Scalable Private Set Union Beyond Uniform Weighting

Justin Y. Chen
MIT
justc@mit.edu

Vincent Cohen-Addad
Google Research
cohenaddad@google.com

Alessandro Epasto
Google Research
aepasto@google.com

Morteza Zadimoghaddan
Google Research
zadim@google.com

May 7, 2024

Abstract

In the differentially private set union problem, users contribute sets of items as input, and the output is a subset of the union of all items. Algorithms for this problem seek to output as many items as possible while maintaining differential privacy with respect to the addition or removal of an individual user.

The basic solution to this problem maintains a weight over each item. Each user contributes uniformly to the items in their set, random noise is added to the weights, and items with noisy weight above a certain threshold are output. The only scalable (i.e., distributed) algorithms for this problem from prior work are this basic algorithm and an iterative method which repeatedly calls the basic algorithm, ignoring items found in prior invocations.

In this work, we give an improved weighting algorithm over basic uniform weighting. Our algorithm reroutes weight from items with weight far above the threshold to items with smaller weight, thereby increasing the probability that less frequent items are output. The algorithm is scalable and does not suffer any privacy loss when compared to the basic algorithm. We prove that our algorithm will never underperform the basic algorithm and show experimentally that replacing the basic algorithm with ours yields the best results among scalable algorithms for the private set union problem.

1 Introduction

In the setting of the private set union problem (also known as private partition selection), users each have a subset of items from an unknown and possibly infinite universe of items. The goal is to output as many of the items which appear in the union of the users' sets as possible while maintaining user-level differential privacy. This problem generalizes several tasks in differential privacy such as outputting queries asked to a search engine [KKMN09], discovering words and phrases used in emails or text messages [GGK+20], or as a prerequisite step to run GROUPBY operations over items in a SQL system [DVGM22]. Real-world datasets for these applications can be massive, precluding solutions which require storing the input in-memory on a single machine. In this work, we focus on scalable solutions to private set union that can be implemented in a distributed framework (e.g., MapReduce).

Differential privacy guarantees that the output of any algorithm for this problem cannot depend much on any given user's set. In particular, for any reasonable setting of privacy parameters, private algorithms cannot output items which appear in only a single set, implying that essentially no items can be output if the users' sets are disjoint. On the other hand, it is possible for a private algorithm to output items which appear in many different sets. This intuition is operationalized in the following approach.

Algorithms for private set union start by subsampling each user's set to bound the maximum number of items per set. Then, most algorithms proceed by assigning a weight to each item in the union, adding Gaussian (or Laplace) noise to each weight, and outputting all items with noised weight above a certain

threshold [KKMN09, GGK+20, CWG22, SDH23]. The amount of noise and value of the threshold depend on the privacy parameters and the sensitivity of the weighting function to the addition or removal of any individual user's set. Loosely, the contribution of each user to the item weights must be bounded in order to achieve differential privacy. Algorithms within this framework differ in the choice of how to assign item weights given a collection of users' sets.

A basic strategy is for each user to contribute equal weight to each of the items in their set [KKMN09]. It is easy to bound the sensitivity of this basic weighting and thus to prove differential privacy. On the other hand, the basic weighting algorithm is lossy in that it may overallocate weight far above the threshold to high frequency items, missing an opportunity to boost the weight of items closer to the decision boundary.

Despite its simplicity, the basic weighting algorithm is essentially the only known solution to the private set union problem which is amenable to implementation in a distributed framework. Other weighting schemes are greedy and inherently sequential [GGK+20, CWG22]. To our knowledge, the sole exception is the scalable, iterative partition selection (SIPS) scheme of [SDH23]. Even so, the core computation of this algorithm is repeated invocations of the basic weighting algorithm.

In this work, we design an adaptive, non-uniform weighting algorithm that reroutes overallocated weight to less frequent items. Our algorithm is computationally efficient and can be implemented in a distributed framework. Furthermore, it suffers no privacy loss compared to basic weighting—given the same privacy parameters, both algorithms utilize exactly the same amount of noise and the same threshold. Designing an adaptive algorithm with this property presents a non-trivial challenge. As a corollary, we prove that our algorithm will never underperform the basic algorithm, and we show examples where it provably outperform SIPS by a large margin. Finally, we conduct experiments on nine datasets with size up to the billion-scale. Our algorithm, either by itself or as a plug-in replacement for the basic weighting algorithm in SIPS, performs the best out of all scalable baselines on every dataset and is competitive with the sequential baselines.

