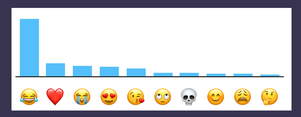
Frequency Estimation under Local Differential Privacy

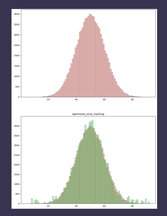
Graham Cormode, Samuel Maddock, Carsten Maple
University of Warwick

Private Frequency Estimation

- Local Differential Privacy (LDP) is a popular way to collect private statistics from a set of users with large-scale deployments by companies such as Apple and Google.
- Our main goal is to present an unbiased evaluation of current state-of-the-art frequency estimation and heavy hitter LDP methods.
- We use a four-layer framework, analysing popular techniques experimentally in each.
- We find more effective combinations of techniques with clear improvements over existing deployments.

1 + 2 + 3 + 4





Experimental Setup

We consider n clients over a domain size of d, with privacy budget ϵ. Experiments on both synthetic and real-world data:

- **Synthetic data:** Zipf distribution s=1.1 varying n and d depending on the experiment. We measure efficacy via Mean Square Error (MSE) and k-MSE which is calculated on only the top-k frequent items (fixing k=50)
- **AOL Dataset:** Search query dataset containing clicked URLs. We consider each clicked URL to belong to a single user. Here n =1,935,614 with d=383,467 unique URLs

Implemented in Python 3.7.4, code available: https://github.com/Samuel-Maddock/pure-LDP

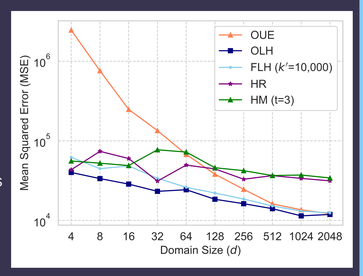
1 Frequency Oracles (FO)

Frequency Oracles give frequency estimates for any item:

- **Direct Encoding (DE) -** Basic extension of Randomised Response (RR) to more than 2 outputs
- **Unary Encoding (SUE, OUE)**
  - **Symmetric Unary Encoding (SUE)** [Erlingsson et al 2014]
  - **Optimised Unary Encoding (OUE)** [Wang et al 2017]
- **Local Hashing (BLH, OLH, FLH)**
  - **Binary Local Hashing (BLH)** [Bassily and Smith 2015]
  - **Optimised Local Hashing (OLH)** [Wang et al 2017]
  - **Fast Local Hashing (FLH)** - Heuristic approach
- **Hadamard Encodings (HM, HR)**
  - **Hadamard Mechanism (HM)**
  - **Hadamard Response (HR)** [Acharya et al 2018]

Experiments and Results

- For Hadamard methods, when $\epsilon > 1$ sampling more coefficients gives more accurate results. However, HR remained optimal in terms of aggregation speed and communication cost
- Figure varies domain size over a Zipf distribution, n=100k and ϵ=3
- OLH achieves the smallest MSE. FLH has near identical performance as d increases with substantial speed-ups (10x) in server aggregation



- Clear grouping between the Hadamard methods and local hashing/OUE methods illustrates the price of accuracy for using the Hadamard transform.

2 Domain Reduction

Larger domains are handled by domain reduction methods:

1. Bloom Filters (BFs) [Erlingsson et al 2014], Used in Google's RAPPOR

2. Count-Min Sketches (CM) [Apple 2017], Used in Apple's (H)CMS

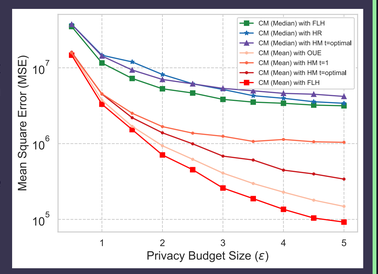
3. Count-Sketches (CS) [Bassily et al 2017], Similar to CM, uses two sets of hash functions

Sketch estimation methods: Minimum, Mean and **Median**

Popular LDP implementations can be expressed in the framework:
- **Google's RAPPOR** = Bloom Filter + SUE
- **Apple's CMS** = CM Mean + S/OUE
- **Apple's HCMS** = CM Mean + HM (sampling t=1 coefficient)

Experiments and Results

- Experiments detailing effects of sketch sizes, estimation methods and comparisons between BFs and sketches showed BFs were competitive but tricky to use as the domain increases, so sketching methods are preferred
- Figure shows experiment on Zipf data comparing most competitive FOs with most competitive sketch methods
- Clear grouping of the mean and median techniques, with mean performing best



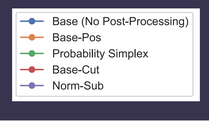
- Sampling more Hadamard coefficients gives improvements over Apple's HCMS
- CM (Mean) with FLH beats Apple's CMS implementations
- Overall, clear improvements to Apple's deployments can be made by modifying the underlying frequency oracles being used.

