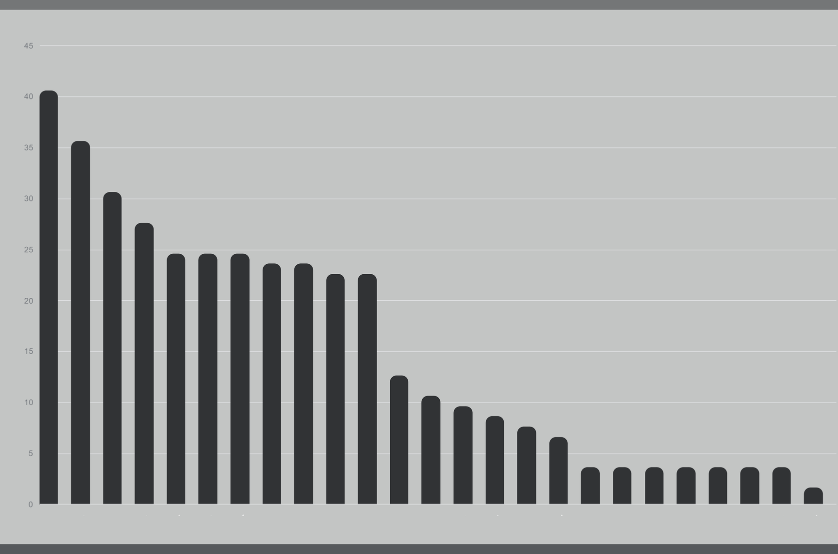
# Practical Differentially Private Top-$k$ Selection with Pay-what-you-get Composition

David Durfee, Ryan Rogers

## Task: Top-$k$

- Return the $k$ most frequent data elements from a large dataset with $d$ dimension.
- Example: What are the top-**10** most popular articles among data scientists in the Bay Area?



## Including Differential Privacy

- Data may contain sensitive information.
- The presence of a data element could be the result of a single user's data.
- Simple thresholding does not provide formal privacy guarantees.
- Use differential privacy (DP) [DMNS] to protect the individual's in the data set.
- An algorithm $M : \mathcal{X} \to \mathcal{Y}$ is $\epsilon$-DP if for all neighboring inputs $x, x'$ and any outcome sets $S \subset \mathcal{Y}$, we have

$$\Pr[M(x) \in S] \leq e^{\epsilon} \Pr[M(x') \in S] + \delta.$$

**Modified Goal**: Select the top-$k$ data elements subject to differential privacy.

## Previous Work

### Hasn't this problem already been solved?

- Although it is well known that the exponential mechanism [MT] and report noisy max [DR] can both solve this problem, the algorithms require knowing the full data universe in advance.
- Other works have proposed solutions in the frequent *itemset* setting, where all domain elements come from a sequence of known length and each element comes from a known alphabet [BLST, LQSC, ZNC, LC**?** ].
- However, what if a data analyst has no knowledge of the data domain, as is the case in exploratory analyses.

## Various Settings for User Level Privacy

| | $\Delta$-Restricted Sensitivity | Unrestricted Sensitivity |
|---|---|---|
| Known Domain | Laplace Mechanism [DMNS] | Exp Mechanism [MT] |
| Unknown Domain This work | Algorithm 1 | Algorithm 2 |

- Known domain setting is where the algorithm knows all possible data elements any user can have, e.g. top-10 countries with a certain skill.
- Unknown domain setting is where algorithms have no knowledge of data domain, e.g. top-**10** articles viewed.
- Sensitivity of a histogram $h$ is how much any one user can impact the counts when her data is removed.
- We assume for all cases that for each $h, h' \in \mathbb{N}^d$ that are neighbors,

$$||h - h'||_\infty \leq 1$$

- $\Delta$-restricted sensitivity means for any neighbors $h, h'$:

$$||h - h'||_0 \leq \Delta$$

- Unrestricted sensitivity means for any neighbors $h, h'$:

$$||h - h'||_0 \leq d$$

- Laplace Mechanism: Add $\mathrm{Lap}(1/\epsilon)$ to each count. Can release top-$k$ and counts while being $\Delta\epsilon$-DP.
- Exponential Mechanism: Add $\mathrm{Gumbel}(1/\epsilon)$ to each count. Only release top-$k$ (no counts) to ensure $k\epsilon$-DP.

## Algorithm 1 - Unknown Domain with $\Delta$-Restricted Sensitivity

- First find the top-$(\bar{k}+1)$: $h_{(1)} \geq \cdots \geq h_{(\bar{k}+1)}$.
- Make noisy threshold: $\hat{h}_\perp = h_{(\bar{k}+1)} + 1 + \log(\Delta/\delta) + \mathrm{Lap}(1/\epsilon)$.
- Add noise to each top-$\bar{k}$ count:
  $\hat{h}_1 = h_{(1)} + \mathrm{Lap}(1/\epsilon), \cdots, \hat{h}_{\bar{k}} = h_{(\bar{k})} + \mathrm{Lap}(1/\epsilon)$.
- Sort noisy counts $\{\hat{h}_1, \cdots, \hat{h}_{\bar{k}}, \hat{h}_\perp\}$ and release indices and counts that are larger than $\hat{h}_\perp$.