1.1 Related Work

The differentially private set union problem was first studied in [KKMN09]. They utilized the now-standard approach of subsampling to limit the number of items in each users' set, constructing weights over items, and thresholding noised weights to produce the output. They proposed a version of the basic weighting algorithm which uses the Laplace mechanism rather than the Gaussian mechanism. This algorithm was also used in [WZL+20] within the context of a private SQL system.

The problem received renewed study in [GGK+20] where the authors propose a generic class of greedy, sequential weighting algorithms which empirically outperform basic weighting (with either the Laplace or Gaussian mechanism). [CWG22] gave an alternative greedy, sequential weighting algorithm which leverages item frequencies in cases where each user has a multiset of items. [DVGM22] analyzed in depth the optimal strategy when each user has only a single item (all sets have cardinality one). This is the only work that does not utilize the weight and threshold approach, but it is tailored only for this special case.

The work most related to ours is SIPS [SDH23] which proposes the first algorithm other than basic weighting which is amenable to implementation in a distributed environment. SIPS splits the privacy budget over a small number of rounds, runs the basic algorithm as a black box each round, and iteratively removes the items found in previous rounds for future computations. This simple idea leads to large empirical improvements, giving a scalable algorithm that has competitive performance with sequential algorithms.

2 Preliminaries

Definition 2.1 (Private Set Union)**.** In the private set union (aka private partition selection) problem, there are n users with each user u having a set S_u of items from an unknown and possibly infinite universe of items: the input is of the form $\mathcal{S} = \{(u, S_u)\}_{u\in[n]}$. The goal is to output a subset of the union of the users' sets $\mathcal{U} = \cup_{u\in[n]} S_u$ of maximum cardinality while maintaining user-level differentially privacy.

Definition 2.2 (Neighboring Datasets)**.** We say that two input datasets $\mathcal{S}$ and $\mathcal{S}'$ are neighboring if one can be obtained by removing a single user's set from the other, i.e., $\mathcal{S}' = \mathcal{S} \cup \{(v, S_v)\}$ for some new user v.

Definition 2.3 (Differential Privacy [DR14])**.** A randomized algorithm $\mathcal{M}$ is (ε, δ)-differentially private, or (ε, δ)-DP, if for any two neighboring datasets $\mathcal{S}$ and $\mathcal{S}'$ and for any possible subset of outputs $\mathcal{O} \subseteq \{U : U \subseteq \mathcal{U}\}$,

$$\Pr(\mathcal{M}(\mathcal{S}) \in \mathcal{O}) \leq e^{\varepsilon} \cdot \Pr(\mathcal{M}(\mathcal{S}') \in \mathcal{O}) + \delta.$$

Let $\Phi : \mathbb{R} \to \mathbb{R}$ be the standard Gaussian cumulative density function.

Proposition 2.4 (Gaussian Mechanism [BW18])**.** *Let $f : \mathcal{D} \to \mathbb{R}^d$ be a function with ℓ_2 sensitivity Δ_2. For any $\varepsilon \geq 0$ and $\delta \in [0, 1]$, the mechanism $M(x) = f(x) + Z$ with $Z \sim \mathcal{N}(0, \sigma^2 I)$ is (ε, δ)-DP if and only if*

$$\Phi\left(\frac{\Delta_2}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2}\right) - e^{\varepsilon}\Phi\left(-\frac{\Delta_2}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2}\right) \leq \delta.$$

3 Technical Overview

3.1 Weight and Threshold Meta-Algorithm

Solutions to the private set union problem follow a weight and threshold meta-algorithm. Input is given as a set of user sets $\mathcal{S} = \{(u, S_u)\}_{u \in [n]}$, privacy parameters ε and δ, a maximum degree cap Δ_0, and a weighting algorithm `WeightingAlgo`. First, each user's set is randomly subsampled so that the size of each resulting set is at most Δ_0 (the necessity of this step will be further explicated). Then, the `WeightingAlgo` takes in the cardinality-capped sets and produces a set of weights over all items in the union. Independent Gaussian noise with standard deviation σ is added to each coordinate of the weights, and items with weight above a certain threshold ρ are output.

