A Joint Exponential Mechanism for Differentially Private Top-k Set

Jenny Gillenwater
Google New York

Matthew Joseph
Google New York

Andrés Muñoz Medina
Google New York

Mónica Ribero
UT Austin*

Abstract

We present a novel differentially private algorithm for releasing the set of k elements with the highest counts from a data domain of d elements. We define a "joint" instance of the exponential mechanism (EM) whose output space consists of all $O(d^k)$ size-k subsets; yet, we are able to show how to sample from this EM in only time $\tilde{O}(dk^3)$. Experiments suggest that this joint approach can yield utility improvements over the existing state of the art for small problem sizes.

1 Introduction

Top-k selection is the problem of identifying the k elements with the highest counts from a data domain of d elements. This is a basic data analysis question that arises in machine learning tasks such as recommender systems, basket market analysis, and language learning tasks. In an effort to solve these problems while guaranteeing privacy to the individuals contributing data, several works have studied top-k under the additional constraint of differential privacy (DP) [5].

One popular approach to DP top-k selection uses "peeling" mechanisms [2, 4], which adaptively applies the exponential mechanism (EM) k times to repeatedly "peel" off and remove from consideration the highest-count element. Another approach [12] adds Laplace noise to each element count and outputs the elements with the k highest noisy counts. These algorithms each admit an efficient implementation in time $O(d)$ and also possess the advantage of releasing an ordered sequence (not set) of top-k items. We use them as baselines; more details appear in Section 4.

We depart from these approaches by applying an EM that chooses directly from sets of k items. Past work has used this style of "joint EM" to privately estimate multiple quantiles [7] and output password frequency lists [3], but it is not obvious how to directly extend either to the top-k problem. As with previous work on joint EMs, our main contribution is showing that it is possible to sample exactly from the mechanism in polynomial time, even though the size of the output space is exponential (Theorem 8). We find that our joint approach can offer significant accuracy improvements (Section 4).

2 Preliminaries

We start by formally describing the top-k problem before moving on to general privacy preliminaries. In keeping with past work on private top-k [4], we use an ℓ_∞ error metric.

Definition 1. *Let $\mathcal{X}$ be a data domain of d items. In an instance of the top-k problem, there are n users and each user i has an associated vector $x_i \in \{0,1\}^d$. Given $D = \{x_i\}_{i=1}^n$, let $c_j = \sum_{i=1}^n x_{i,j}$ denote the count of item $j \in [d]$, and let D_c denote the (decreasing) vector of the top k counts. The goal is to output a set of k items* $\text{TOP-K}(D) = \arg\min_{S \subset [d]^k} \|S_c - D_c\|_\infty$.

Note that in this setting each user contributes at most one to each item count, but may contribute to up to arbitrarily many items. In general, we describe database D by the vector of counts for its

*Part of this work done while an intern at Google New York.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

domain, $D = (c_1, \ldots, c_d)$. We also assume that the counts are distinct; a small amount of noise can always be added to achieve this. Finally, note that our output is an unordered set, not a sequence. Extending to sequences is a goal for future work. We now cover necessary privacy prerequisites.

Definition 2 ([5]). *Two data sets $D, D' \in \mathcal{X}^*$ are neighbors if D' can be obtained from D by adding or removing some data point x. We define $\mathcal{N}(D)$ as the set of neighboring data sets for D. Mechanism $M \colon \mathcal{X}^* \to Y$ is ε-differentially private (DP) if, for any two neighboring datasets D and D', and any $S \subseteq Y$, it holds that $P(M(D) \in S) \leq e^{\varepsilon} P(M(D') \in S)$.*

The exponential mechanism (EM) is a general-purpose DP algorithm for selecting a "good" output given some utility function over outputs.

Definition 3 ([10, 6]). *Given utility function $u \colon \mathcal{X}^* \times O \to \mathbb{R}$ with ℓ_1 sensitivity $\Delta_u = \max_{X \sim X', o \in O} |u(X, o) - u(X', o)|$, the exponential mechanism (EM) M has output distribution*

$$P_M(M(X) = o) \propto \exp\left(\varepsilon u(X, o)/(2\Delta_u)\right),$$

where $\propto$ elides the normalization factor. This mechanism is ε-DP.

One cost of this generality is that the EM definition provides no guidance for actually sampling from the mechanism. As subsequent sections demonstrate, this can be the main obstacle to applying it.