## Algorithm 2 - Unknown Domain with Unrestricted Sensitivity

- First find the top-$(\bar{k}+1)$: $h_{(1)} \geq \cdots \geq h_{(\bar{k}+1)}$.
- Make noisy threshold: $\hat{h}_\perp = h_{(\bar{k}+1)} + 1 + \log(\bar{k}/\delta) + \mathrm{Gumbel}(1/\epsilon)$.
- Add noise to each top-$\bar{k}$ count:
  $\hat{h}_1 = h_{(1)} + \mathrm{Gumbel}(1/\epsilon), \cdots, \hat{h}_{\bar{k}} = h_{(\bar{k})} + \mathrm{Gumbel}(1/\epsilon)$.
- Sort noisy counts $\{\hat{h}_1, \cdots, \hat{h}_{\bar{k}}, \hat{h}_\perp\}$ and release at most $k$ indices with counts larger than $\hat{h}_\perp$ in sorted order.

## Pay-what-you-get Composition

- Note that Algorithm 2 can return fewer than $k$ elements.
- It turns out that despite asking for a top-$k$ query, the privacy loss need only increase by the amount of elements that are returned (plus 1).
- Given an overall privacy loss budget of $k^*$ many indices that can be returned, an analyst can continue asking top-$k$ queries until $k^*$ many indices have been returned.
- Analysis follows from stringing together adaptively chosen exponential mechanisms with privacy parameter $\epsilon$.
- Need to also account for how many top-$k$ queries are asked $\ell^*$, not just the total number of indices returned. This impacts the total $\delta$.
- With each top-$k$ query we update the privacy budget as

$$k^* \leftarrow k^* - (\#\text{ of outcomes} + 1) \qquad \ell^* \leftarrow \ell^* - 1$$

- Continue until either $k^*$ is 0 or $\ell^*$ is 0.
- The full system of top-$k$ queries is $(\epsilon^*, \ell^*\delta)$-DP where

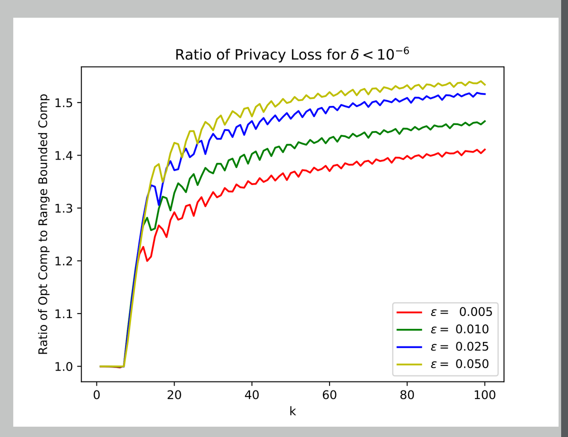
$$\epsilon^* \approx \epsilon\sqrt{k^* \log(1/\delta)}$$

## Improved Composition with Bounded Range Mechanisms

- Exp. mechanisms satisfy a stronger condition than DP.
- A mechanism $M : \mathcal{X} \to \mathcal{Y}$ is $\epsilon$-bounded range (BR) if for any neighbors $x, x' \in \mathcal{X}$ and outcome pairs $y, y' \in \mathcal{Y}$,

$$\frac{\Pr[M(x) = y]}{\Pr[M(x') = y]} \leq e^{\epsilon} \frac{\Pr[M(x) = y']}{\Pr[M(x') = y']}$$

- $\epsilon$-BR $\implies$ $\epsilon$-DP and $\epsilon$-DP $\implies$ $2\epsilon$-BR.
- Leads to more than **50%** improvement in overall privacy loss compared to the optimal DP composition.



**Recent optimal composition bounds:** `arxiv.org/abs/1909.13830`

## References


[BLST] Bhaskar, Laxman, Smith, and Thakurta. Discovering frequent patterns in sensitive data. In *KDD'10*.

[DMNS] Dwork, McSherry, Nissim, and Smith. Calibrating noise to sensitivity in private data analysis. In *TCC'06*.

[DR] Dwork and Roth. The algorithmic foundations of DP. *Foundations and Trends in Theoretical Computer Science*, 9(3 & 4).

[LC] Lee and Clifton. Top-k frequent itemsets via DP FP-trees. In *KDD'14*.

[LQSC] Li, Qardaji, Su, and Cao. Privbasis: Frequent itemset mining with DP. *PVLDB'12*.

[MT] McSherry and Talwar. Mechanism design via DP.

[ZNC] Zeng, Naughton, and Cai. On DP frequent itemset mining. *VLDB'12*.