The privacy of this algorithm depends on certain "sensitivity" properties of the `WeightingAlgo` as well as our choice of σ and ρ. Consider any pair of neighboring inputs $\mathcal{S}$ and $\mathcal{S}' = \mathcal{S} \cup \{(v, S_v)\}$, let $\mathcal{U}$ and $\mathcal{U}'$ be the corresponding unions, and let w and w' be the item weights assigned by `WeightingAlgo` on the two inputs, respectively. The ℓ_2 sensitivity of `WeightingAlgo` is defined as an upper bound on the Euclidean distance between w and w': $\sqrt{\sum_{i \in \mathcal{U}} (w(i) - w'(i))^2}$. Given bounded ℓ_2 sensitivity, choosing the scale of noise σ appropriately for the Gaussian mechanism in Proposition 2.4 ensures that outputting the noised weights on items in $\mathcal{U}$ satisfies $\left(\varepsilon, \frac{\delta}{2}\right)$-DP. So if we knew $\mathcal{U}$, then the output of the algorithm after thresholding would be private via post-processing.

However, knowledge of the union $\mathcal{U}$ is exactly the problem we want to solve. The challenge is that there may be items in $\mathcal{U}'$ which do not appear in $\mathcal{U}$. Let $T = \mathcal{U}' \setminus \mathcal{U}$ be these "novel" items with $t = |T|$. As long as the probability that any of these items are output by the algorithm is at most $\frac{\delta}{2}$, (ε, δ)-DP will be maintained. Consider a single item $i \in T$ which has zero probability of being output by a weight and threshold algorithm run on $\mathcal{S}$ but is given some weight $w'(i)$ when the `WeightingAlgo` is run on $\mathcal{S}'$. The item will be output only if after adding the Gaussian noise with standard deviation σ, the noised weight exceeds ρ. The probability that any item in T is output follows from a union bound. In order to union bound only over finitely many events, we rely on the fact that $t \leq \Delta_0$; this is why the cardinalities must be capped. The second important sensitivity measure of a `WeightingAlgo` is the novel ℓ_∞ sensitivity, which is an upper bound, parameterized by t, on the value of $w'(i)$ for $i \in T$. Then, the calculation of ρ to obtain (ε, δ)-DP is obtained based on the novel ℓ_∞ sensitivity, δ, σ, and Δ_0.

It is straightforward to show that for the uniform weighting algorithm `BasicWeighting` in which each user u contributes $1/\sqrt{|S_u|}$ weight to each of the items in S_u has ℓ_2 sensitivity equal to 1 and novel ℓ_∞ sensitivity equal to $1/\sqrt{t}$.

3.2 Max Degree Adaptive Weighting

Our main result is an adaptive weighting algorithm `MaxDegreeAdaptiveWeighting` which is amenable to distributed implementations and has the exact same ℓ_2 and novel ℓ_∞ sensitivities as `BasicWeighting`. Therefore, within the weight and threshold meta-algorithm, both algorithms utilize the same noise σ and threshold ρ to maintain privacy.

Our algorithm takes two additional parameters: a maximum adaptive degree $d_{max} \in [1, \Delta_0]$ and an adaptive threshold $\tau = \rho + \beta\sigma$ for a free parameter $\beta \geq 0$. Users with set cardinalities greater than d_{max} are set aside and contribute basic uniform weights to their items at the end of the algorithm. The rest of the users participate in adaptive reweighting. We start from a uniform weighting where each user sends $1/|S_u|$ weight to each of their items. Items have their weights truncated to τ and any excess weight is sent back to the users proportional to the amount they contributed. Users then reroute a carefully chosen fraction (depending on d_{max}) of this excess weight across their items. Finally, each user adds $1/\sqrt{|S_u|} - 1/|S_u|$ to the weight of each of their items. Each of these steps are straightforward to implement within a distributed framework.