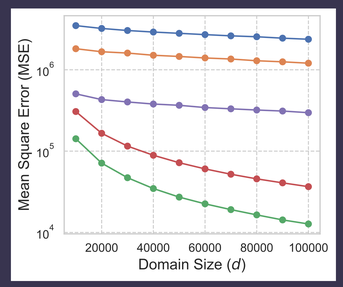
3 Post-Processing

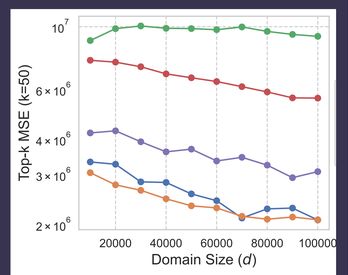
[Wang et al 2020] **present an experimental comparison of a large range of post-processing techniques. For our experiments we choose a subset of the best performing.**

Experiments and Results

- We find post-processing methods affect both MSE and k-MSE separately
- For example, threshold based methods may reduce MSE by rounding many frequencies to 0 but inflate k-MSE if they accidentally round frequent items to 0
- Rounding negative values to zero always improves MSE and that Norm-Sub achieves best balance overall. Results corroborate those by [Wang et al 2020]
- Figure shows example, sampling from a Zipf with d =100,000 ϵ=3, n =1,000,000 and fixing the oracle to be FLH







- Further experiments were performed with sketching and conclusions were generally the same.

4 Heavy Hitter Protocols

When the domain is very large or unknown we can use heavy hitter protocols to discover the most popular items:

Sequence Fragment Puzzle (SFP) [Apple 2017] - Frequent strings in a population will also have frequent substrings, so heavy hitters are 'reconstructed' by privatising substrings

Hierarchical Search Methods -Prefixes (instead of substrings) are used to build up heavy hitters

- **Prefix Extending Methods (PEM)** [Wang et al 2019] - Randomly sample the length of the prefix
- **TreeHistogram (TH)** [Bassily et al 2017]- Split privacy budget evenly across privatising the whole string and a randomly chosen prefix

Experiments and Results

- Goal is to recover the top-10 most frequent URLs in the AOL data
- We consider combinations of parameters:
  - **Top-T:** Top-10 or Top-20 prefixes used to build heavy-hitters
  - **Frequency Oracles:** OUE, FLH
  - **Domain Reduction:** CM Mean and CM Median over 3 different sketch sizes (32,1024), (128,1024), (1024,2048)
- Table below shows top-10 results of our experiment, sorted by F1 score. PEM dominates the other techniques
- FLH and OUE are similar in performance but FLH has a smaller communication cost
- Can achieve best performance with FLH and at much smaller sketch sizes than those proposed by Apple.

	Heavy Hitter	Sketch Method	Frequency Oracle	Parameters	Precision	Recall	F1 Score
1	PEM	CM (Median)	FLH	T=20, r=1024, c=2048	0.822	0.74	**0.779** (0.052)
2	PEM	CM (Median)	FLH	T=20, r=128, c=1024	0.789	0.74	0.763 (0.051)
3	PEM	CM (Mean)	OUE	T=10, r=1024, c=2048	0.626	**0.96**	0.757 (0.057)
4	PEM	CM (Mean)	OUE	T=20, r=32, c=1024	0.782	0.72	0.749 (0.025)
5	PEM	CM (Median)	FLH	T=10, r=1024, c=2048	0.758	0.74	0.748 (0.061)
6	PEM	CM (Mean)	OUE	T=20, r=128, c=1024	0.786	0.70	0.738 (0.035)
7	PEM	CM (Mean)	OUE	T=10, r=32, c=1024	0.782	0.70	0.738 (0.025)
8	PEM	CM (Median)	FLH	T=20, r=32, c=1024	0.792	0.68	0.731 (0.036)
9	PEM	CM (Mean)	OUE	T=20, r=1024, c=2048	0.766	0.70	0.73 (0.029)
10	PEM	CM (Median)	FLH	T=10, r=32, c=1024	0.744	0.70	0.721 (0.055)