3 A Joint Exponential Mechanism for Top-k

Our mechanism employs a utility function u^* measuring how far a candidate collection of k items is from being the true answer. This requires the following notion of distance between databases:

Definition 4. *Given databases $D, D' \in \mathcal{X}^*$, let $d(D, D')$ denote the smallest integer m such that there is a sequence $D_0 = D, D_1, D_2, \ldots, D_m = D'$ where $D_1 \in \mathcal{N}(D_0), \ldots, D' \in \mathcal{N}(D_{m-1})$.*

For candidate set of items $S \subseteq [d]$ such that $|S| = k$, our utility function is

$$u^*(D, S) = - \min_{D' | \textsc{top-k}(D') = S} d(D, D').$$

$u^*(S, D)$ thus penalizes candidate sets that are far from being optimal on D. This general approach of a utility based on database distances has appeared in the DP literature under several names [9, 1, 11], but it has not previously been applied to top-k. u^* has two helpful properties: it is low-sensitivity and is easy to compute. Let the true top-k counts be denoted $c_1 > \ldots > c_k$, and consider a set S with sorted counts $c_{j_1} > \ldots > c_{j_k}$. Then, because each user can contribute 1 to every count, we need exactly as many users as the largest count difference to make S contain the true top-k counts.

Lemma 5. *$\Delta_{u^*} = 1$ and, for S with element counts sorted in decreasing order $c_{j_1} > \cdots > c_{j_k}$, and true top k counts $c_1 > \cdots > c_k$, we have that $u^*(S, D) = -\max_{i \in [k]} [c_i - c_{j_i}]$.*

We call this EM with utility u^* JOINT. As an instance of the EM, its ε-DP privacy guarantee is immediate from Definition 3. However, a naive implementation of JOINT requires computing $O(d^k)$ output probabilities, one for each candidate output set S. The next section sketches a much more efficient sampling algorithm.

3.1 Efficiently Sampling JOINT

Our sampling approach appears in Algorithm 1. The basic idea is to take advantage of the fact that, although there are exponentially many possible output sets, there are only $O(dk)$ possible utility values for those sets. This is because, by Lemma 5, each utility value takes the form $-(c_{i_1} - c_{i_2})$ for some $i_1 \in [k]$ and some $i_2 \in [d]$. We compute and store these utility values in matrix M (Line 1) and sample a utility value for the eventual output set in Line 3. Once a utility value is selected we select a set with the given utility uniformly at random (Line 4).

We now sketch proofs of the runtime and correctness of Algorithm 1. Line 1 takes time $O(dk)$. Once we have calculated U, Line 3 also takes time $O(dk)$. It remains to explain Lines 2 and 4. For Line 2, first observe that it suffices to compute the number of sets with utility at least $-M_{ij}$ for each M_{ij}, since we can recover the number of sets with utility exactly $-M_{ij}$ by taking differences afterward. The next step reduces this problem to counting paths through a matrix derived from M. First, define

Algorithm 1: An efficient implementation of JOINT

Input : Vector of (sorted) counts $(c_1, ..., c_d)$, number of items to return k, privacy budget ε
Output : Differentially private set of k item indices $S = \{i_1, \ldots, i_k\}$

Compute $k \times d$ matrix M where $M_{ij} = c_i - c_j$ if $c_i - c_j \geq 0, d - j \geq k - i$; $\perp$ otherwise
For each entry M_{ij} of M, compute the number U_{ij} of sets with utility $-M_{ij}$
Sample a utility value $m = -M_{ij}$ with probability $\propto U_{ij} \exp\left(\frac{-\varepsilon M_{ij}}{2}\right)$
Output a set sampled uniformly from sets with utility m

$(k+1) \times (d-k+1)$ matrix $\hat{M}$ to be M "left-aligned" to remove $\perp$s, with a row of 1s appended to the bottom. Second, define matrix A^{ij} by $A^{ij}_{\ell r} = \mathbb{1}_{\widehat{M}_{\ell r} < \widehat{M}_{ij}}$. For example, for $k = 3$ and $d = 5$,

$$M = \begin{bmatrix} c_1 - c_1 & c_1 - c_2 & c_1 - c_3 & \perp & \perp \\ \perp & c_2 - c_2 & c_2 - c_3 & c_2 - c_4 & \perp \\ \perp & \perp & c_3 - c_3 & c_3 - c_4 & c_3 - c_5 \end{bmatrix}, \qquad \widehat{M} = \begin{bmatrix} c_1 - c_1 & c_1 - c_2 & c_1 - c_3 \\ c_2 - c_2 & c_2 - c_3 & c_2 - c_4 \\ c_3 - c_3 & c_3 - c_4 & c_3 - c_5 \\ 1 & 1 & 1 \end{bmatrix},$$

and with $c_3 - c_4 < c_2 - c_3$ and $c_1 - c_2$, we have the following A^{34} matrix: $A^{34} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$.