Bounding the sensitivity of this weighting algorithm is significantly more involved than for basic weighting. Algorithm design choices such as using an initial uniform weighting inversely proportional to cardinality rather than square root of cardinality, using a maximum adaptive degree, and the fraction of how much excess weight to reroute are all required for the following theorem.

Theorem 3.1 (Informal)**.** *`MaxDegreeAdaptiveWeighting` has ℓ_2 sensitivity equal to 1 and novel ℓ_∞ sensitivity equal to $1/\sqrt{t}$. Thus, using this algorithm within the weight and threshold meta-algorithm maintains (ε, δ)-DP.*

As a direct result of the design of `MaxDegreeAdaptiveWeighting`, for items with weight less than τ, our algorithm dominates the basic weighting algorithm in the sense that the weights on those items will only increase under our algorithm. While weights on overallocated items may decrease due to truncation, this only marginally affects the probability that these items are output as long as β is large enough, formalized in the following theorem statement.

Theorem 3.2 (Informal)**.** *Let U be the set of items output when using `BasicWeighting` as the weighting algorithm in the weight and threshold meta-algorithm and let U^* be the set of items output when using `MaxDegreeAdaptiveWeighting` as the weighting algorithm. Then, for items $i \in \mathcal{U}$, $\Pr(i \in U^*) \geq \min\{\Pr(i \in U), \Phi(\beta)\}$.*

3.3 Experiments

We compare our algorithm to several distributed and sequential baselines. The results for our algorithm include the algorithm run by itself or as a repalcement for basic weighting in the SIPS framework of iteratively running a weight and threshold algorithm while removing items output in previous rounds. The baselines are basic weighting [KKMN09, GGK+20], SIPS [SDH23], PolicyGaussian [GGK+20], and GreedyUpdate [CWG22].

Dataset	*Distributed*			*Sequential*	
	MaxDegreeAdaptive	Basic	SIPS	PolicyGaussian	GreedyUpdate
higgs	1933* (±5)	1907 (±17)	1907 (±17)	2052 (±10)	2811† (±12)
imdb	3102* (±19)	2504 (±7)	3076 (±16)	3578† (±19)	1356 (±2)
reddit	5826* (±30)	4062 (±21)	5783 (±30)	7170† (±39)	6364 (±8)
finance	17421* (±21)	12677 (±28)	17295 (±31)	20566 (±10)	23563† (±42)
wiki	9848* (±34)	7753 (±36)	9795 (±21)	11455† (±21)	4769 (±20)
twitter	13609* (±12)	8859 (±22)	13499 (±50)	15907 (±30)	16003† (±28)
amazon	66816*† (±92)	35256 (±23)	66158 (±59)	−−	−−
memetracker	531530*† (±307)	527611 (±384)	527611 (±384)	−−	−−
clueweb	34914880*† (±1705)	34603077 (±1618)	34889210 (±1873)	−−	−−

Table 1: Comparison of size of output of the weight and threshold approach using different weighting algorithms with $\varepsilon = 1$ and $\delta = 10^{-5}$. The results for all algorithms are the best among a grid search of parameters each averaged over five trials with standard deviation shown in parentheses. For each dataset, the best distributed result is denote by $(*)$ and the best distributed or sequential result is denoted by $(\dagger)$.

References

[BW18] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*. PMLR, 2018.

[CWG22] 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*, 2022.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[DVGM22] Damien Desfontaines, James Voss, Bryant Gipson, and Chinmoy Mandayam. Differentially private partition selection. In *Proceedings on Privacy Enhancing Technologies*, 2022.

[GGK+20] Sivakanth Gopi, Pankaj Gulhane, Janardhan Kulkarni, Judy Hanwen Shen, Milad Shokouhi, and Sergey Yekhanin. Differentially private set union. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020.

[KKMN09] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. In *Proceedings of the 18th international conference on World wide web*, 2009.

[SDH23] Marika Swanberg, Damien Desfontaines, and Samuel Haney. DP-SIPS: A simpler, more scalable mechanism for differentially private partition selection. In *Proceedings on Privacy Enhancing Technologies*, 2023.

[WZL+20] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. Differentially private SQL with bounded user contribution. In *Proceedings on Privacy Enhancing Technologies*, 2020.