ILLINOIS INSTITUTE OF TECHNOLOGY

CS 528 (Fall 2021)
Data Privacy & Security

Yuan Hong
Department of Computer Science
Illinois Institute of Technology

Chapter 4
Local Differential Privacy

RECAP

- **Differential Privacy**
  - Adversarial Model
  - Interactive
  - Non-Interactive
  - Laplace Mechanism
  - Exponential Mechanism
  - Composition

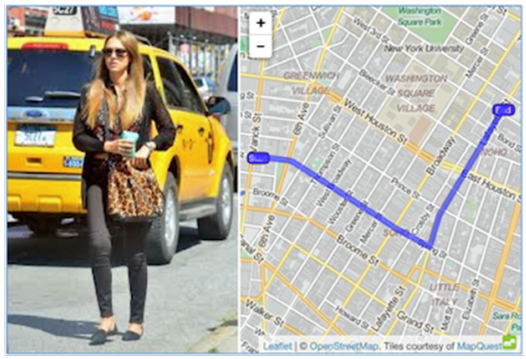
OUTLINE

Local Differential Privacy for Distributed Data

1. Local Differential Privacy in Practice
2. LDP Guarantees
3. LDP Applications

DATA RELEASE HORROR STORIES

NYC Taxi & Limousine Commission

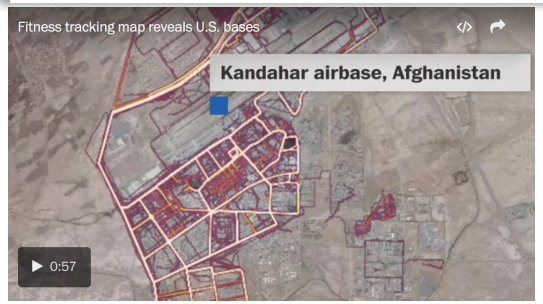


Jessica Alba (actor)

