada

Ke Wang
wangk@sfu.ca
Simon Fraser University
Canada

ABSTRACT

We provide a "framework" for differentially private top-k selection with three main properties that make it highly practical. First, it works over a *restricted domain*, i.e. it selects k elements by choosing from the elements that have the top-$\bar{k}$ scores, for a chosen $\bar{k} \geq k$. This means that to privately choose k elements, it will first query a database only for $\bar{k}$ elements. In contrast, traditional mechanisms need to query all of the elements, which for large databases degrades performance. Second, while selecting from the *restricted domain* of top-$\bar{k}$ elements, our framework still provides differential privacy guarantees for the *full domain* of all elements. In other words, it ensures privacy for the entire database irrespective of the elements in the top-$\bar{k}$. Third, our framework is *mechanism-agnostic*, i.e. it automatically adapts *any* traditional differentially private top-k selection mechanism to work on *restricted domain* while maintaining the original privacy guarantees. Unlike previous work that modified specific mechanisms internally to make them work on a restricted domain, our framework takes *as a black-box* any given private top-k mechanism and converts it into a *restricted domain* mechanism. This property greatly simplifies adoption in practice and eliminates the need for complex privacy analyses to work on *restricted domain*, as the framework provides a unified proof of privacy already valid by default for any given mechanism. Finally, we also show, both formally and empirically, that our framework gives superior utility guarantees compared to previous restricted domain mechanisms.

KEYWORDS

Differential privacy, selection, top-k, restricted domain

1 INTRODUCTION

Arising from countless real-world applications, top-k selection can be framed both as the end goal, as in exploratory analysis, such as selecting k most rated movies or k terms with highest TF-IDF scores, or as an intermediate step of complex computations, such as obtaining most relevant gradients of a large neural network to reduce the communication cost of distributed systems [30, 32]. Top-k selection is also a key subroutine for synthetic data generation [31], recommender systems [24], language models [10, 11] and search engines [21]. Regardless of the scenario, if sensitive data are used, we need to take privacy into account, as the result of a top-k selection may leak information [22]. In this context, Differential Privacy (DP) [15] has become one of the most adopted privacy definitions. Use-cases are increasingly more common, with production applications from Amazon [19], Google [18], Microsoft [12] and LinkedIn [23]. In this work, we consider user-level DP, i.e. a dataset D contains records of n users, where every user has a set of elements from the d elements in the *full domain*, and each element has an associated "score" that quantifies its importance in D.

Traditional DP top-k selection mechanisms [3–5, 16, 17, 28], such as the Exponential Mechanism [28] and Report Noisy Max [16], need access to the scores of all elements in the full domain . In general, they also typically perform other operations, like adding noise, to these scores in order to select k elements. However, in real-world scenarios that work with very large domains, querying the database for the scores of all of the elements and adding noise to them tend to degrade performance and undermine the use of these traditional mechanisms in practice.

In this work we focus on the *restricted domain* setting, recently introduced by [14]. In this setting, private mechanisms only need to query a database for the scores of the top-$\bar{k}$ elements, for any chosen $\bar{k} \geq k$. Restricting to these $\bar{k}$ elements tends to improve performance by reducing the amount of data handled by the private mechanisms and by leveraging existing optimized database query plans for "top" queries – see an example in Figure 1 below.

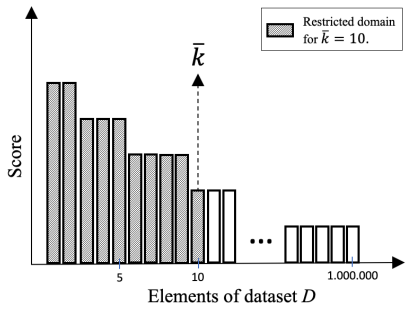

Figure 1: Histogram with the scores of the 1.000.000 elements of a dataset D. Full domain selection mechanisms query and add noise to all 1.000.000 elements to select k. Restricted domain top-k mechanisms only process the top-$\bar{k}$ to select k, for any $\bar{k} \geq k$. The image shows an example for $\bar{k} = 10$.

Moreover, in the *restricted domain* setting, no structure is assumed on the domain – for example, this is different from scenarios with itemsets where properties like the anti-monotonicity of the support measure can be used to reduce the domain analyzed [25]. For these reasons, restricted domain mechanisms are highly desirable in practice. Additionally, they can be directly implemented on top of any existing database system without modifying it, and are consistent with the *principle of least privilege* that aims to minimize the access to sensitive data.

The challenge in the *restricted domain* setting is that even though the private selection only accesses the restricted domain, it still must guarantee DP on the *full domain*. The reason is that the restricted domain defined by the top-$\bar{k}$ elements is itself *sensitive*, as it depends

Under Review, 2023, PLEASE DO NOT DISTRIBUTE
2023. https://doi.org/XXXXXXX.XXXXXXX

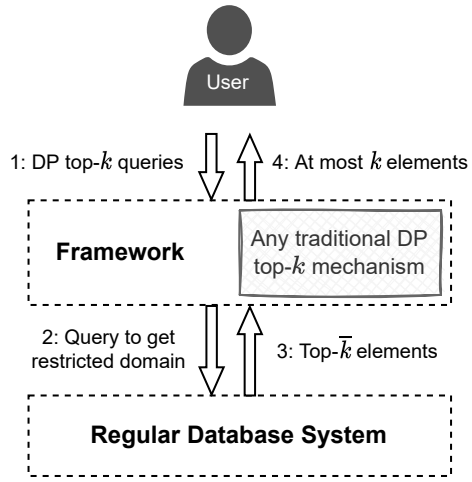

Figure 2: Our framework is built on top of any regular database system. When (1) a user requests a DP top-k query, our framework first (2) queries the database system, which (3) returns the true top-$\bar{k}$ elements. Then the framework applies, as a blackbox, any traditional DP top-k selection mechanism to the top-$\bar{k}$ together with additional private computation, to (4) return at most k elements while guaranteeing DP by default for the full domain.

on the result of querying the sensitive scores of the elements in a private dataset. More precisely, the presence of some element in the top-$\bar{k}$ might be the result of a single user's sensitive data. Thus, simply enforcing DP on the restricted domain is not sufficient to guarantee privacy. In general, we have to limit the probability of selecting elements that would *not* be included in the restricted domain if we removed any individual user's data. These events must only occur with at most a small probability δ to guarantee DP on the full domain of all elements. In this context, previous work [7, 14] on restricted domain enforced DP by creating specific mechanisms and thoroughly working with their inner properties in complex privacy analyses to ensure DP holds on the full domain.

Instead of following a *mechanism-specific* approach and designing another restricted domain mechanism for top-k selection that makes use of the particular properties of a full domain DP mechanism like work [7, 14], in this work we propose a *mechanism-agnostic* approach, denoted as "framework".

As shown in Figure 2, our framework takes as a black-box *any* traditional DP top-k mechanism and applies it to select elements from the restricted domain of the top-$\bar{k}$ elements, which are provided by any regular database system without internal modification, leveraging existing optimized data infrastructure. So in practice our solution is converting *any* given traditional mechanism into its restricted domain counterpart. The novelty is that our framework can be used to easily create new restricted domain mechanisms from *any* traditional DP top-k mechanism, while guaranteeing DP for the full domain by default and without needing any additional DP proof, effectively contributing to the use of differentially private top-k selection in practical scenarios.

Therefore, our approach eliminates the hassle involved in the DP analysis of individual mechanisms. This means that every time a new traditional DP selection mechanism is proposed in the literature, e.g. [27, 29], we can already directly leverage it on *restricted domain* by using our framework. In contrast, previous work [7, 14] enforced privacy by incorporating and modifying the detailed steps of particular mechanisms under consideration, thus, being dependent on mechanism-specific properties. The mechanism-specific modifications and proofs designed by previous work cannot be generalized to other algorithms.

To summarize, our contributions are the following:

- **Theoretical foundation** (Section 4): We present a *theoretical* analysis showing a probability condition that is sufficient to guarantee DP on the full domain when adapting *any* traditional DP top-k mechanism to the restricted domain of the top-$\bar{k}$ elements. This analysis lays the foundation for the generic proof of DP guarantees for the constructive framework in the next contribution.
- **Constructive framework** (Section 5): We provide a constructive framework that is *mechanism-agnostic*, i.e. it takes **any** traditional DP top-k mechanism, applies it on the restricted domain defined by top-$\bar{k}$ elements, and by default satisfies the above-mentioned sufficient condition to remain DP on the full domain of all elements.
- **Utility analysis** (Section 6): We formally prove that, for a fixed privacy budget, the mechanisms obtained from our framework have a better utility compared to previous restricted domain mechanisms [14].
- **Applications** (Section 7): To demonstrate the use of our framework in practice we derive a new top-k selection mechanism that works on restricted domain, denoted as `RestrictedGumbel`, by passing as input the traditional mechanism *top-k with Gumbel noise mechanism* [14].
- **Empirical evaluation** (Section 8): We also provide experimental evaluation on three real-world datasets by comparing `RestrictedGumbel` with the state-of-the-art on the restricted domain scenario [14] for various settings.

2 RELATED WORK

Restricted domain mechanisms address the scenario where the domain of elements to be selected is typically very large and no specific structure information is available to aid top-k selection algorithms. For this purpose, previous works [7, 9, 14] considered the restricted domain defined by the elements with the top-$\bar{k}$ largest scores, for any $\bar{k} \geq k$.

The Large Margin Mechanism (LMM) [9] iteratively searches for a safe margin (similar to a top-$\bar{k}$ cutoff) to limit the selection to a restricted domain. However, this process frequently examines the entire domain while searching for this point. Thus, even though LMM does not sample from elements after the margin when it finds a valid one, if the search for a margin reaches the full domain, LMM in the worst case may not restrict the domain.

The Top-Stable (TS) mechanism [7] searches for *stability* on a given domain to allow an unordered top-k selection. This *stability* is observed when there is a large gap between the scores of adjacent elements in the domain. While it is possible to restrict TS to search

gaps only on the top-$\bar{k}$ elements, if a gap is not found, i.e. the data has no *stability*, TS will return an empty set.

[14] proposed the LimitDomain (LD) mechanism. To avoid breaching privacy due to accessing a restricted domain, LD creates a $\perp$ element with a data-dependent score and includes this element in the top-k selection. Whenever LD selects $\perp$, it only returns the elements selected until then. Thus, for top-k selection, LD will either select k elements or less than k if $\perp$ is among the selected.

However, [14] used the inner properties of the specific mechanisms they adapted to restricted domain to prove DP on the full domain. To create LD, they used the *top-k with Gumbel noise mechanism*, which is the full domain mechanism they modified to work on restricted domain and is equivalent to the Exponential Mechanism [28]. In this context, adapting the mechanisms using their internal properties leads to mechanisms with limited application, as well as complex privacy proofs that are tightly coupled with the specific properties of the underlying mechanism. Our framework takes a given full domain DP mechanism as a blackbox and turns it into a restricted domain DP mechanism. Thus, our approach can adapt any existing or future DP top-k mechanism without requiring additional proofs. We compare our framework with LD in more detail in Section 6.1.

Finally, recently, [6] proposed algorithms to release histograms for several settings, including restricted domain. However, their mechanisms are not directly comparable to this work, as they consider the release of histograms under *continual observation*, while our work is focused on a *given full dataset*. Nonetheless, some algorithms in their work use base mechanisms as a starting point to build upon. Thus, the mechanisms constructed by our framework may be used by their algorithms to derive more solutions for the scenario of continual release of histograms.

3 PRELIMINARIES

Consider a dataset $D \in \mathcal{X}$ containing records of n users for d elements, where the full **domain** of elements is represented by the indices of elements $[d] := \{1, ..., d\}$, and a score function $f : [d] \times \mathcal{X} \to \mathbb{R}$ is such that one user can only alter the score of any given element by at most 1. Moreover, a score function is *monotonic* if adding a user's record never decreases the score for any element, whereas removing a user's record never increases the score for any element. For example, if each user contributes with 1 when the user bought a product and 0 otherwise, then the function f that, when applied to a product i, returns the number of users that bought i is monotonic. In this work, we consider only monotonic f.

Definition 3.1 (Mechanism for top-k selection)**.** *For a given dataset $D \in \mathcal{X}$ having elements with indices on a input domain $[d]$, a parameter $k > 0$, and a (monotonic) score function $f : [d] \times \mathcal{X} \to \mathbb{R}$, we denote $\mathcal{M}_f^k$ as a mechanism for top-k selection applied to D that:*
(i) returns a set of at most k indices as the outcome; and
(ii) for any element $j \in [d]$, has a probability of returning j that is monotonic with respect to the score $f(j, D)$ – that is, increasing the score $f(j, D)$ (while keeping the scores of the other elements constant) will increase the probability of selecting j.

The above definition is general enough to cover all randomized top-k selection mechanisms that are DP, such as the Exponential Mechanism [28] and Report Noisy Top-k [16]. Note that DP mechanisms may return less[1] than k elements in order to guarantee DP. For this reason, we purposefully leave the possibility of returning less than k elements and simply require that $\mathcal{M}_f^k$ is applied to elements in $[d]$ with a *probability* of selecting an element *monotonic* with respect to the score of the element.

Moreover, we refer to *traditional* or *full domain* DP selection mechanisms as those that select k elements querying the scores of all elements in the *full domain*, in contrast to *restricted domain* mechanisms that only query the top-$\bar{k}$ elements in order to select k, for a given $\bar{k} \geq k$. We go into more detail about such mechanisms in the next section.

Let datasets D and D' be *neighbors* if they differ in the addition or removal of one user's data.

Definition 3.2. *(Differential Privacy (DP), [15]) A randomized mechanism $\mathcal{M}$ is (ε, δ)-differentially private, or (ε, δ)-DP, if for all neighbors $D, D' \in \mathcal{X}$ and any possible outcome set $O \subseteq Range(M)$:*

$$\Pr[\mathcal{M}(D) \in O] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in O] + \delta$$

The inequality above ensures that removing or adding any user will have little impact on the outcome of a mechanism satisfying DP. The ε and δ serve as the multiplicative and additive factors in measuring this impact, respectively, and smaller values mean a tighter privacy requirement.

For neighboring datasets D and D' having elements with binary values, and any element $i \in [d]$, we have $|f(i, D) - f(i, D')| \leq 1$, i.e., one user can only alter the score of a given element by at most 1. Moreover, we refer to Δ as the maximum number of elements one user can impact, i.e.

$$\max_{\forall D, D'} \sum_{i=1}^{d} |f(i, D) - f(i, D')| \leq \Delta$$

Δ can either be unknown or equal to some known fixed value, which leads to different algorithms. Unless stated otherwise, we consider Δ to be unknown, as it is the most general setting.

4 THEORETICAL RESULTS

Here we present our theoretical foundation with a general theorem for top-k selection on restricted domain. It provides a relationship between an important probability bound explained below and the DP guarantees needed in the full domain. A general enforcement of this probability bound that works for any DP top-k mechanism will be given in Section 5.