The value of constructing A^{ij} comes from the following lemma.

Lemma 6. *The number of paths in A^{ij} from $A^{ij}_{1,1}$ to $A^{ij}_{k+1,d-k+1}$, going only right and down and avoiding 0s, is the number of sequences of decreasing counts with utility at least $-M_{ij}$.*

Proof. (Sketch): A sequence with utility $\geq -M_{ij}$ can be viewed as a path through A^{ij} that only goes right and down (since the sequence must have decreasing counts) and avoids 0s (since the sequence has utility $\geq -M_{ij}$). The column indices where the path goes down determine the indices of the chosen elements. For example, consider the path $\widehat{M}_{11}, \widehat{M}_{21}, \widehat{M}_{31}, \widehat{M}_{41}$, whose first three entries correspond to M_{11}, M_{22}, M_{33}. The column indices from these M entries indicate that we are considering the sequence with item indices $(1, 2, 3)$. Similarly, $\widehat{M}_{11}, \widehat{M}_{21}, \widehat{M}_{22}, \widehat{M}_{32}, \widehat{M}_{42}$ corresponds to entries $M_{11}, M_{22}, M_{23}, M_{34}$, so the item indices in this case are $(1, 3, 4)$. □

Denote the number of such paths through A^{ij} by $\mathcal{P}_{A^{ij}}$. We now turn to computing $\mathcal{P}_{A^{ij}}$ quickly.

Lemma 7. *Given A^{ij} constructed from $k \times d$ matrix M, we can compute $\mathcal{P}_{A^{ij}}$ in time $O(k^2 \log(k))$.*

Proof. (Sketch): We count paths through A^{ij} by splitting it into $O(k)$ submatrices with no 0s, counting the paths going from left to right in each submatrix, and then stitching these paths together to count the number of paths overall through A^{ij}. For example, given some matrix

$$A^{ij} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & 0 & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

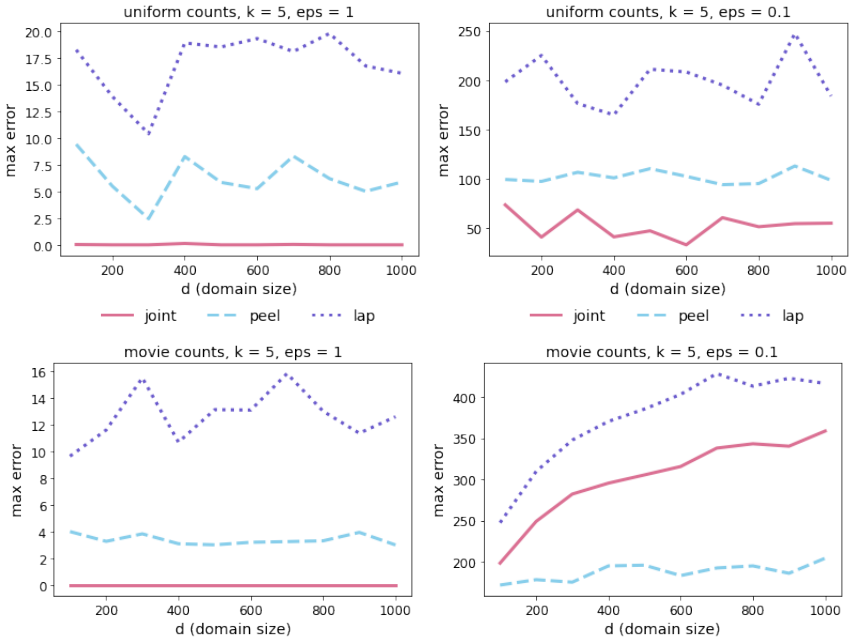
the boundary columns for the submatrices are $e_1 = [a_{23}, a_{33}, a_{43}, 1]^T$, $e_2 = [a_{35}, a_{45}, 1]^T$, $e_3 = [a_{46}, 1]^T$, and $e_4 = [1]^T$. We define a vector Q_1, of length 4, that counts the number of paths from a_{11} to elements of e_1. Then we define a matrix Q_2 of size 4×3 counts the number of paths from each element of e_1 to each element of e_2, and so on. $\mathcal{P}_{A^{ij}}$ can thus be computed as $Q_1 \cdot Q_2 \cdots Q_{O(k)}$. The elements of each Q_j come directly from Pascal's triangle, so we can compute each Q_j in time $O(k)$. Moreover, each Q_j is Toeplitz, so we can use the fast Fourier transform (FFT) to perform each matrix-vector Toeplitz multiplication in time $O(k \log(k))$. Hence, we can compute $\mathcal{P}_{A_{ij}} = Q_1 \cdot Q_2 \cdots Q_{O(k)}$ in total time $O(k^2 \log(k))$. □

By Lemmas 6 and 7, Line 2 of Algorithm 1 takes time $O(dk^3 \log(k))$. After sampling a utility value $m = M_{ij}$ in Line 3, we can use a similar counting argument in Line 4 to sample an actual set of outputs with the given utility in time $O(dk)$. We collect these guarantees in the following theorem.