Strava's fitness tracker heat map reveals the location of military bases

Geolocation isn't a new problem for the military

By Andrew Liptak | @AndrewLiptak | Jan 28, 2018, 3:51pm EST



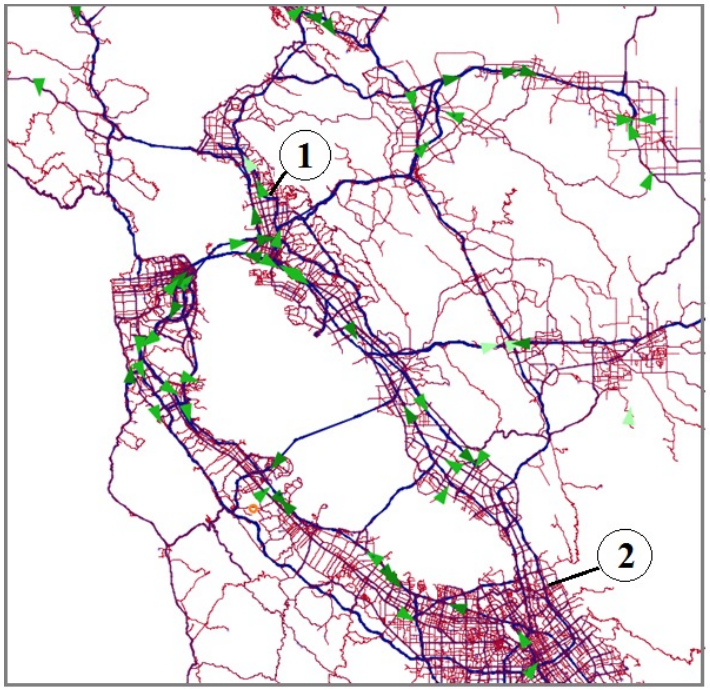
AOL

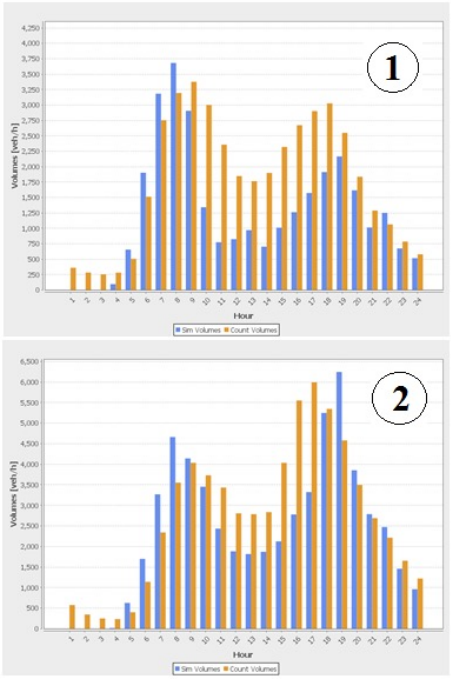
NETFLIX

Commonwealth of Massachusetts
Group Insurance Commission
Your Benefits Connection

We need to solve this data release problem ...

BUT WHY RELEASE DATA?





For example, urban mobility modeling at scale

- Smart traffic design is critical, congestion is getting worse

DIFFERENTIAL PRIVACY (RECAP)

A randomized algorithm A satisfies ε-differential privacy if:

Given two data sets that differ by one individual, D and D', any property S, and $\varepsilon > 0$:

$$Pr[A(D) \in S] \leq e^{\varepsilon} Pr[A(D') \in S]$$

Can achieve ε-DP for counts by adding a random noise value

Uncertainty due to noise → plausible deniability

- Hides whether someone is present in the data

DIFFERENTIAL PRIVACY (RECAP)

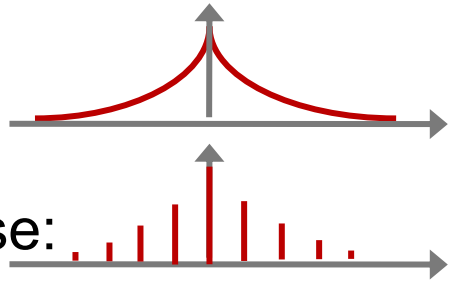
(Global) Sensitivity of publishing:

$s = \max_{D,D'} |F(D) - F(D')|$, D, D' differ by 1 individual

E.g., count individuals satisfying property P: s = 1

For every value that is output:

- Add Laplacian noise Lap(s/ε):
- Or Geometric noise for discrete case:



Simple rules for composition of differentially private outputs:

Given output O_1 that is ε_1-DP and O_2 that is ε_2-DP

- (Sequential composition) If inputs overlap, result is $(\varepsilon_1 + \varepsilon_2)$-DP
- (Parallel composition) If inputs disjoint, result is $\max(\varepsilon_1, \varepsilon_2)$-DP

TRYING TO REDUCE TRUST

Traditional differential privacy assumes a trusted party

- Data aggregator (e.g., organizations) that sees the true, raw data
- Can compute exact query answers, then perturb for privacy

A reasonable question: can we reduce the amount of trust?

- Can we remove the trusted party from the equation?
- Users produce locally private output, aggregate to answer queries

One approach is to use SMC or homomorphic encryption

- Merge encrypted data, and add noise for privacy inside encryption
- Complex to get right, and higher computational overheads

LOCAL DIFFERENTIAL PRIVACY

What about having each user run a DP algorithm on their data?

- Then combine all the results to get a final answer

On first glance, this idea seems crazy

- Each user adds noise to mask their own input
- So surely the noise will always overwhelm the signal?