4.1 Main Theorem

First we describe the restricted domain setting and the definition of possible outcomes of a restricted mechanism. After that we use these definitions to formalize our main theoretical result.

We are interested in mechanisms that perform top-k selection by considering only a restricted subset of the elements of the highest quality. More specifically, for any given dataset D, a score function $f : [d] \times \mathcal{X} \to \mathbb{R}$ and the parameter $\bar{k} \geq k$, our *restricted domain*

[1] Note that returning more than k elements would be a different problem, e.g. differentially private set union [8, 20], as the privacy budget should be better allocated to maximize output size.

is defined as the set of top-$\bar{k}$ elements, denoted by $\mathcal{I}_D^{\bar{k}} := \{i_{(j)} \in \{1,2,...,d\} : 1 \leq j \leq \bar{k} \text{ and } f(i_{(1)},D) \geq ... \geq f(i_{(d)},D)\}$, i.e., the set of indices for the $\bar{k}$ elements with the largest scores.

The main problem with restricting the domain based on the data is that the restricted domain itself becomes sensitive and thus may change on neighboring datasets. This way the possible outcomes of a mechanism $\mathcal{M}_f^k$ applied on neighboring datasets can be considerably different. Thus, while we aim to perform top-k selection by accessing the restricted domain, DP is required to hold on the *full domain* containing all elements, not just on the top-$\bar{k}$ elements. With this in mind, we define the possible outcomes in our analysis. Recall that, as seen in Definition 3.1, an outcome is a set of elements being returned, so the outcomes O^S and O^U defined next are sets where each element is an outcome (thus, set of sets).

Definition 4.1. *For any given dataset D, and any possible D' neighbor of D, with restricted domains respectively $\mathcal{I}_D^{\bar{k}}$ and $\mathcal{I}_{D'}^{\bar{k}}$, we define:*
*(i) the **unstable** outcomes in D, denoted as O^U, as the set of outcomes where each outcome in O^U contains at least one element in $\mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}_{D'}^{\bar{k}}$, i.e. one element that is in $\mathcal{I}_D^{\bar{k}}$ but **not** in at least one possible $\mathcal{I}_{D'}^{\bar{k}}$, from some D' neighbor of D; and*
*(ii) the **stable** outcomes, denoted as O^S, are the set of outcomes where each outcome in O^S has only elements in $\mathcal{I}_D^{\bar{k}} \cap \mathcal{I}_{D'}^{\bar{k}}$.*

Basically, each *unstable* outcome in O^U contains some indices of elements present in the restricted domain of D but **not** present in the restricted domain of some D'. In other words, when determining O^U, for any given D, it contains outcomes with at least one element that is in the top-$\bar{k}$ of D but is **not** in the top-$\bar{k}$ of at least one possible D' neighbor of D. In contrast, each *stable* outcome in O^S contains only elements present in both restricted domains of D and D'.

Clearly, returning *unstable* outcomes in D will affect privacy as that would reveal the use of D instead of D'. Thus, *unstable* outcomes in D should be outputted with at most a very small probability to guarantee DP. Let δ_R denote the threshold on this small probability[2].

Theorem 1 below gives a relationship between the upper bound δ_R on the probability of outputting *unstable* outcomes and a DP guarantee on the full domain. For convenience, we denote $\mathcal{M}_f^k$ applied to the restricted domain $\mathcal{I}_D^{\bar{k}}$ as $\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}})$.

Theorem 1 (Main Theoretical Result)**.** *Given a mechanism $\mathcal{M}_f^k$ for top-k selection that is $(\varepsilon_M, \delta_M)$-DP on the full domain $[d]$, an integer $\bar{k} \geq k$, and any dataset D, if the probability of outputting unstable outcomes satisfies $\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^U] \leq \delta_R$, then $\mathcal{M}_f^k$ applied to the restricted domain $\mathcal{I}_D^{\bar{k}}$ satisfies $(\varepsilon_M, \delta_M + \delta_R)$-DP.*

The theorem says that the only "loss" due to adapting the DP mechanism $\mathcal{M}_f^k$ to the restricted domain is the added δ_R in the DP guarantee, and this loss is *independent* of the mechanism $\mathcal{M}_f^k$ being adapted. Moreover, as a DP guarantee, it holds for any given D, and any possible D' neighbor of D, which define the *unstable* outcomes as seen in Definition 4.1.

[2]Note that δ_R cannot be zero, because when using restricted domain mechanisms we have to account for the possibility of outputting *unstable* outcomes.

Therefore, Theorem 1 allows simplified development of mechanisms that want to work over restricted domains: if we want any top-k selection mechanism $\mathcal{M}_f^k$, which is DP on the full domain, to work on our restricted domain and remain DP on the full domain, our only effort goes into making sure that, for any D, the mechanism $\mathcal{M}_f^k$ applied to $\mathcal{I}_D^{\bar{k}}$ outputs *unstable* outcomes with a probability of at most δ_R. A general constructive enforcement of this probability bound that works for any DP top-k mechanism $\mathcal{M}_f^k$ will be given in Section 5.

At first glance, Theorem 1 seems like a straightforward composition [15], reaching the overall privacy guarantee by sequentially combining the mechanism $\mathcal{M}_f^k$ with an additional step that would limit the probability of outputting *unstable* outcomes. However, this simple composition works only if the restricted domains (i.e., the top-$\bar{k}$ elements) were the same on neighboring datasets. Unfortunately, the restricted domain depends on the scores of the elements on each dataset. So, even though we were able to reach a concise result in Theorem 1, the analysis with a data-dependent restricted domain is not trivial. Moreover, the techniques developed by previous work [9, 14] leverage mechanism-specific properties, thus, they cannot be applied to obtain Theorem 1 that assumes no particular property of the mechanism $\mathcal{M}_f^k$ other than being DP on full domain.

4.2 Proof of Theorem 1

We have to show the DP inequality required in Definition 3.2 for all possible outcomes, which can be either *stable* (O^S) or *unstable* (O^U), as shown in Definition 4.1 – note that O^S and O^U are each a set of outcomes, where each outcome is also a set and contains elements. First, Lemma 1 proves the inequality for the *stable* outcomes.

Lemma 1. *Given a mechanism $\mathcal{M}_f^k(D, \mathcal{I})$ for top-k selection satisfying $(\varepsilon_M, \delta_M)$-DP on any given input domain $\mathcal{I} \subseteq \{1,2,...,d\}$, and a parameter $\bar{k}$, if we apply the mechanism on restricted domains dependent on dataset, for any pair of neighboring datasets D and D', and stable outcomes O^S, we have that:*

$$\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^S] \leq e^{\varepsilon_M} \cdot \Pr[\mathcal{M}_f^k(D', \mathcal{I}_{D'}^{\bar{k}}) \in O^S] + \delta_M$$

Proof. First we prove a result on outputting O^S for the **same** dataset D on different indices of restricted domain elements:

$$\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^S] \leq \Pr[\mathcal{M}_f^k(D, \mathcal{I}_{D'}^{\bar{k}}) \in O^S] \quad (1)$$

Let $\mathcal{I}^S = \mathcal{I}_D^{\bar{k}} \cap \mathcal{I}_{D'}^{\bar{k}}$, i.e. the indices of the elements belonging to both restricted domains of neighboring datasets D and D'.

To show Equation (1) we have to note that:
(i) outcomes in O^S only have elements in $\mathcal{I}^S$, which by definition are the same elements in both $\mathcal{I}_D^{\bar{k}}$ and $\mathcal{I}_{D'}^{\bar{k}}$;
(ii) Any other outcome $\notin O^S$ must have at least one element either in $\mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}^S$ or $\mathcal{I}_{D'}^{\bar{k}} \setminus \mathcal{I}^S$, and the remaining elements in $\mathcal{I}^S$.
(iii) $f(i,D) \leq f(j,D)$ for every possible $i \in \mathcal{I}_{D'}^{\bar{k}} \setminus \mathcal{I}^S$ and $j \in \mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}^S$, since, by definition, the indices $\mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}^S$ are *inside* of the top-$\bar{k}$ on D while the indices $\mathcal{I}_{D'}^{\bar{k}} \setminus \mathcal{I}^S$ are *outside* of the top-$\bar{k}$ on the *same* D.

The intuition of Equation (1) is that to have outcomes in O^S (formed only by elements in $\mathcal{I}^S$), we see that in $\mathcal{I}_{D'}^{\bar{k}}$ (right hand

side), the elements $\mathcal{I}^S$ are "competing" against elements that have smaller scores than those in $\mathcal{I}_D^{\bar{k}}$ (left hand side), therefore $\mathcal{I}^S$ should be selected with higher probability to output O^S in $\mathcal{I}_{D'}^{\bar{k}}$ (right hand side). In Appendix A we give an illustration to help this intuition.

Now to formally show Equation (1), since we are dealing with probabilities (which sum to 1), we have:

$$\begin{aligned}&\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^S] + \Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \notin O^S] \\ &= \Pr[\mathcal{M}_f^k(D, \mathcal{I}_{D'}^{\bar{k}}) \in O^S] + \Pr[\mathcal{M}_f^k(D, \mathcal{I}_{D'}^{\bar{k}}) \notin O^S] \qquad (2)\\ &= 1\end{aligned}$$

For outcomes $\notin O^S$, with elements defined in (ii), combining the relation of the scores in (iii) with the fact that the probability of selecting elements is monotonic with respect to the scores of these elements (as given by Definition 3.1), we get:

$$\Pr[\mathcal{M}_f^k(D, \mathcal{I}_{D'}^{\bar{k}}) \notin O^S] \leq \Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \notin O^S] \qquad (3)$$

Given (i), replacing the result of Equation (3) in Equation (2) gives Equation (1), that is:

$$\begin{aligned}&\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^S] + \Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \notin O^S] \\ &\leq \Pr[\mathcal{M}_f^k(D, \mathcal{I}_{D'}^{\bar{k}}) \in O^S] + \Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \notin O^S] \\ &\Rightarrow \Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^S] \leq \Pr[\mathcal{M}_f^k(D, \mathcal{I}_{D'}^{\bar{k}}) \in O^S]\end{aligned}$$

Now to complete the proof of this lemma, we use the property that $\mathcal{M}_f^k$ is $(\varepsilon_M, \delta_M)$-DP. Since the inequality in Definition 3.2 works for any fixed input domain and outcome set, in particular, it works for the domain $\mathcal{I}_{D'}^{\bar{k}}$ and all possible *stable* outcomes O^S:

$$\begin{aligned}&\Pr[\mathcal{M}_f^k(D, \mathcal{I}_{D'}^{\bar{k}}) \in O^S] \\ &\leq e^{\varepsilon_M} \cdot \Pr[\mathcal{M}_f^k(D', \mathcal{I}_{D'}^{\bar{k}}) \in O^S] + \delta_M \qquad (4)\end{aligned}$$

Finally, the inequality required by this Lemma is obtained by the transitivity of Equations (1) and (4). □

Now we restate Theorem 1 to give its proof using Lemma 1.

Theorem 1 (Main Theoretical Result). *Given a mechanism $\mathcal{M}_f^k$ for top-k selection that is $(\varepsilon_M, \delta_M)$-DP on the full domain $[d]$, an integer $\bar{k} \geq k$, and any dataset D, if the probability of outputting unstable outcomes satisfies $\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^U] \leq \delta_R$, then $\mathcal{M}_f^k$ applied to the restricted domain $\mathcal{I}_D^{\bar{k}}$ satisfies $(\varepsilon_M, \delta_M + \delta_R)$-DP.*

Proof. We need to prove, for any $O \subseteq \text{Range}(\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}))$:

$$\begin{aligned}&\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O] \\ &\leq e^{\varepsilon_M} \Pr[\mathcal{M}_f^k(D', \mathcal{I}_{D'}^{\bar{k}}) \in O] + \delta_M + \delta_R \qquad (5)\end{aligned}$$

Let $O^S = O \cap O^S$ and $O^U = O \cap O^U$. Since O is a subset of all possible outcomes $O^S \cup O^U$, and $O^S \cap O^U = \emptyset$, we have:

$$\begin{aligned}&\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O] = \\ &(\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^S]) + (\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^U])\end{aligned}$$

Using Lemma 1, and since we assumed $\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^U] \leq \delta_R$, the above becomes:

$$\begin{aligned}&\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O] \leq \\ &(e^{\varepsilon_M} \cdot \Pr[\mathcal{M}_f^k(D', \mathcal{I}_{D'}^{\bar{k}}) \in O^S] + \delta_M) + (\delta_R)\end{aligned}$$

Finally, as $O^S \subseteq O$, then $\Pr[\mathcal{M}_f^k(D', \mathcal{I}_{D'}^{\bar{k}}) \in O^S] \leq \Pr[\mathcal{M}_f^k(D', \mathcal{I}_{D'}^{\bar{k}}) \in O]$. Replacing this on the last inequality above gives Equation (5). □

It is worth noting that Theorem 1, and Lemma 1 in particular, is effectively **not** requiring any specific property of the mechanism $\mathcal{M}_f^k$ being adapted to restricted domain, aside from it being DP on full domain. Therefore, Theorem 1 applies to any general top-k selection mechanism $\mathcal{M}_f^k$ as given in Definition 3.1. In contrast, the workaround of previous work [7, 14] for analyzing *stable* outcomes required using specific properties of the mechanisms being adapted to restricted domain, such as the exact probability formulations given by the exponential mechanism [28].

5 PRACTICAL CONSTRUCTION

Although theoretically useful, Theorem 1 does not give a constructive way of limiting the probability of outputting *unstable* outcomes as required, i.e. $\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^U] \leq \delta_R$. With this in mind, we present a constructive framework that ensures this condition *by default* and is *mechanism-agnostic*, i.e. it works for **any** traditional DP top-k mechanism $\mathcal{M}_f^k$ being adapted to restricted domain.

5.1 Constructive Framework

As shown in Figure 3, we propose a framework $\mathcal{F}(\mathcal{M}_f^k, \cdot)$ that takes as input **any** traditional DP top-k mechanism $\mathcal{M}_f^k$, with a score function f. Our framework applies $\mathcal{M}_f^k$ to the **restricted domain** defined by the top-$\bar{k}$ elements, purposefully ignoring the fact that it may cause privacy breach. Then it performs an additional step on the result to ensure differential privacy guarantees on the **full domain** defined by all the elements.

In all steps and the DP analysis, the framework treats the input mechanism $\mathcal{M}_f^k$ as a *blackbox*, i.e., without accessing the internal working of $\mathcal{M}_f^k$. So it provides a unified construction of restricted domain mechanisms, such that the DP proof is valid for any traditional top-k mechanism $\mathcal{M}_f^k$ being adapted to restricted domain.

Our constructive $\mathcal{F}(\mathcal{M}_f^k, \bar{k}, \varepsilon_R, \delta_R, \cdot)$ is formally given in Algorithm 1. It considers that the input top-k mechanism $\mathcal{M}_f^k$ is $(\varepsilon_M, \delta_M)$-DP on the full domain. The parameters ε_R and δ_R are the additional privacy budget allocated for adapting $\mathcal{M}_f^k$ to restricted domain. As part of the input, the restricted domain defined by the top-$\bar{k}$ elements $\mathcal{I}_D^{\bar{k}}$ is provided to the framework by the query solver of any existing database system, so the framework does not access the full domain.