Theorem 8. JOINT *is an* ε*-DP algorithm for* TOP-K *that runs in time* $O(dk^3 \log(k))$.

4 Experiments

We compare to the peeling mechanism [2, 4] (PEEL) and Laplace noise [12] (LAPLACE).[2] In all cases, we use the ε-DP variant of the algorithm. For PEEL, we use the Gumbel noise implementation: add Gumbel noise with parameter k/ε to each count (see Section 4.1 in [4]) and take the k items with the highest noisy counts. For LAPLACE, we instead add Laplace noise with parameter $2k/\varepsilon$. The first dataset is synthetic uniform data: given domain size d, we draw item counts from $U(0, 10d)$. The second dataset consists of movie ratings from the MovieLens 100k database [8]: given d, we use the rating counts of the d most-rated movies. For each dataset, we evaluate JOINT, PEEL, and LAPLACE at $\varepsilon = 1$ and $\varepsilon = 0.1$ for $k = 5$ and a range of d. Each point averages 100 trials, and we measure l_∞ error between true and estimated top k.



Overall, JOINT obtains the strongest utility. However, this advantage sometimes reverses for small ε. This is possibly because the smaller ε most heavily penalizes JOINT's extremely large output space. In more detail: Note that the baseline mechanisms are essentially outputting vectors of noisy counts in $\mathbb{R}^d$ (which we then post-process to get a top-k set). Normalizing over $\mathbb{R}^d$ can have very different behaviour than normalizing over all size-k subsets of $[d]$.

5 Future Directions

As the preliminary experiments above demonstrate, JOINT offers strong accuracy potential for top-k (set). However, we close by highlighting a nontrivial obstacle to using it: the relevant quantities are extremely large. Specifically, while each quantity in the submatrices $Q_1, \ldots, Q_{O(k)}$ described in the proof sketch of Lemma 7 can be computed in constant time, the quantities themselves can be as large as $O(d^k)$, which leads to numerical issues in practical implementations. It is not clear how to solve this by working with logarithmic quantities either, since this requires computing matrix multiplication $\mathcal{P}_{A^{ij}} = Q_1 \cdot Q_2 \cdots Q_{O(k)}$ in logspace. At the same time, we emphasize that this is a numerical issue, not an algorithmic one. Resolving it is nonetheless necessary for applying JOINT to large databases.

[2] We also experimented with the Gamma mechanism [13], but it achieved poor error.

References

[1] Hilal Asi and John C Duchi. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. Neural Information Processing Systems (NeurIPS), 2020.

[2] Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. Discovering frequent patterns in sensitive data. In Knowledge Discovery and Data Mining (KDD), 2010.

[3] Jeremiah Blocki, Anupam Datta, and Joseph Bonneau. Differentially private password frequency lists. In Network and Distributed System Security (NDSS), 2016.

[4] David Durfee and Ryan Rogers. Practical differentially private top-k selection with pay-what-you-get composition. In International Conference on Machine Learning (ICML), 2019.

[5] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography Conference (TCC), 2006.

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

[7] Jennifer Gillenwater, Matthew Joseph, and Alex Kulesza. Differentially private quantiles. In International Conference on Machine Learning (ICML), 2021.

[8] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. Transactions on Interactive Intelligent Systems (TiiS), 2015.

[9] Aaron Johnson and Vitaly Shmatikov. Privacy-preserving data exploration in genome-wide association studies. In Knowledge Discovery and Data Mining (KDD), 2013.

[10] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In Foundations of Computer Science (FOCS), 2007.

[11] Andrés Muñoz Medina and Jenny Gillenwater. Duff: A dataset-distance-based utility function family for the exponential mechanism. arXiv preprint arXiv:2010.04235, 2020.

[12] Gang Qiao, Weijie J Su, and Li Zhang. Oneshot differentially private top-k selection. In International Conference on Machine Learning (ICML), 2021.

[13] Thomas Steinke and Jonathan Ullman. Between pure and approximate differential privacy. Journal of Privacy and Confidentiality (JPC), 2015.