But … noise can cancel out or be subtracted out

- We end up with the true answer, plus noise which can be smaller
- However, noise is still larger than in the centralized case

LOCAL DIFFERENTIAL PRIVACY: EXAMPLE

Each of N users has 0/1 value, estimate total population sum

- Each user adds independent Laplace noise: mean 0, variance $2/\varepsilon^2$

Adding user results: true answer + sum of N Laplace distributions

- Error is random variable, with mean 0, variance $2N/\varepsilon^2$
- Confidence bounds: ~95% chance of being within 2σ of the mean
- So error looks like $\sqrt{N}/\varepsilon$, but true value may be proportional to N

Numeric example: suppose true answer is N/2, $\varepsilon = 1$, N = 1M

- We see 500K ± 2800 : about 1% uncertainty
- Error in centralized case would be close to 1 (0.001%)

LOCAL DIFFERENTIAL PRIVACY

We can achieve LDP, and obtain reasonable accuracy (for large N)

- The error typically scales with $\sqrt{N}$

Generic approach: apply centralized DP algorithm to local data

- But error might still be quite large
- Unclear how to merge private outputs (e.g., private clustering)

So we seek to design new LDP algorithms

- Maximize the accuracy of the results
- Minimize the costs to the users (space, time, communication)
- Ensure that there is an accurate algorithm for aggregation

FROM DP TO LDP

ε is also called privacy budget
Smaller ε ➔ stronger privacy

Idea of DP: Any output should be about as likely regardless of whether or not I am in the

A randomized algorithm $\boldsymbol{A}$ satisfies $\boldsymbol{\varepsilon}$-differential privacy, iff for any two neighboring datasets $\boldsymbol{D}$ and $\boldsymbol{D'}$ and for any output $\boldsymbol{O}$ of $\boldsymbol{A}$,

$$\Pr[\boldsymbol{A}(\boldsymbol{D}) = \boldsymbol{O}] \leq \exp(\boldsymbol{\varepsilon}) \cdot \Pr[\boldsymbol{A}(\boldsymbol{D'}) = \boldsymbol{O}]$$

Run by the server

A randomized algorithm $\boldsymbol{A}$ satisfies $\boldsymbol{\varepsilon}$-local differential privacy, iff for any two inputs $\boldsymbol{x}$ and $\boldsymbol{x'}$ and for any output $\boldsymbol{y}$ of $\boldsymbol{A}$,

$$\Pr[\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{y}] \leq \exp(\boldsymbol{\varepsilon}) \cdot \Pr[\boldsymbol{A}(\boldsymbol{x'}) = \boldsymbol{y}]$$

Run by each single person

Idea of LDP: Any output should be about as likely regardless of my secret

PROPERTIES OF CENTRALIZED DP

A randomized algorithm $\boldsymbol{A}$ satisfies $\boldsymbol{\varepsilon}$-differential privacy, iff for any two neighboring datasets $\boldsymbol{D}$ and $\boldsymbol{D'}$ and for any output $\boldsymbol{O}$ of $\boldsymbol{A}$,

$$\Pr[\boldsymbol{A}(\boldsymbol{D}) = \boldsymbol{O}] \leq \exp(\boldsymbol{\varepsilon}) \cdot \Pr[\boldsymbol{A}(\boldsymbol{D'}) = \boldsymbol{O}]$$

Post-processing (of the output) is free

- does not consume privacy budget

What about LDP?

Parallel composition

- partition the dataset into subsets, each applying an ε_i-DP algorithm, the overall result satisfies $\max(\varepsilon_i)$-DP

Sequential composition

- apply k DP algorithms, each using ε_i, result satisfies $\sum \varepsilon_i$-DP

PROPERTIES OF LDP

A randomized algorithm $\boldsymbol{A}$ satisfies $\boldsymbol{\varepsilon}$-local differential privacy, iff for any two inputs $\boldsymbol{x}$ and $\boldsymbol{x'}$ and for any output $\boldsymbol{y}$ of $\boldsymbol{A}$,

$$\Pr[\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{y}] \leq \exp(\boldsymbol{\varepsilon}) \cdot \Pr[\boldsymbol{A}(\boldsymbol{x'}) = \boldsymbol{y}]$$

Post-processing is also free

- does not consume privacy budget

No direct parallel composition

- because each user only has one record, which cannot be partitioned
- but one can apply different questions to different subsets of users

Sequential composition

- apply k LDP algorithms, each using ε_i, result satisfies $\sum \varepsilon_i$-LDP