On STEP 1, the framework calls the given mechanism $\mathcal{M}_f^k$ to select the top-k elements from the restricted domain (i.e., the top-$\bar{k}$ elements). Then, STEP 2 goes through the selected elements to

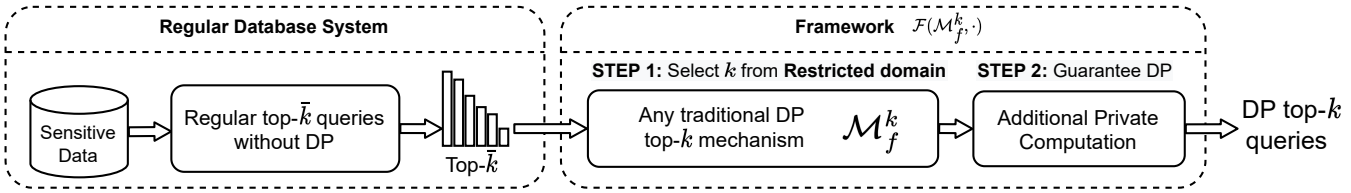

Figure 3: Our framework $\mathcal{F}(\mathcal{M}_f^k, \cdot)$ is built on top of a regular database system. On the left, the database system generates the restricted domain defined by the top-$\bar{k}$ elements. On the right, $\mathcal{F}$ has two steps. In STEP 1, $\mathcal{F}(\mathcal{M}_f^k, \cdot)$ takes as a blackbox any traditional DP top-k selection mechanism $\mathcal{M}_f^k$ and applies the mechanism to select k elements from the restricted domain of the top-$\bar{k}$ elements given by the database system. STEP 2 has additional computation on the k elements selected by $\mathcal{M}_f^k$ to ensure differential privacy for the full domain, even though $\mathcal{M}_f^k$ only selected elements from the restricted domain.

Algorithm 1 $\mathcal{F}(\mathcal{M}_f^k, \bar{k}, \varepsilon_R, \delta_R, D, \mathcal{I}_D^{\bar{k}})$ – Framework for top-k selection from the restricted domain of the top-$\bar{k}$ elements.

Input: Dataset D, selection mechanism $\mathcal{M}_f^k$ that is $(\varepsilon_M, \delta_M)$-DP with monotonic score function $f(\cdot, D)$, parameters $k, \bar{k} \geq k$, ε_R, δ_R, and the indices $\mathcal{I}_D^{\bar{k}}$ of the true top-$\bar{k}$ elements in D.
Output: Ordered set of indices.

STEP 1: Return k elements by applying $\mathcal{M}_f^k$ as a **blackbox** to **the restricted domain** $\mathcal{I}_D^{\bar{k}}$.

1: $S_1 \longleftarrow \mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}})$

SETUP FOR NEXT STEP:

2: Set $\delta_q = \underset{\delta}{\text{argmax}} \left[\frac{1}{4}\delta \cdot (3 + \ln(1/\delta)) \leq \delta_R\right]$
3: Set $T = \log(\frac{1}{\delta_q})/(\varepsilon_R/2)$ and $\hat{T} = T + \text{Lap}(\frac{1}{\varepsilon_R/2})$
4: Let $f_{\bar{k}+1}(D)$ be the $(\bar{k}+1)$th largest score $f(\cdot, D)$
5: Let $S_2 = \emptyset$ be an empty ordered set.

STEP 2: Bound the probability of outputting *unstable*

6: **for** each $i \in S_1$ in order of selection in STEP 1 **do**
7: Let $\nu_i = Lap(\frac{1}{\varepsilon_R/2})$
8: **if** $\left(f(i, D) - f_{\bar{k}+1}(D) - 1\right) + \nu_i > \hat{T}$ **then**
9: $S_2 = S_2 \cup \{i\}$
10: **else**
11: $S_2 = S_2 \cup \{\perp\}$ and **Halt**
12: Return S_2

verify if they are *unstable*, and ensures that *unstable* elements are only outputted with probability at most δ_R. This is achieved by adding noise to the gap "$f(i, D) - f_{\bar{k}+1}(D) - 1$" for each element i selected on STEP 1 – where $f_{\bar{k}+1}(D)$ is the $(\bar{k}+1)$th largest score $f(\cdot, D)$ – and requiring the noisy gap to be larger than a carefully chosen threshold $\hat{T}$ (i.e., lines 7 and 8 of Algorithm 1). The output is an ordered set of indices, as determined by the selection order of $\mathcal{M}_f^k$. We now state the privacy guarantee of Algorithm 1.

Theorem 2 (Constructive Framework). *The framework $\mathcal{F}(\mathcal{M}_f^k, \cdot)$ in Algorithm 1 is $(\varepsilon_M + \varepsilon_R, \delta_M + \delta_R)$-DP.*

Together, Algorithm 1 and Theorem 2 give a constructive framework that adapts any traditional DP selection mechanism $\mathcal{M}_f^k$ to be able to work on a restricted domain while still providing DP on the full domain. The novelty of this approach is that all steps treat the mechanism $\mathcal{M}_f^k$ as a blackbox (i.e., making no access to the internal working of $\mathcal{M}_f^k$), thus, the framework is useful to adapt any DP top-k mechanism $\mathcal{M}_f^k$. Importantly, the DP proof of the resulting restricted domain mechanism is "once for all": once Theorem 2 is proved, immediately we have the stated DP guarantee on the resulting restricted domain mechanism for any existing or future $\mathcal{M}_f^k$. Despite this generality, Section 6 shows our framework with better utility compared to [14], which enforced privacy by directly modifying a particular mechanism, using its specific properties. Note that the mechanism-specific modifications they used cannot be generalized to other mechanisms.

Finally, compared to Theorem 1, Theorem 2 adds a second step to enforce privacy, which was not considered in Theorem 1. This second step is responsible for the added ε_R in the privacy guarantee of Theorem 2 compared to Theorem 1.

5.2 Proof of Theorem 2

Before proving Theorem 2, we describe the Sparse Vector Technique (SVT) [26], which is used in STEP 2 of Algorithm 1.

Algorithm 2 SVT(D, $\{q_1, ..., q_m\}$, T, ε) [26]: Test sensitivity-1 **monotonic** queries against a threshold

1: Let $\hat{T} = T + \text{Lap}(\frac{1}{\varepsilon/2})$
2: **for** each query q_i **do**
3: Let $\nu_i = \text{Lap}(\frac{1}{\varepsilon/2})$
4: **if** $q_i(D) + \nu_i > \hat{T}$, **then** Output $\top$
5: **else** Output $\perp$; **Halt**

In general, SVT works by testing multiple queries against a threshold. Intuitively, it assumes that the test results will be "true". So basically the algorithm accumulates privacy loss only when the result is "false". In the case of Algorithm 2, we allow just one "false" result, as it halts when the test result is $\leq \hat{T}$ (Line 5). That is the reason the ε parameter in Algorithm 2 is *not* multiplied by

any factor related to the number of tests, as formalized below in Theorem 3.

Theorem 3. *SVT in Algorithm 2 is $(\varepsilon, 0)$-DP.*

Proof. Algorithm 2 is Algorithm 7 of [26] with a "below the threshold" test (as it halts when the test result is $\leq \hat{T}$), queries that can change by at most 1 on neighboring datasets (which [26] denotes as $\Delta = 1$) and $c = 1$ (as we halt in the first "below" test). Thus, with monotonic queries, Theorem 7 in [26] for $\varepsilon_3 = 0$, $\varepsilon/2 = \varepsilon_1 = \varepsilon_2$ gives Algorithm 2 as $(\varepsilon, 0)$-DP. □

Now the main property needed for proving Theorem 2 is in Lemma 2, which bounds the probability of *unstable* outcomes in Algorithm 1.

Lemma 2. *Algorithm 1, denoted as $\mathcal{F}(\mathcal{M}_f^k, D, \mathcal{I}_D^{\bar{k}})$, for a given $\bar{k}$ and any dataset D, satisfies $\Pr[\mathcal{F}(\mathcal{M}_f^k, D, \mathcal{I}_D^{\bar{k}}) \in O^U] \leq \delta_R$.*

Proof. Here we will bound the probability of outputting *unstable* outcomes given any dataset D. Recall from Definition 4.1 that O^U is defined for any given D by outcomes that have at least one element in the top-$\bar{k}$ of D which are **not** in the top-$\bar{k}$ of at least one possible D' neighbor of D – as we will see below, in this proof we will look at the worst case scenario to define the possible elements in the outcomes in O^U, which is a common approach in differential privacy proofs.

We will prove this Lemma by induction on the output size t, which is at most k. Therefore, we will bound the probability of *unstable* outcomes for all possible t. Note that we do **not** need an iterative mechanism in STEP 1, as our induction is on the output size t, which is defined by the tests in the iterations of STEP 2 ($t \leq k$). The induction is not on the number of elements selected in STEP 1, which is always k.

The base case ($t = 1$): To have one *unstable* element outputted, first we have some probability for selecting *unstable* elements by STEP 1 of Algorithm 1 using any DP top-k mechanism $\mathcal{M}_f^k$. Then we have just one successful/true test on line 8 of Algorithm 1. Since on any D the largest score possible for any *unstable* element is $f_{\bar{k}+1}(D) + 1$ (Lemma 5.2 in [14]), to bound the probability of selecting one *unstable* element we assume w.l.o.g. the worst case scenario of having such largest score for the element i selected by STEP 1. This is done by replacing $f(i, D)$ with $f_{\bar{k}+1}(D) + 1$ in line 8 of Algorithm 1. This way, the probability over the test on line 8 becomes:

$$\Pr[Lap(\frac{1}{\varepsilon_R/2}) > \log(\frac{1}{\delta_q})/(\varepsilon_R/2) + Lap(\frac{1}{\varepsilon_R/2})]$$

A general and independent probability result of the Laplace distribution, already stated in previous works, such as Lemma D.1 in [14] and Lemma 3.1 in [7], implies that the probability above is:

$$\Pr[Lap(\frac{1}{\varepsilon_R/2}) > \log(\frac{1}{\delta_q})/(\varepsilon_R/2) + Lap(\frac{1}{\varepsilon_R/2})] \leq \frac{1}{4}\delta_q \cdot (3 + \ln(1/\delta_q)) \quad (6)$$

Finally, according to line 2 of Algorithm 1, the right-side of (6) is bounded by δ_R, that is: $\frac{1}{4}\delta_q \cdot (3 + \ln(1/\delta_q)) \leq \delta_R$. Since the probability of selecting any element on STEP 1 is, by definition, ≤ 1, this proves the result for $t = 1$.

Inductive step ($t > 1$): The privacy test on STEP 2 of Algorithm 1 seen in line 8 has **at most** k iterations and the elements outputted will be those that pass such test. Let p_1^U and p_1^S be the maximum probability that, respectively, an *unstable* or *stable* element has a successful/true test on line 8 on the *first* iteration. So for $\hat{T}$ given in line 3, we define:

$$p_1^U = \max_{i \in \mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}_{D'}^{\bar{k}}} \Pr[f(i, D) - f_{\bar{k}+1}(D) - 1 + \nu_i > \hat{T}]$$

$$p_1^S = \max_{i \in \mathcal{I}_D^{\bar{k}} \cap \mathcal{I}_{D'}^{\bar{k}}} \Pr[f(i, D) - f_{\bar{k}+1}(D) - 1 + \nu_i > \hat{T}]$$

Recalling that the tests in line 8 of Algorithm 1 (STEP 2) follow the order of the elements from STEP 1, we denote e_1 as the first element in the ordered list of k elements returned by the mechanism in STEP 1. This way the overall bound for output size $t + 1$ is:

$$\Pr[\mathcal{F}(\mathcal{M}_f^k, D, \mathcal{I}_D^{\bar{k}}) \in O^U] \leq$$

$$\Big(\sum_{j \in \mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}_{D'}^{\bar{k}}} \Pr[e_1 = j]\Big) \cdot \big(p_1^U\big) \cdot 1 \quad (7)$$

$$+ \Big(\sum_{i \in \mathcal{I}_D^{\bar{k}} \cap \mathcal{I}_{D'}^{\bar{k}}} \Pr[e_1 = i]\Big) \cdot \big(p_1^S\big) \cdot \big(\delta_R\big) \quad (8)$$

The first term (7) accounts for the case of e_1 being an *unstable* element (i.e. in $\mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}_{D'}^{\bar{k}}$), then it passing the test on line 8 (p_1^U) and 1 as the probability of any element passing or not tests in the next t iterations of STEP 2. The second term (8) accounts for the case of e_1 being a *stable* element (i.e., in $\mathcal{I}_D^{\bar{k}} \cap \mathcal{I}_{D'}^{\bar{k}}$), it passing the test on line 8 (p_1^S), and then an *unstable* element passing a test in any of the next t iterations of STEP 2, which has a probability bounded by δ_R as assumed by the inductive step.

As discussed in the base case above, $p_1^U \leq \delta_R$. Obviously $p_1^S \leq 1$ (as it is a probability), and since the first element selected is from the restricted domain $\mathcal{I}_D^{\bar{k}}$, which is $\mathcal{I}_D^{\bar{k}} = (\mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}_{D'}^{\bar{k}}) \cup (\mathcal{I}_D^{\bar{k}} \cap \mathcal{I}_{D'}^{\bar{k}})$, then:

$$\sum_{j \in \mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}_{D'}^{\bar{k}}} \Pr[e_1 = j] + \sum_{i \in \mathcal{I}_D^{\bar{k}} \cap \mathcal{I}_{D'}^{\bar{k}}} \Pr[e_1 = i] = 1$$

Using these properties in (7) and (8), we get the result of this proof:

$$\Pr[\mathcal{F}(\mathcal{M}_f^k, D, \mathcal{I}_D^{\bar{k}}) \in O^U] \leq \sum_{j \in \mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}_{D'}^{\bar{k}}} \Pr[e_1 = j] \cdot \delta_R + \sum_{i \in \mathcal{I}_D^{\bar{k}} \cap \mathcal{I}_{D'}^{\bar{k}}} \Pr[e_1 = i] \cdot \delta_R = \delta_R$$

□

Now we restate Theorem 2 to give its proof using Lemma 2.