KEY DIFFERENCE BETWEEN DP AND LDP

DP concerns two neighboring datasets

LDP concerns any two values

As a result, the amount of noise is different: In aggregated result for counting queries

- Noise in DP is $\Omega(1)$ (sensitivity is constant)
- But in LDP, even noise for each user is constant, the aggregated result is $\Omega(\sqrt{n})$ [1]
- If the result is normalized (divide the result with n), noise is $\Omega\left(\frac{1}{n}\right)$ versus $\Omega\left(\frac{1}{\sqrt{n}}\right)$

[1] Optimal lower bound for differentially private multi-party aggregation by T.-H. H. Chan, E. Shi, and D. Song

PRIVACY WITH A COIN TOSS: RANDOMIZED RESPONSE

Each user has a single bit of private information

- Encoding e.g., political/sexual/religious preference, illness, etc.

Randomize Response (RR): toss an unbiased coin [Warner 65]

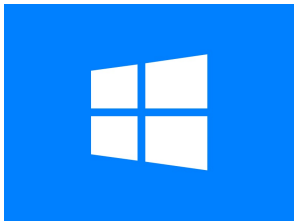
- If Heads (probability $p = ½$), report the true answer
- Else, toss unbiased coin again: if Heads, report true answer, else lie

Collect responses from N users, subtract noise

- Error in the estimate is proportional to $1/\sqrt{N}$
- Simple LDP algorithm with parameter $\varepsilon=\ln((¾)/(¼)) = \ln(3)$
- Generalization: allow biased coins ($p \neq ½$)

PRIVACY IN PRACTICE





Differential privacy based on coin tossing is widely deployed!

- In Google Chrome browser, to collect browsing statistics
- In Apple iOS and MacOS, to collect typing statistics
- In Microsoft Windows to collect telemetry data over time
- From Snap to perform modeling of user preference
- This yields deployments of over 100 million users each

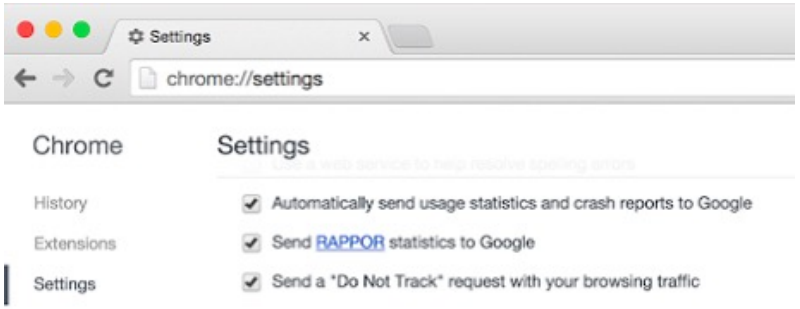
All deployments are based on RR, but extend it substantially

- To handle the large space of possible values a user might have

Local Differential Privacy is state of the art in 2021

- Randomized response invented in 1965: five decades ago!

GOOGLE'S RAPPOR



Each user has one value out of a very large set of possibilities

- E.g., their favourite URL, www.nytimes.com

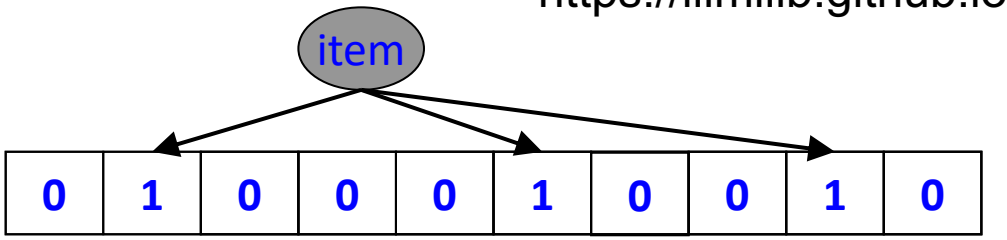
Attempt 1: run RR for all values: google.com? Bing.com? …

- User has sparse binary vector with one 1: run RR on every bit
- Satisfies 2ε-LDP: change user's choice, 2 bits change: 1 → 0, 0 → 1
- Slow: sends 1 bit for every possible index in the vector
- Limited: can't handle new possibilities being added

Try to do better by reducing domain size through hashing

BLOOM FILTERS

https://en.wikipedia.org/wiki/Bloom_filter
https://llimllib.github.io/bloomfilter-tutorial/



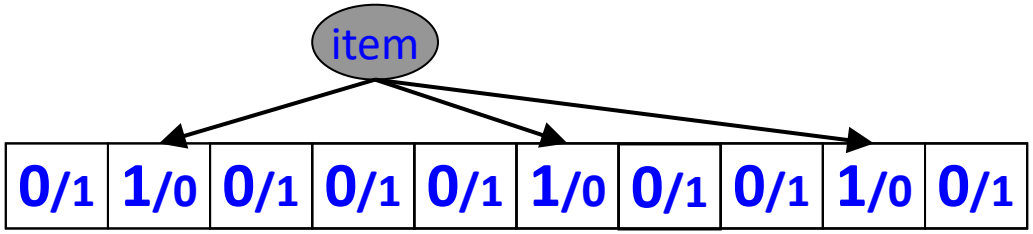
Bloom filters compactly encode set membership

- k hash functions map items to m-bit vector k times
- Update: set all k entries to 1 to indicate item is present
- Query: can lookup items, store set of size n in O(n) bits
- Analysis: choose k and m to obtain small false positive probability
  - The false positive rate will be approximately $(1-e^{-kn/m})^k$, so you can just plug the number *n* of elements you expect to insert, and try various values of *k* and *m* to configure your filter for your application.

Can be merged by OR-ing vectors (of same size)

- Duplicate insertions do not change Bloom filters

COUNTING BLOOM FILTERS & RR



Idea: apply Randomized Response to the bits in a Bloom filter

- Not too many bits in the filter compared to all possibilities

Each user maps their input to at most k bits in the filter

- Privacy guarantee with parameter $2k\varepsilon$

Combine all user reports and observe how often each bit is set

RAPPOR IN PRACTICE

The RAPPOR approach is implemented in the Chrome browser

- Collects data from opt-in users, tens of millions per day
- Open source implementation available

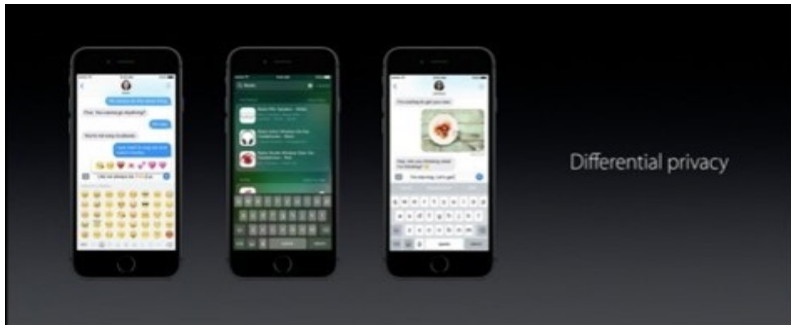
Tracks settings in the browser, e.g., home page, search engine

- Many users unexpectedly change home page → possible malware

Typical configuration:

- 128 bit Bloom filter, 2 hash functions, privacy parameter ~0.5
- Needs about 10K reports to identify a value with confidence

APPLE: SKETCHES AND TRANSFORMS



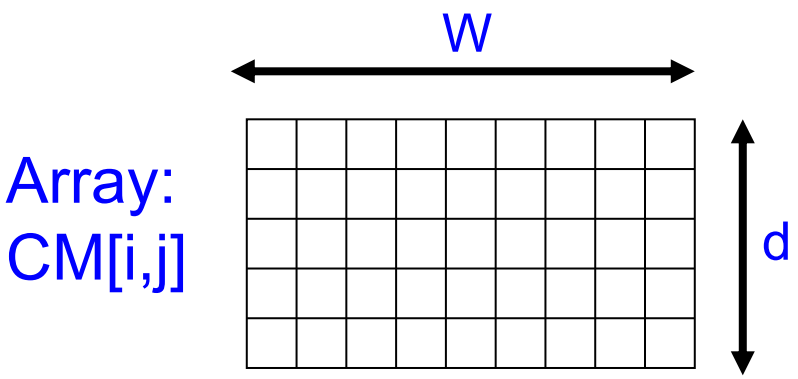
Similar problem to RAPPOR: count frequencies of many items

- For simplicity, assume that each user holds a single item
- To reduce burden of collection, can size of summary be reduced?