Theorem 2 (Constructive Framework). *The framework $\mathcal{F}(\mathcal{M}_f^k, \cdot)$ in Algorithm 1 is $(\varepsilon_M + \varepsilon_R, \delta_M + \delta_R)$-DP.*

Proof. This proof is a direct application of Theorem 1 with the general input mechanism $\mathcal{M}_f^k$ being $\mathcal{F}(\mathcal{M}_f^k, \cdot)$ given by Algorithm 1. First, Theorem 1 requires that the input mechanism is DP on any fixed domain. This DP guarantee for Algorithm 1 comes from the composition of STEP 1 and STEP 2. STEP 1 has a mechanism $\mathcal{M}_f^k$ that is $(\varepsilon_M, \delta_M)$-DP on any given fixed domain, and STEP 2 is an application of SVT in Algorithm 2 with the queries q_i being the gap $f(i, D) - f_{\bar{k}+1}(D) - 1$ and the parameter ε_R, which satisfies $(\varepsilon_R, 0)$-DP according to Theorem 3. Thus the sequential composition gives $(\varepsilon_M + \varepsilon_R, \delta_M)$-DP for Algorithm 1 on any fixed input domain. So using this DP guarantee for Algorithm 1 as the input in Theorem 1 and the probability bound δ_R proved in Lemma 2, we complete this proof with Algorithm 1 as $(\varepsilon_M + \varepsilon_R, \delta_M + \delta_R)$-DP. □

6 UTILITY ANALYSIS

Here we show a theoretical utility result that gives a condition for our framework to output k elements with high probability. Aside from giving insights about utility, the result below will also be used in Section 6.1 to formally compare the utility of our framework and LD [14].

By restricting to the top-$\bar{k}$ elements, restricted domain algorithms usually return elements with asymptotically better scores than full domain mechanisms [1, 14]. Like previous work [14], in our framework this quality improvement comes at the cost of possibly outputting fewer than k elements due to the threshold test on line 8 of Algorithm 1. Thus, to measure utility we derive a sufficient condition to output k elements with a high probability.

Theorem 4. *Let δ_q be computed on line 2 of Algorithm 1. The framework outputs k elements with probability at least $1 - \beta$ if:*

$$f_{\bar{k}}(D) - f_{\bar{k}+1}(D) \geq 1 + \log(k/(\sqrt{\delta_q}\beta))/(\varepsilon_R/4)$$

Proof. To output k elements, every element selected on STEP 1 of Algorithm 1 should pass the test on line 8 of STEP 2. If that should happen with overall probability of at least $1 - \beta$, then it should fail with overall probability of at most β. This means we have to show that each individual test will only fail with probability at most β/k. To obtain a probability bound for failing a single test, we look at the smallest possible score on the top-$\bar{k}$: $f_{\bar{k}}(D)$, i.e. the score of the element with the $\bar{k}$th largest score.

In this context, writing out the test from line 8 of Algorithm 1 for $f(i, D) = f_{\bar{k}}(D)$, we need to show:

$$\begin{aligned}\Pr[\big(f_{\bar{k}}(D) - f_{\bar{k}+1}(D) - 1\big) + Lap(\frac{1}{\varepsilon_R/2}) \\ < \log(\frac{1}{\delta_q})/(\varepsilon_R/2) + Lap(\frac{1}{\varepsilon_R/2})] \\ \leq \frac{\beta}{k}\end{aligned} \tag{9}$$

under the assumed condition:

$$f_{\bar{k}}(D) - f_{\bar{k}+1}(D) \geq 1 + \log(\frac{k}{\sqrt{\delta_q}\beta})/(\varepsilon_R/4)$$

We write this condition into:

$$\begin{aligned}f_{\bar{k}}(D) - f_{\bar{k}+1}(D) - 1 \\ \geq \log(\frac{1}{\delta_q})/(\varepsilon_R/2) + \log(\frac{1}{\beta^2/k^2})/(\varepsilon_R/2)\end{aligned} \tag{10}$$

and replace it in the left side of Equation (9):

$$\Pr[\log(\frac{1}{\beta^2/k^2})/(\varepsilon_R/2) + Lap(\frac{1}{\varepsilon_R/2}) < Lap(\frac{1}{\varepsilon_R/2})]$$

Letting $\beta_x = \beta^2/k^2 < 1$ for simplicity, and using the general probability result from the Laplace distribution applied in (6):

$$\begin{aligned}\Pr[\log(\frac{1}{\beta^2/k^2})/(\varepsilon_R/2) + Lap(\frac{1}{\varepsilon_R/2}) < Lap(\frac{1}{\varepsilon_R/2})] \\ \leq \frac{1}{4}\beta_x \cdot (3 + \ln(1/\beta_x)) \leq \frac{\beta}{k}\end{aligned} \tag{11}$$

The last $\leq$ above comes from using $\ln(1/x) \leq (1/x - 1)/(\sqrt{1/x})$ for $0 \leq x \leq 1$, and knowing $0 \leq \beta_x \leq 1$, as follows:

$$\begin{aligned}\frac{1}{4}\beta_x \cdot (3 + \ln(1/\beta_x)) \leq \frac{1}{4}\beta_x \cdot (3 + \frac{1/\beta_x - 1}{\sqrt{1/\beta_x}}) \\ = \frac{1}{4}\beta_x \cdot (3 + \frac{1 - \beta_x}{\sqrt{\beta_x}}) \leq \frac{1}{4}\beta_x \cdot (3 + \frac{1}{\sqrt{\beta_x}}) \\ = \frac{3}{4}\beta_x + \frac{1}{4}\sqrt{\beta_x} \leq \frac{3}{4}\sqrt{\beta_x} + \frac{1}{4}\sqrt{\beta_x} = \sqrt{\beta_x} = \frac{\beta}{k}\end{aligned}$$

This has proved the inequality in Equation (9) under the assumed condition, as required. Consequently, the above gives at least $1 - \beta$ probability of outputting k elements, as explained in the first paragraph of this proof.

□

This theorem gives a general worst-case condition, in terms of the minimum gap $1 + \log(k/(\sqrt{\delta_q}\beta))/(\varepsilon_R/4)$, for Algorithm 1 to return k elements with probability at least $1 - \beta$. The smaller the minimum gap, the more likely this probability holds. Surprisingly, this gap requirement does **not** contain $\bar{k}$. This means that, for our framework, the utility measured in terms of outputting k elements is independent of the actual value of $\bar{k}$. This independence gives the benefit of choosing a large $\bar{k}$ to maximize the output size of Algorithm 1. See more details on choosing a value for $\bar{k}$ and the impact on the output size in Section 6.2. In contrast, previous work had a similar utility statement but with a minimum gap that was $O(\log(k\bar{k}))$ (Lemma 8.1 in [14]).

Finally, another way of looking at Theorem 4 is that it gives parameters that minimizes false negatives (i.e. the true top-k elements that are not returned by Algorithm 1) with high probability. On the other hand, false positives (i.e. the elements returned by Algorithm 1 that are not among the true top-k) are basically already minimized by default when selecting from a restricted domain, as selecting only from the top-$\bar{k}$ elements returns elements with high scores. To complement this discussion, in Appendix G we include additional experiments using the "F1 score" metric, which weights true/false positives and false negatives.

6.1 Comparison to Previous Work

Now we compare our framework with the `LimitDomain` (LD) mechanism [14]. First we briefly introduce LD, and then we compare the utility theoretical guarantees of LD given in [14] and those from our framework derived above in Theorem 4.

`LimitDomain` (LD) mechanism [14]: LD is $(\varepsilon_{LD}, \delta_M + \delta_R)$-DP, where using the current most optimal composition for LD [13], we have ε_{LD} as:

$$\min\left(k\cdot\varepsilon,\quad k\left(\frac{\varepsilon}{1-e^{-\varepsilon}}-1-\ln\left(\frac{\varepsilon}{1-e^{-\varepsilon}}\right)\right)+\varepsilon\sqrt{\frac{k}{2}\ln\frac{1}{\delta_M}}\right) \tag{12}$$

Essentially, LD internally customizes the full domain *top-k Gumbel noise mechanism* [14] to the restricted domain setting by creating a $\perp$ element to include in the selection and forcing the selection to halt once $\perp$ is returned. $\perp$ is embedded in the process to ensure returning *unstable* elements with probability of at most δ_R. Thus LD deals with the *unstable* elements by directly modifying an existing DP mechanism internally.

In contrast, our framework in Algorithm 1 first selects k elements by calling a *given* top-k mechanism $\mathcal{M}_f^k$ as a blackbox without any internal modification, and deals with the *unstable* elements by doing an additional step to the k elements selected. This design gives us a general construction that works for basically any given DP mechanism $\mathcal{M}_f^k$ and allows us to separate the selection from the privacy issues related to restricting the domain.

Condition to output k elements: Now we show that our framework has a higher probability of outputting k elements compared to LD.

Theorem 5. *Suppose that LimitDomain (LD) [14] and Algorithm 1 with privacy given in Theorem 2 have the same overall privacy guarantee, by letting $\varepsilon_{LD} = \varepsilon_M + \varepsilon_R$. Then for δ_q defined on line 2 of Algorithm 1, such that $\delta_R^2 \le \delta_q \le 1$, the probability of Algorithm 1 outputting k elements on D, given by Theorem 4, is higher than that of LD, under the same conditions, if we set $\varepsilon_R \ge 4\varepsilon$, where ε is the per iteration privacy budget used by LD.*

Proof. The gap requirement for the utility guarantees of our framework comes from Theorem 4. For `LimitDomain` (LD) [14] the same gap is straightforward to obtain by looking at their Lemma 8.1 and getting the gap between the largest $\bar{k}+1$ and $\bar{k}$ elements (instead of $\bar{k}+1$ and k). Putting these two gaps side to side, we have that a mechanism from our framework and LD will output k elements with probability respectively at least $1-\beta_{\mathcal{F}}$ and $1-\beta_{LD}$ if:

$$\text{Ours (Theorem 4):}\quad f_{\bar{k}}(D) - f_{\bar{k}+1}(D) - 1 \ge \ln\left(\frac{k}{\sqrt{\delta_q}\beta_{\mathcal{F}}}\right)/(\varepsilon_R/4)$$

$$\texttt{LimitDomain:}\quad f_{\bar{k}}(D) - f_{\bar{k}+1}(D) - 1 \ge \ln\left(\frac{k\bar{k}}{\delta_R\beta_{LD}\cdot(\bar{k}-k+1)}\right)/\varepsilon$$

Now to compare the probabilities we fix the same gap requirement for both our framework and LD:

$$\ln\left(\frac{k\bar{k}}{\delta_R\beta_{LD}\cdot(\bar{k}-k+1)}\right)/\varepsilon = \ln\left(\frac{k}{\sqrt{\delta_q}\beta_{\mathcal{F}}}\right)/(\varepsilon_R/4) \tag{13}$$

Using the assumptions that $\delta_q \ge \delta_R^2$, or $\sqrt{\delta_q} \ge \delta_R$, and $\varepsilon_R \ge 4\varepsilon$, we have:

$$\ln\left(\frac{k}{\sqrt{\delta_q}\beta_{\mathcal{F}}}\right)/(\varepsilon_R/4) \le \ln\left(\frac{k}{\delta_R\beta_{\mathcal{F}}}\right)/(\varepsilon_R/4) \le \ln\left(\frac{k}{\delta_R\beta_{\mathcal{F}}}\right)/\varepsilon$$

Since $\bar{k}-k+1 \le \bar{k}$ or equivalently $1 \le \bar{k}/(\bar{k}-k+1)$, the above becomes:

$$\ln\left(\frac{k\bar{k}}{\delta_R\beta_{\mathcal{F}}(\bar{k}-k+1)}\right)/\varepsilon \ge \ln\left(\frac{k}{\delta_R\beta_{\mathcal{F}}}\right)/\varepsilon$$

Connecting this result to Equation (13), we get:

$$\ln\left(\frac{k\bar{k}}{\delta_R\beta_{LD}\cdot(\bar{k}-k+1)}\right)/\varepsilon = \ln\left(\frac{k}{\sqrt{\delta_q}\beta_{\mathcal{F}}}\right)/(\varepsilon_R/4) \le \ln\left(\frac{k\bar{k}}{\delta_R\beta_{\mathcal{F}}(\bar{k}-k+1)}\right)/\varepsilon$$

Simplifying the result above gives us:

$$\beta_{\mathcal{F}} \le \beta_{LD}$$
$$1-\beta_{\mathcal{F}} \ge 1-\beta_{LD}$$

Thus, this proves our framework on Algorithm 1 has higher probability of returning k elements. □

Theorem 5 formally gives a guarantee that, for a reasonable choice of parameters, our approach has a higher probability to return k elements than LD. $\delta_R^2 \le \delta_q \le 1$ can be easily satisfied even for small datasets, since usually in DP we set $\delta_R \ll 1/n$ for a dataset with n users. Moreover, ε_R is a choice of the analyst, thus being free to set as suggested by Theorem 5. In practice, Theorem 5 gives us a general way of choosing a value for ε_R – it suggests ε_R to be at least equal to four times the value of ε, which is the budget per iteration used by LD, **not** the total budget. Thus, ε_R is a fraction of the total privacy budget, which is given by Theorem 2.

6.2 Choice of the $\bar{k}$ Parameter

The choice of $\bar{k}$, i.e. the restricted domain size, might affect the number of elements returned. To output more elements we need to increase the probability of success of the tests in line 8 of Algorithm 1. For fixed $\hat{T}$ and ν_i, smaller values of the score $f_{\bar{k}+1}(D)$ will improve the probability of success, as the value tested against $\hat{T}$ will be larger. Since $\hat{T}$ and ν_i do not depend on $\bar{k}$, we can simply choose a large $\bar{k}$ as a heuristic to decrease $f_{\bar{k}+1}(D)$ and improve the probability of outputting more elements. For example, for a given k we can set $\bar{k} = 50\cdot k$.

In contrast, LD [14] had $\bar{k}$ in the score of their $\perp$ element, denoted as $h_\perp$ in their paper, which works as a threshold for outputting elements. Thus, they had a trade-off when choosing $\bar{k}$, as a larger $\bar{k}$ would decrease $f_{\bar{k}+1}(D)$ but also increase the $\bar{k}$ term in the threshold. For this reason, previous work suggests to spend extra privacy budget to choose a $\bar{k}$ between k and a $\bar{k}_{max}$ using standard DP tools to solve an optimization problem. The heuristic choice of large $\bar{k}$ for our framework, as discussed above, avoids spending this extra privacy budget.

7 APPLICATIONS

To demonstrate the power of our framework, we apply Algorithm 1 and Theorem 2 to develop novel differentially private top-k mechanisms for restricted domain.

We consider the scenario of unknown Δ, i.e., a single user can impact an arbitrary number of elements, and derive a novel mechanism for restricted domain by instantiating $\mathcal{M}_f^k$ in Algorithm 1 with the *top-k Gumbel noise mechanism* [14]. This will allow a direct comparison with LD, in our experiments in Section 8, because LD (as explained in Section 6.1) also modifies the *top-k Gumbel noise mechanism* to obtain their restricted mechanism.

`RestrictedGumbel` (RG): Consider the *top-k Gumbel noise mechanism*, detailed in Algorithm 4 of Appendix B, which satisfies $(\varepsilon_M, \delta_M)$-DP for any $\delta_M > 0$. The `RestrictedGumbel` (RG) is defined in Algorithm 3 as the new restricted domain version of *top-k Gumbel noise mechanism* obtained by instantiating $\mathcal{M}_f^k$ with such mechanism in Algorithm 1, i.e. $\mathcal{F}(\mathcal{M}_f^k, \cdot)$.