Instead of Bloom Filter, make use of sketches

- Similar idea, but better suited to capturing frequencies

COUNT-MIN SKETCH



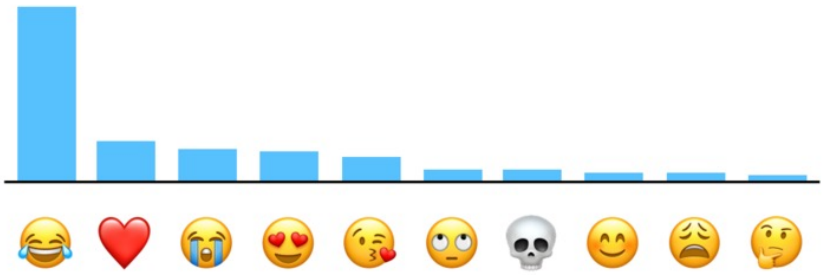
Count-Min sketch encodes item counts

- Allows estimation of frequencies (e.g., for selectivity estimation)
- Some similarities in appearance to Bloom filters

Model input data as a sparse frequency vector x of dimension U

- Create a small summary as an array of w × d in size
- Use d hash function to map vector entries to [1..w]

APPLE (CON'T)



The Count Mean Sketch technique allows Apple to determine the most popular emoji to help design better ways to find and use our favorite emoji. The top emoji for US English speakers contained some surprising favorites.

Apple uses their system to collect data from iOS and OS X users

- Popular emojis: (heart) (laugh) (smile) (crying) (sadface)
- "New" words: bruh, hun, bae, tryna, despacito, mayweather
- Which websites to mute, which to autoplay audio on!

Deployment settings: w=1000, d=1000, ε=2-8 (some criticism)

MICROSOFT TELEMETRY DATA COLLECTION

Microsoft wants to collect data on app usage

- How much time was spent on a particular app today?
- Allows finding patterns over time

Makes use of multiple subroutines:

- 1BitMean to collect numeric data
- dBitFlip to collect (sparse) histogram data
- Memoization and output perturbation to allow repeated probing

Has been implemented in Windows since 2017

1BitMean

Problem 1: want to estimate the mean f value of all users

- Each user has a value f between 0 and 1

Simple approach: build a histogram

- How to decide bucket boundaries? Introduces rounding error

1bitMean: modify randomized response

- User outputs 1 with probability linear in f: $(1 + f \cdot (e^{\varepsilon} - 1))/(e^{\varepsilon} + 1)$
- f=0 gives $Pr[1] = 1/(e^{\varepsilon} + 1)$, while f=1 gives $Pr[1] = e^{\varepsilon}/(e^{\varepsilon} + 1)$
- Taking the average of (unbiased version of) reports gives mean
- Error can be bounded proportional to $1/(\varepsilon\sqrt{n})$

dBitFlip

Problem 2: build a histogram over data items from a population

- Each user has a single item value out of k possibilities
- Essentially the same problem that Google and Apple tackle