Algorithm 3 $RG(D, k, \bar{k}, \varepsilon_M, \delta_M, \varepsilon_R, \delta_R)$ – `RestrictedGumbel`

Input: Parameters $k, \bar{k} \geq k$, $\varepsilon_M, \delta_M, \varepsilon_R, \delta_R$, dataset D and the indices $\mathcal{I}_D^{\bar{k}}$ of the true top-$\bar{k}$ elements in D.;
Output: Ordered set of indices.

1: Let $\mathcal{M}_f^k \leftarrow GK(D, k, \bar{k}, \varepsilon_M)$ from Algorithm 4 of Appendix B
2: Return output of $\mathcal{F}(\mathcal{M}_f^k, \bar{k}, \varepsilon_R, \delta_R, D, \mathcal{I}_D^{\bar{k}})$ from Algorithm 1

The following Theorem 6 follows immediately from Theorem 2 with $\mathcal{M}_f^k$ being $(\varepsilon_M, \delta_M)$-DP, serving as a concrete illustration of our framework in simplifying the privacy analysis of new restricted domain mechanisms.

Theorem 6. *For any $\delta_M > 0$ and ε_M from Equation* (15)*, RG is $(\varepsilon_M + \varepsilon_R, \delta_M + \delta_R)$-DP.*

An empirical comparison between RG with LD in various settings is performed in Section 8.

Adapting other recent mechanisms: Recently two novel traditional mechanisms for DP selection with state-of-the-art properties were proposed: Permute-and-Flip [27] and the Oneshot Laplace [29]. Instead of working with their internal properties to modify them in order to select from a restricted domain, **our framework can directly adapt these mechanisms to restricted domain** without needing any additional DP analysis, providing a direct use with proven DP guarantees by default. We brought these two examples to emphasize the practicality of our framework, which makes it easily adoptable by DP practitioners.

Mechanisms that consider fixed Δ: So far, we have considered the general case of unknown Δ for deriving new restricted domain mechanisms. While mechanisms that consider Δ unknown have privacy loss scaling with k, as seen on Equation (12), the benefit of considering a fixed $\Delta < k$ is that we can take advantage of Δ and have a mechanism with privacy loss scaling with a value smaller than k. As an example in this scenario, consider the *Gaussian mechanism* [2], which adds Gaussian noise based on Δ to the scores of the elements and returns the elements with their noisy scores. Selecting the k elements with the largest noisy scores gives us the top-k elements. To create a restricted domain version we apply Algorithm 1 with $\mathcal{M}_f^k$ being the above top-k *Gaussian mechanism*. A formal definition of this mechanism, including the privacy guarantees, is detailed in Appendix E. Interestingly, the privacy loss of this new mechanism scales with $\sqrt{\Delta}$, since the *Gaussian mechanism* uses L2 sensitivity. In contrast, the existing restricted domain mechanism for the setting of known Δ [14] has a privacy loss scaling with Δ.

8 EXPERIMENTS

As discussed in Section 2, most of the previous work have drawbacks and do not practically restrict the domain. Thus, following recent work [7], here we compare the mechanism RG derived from our framework in Algorithm 1 with the mechanism considered to be the current state-of-the-art on restricted domain: LD [14]. For completion we also include in Section 8.5 a performance comparison between RG and a traditional mechanism that works on full domain: the Exponential Mechanism (EM) [28]. Additionally, in Appendix F we also compare RG with the recently introduced restricted domain mechanism TS [7].

The experiments were executed using Python 3.9.10 on a 4-core Intel i7 1.7GHz processor with 8GB of RAM.

All of the package requirements are described with the code. The code and instructions on where to download the datasets are publicly available at an anonymized repository[3].

8.1 Datasets

We compare the results on three real-world datasets: Yelp (D_1), Foursquare (D_2) and Gowalla (D_3). Table 1 gives an overview of each dataset. To help gauge utility depending on the dataset, in Appendix H we include a discussion on the applicability of restricted domain mechanisms depending on the dataset used.

Table 1: Overview of datasets. The number of users, elements, elements with count 1 and percentiles of counts.

Dataset		Users	Elements	Perc. of 1	Percentile 50^{th}	99^{th}	100^{th}
D_1	Yelp	365,869	132,700	35%	3	96	2262
D_2	Foursquare	2,293	100,191	60%	1	28	1274
D_3	Gowalla	107,092	1,280,969	49%	2	24	2931

The first dataset, D_1, comes from the Yelp Dataset Challenge[4], and contains businesses' tips given by users. We perform top-k selection out of $d = 132,700$ businesses given by $n = 365,869$ users. The second dataset, D_2, is from Foursquare[5], a mobile app where users check-in and write recommendations about venues. The dataset includes long-term check-in data in New York City and Tokyo collected from Foursquare from April 2012 to February 2013. In the experiments we perform top-k selection of $d = 100,191$ venues from check-ins made by $n = 2,293$ users. The last dataset, D_3,

[3] Anonymized repository with code and instructions to download datasets for experiments: https://anonymous.4open.science/r/supplementary-material-40/.

[4] Yelp Dataset Challenge, available at: https://www.kaggle.com/yelp-dataset/yelp-dataset?select=yelp_academic_dataset_tip.json.

[5] Foursquare dataset, available at: https://sites.google.com/site/yangdingqi/home/foursquare-dataset.

is the Gowalla dataset[6], containing time and location information of check-ins made by users. We perform top-k selection from $d = 1,280,968$ locations based on check-ins of $n = 107,092$ users.

For every dataset considered, we always preprocess the elements to be selected so that each user only contributes a single value of 1 for an element when the user has 1 or more activities on the element, and 0 otherwise. Table 1 also gives an overview of the counts encountered on each dataset used. Take Gowalla dataset as an example, 49% out of the 1,280,968 locations had only 1 user checking-in, 50% had less than or equal to 2 unique users checking-in, 99% less than or equal to 24, and the maximum number of unique users checking-in a single location was 2931.

8.2 Settings

We show the average result of 100 independent trials for k equal to 10 (small), 50 (medium) and 100 (large). Regarding $\bar{k}$, as discussed in Section 6.2, our framework would **not** need to spend privacy budget to choose $\bar{k}$, and a simple heuristic like $\bar{k} = 100 \cdot k$ would be enough. In contrast, LD proposes to allocate privacy budget specifically to choose $\bar{k}$, leaving less privacy budget allocated to the actual top-k selection, which could decrease the utility of their results. Although this represents an advantage of our framework, we chose to simply use fixed values of $\bar{k}$ for a clear and direct comparison. Therefore, for each k we set $\bar{k}$ equal to k (small), $10 \cdot k$ (medium) and $50 \cdot k$ (large).

To fairly allocate the same privacy budget for all mechanisms, we fix every mechanism with the same overall $(\varepsilon_{total}, \delta_{total})$-DP guarantee, for $\delta_{total} = 1/(2n)$, where n is the number of users on Table 1 for each dataset, and ε_{total} varies from 0.01 to 1.00. Below we detail the specific individual parameters used by each mechanism.

Usually, composition methods [13, 14] give a final ε_{total} given the privacy budget per iteration. So to take the inverse path, we use the simple algorithm denoted as **Ind**, described in Appendix C, to derive the ε of individual iterations retroactively given a value of ε_{total}, which comes from the composition in [13], seen in Equation (12).

Therefore, with the fixed overall $(\varepsilon_{total}, \delta_{total})$-DP guarantee for all of the mechanisms evaluated, their individual settings are:

- **LD**: From Section 6.1 it is $(\varepsilon_{LD}, \delta_M + \delta_R)$-DP, so we use:
  - $\delta_M = \delta_R = \delta_{total}/2$
  - $\varepsilon_{LD} = \varepsilon_{total}$ with the privacy budget for each individual selection iteration calculated as $\varepsilon = \textbf{Ind}(\varepsilon_{LD}, k, \delta_M)$
- **RG**: From Theorem 6 it is $(\varepsilon_M + \varepsilon_R, \delta_M + \delta_R)$-DP, so we use:
  - $\delta_M = \delta_R = \delta_{total}/2$
  - $\varepsilon_R = 4 \cdot \textbf{Ind}(\varepsilon_{total}, k, \delta_M)$
  - $\varepsilon_M = \varepsilon_{total} - \varepsilon_R$ with the privacy budget for each individual selection iteration calculated as $\varepsilon = \textbf{Ind}(\varepsilon_M, k, \delta_M)$
- **EM**: For top-k selection we apply EM k times with peeling[7]. From [28] and composition from [14] it is $(\varepsilon_M, \delta_M)$-DP, where
  - $\delta_M = \delta_{total}$
  - $\varepsilon_M = \varepsilon_{total}$ with the privacy budget for each individual selection iteration calculated as $\varepsilon = \textbf{Ind}(\varepsilon_M, k, \delta_M)$

[6] Gowalla dataset, available at: https://snap.stanford.edu/data/loc-gowalla.html.

[7] EM with peeling for top-k selection is the Exponential Mechanism [28] applied k times, where after each iteration we "peel off" from the selection those already selected previously. See [14, 28] for more details.

The choice of ε_R for RG above comes from Theorem 5, where we outperform LD for $\varepsilon_R \geq 4\varepsilon$. Thus, that sets ε_R according to four times the value that is used per iteration by LD. This provides a general way to set ε_R, which is the budget only for the second step of Algorithm 1. Moreover, the actual individual budget for each single selection on RG (first step of Algorithm 1) will come from splitting the remaining budget $\varepsilon_{total} - \varepsilon_R$ over the k iterations, as showed above by the use of **Ind** detailed in Appendix C.

8.3 Metrics

We evaluate two metrics for the output utility[8]. Note that restricted domain mechanisms already return elements with high scores, because the selection is restricted to the top-$\bar{k}$. However this improvement comes at the cost of not guaranteeing to return k elements. Therefore, our first metric is the "number of elements returned".

Additionally, similar to recent work [7], we also measure the "Score Ratio" ($\mathcal{SR}$), defined as the ratio between the "sum of the scores of the elements returned" and the "sum of the scores of the true top-k elements". $\mathcal{SR}$ gives an idea of the scores each mechanism outputs compared to the best possible scores, which are those from the true top-k scores.

8.4 Results of RG versus LD

Figure 4 (number of elements returned) and Figure 5 ($\mathcal{SR}$) show the experimental evaluation comparing our mechanism RG (Section 7) with LD [14].

From Figure 4 we can see that RG consistently returns more elements than LD on all three datasets and settings. Moreover we see small values of $\bar{k}$ having significantly fewer elements returned, and that RG has greater improvements than LD for larger $\bar{k}$, which supports our theoretical findings in Section 6.2. We also note RG giving very significant advantages on some settings. For example, Figure 4 shows for D_1, $k = 100$, $\bar{k} = 50k$ and $\varepsilon_{total} = 1.0$, RG returning over three times more elements than LD.

Regarding the "Score Ratio" $\mathcal{SR}$, Figure 5 shows RG consistently outperforming LD, outputting valuable elements of larger scores for all three datasets in our experiments. We see RG needing less privacy budget than LD for comparable utility. For example, for D_1, $k = 50$ and $\bar{k} = 50k$, the total budget $\varepsilon_{total} = 0.7$ is enough to have RG achieving "Score Ratio" $\mathcal{SR} = 1.0$, while LD only reaches such value with $\varepsilon_{total} = 1.0$. Therefore, we note that in general, RG has similar improvements compared to LD in terms of both "number of elements returned" and $\mathcal{SR}$.

Variance: In all figures, the vertical bar represents one standard deviation of the mean. Note that RG may present a larger variance on the results compared to LD due to the privacy testing on the second step of Algorithm 1 following the noisy order of selection of the first step. Nonetheless, the utility of RG still remains considerably higher than LD, empirically as well as theoretically from Theorem 5. For the settings where RG has a larger variance, it also gets substantially more elements outputted than LD. For example, from Figure 4 on D_1 with $k = \bar{k} = 50$ and all values of ε_{total}, RG has the standard deviation of 7.3 and LD of 4.3, but the mean output size of RG is 10.9 and LD is 5.2.

[8] In Appendix G we include additional experiments using the "F1 score" metric.

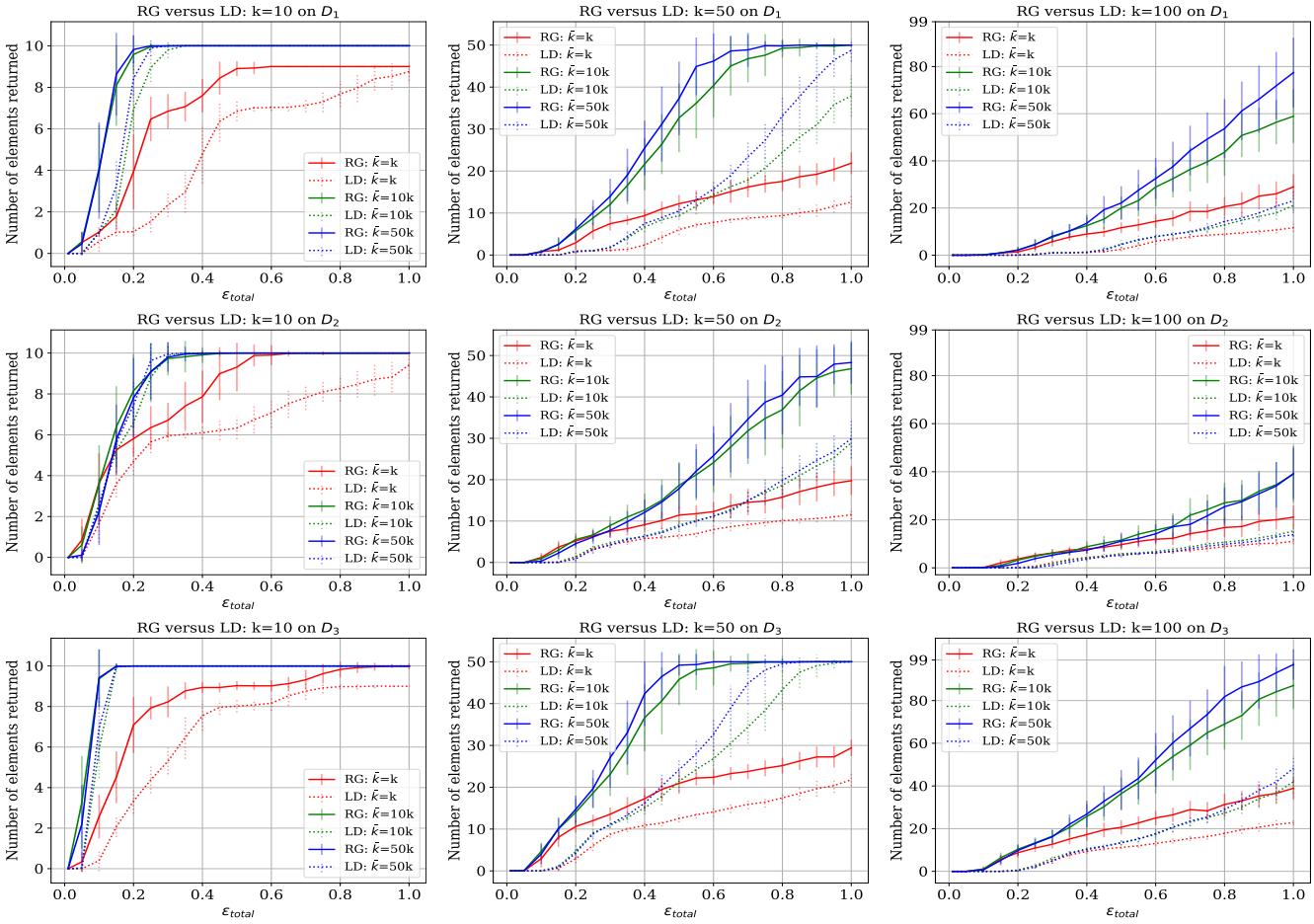

Figure 4: Number of elements returned (on 100 independent trials) for restricted domain mechanisms RG and LD and various k, $\bar{k}$ and ϵ_{total} and three datasets. Each row compares mechanisms on a different dataset. Our mechanism RG has consistently better utility than the previous work LD on every scenario.