Simple approach: apply RR to every bucket, costly when k is large

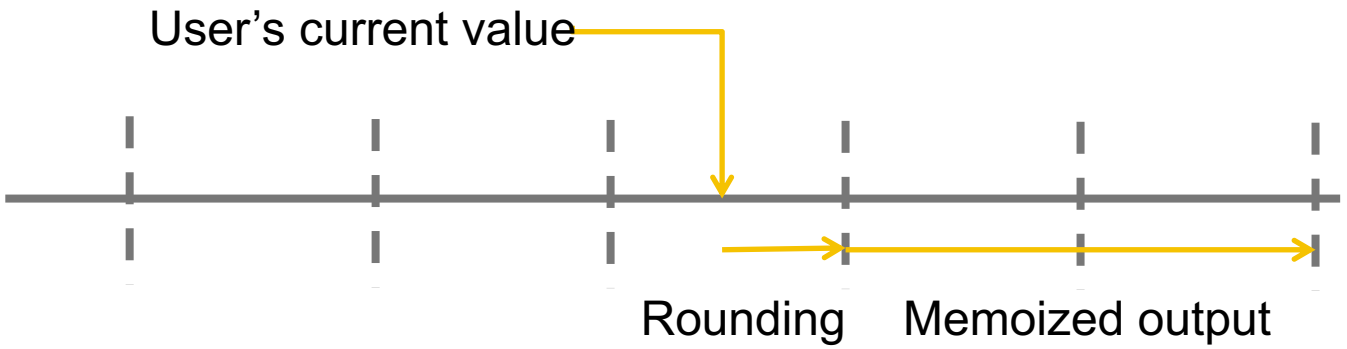
dBitFlip: apply Randomized Response to a sample of buckets

- User samples d buckets without replacement, applies RR to each
- Aggregator sums and unbiases the noisy reports
- Error proportional to $\sqrt{(k/d)}$: trades off error for speed
- Many alternative solutions for this “frequency oracle” problem

Memoization

Collecting data continuously erodes the privacy guarantee

- E.g., if we know a user's value didn't change, we can estimate it



Memoization: a heuristic to reduce privacy loss

- Each user rounds their current value to a discrete range
- Each user precomputes their output for each range
- Choosing a fixed random shift for each user reduces error

OUTPUT PERTURBATION

Note however "memoization violates differential privacy"!

- Can tell the difference between change and no change
- But at least this process hides what the value was

Output perturbation aims to address this problem

- For each user output bit, flip it with small probability γ, e.g., 0.2
- Increases error by a factor of $1/(1 - 2\gamma)$ (so 5/3 for γ=0.2)

Increases uncertainty about exactly when a change happened

- After long enough, can be certain that some change happened

MS Telemetry Collection in Practice

Deployed in Windows 10 Fall Creators Update (October 2017)

- Collects number of seconds users spend in different apps
- Parameters: $\varepsilon = 1$ and $\gamma = 0.2$
- Collection period: every 6 hours

Collects data on all app usage, not just one at a time

- Can analyze based on the fact that total time spent is limited
- Gives overall guarantee of $\varepsilon = 1.672$ for a round of collection

SNAP PRIVATE MACHINE LEARNING

Snap (parent company of Snapchat) proposes LDP ML

- "Federated Machine Learning", with privacy
- Maintain k current models
- Each step, send one user a randomly chosen model
- Receive an updated model, randomly replace one of the k

Deployment example: detecting phishing URLs

- Model: Logistic regression, 1000 features
- Parameters: k=20, ε = ???
- Accuracy: reached 93% accuracy, after 1 billion updates