8.5 Results of RG versus a Traditional Full Domain Mechanism

To illustrate the efficiency gains from using restricted domain mechanisms, here we compare the performance of our RG and the traditional *full domain* mechanism for differentially private selection, the Exponential Mechanism (EM) [28]. The idea is to evaluate the execution time of running both algorithms for fixed settings. The privacy budget and settings were described in Section 8.2. On each setting we use the same overall privacy budget of ϵ_{total} for both mechanisms.

Both EM and RG work over the result of querying a database system for elements and their aggregated scores. The difference is that EM will select k elements from the complete data, while RG selects k elements only from the top-$\bar{k}$ elements, for $\bar{k} \geq k$, as seen in Figure 2. For simplicity and due to the results being highly dependent on the database system used, we will not include the time to query the database. This way, for RG the time will include the selection of k elements given the restricted domain of top-$\bar{k}$. Whereas for EM the time will be the iterative top-k selection step given the full domain. Note that the restricted domain can be obtained from any existing database system that solves "top" queries. Thus, including this step would give advantages to RG, as in practice it would just query the system for the top-$\bar{k}$ elements. On the other hand, EM would always need to query and obtain the complete aggregated data for the selection, increasing the communication costs and latency.

Table 2 shows the comparison of execution time between RG and EM. Even for relatively small datasets such as D_1 and D_2, which have around 100,000 elements, RG is already 6 to 20 times faster than EM. When we look at the results of D_3, which has around one million elements, the execution time of RG can be more than 200 times faster. For example, on D_3 for $k = 10$, EM takes an average time of 475.94 milliseconds, and RG only around 2.27 milliseconds. This means EM is taking an average of around 0.5 seconds for each top-k selection, which is not practical for real-world systems, especially when the selection is part of a larger and more complex analysis pipeline.

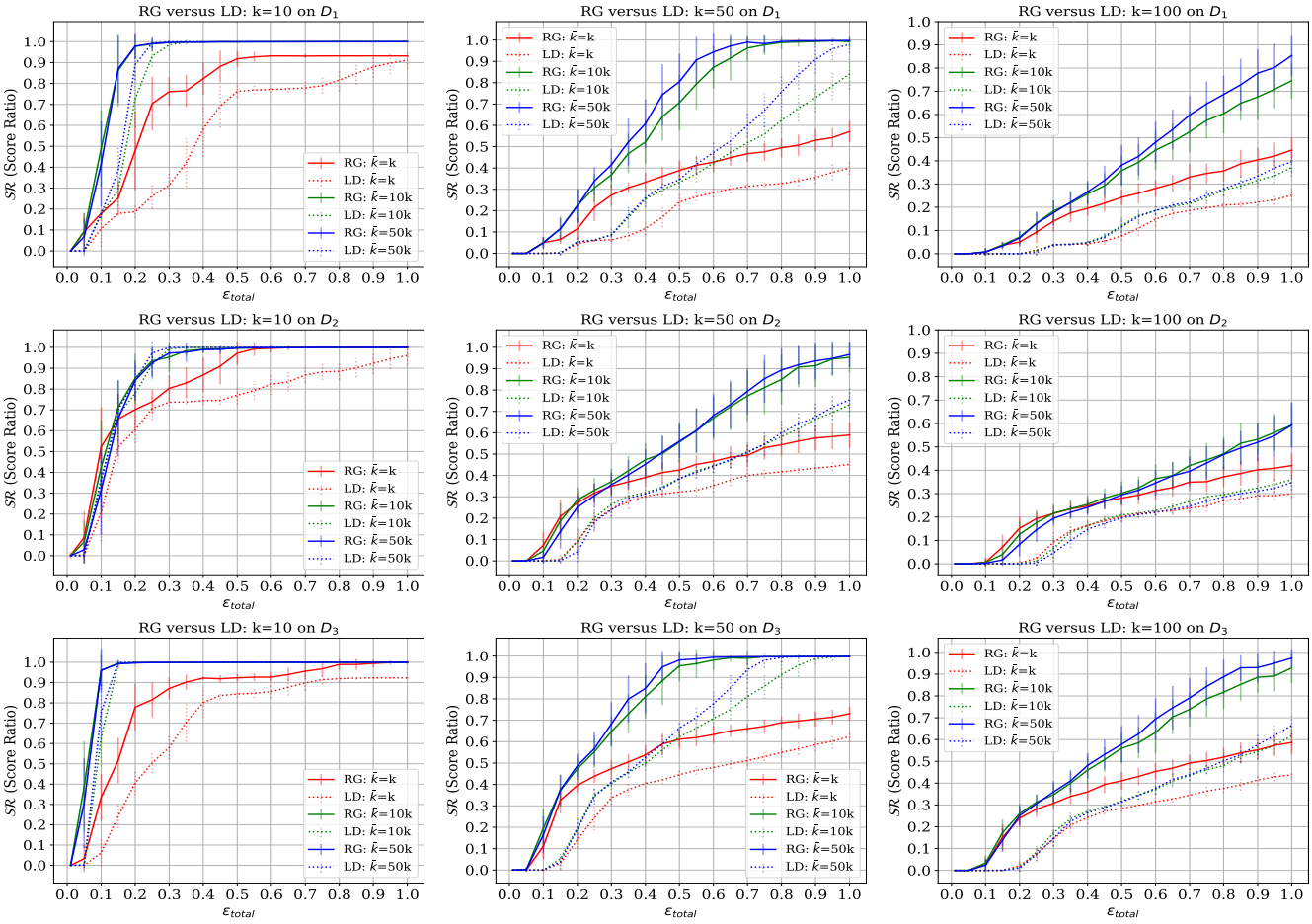

Figure 5: Metric $\mathcal{SR}$, average on 100 independent trials, for mechanisms RG and LD and various k, $\bar{k}$ and ε_{total} and three datasets. Each row compares the mechanisms on a different dataset. Our mechanism RG consistently outperforms LD.

Finally, in Appendix D we also compare these two mechanisms regarding *utility* for very large domains, and note that the utility of traditional top-k methods tends to deteriorate with increasing domain size, while restricted domain mechanisms are not affected. This shows that in domains with a huge number of elements, the use of restricted domain mechanisms can be beneficial to both performance and utility.

9 CONCLUSION

We proposed a framework for *practical* differentially private top-k selection, that can be easily adopted by practitioners applying DP in real-world systems. Our framework works on restricted domain, only needing the top-$\bar{k}$ elements returned by a database system. More importantly, it is useful to make traditional DP selection mechanisms work on restricted domain. The framework adapts any given traditional DP top-k mechanism to restricted domain. The main novelty is in the *mechanism-agnostic* approach, i.e. our framework treats the given traditional DP mechanism as a blackbox when selecting elements from the restricted domain while keeping DP on the full domain. This approach enables converting existing and future DP selection mechanisms into restricted domain mechanisms. Finally, we also showed our framework with improved utility compared to previous work, both formally and empirically.

Table 2: Average execution time of top-k selection in milliseconds. Each value is computed as the mean time for ε_{total} from 0.1 to 1.0 with 50 trials on each individual setting. RG is up to 200 times more computationally efficient than EM.

	k	EM	Our Framework (RG)		
			$\bar{k} = k$	$\bar{k} = 10k$	$\bar{k} = 50k$
D_1	10	41.94	1.91	1.87	1.93
	50	36.66	4.57	4.21	4.40
	100	38.41	3.61	3.99	4.82
D_2	10	29.48	4.73	4.71	4.24
	50	31.17	5.14	5.18	5.87
	100	30.39	4.64	4.93	5.74
D_3	10	475.94	2.11	2.37	2.29
	50	466.73	4.11	4.26	4.82
	100	481.90	3.55	3.73	4.66

REFERENCES

[1] Mitali Bafna and Jonathan Ullman. The price of selection in differential privacy. In Satyen Kale and Ohad Shamir, editors, *CLT*, volume 65 of *Proceedings of Machine Learning Research*, pages 151–168, Amsterdam, Netherlands, 07–10 Jul 2017. PMLR.

[2] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising, 2018.

[3] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. Practical locally private heavy hitters. *Journal of Machine Learning Research*, 21(16):1–42, 2020.

[4] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 127–135, New York, NY, USA, 2015. Association for Computing Machinery.

[5] Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. Discovering frequent patterns in sensitive data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 503–512, 2010.

[6] Adrian Rivera Cardoso and Ryan Rogers. Differentially private histograms under continual observation: Streaming selection into the unknown. *arXiv preprint arXiv:2103.16787*, 2021.

[7] Ricardo Silva Carvalho, Ke Wang, Lovedeep Gondara, and Chunyan Miao. Differentially private top-k selection via stability on unknown domain. In *Conference on Uncertainty in Artificial Intelligence*, pages 1109–1118. PMLR, 2020.

[8] Ricardo Silva Carvalho, Ke Wang, and Lovedeep Singh Gondara. Incorporating item frequency for differentially private set union. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9504–9511, 2022.

[9] Kamalika Chaudhuri, Daniel J Hsu, and Shuang Song. The large margin mechanism for differentially private maximization. In *Neur IPS*, pages 1287–1295, 2014.

[10] Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[11] Budhaditya Deb, Peter Bailey, and Milad Shokouhi. Diversifying reply suggestions using a matching-conditional variational autoencoder. *arXiv preprint arXiv:1903.10630*, 2019.

[12] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Neur IPS*, pages 3571–3580, 2017.

[13] Jinshuo Dong, David Durfee, and Ryan Rogers. Optimal differential privacy composition for exponential mechanisms and the cost of adaptivity, 2019.

[14] David Durfee and Ryan M Rogers. Practical differentially private top-k selection with pay-what-you-get composition. In *NeurIPS*, pages 3527–3537, 2019. Available at https://arxiv.org/pdf/1905.04273.pdf, version 2.

[15] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[16] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 2014.

[17] Cynthia Dwork, Weijie Su, and Li Zhang. Private false discovery rate control. *arXiv preprint arXiv:1511.03803*, 2015.

[18] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. ACM, 2014.

[19] Oluwaseyi Feyisetan, Borja Balle, Thomas Drake, and Tom Diethe. Privacy-and utility-preserving textual analysis via calibrated multivariate perturbations. In *WSDM*, pages 178–186, 2020.

[20] Sivakanth Gopi, Pankaj Gulhane, Janardhan Kulkarni, Judy Hanwen Shen, Milad Shokouhi, and Sergey Yekhanin. Differentially private set union. *International Conference on Machine Learning (ICML)*, 2020.

[21] Ihab F Ilyas, George Beskales, and Mohamed A Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*, 40(4):11, 2008.

[22] Murat Kantarcioğlu, Jiashun Jin, and Chris Clifton. When do data mining results violate privacy? In *SIGKDD*, pages 599–604. ACM, 2004.

[23] Krishnaram Kenthapadi and Thanh TL Tran. Pripearl: A framework for privacy-preserving analytics and reporting at linkedin. In *CIKM*, pages 2183–2191. ACM, 2018.

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[25] Jaewoo Lee and Christopher 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*, pages 931–940, 2014.

[26] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *VLDB*, 10(6):637–648, 2017.

[27] Ryan McKenna and Daniel R Sheldon. Permute-and-flip: A new mechanism for differentially private selection. *Advances in Neural Information Processing Systems*, 33, 2020.

[28] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, volume 7, pages 94–103, 2007.

[29] Gang Qiao, Weijie Su, and Li Zhang. Oneshot differentially private top-k selection. In *International Conference on Machine Learning*, pages 8672–8681. PMLR, 2021.

[30] Shaohuai Shi, Qiang Wang, Kaiyong Zhao, Zhenheng Tang, Yuxin Wang, Xiang Huang, and Xiaowen Chu. A distributed synchronous sgd algorithm with global top-k sparsification for low bandwidth networks. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019.

[31] Giuseppe Vietri, Grace Tian, Mark Bun, Thomas Steinke, and Zhiwei Steven Wu. New oracle-efficient algorithms for private synthetic data release. *arXiv preprint arXiv:2007.05453*, 2020.

[32] Benjamin Weggenmann and Florian Kerschbaum. Syntf: Synthetic and differentially private term frequency vectors for privacy-preserving text mining. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 305–314, 2018.

A INTUITION OF FIRST PART OF PROOF OF LEMMA 1

In the fist part of Lemma 1 we prove a result[9] on outputting O^S for the **same** dataset D on different indices of restricted domain elements:

$$\Pr[\mathcal{M}_f^k(D, \mathcal{I}_D^{\bar{k}}) \in O^S] \leq \Pr[\mathcal{M}_f^k(D, \mathcal{I}_{D'}^{\bar{k}}) \in O^S] \tag{14}$$

For this we let $\mathcal{I}^S = \mathcal{I}_D^{\bar{k}} \cap \mathcal{I}_{D'}^{\bar{k}}$, i.e. the indices of the elements belonging to both restricted domains of neighboring datasets D and D'.

To understand this intuition better, please see an example of a histogram of scores in Figure 6 below.

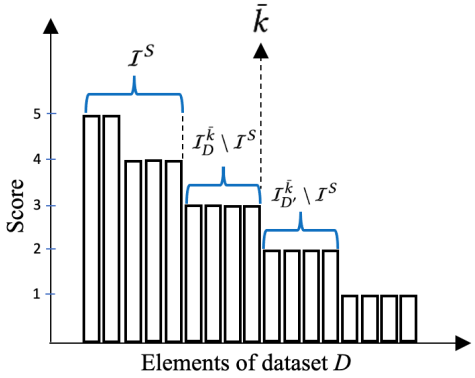

Figure 6: Intuition on sets of elements of the same dataset D with indices defined by restricted domains.

Although the histogram above is a generic example, the order shown remains in any scenario of restricted domains for any given D. Basically the elements in $\mathcal{I}^S$ have scores greater or equal than the scores of elements in $\mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}^S$, which have scores greater or equal than the scores of elements in $\mathcal{I}_{D'}^{\bar{k}} \setminus \mathcal{I}^S$. This happens by definition, as $\mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}^S$ are inside the top-$\bar{k}$ of D and $\mathcal{I}_{D'}^{\bar{k}} \setminus \mathcal{I}^S$ are outside of the same D.

Now in the left hand side of Equation (14), we can select elements in $\mathcal{I}^S$ or $\mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}^S$, whereas in the right hand side we can select elements in $\mathcal{I}^S$ or $\mathcal{I}_{D'}^{\bar{k}} \setminus \mathcal{I}^S$. Therefore, since $\mathcal{I}_D^{\bar{k}} \setminus \mathcal{I}^S$ have scores greater or equal than the scores of elements in $\mathcal{I}_{D'}^{\bar{k}} \setminus \mathcal{I}^S$, it is easier (i.e. higher probability) to select elements in $\mathcal{I}^S$ in the right hand side (as they are competing against elements with smaller scores when compared to the left hand side). For this reason, and since outcomes O^S only have elements in $\mathcal{I}^S$, the result of Equation (14) holds true.

B TOP-k WITH GUMBEL NOISE

The Top-k with Gumbel Noise (GK) mechanism [14] is presented in Algorithm 4. It works by basically adding Gumbel noise to the scores of the elements and returning the top-k elements with largest noisy scores.

[9] Note that in the main text, this result was reference as Equation (1).

Algorithm 4 $GK(D, k, \bar{k}, \varepsilon)$ – Top-k with Gumbel Noise (GK) for top-k selection [14]

Input: Parameters $k, \bar{k}$, ε, dataset D, monotonic function $f(\cdot, D)$ of sensitivity 1
Output: Ordered set of indices.

1: **for** $j \in [d]$ **do**
2: Set $v_j = f(j, D) + \texttt{Gumbel}(1/\varepsilon)$
3: Sort $\{v_j\}$ and let $v_{i_{(1)}}, ..., v_{i_{(d)}}$ be the sorted noisy results.
4: Return $\{i_{(1)}, ..., i_{(k)}\}$

GK is $(\varepsilon_M, \delta_M)$-DP, where using the current most optimal composition for GK [13], we have ε_M as:

$$\min\left(k \cdot \varepsilon, \quad k\left(\frac{\varepsilon}{1-e^{-\varepsilon}} - 1 - \ln\left(\frac{\varepsilon}{1-e^{-\varepsilon}}\right)\right) + \varepsilon\sqrt{\frac{k}{2}\ln\frac{1}{\delta_M}}\right) \tag{15}$$

Recall that GK is used in Section 7 to define our novel restricted domain mechanism `RestrictedGumbel` (RG), by instantiating $\mathcal{M}_f^k$ in Algorithm 1 with Algorithm 4.

C DERIVE PRIVACY BUDGET PER ITERATION

Below we describe the Algorithm 5, denoted as **Ind**, that derives the ε of individual iterations retroactively given a value of ε_{total}, which comes from the existing composition in [13], as seen on Equation (12) in the main paper. This algorithm was used in the experiments and mentioned in Section 8.2.

Algorithm 5 Calculate the Individual Privacy Budget per Iteration given by Composition from [13]: **Ind**$(\varepsilon_{comp}, k, \delta_{comp})$

Input: Total number of iterations: k, Privacy budget for the composition: ε_{comp} and δ_{comp}.
Output: ε

1: $\varepsilon = \varepsilon_{comp}/k$, $t_{min} = 0$, $p = 0.0001$ (precision)
2: **while** $t_{min} \leq \varepsilon_{comp}$ **do**
3: $\varepsilon = \varepsilon + p$
4: $t_1 = k\varepsilon$
5: $t_2 = k\left(\frac{\varepsilon}{1-e^{-\varepsilon}} - 1 - \ln\left(\frac{\varepsilon}{1-e^{-\varepsilon}}\right)\right) + \varepsilon\sqrt{\frac{k}{2}\ln\left(1/\delta_{comp}\right)}$
6: $t_{min} = \min(t_1, t_2)$
7: Return $\varepsilon - p$

D EXPERIMENTS COMPARING UTILITY OF RG VERSUS A TRADITIONAL FULL DOMAIN MECHANISM

Traditional DP algorithms for top-k selection, such as the Exponential Mechanism [28] and Report Noisy Max [16], have to query the scores of all of the elements in the domain, and then select from the full domain. Generally, these methods first add noise to all counts, sort the noisy values and report the top-k. For large datasets, selecting elements from the full domain becomes impractical, as that can lead to degraded performance for real-time systems, as discussed in Section 8.5. Moreover, in the case of datasets with very large domain, using traditional mechanisms to select elements from the

full domain also impacts negatively the utility, as the long tail of a score distribution with increasing domain size reduces the chance of selecting high quality elements [9].

To evaluate the impact of the domain size, we add multiple new elements with the same small total score of 1. Starting from D_3, which is the dataset with the largest domain size we analyzed, we included from 10^8 (100M) to 10^{10} (10B) new elements. Note that increasing the domain size will only affect EM, as it selects k elements from the full domain. Adding elements with small counts will *not* impact RG, as they will not be part of the restricted domain of the top-$\bar{k}$ elements which RG selects from. For this reason Figure 7 only has one curve for RG.

Figure 7 shows "Score Ratio" $\mathcal{SR}$ (defined in Section 8.3) for RG with $\bar{k} = 50 \cdot k$ and EM for varying number of elements added to D_3. We note that, as expected, in all settings increasing the domain size decreases the utility of EM. For $|D_3| = 100M$, we already have RG with better results than EM. Moreover, for $k = 50$ and $\varepsilon_{total} = 0.4$, the largest domain size used $|D_3| = 10B$ shows EM with a significant lower $\mathcal{SR}$ compared to RG. In this setting, RG has a $\mathcal{SR}$ almost 0.3 larger than that of EM, which represents RG with an absolute improvement of almost 30% in the quality of the scores returned when compared to EM. Finally, it is also important to note that, when asked for top-k, EM always returns k elements and RG may return less than k. So in practice, we see the results on Figure 7 as RG returning more of the clear high quality elements, whereas EM returns some good elements and others with considerably lower score.

E FIXED Δ: RESTRICTEDNORMAL (RN)

When creating new restricted domain mechanism, our main scenario is the general case of an unknown Δ, which was used to derive RG above. Nonetheless, our framework can also be used for known Δ. While mechanisms that consider Δ to be unknown have privacy loss scaling with k, as seen on Equation (12), the main benefit of having a mechanism on restricted domain for fixed known Δ is that we can take advantage of the value of Δ. For example, with $\Delta < k$ we can derive a mechanism with privacy loss scaling with a value smaller than k.

In this scenario, we consider the Gaussian mechanism [2] as shown below in Theorem 7, which consists of two steps: (1) Add noise from Normal distribution with standard deviation defined as $\sigma = \mathcal{G}(\varepsilon, \delta, \sqrt{\Delta})$[1] to the scores of the elements in the defined domain of selection, and (2) return elements with respective noisy scores. For top-k selection, we just output the indices of the k elements with largest noisy scores.

Theorem 7 (Gaussian Mechanism, [2]). *For ε, $\delta_M \in (0, 1)$ and $\sigma = \mathcal{G}(\varepsilon, \delta_M, \sqrt{\Delta})$, the Gaussian Mechanism for top-k selection $\mathcal{N}^k$ adds noise from $\mathcal{N}(0, \sigma^2)$ to each $f(i, D)$ for $i \in \mathcal{I}$ and takes the k indices with the largest noisy scores. For $\Delta_2 = \max \|f(D) - f(D')\|_2$ over all neighboring datasets D, D', we have that $\mathcal{N}^k$ is $(\Delta_2\varepsilon, \delta_M)$-DP.*

We will denote as `RestrictedNormal` (RN) the mechanism created by using our framework for restricted domain on Algorithm 1 with mechanism $\mathcal{M}_f^k$ on second step consisting of the above top-k mechanism using noise from the Normal distribution.

[1] σ is determined by an optimal noise calibration given by [2]. The result is a function of ε, δ and the L2 sensitivity of the score function used which, from our Δ definition, is equivalent to $\sqrt{\Delta}$.

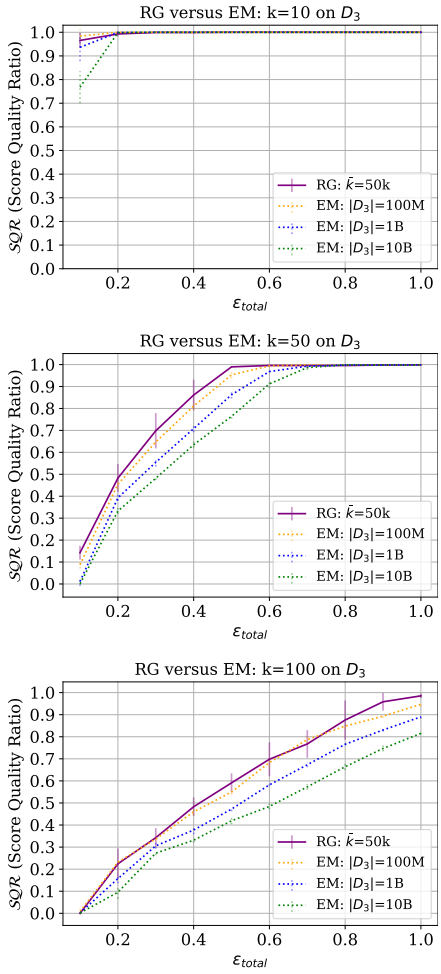

Figure 7: Comparison of our restricted domain mechanism RG with the traditional full domain Exponential Mechanism (EM). We add elements with count 1 to the domain of dataset D_3 to reflect various sizes. Each result is the mean of 5 independent trials. For datasets with domain size $\geq 100M$, we already see RG with better utility than EM in terms of $\mathcal{SR}$.

Theorem 8 (RN). *For any $\delta_M > 0$, the* `RestrictedNormal` *(RN) mechanism with noise from Normal distribution with standard deviation $\sigma = \mathcal{G}(\varepsilon, \delta_M, \sqrt{\Delta})$[1] satisfies $(\sqrt{\Delta} \cdot \varepsilon + \varepsilon_R, \delta_M + \delta_R)$-DP.*

Proof. Follows by using Theorem 2 with $\mathcal{M}_f^k$ defined as the $(\sqrt{\Delta}\varepsilon, \delta_M)$-DP noisy top-$k$ mechanism with Gaussian Noise [2]. □

The result above shows that the privacy loss of RN scales with $\sqrt{\Delta}$. In contrast, previous work [14] created a mechanism for known Δ using Laplace noise, showing privacy loss scaling with Δ. Below we include experimental results comparing RN to the corresponding algorithm for fixed Δ from [14].

E.1 Experiments for Fixed Δ

Working similarly to LD, [14] also considered alternatively adding Laplace noise instead of Gumbel, to obtain `LaplaceLimitDomain` (LL), working with a given known Δ. LL has its privacy loss scaling with Δ instead of $\sqrt{k}$, thus considerably improving over LD for scenarios where $\Delta < k$. More specifically, regarding privacy guarantees, `LaplaceLimitDomain` is $((\Delta+1)\varepsilon, \bar{\delta})$-DP with $\bar{\delta} = (\delta/4)(3+\ln(\Delta/\delta))$.

So here, for **known** Δ and single call, we compare the existing `LaplaceLimitDomain` (LL) [14] to our `RestrictedNormal` (RN) described in Appendix E.

For a fair comparison we fix every mechanism with a overall $(\varepsilon_{total}, \delta_{total})$-DP guarantee, for $\delta_{total} = 1/(2n)$, where n is the number of users on Table 1, and ε_{total} varies from 0.01 to 1.00. We show average results of 100 independent trials for k equal to 10 (small), 50 (medium) and 100 (large). For each k we run the mechanisms with $\bar{k}$ equal to k (small), $10k$ (medium) and $50k$ (large).

We use a fixed **known** $\Delta = 8$ the individual settings for the mechanisms are as follows, to make sure both have the same overall guarantee of $(\varepsilon_{total}, \delta_{total})$-DP:

- **LL**: We use $\varepsilon = \varepsilon_{total}/\Delta$ and δ such that $(e^{\varepsilon_{total}}+1)(\delta/4)(3+\ln(\Delta/\delta))$ is at most δ_{total}.
- **RN**: We use $\delta = \delta_U = \delta_{total}/2$, $\varepsilon_T = 4\cdot\varepsilon_{total}/\Delta$ and $\varepsilon = \varepsilon_{total} - \varepsilon_T$ in order to set $\sigma = \mathcal{F}(\varepsilon, \delta, \sqrt{\Delta})$[10]

The results are showed in Figure 8. Similarly to the unknown Δ scenario, the experiments show that our mechanism RN consistently outperforms the previous mechanism LL.

F EXPERIMENTS COMPARING RG VERSUS TS

Here we compare the recent Top-Stable (TS) mechanism [7] with our mechanism RG created from the framework in Algorithm 1. TS works by searching for *stability* on a given domain to allow an unordered top-k selection.

We note that our framework in Algorithm 1 can be seen as looking for a certain form of *stability*, sort of similar to TS. Nonetheless, our mechanisms test the elements in the (noisy) descending order, whereas TS tests in the ascending order. Moreover, regarding the privacy analysis, TS assumes tests are false and stops when it succeeds, while our framework assumes tests are true and stops when it fails. Finally, in contrast with TS, our threshold formulation does not include a $\bar{k}$ factor. These differences are essential to make our algorithms present better utility, as we see in the experiments below.

For the experiments, we use the "number of elements returned" as metric and show the results for the three datasets introduced in Section 8. For TS we use the official code from the authors, which is publicly available[11]. The privacy parameters are fixed the same way as in Section 8.

The results are showed in Figure 9. Similarly to the previous experiments, results show that our mechanism RG consistently outperforms TS.

[10] σ is determined by an optimal noise calibration given by [2]. The result is a function of ε, δ and the L2 sensitivity of the score function used which, from our Δ definition, is equivalent to $\sqrt{\Delta}$.

[11] Code for Top-Stable (TS) mechanism available at https://github.com/ricardocarvalhods/diff-priv-top-k-stability/

Furthermore, our experiments show that TS usually does not benefit from an increase in $\bar{k}$, which is reasonable as their threshold formulation includes $\bar{k}$, resulting in worse utility as their stability tests become increasingly harder to satisfy. Moreover we see that as k increases, the results of TS become closer to RG's, which is aligned with the findings of [7] for larger k. We also see a "stable range" as defined in [7], for $k = 10$ and D_3, where the results of TS are slightly better than those of RG for $\bar{k} = k$ – however RG's results are still better overall when looking at the three settings analyzed.

G EXPERIMENTS COMPARING F1 SCORE OF RG VERSUS LD

Here we compare the mechanism RG derived from our framework in Algorithm 1 with LD [14] for a metric different from those in Section 8: the F1 score.

The F1 score uses the following three basic definitions:

- TP = True Positives, i.e. elements returned by a mechanism that are among the true top-k elements;
- FP = False Positives, i.e. elements returned by a mechanism that are **not** among the true top-k elements;
- FN = False Negatives, i.e. elements that are among the true top-k elements but were **not** returned by a mechanism.

Now we can define F1 score as:

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{16}$$

Figure 10 below shows the results for the experiments with F1. The privacy parameters were fixed the same way as in Section 8.

Similarly to the experiments with other metrics, the results with F1 show RG consistently outperforming LD. In fact, the plots in Figure 10 are very similar to those seen in Section 8, with just larger variation for smaller values of ε_{total}.

Note that, as discussed in Section 6, with high probability our mechanisms tend to minimize false negatives (see Theorem 4) and also minimize false positives (since we select only from the top-$\bar{k}$ elements), which results in higher values of the F1 score.

H USE OF RESTRICTED DOMAIN MECHANISMS ON DIFFERENT DATA DISTRIBUTIONS

Just like any differentially private mechanism, the use of Algorithm 1 can result in different privacy-utility trade-offs depending on the dataset D on which it is applied. However, we must recall that differential privacy is a property of an algorithm, not a dataset. That being said, if without looking at the data we have an idea of the distribution of the scores (due to e.g. domain knowledge), we are able to assess, without executing any algorithm, if using restricted domain mechanisms is a good idea or not, depending on the goal and privacy budget. Although we cannot cover all possible situations, in this section we aim to give a few examples and the rationale of choices.

Highly skewed score distribution. The case of scores satisfying a highly skewed distribution, e.g. Power law, is the one where the use of restricted domain mechanisms is the most recommended.

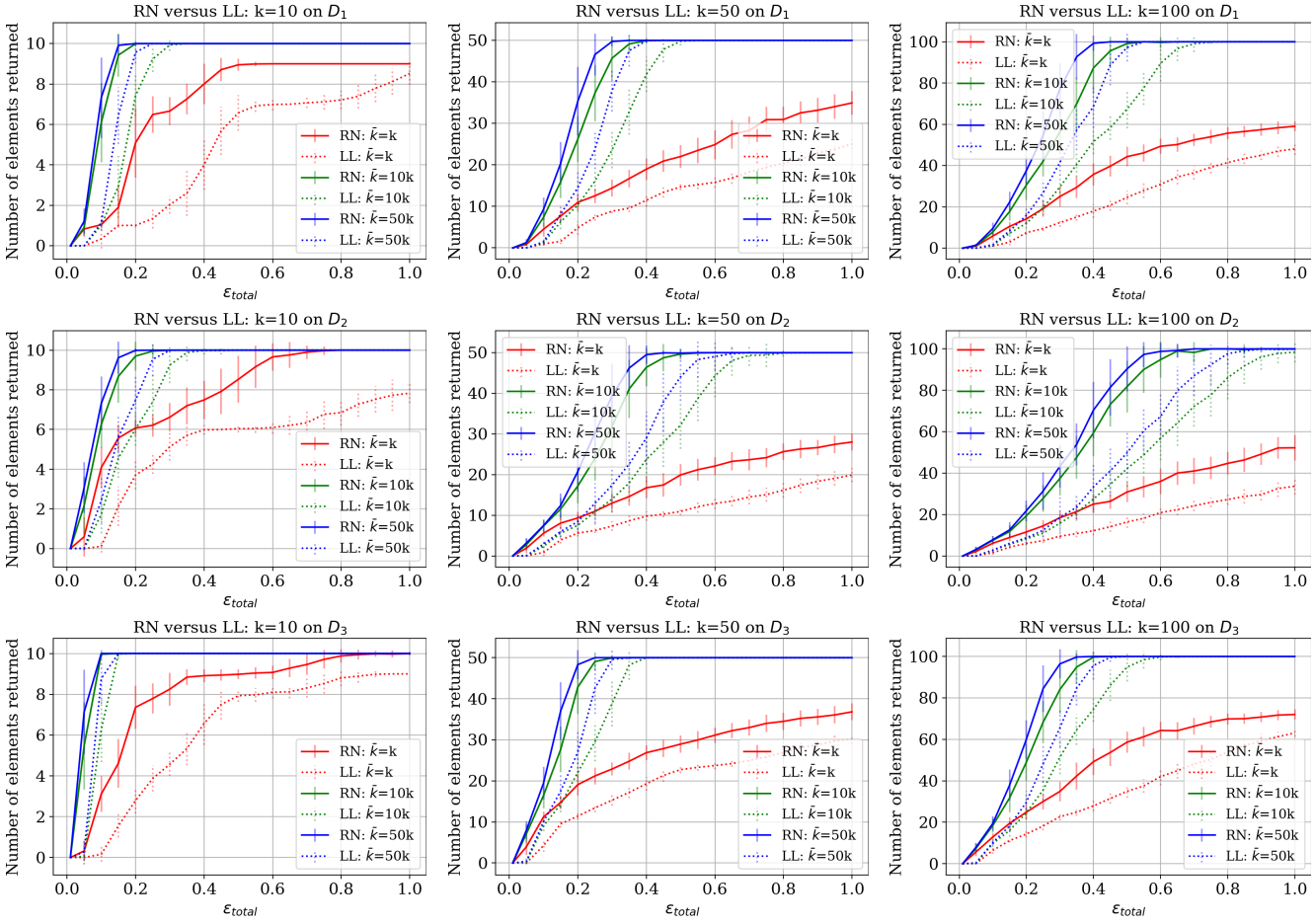

Figure 8: Number of elements returned (on 100 independent trials) for restricted domain mechanisms RN and LL and various k, $\bar{k}$ and ε_{total} and three datasets. Each row compares mechanisms on a different dataset. Our mechanism RN has consistently better utility than the previous work LL on every scenario.

That is because these distributions tend to have larger gaps between top elements and another more distant element such as the top-$\bar{k}+1$. Since this is the distance checked for example in line 8 of Algorithm 1, if this gap is large, we tend to have more elements outputted. Finally, is is important to note that we would not be able to eliminate altogether these tests in line 8 even if the data has large gaps. That is because without looking at the data we do not know the existence of these gaps, and looking at the data will consume privacy budget.

Flatter score distribution. If we follow the arguments above for the highly skewed distribution, it is easy to understand that flat distributions, e.g. Uniform, will make it harder to output elements when using restricted domain mechanisms. However, outputting sets of size much smaller than k does not mean that most elements outputted will be unstable. It is possible to output elements even when all have the same score. In the end, if we want to output more elements, for this case it is generally better to use traditional top-k mechanisms, as these are guaranteed to return k elements. On the other hand, restricted domain mechanisms would be recommended if we agree on outputting fewer elements but that have higher chance of having a high score. Another case for restricted mechanisms is the scenario where performance/running time matters a lot. Additionally, note that, to output more elements, increasing the privacy budget is always another option.

Finally, note that these scenarios do not change the privacy guarantees of restricted domain mechanisms. In other words, the use of a certain dataset will not change the privacy loss of a mechanism, as this is formally defined by differential privacy. In summary, differential privacy will control trade-offs, with the goal of avoiding inferring any characteristic specific to one individual with high probability, independent of the dataset.

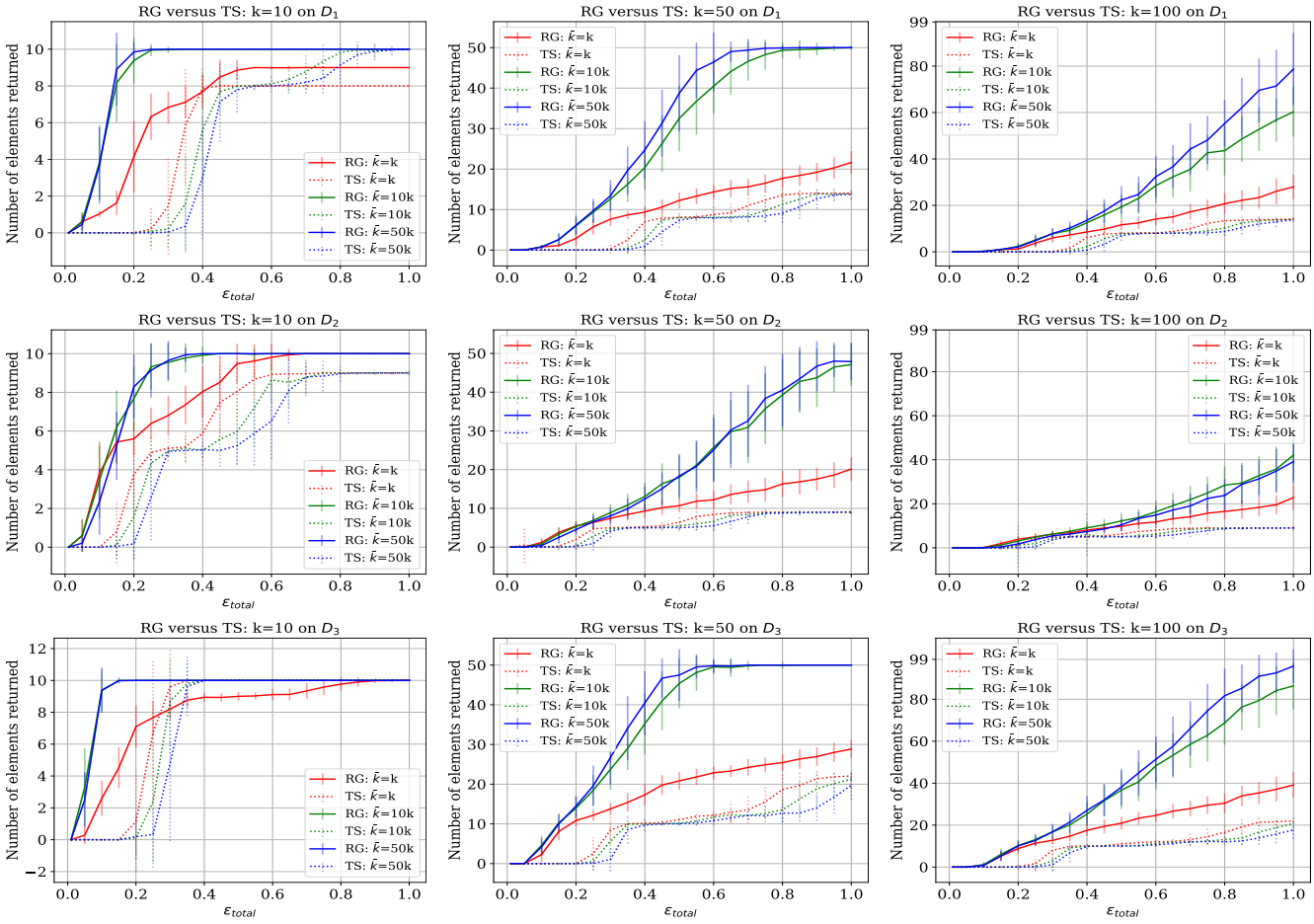

Figure 9: Number of elements returned (on 100 independent trials) for restricted domain mechanisms RG and TS[7] and various k, $\bar{k}$ and ε_{total} and three datasets. Each row compares mechanisms on a different dataset. Our mechanism RG has consistently better utility than the previous work TS.

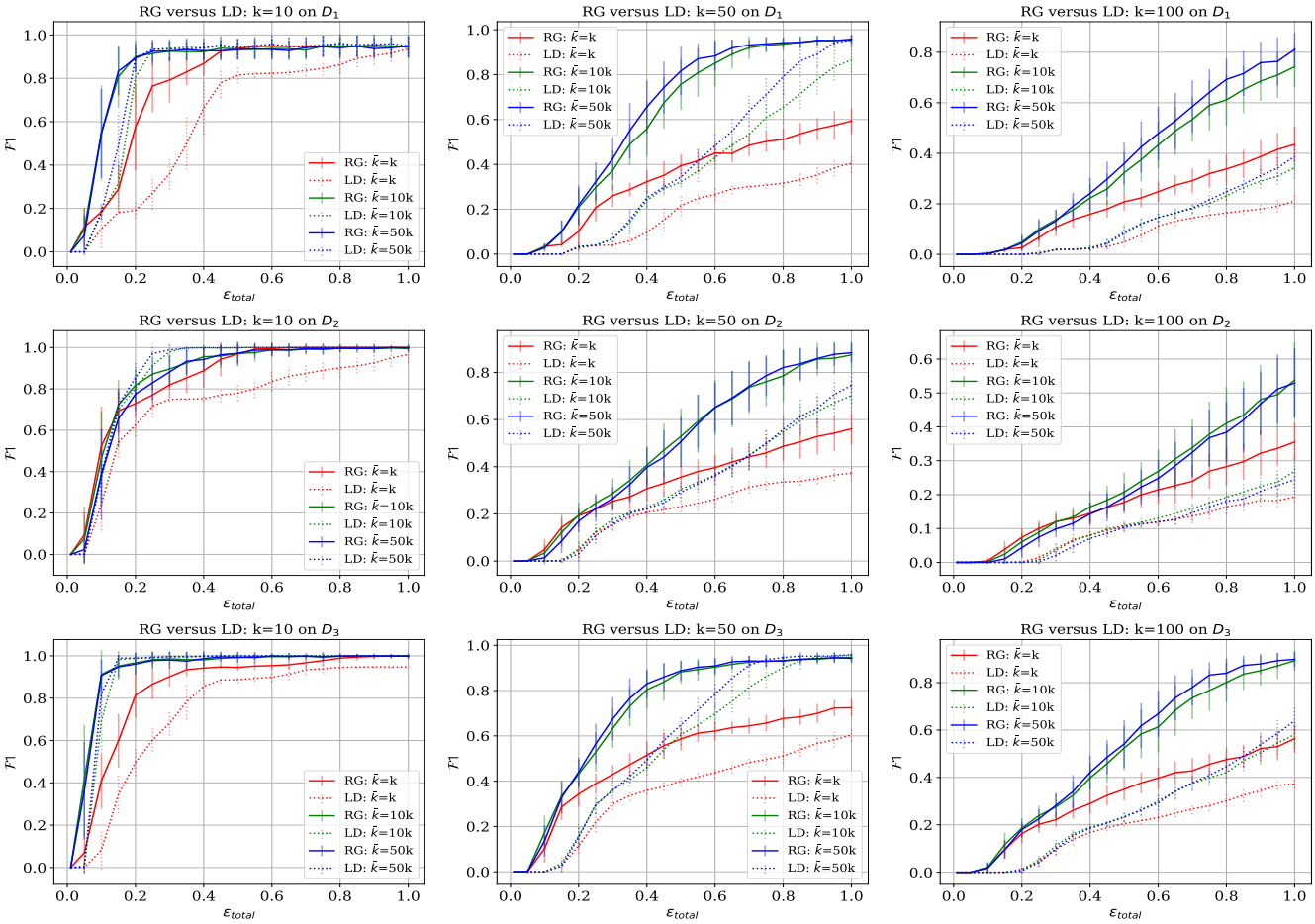

Figure 10: Average F1 score (over 100 independent trials) for restricted domain mechanisms RG and LD and various k, $\bar{k}$ and ε_{total} and three datasets. Each row compares mechanisms on a different dataset. Our mechanism RG has consistently better utility than the previous work LD.