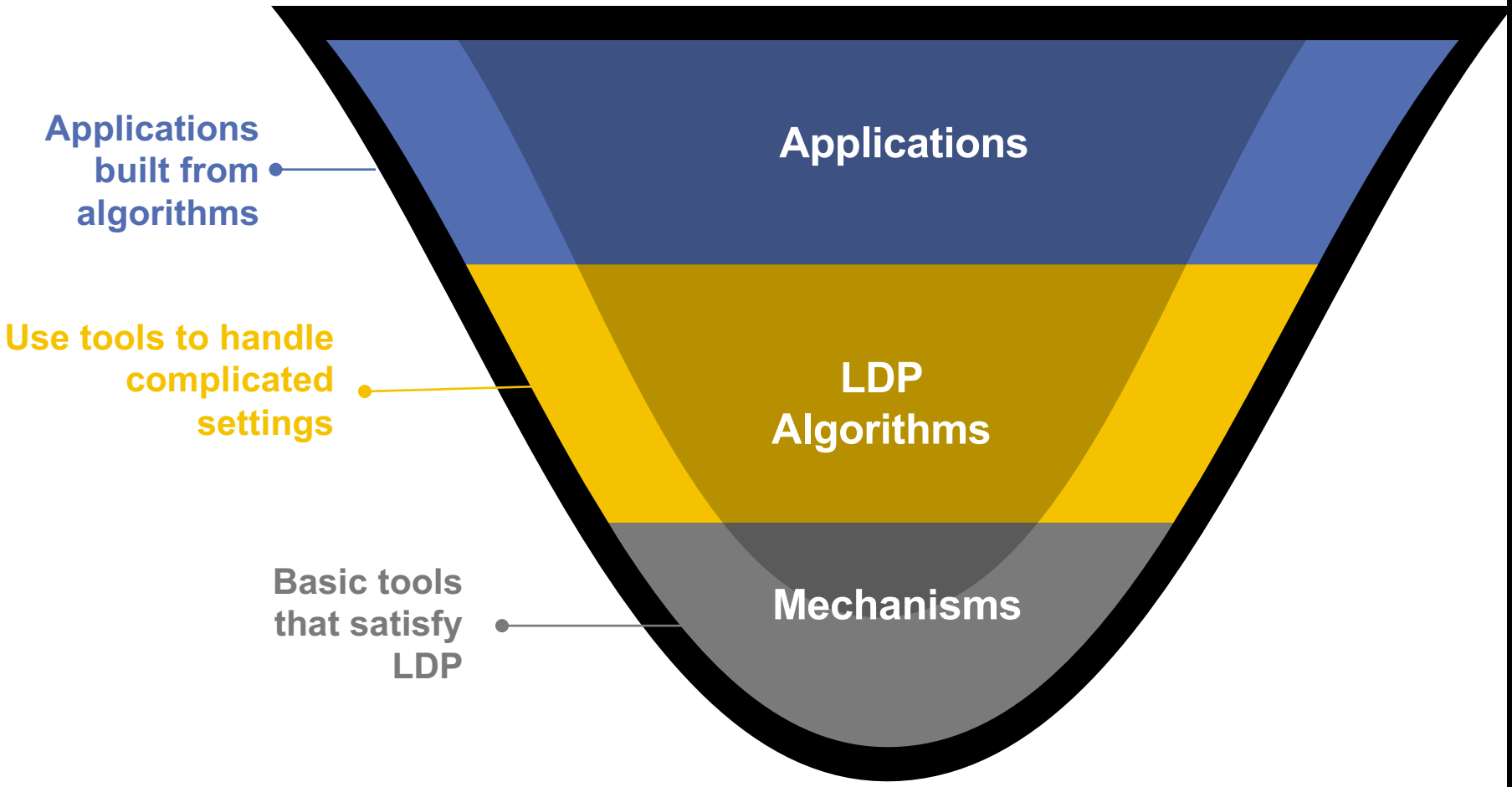
https://eng.snap.com/device-distributed-machine-learning/

OUTLINE

Local Differential Privacy for Distributed Data

1. Local Differential Privacy in Practice
2. LDP Guarantees
3. LDP Applications

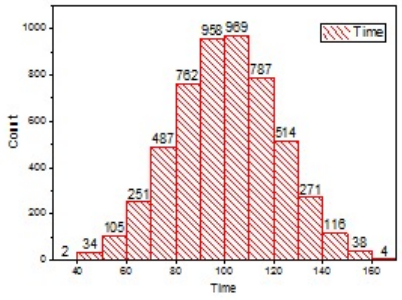
OVERVIEW


Applications built from algorithms
Use tools to handle complicated settings
Basic tools that satisfy LDP
Applications
LDP Algorithms
Mechanisms

FREQUENCY ESTIMATION

Assumption: each user has a single value x from a categorical domain D

Goal: Estimate the frequency of any value in D



Data+Noise Data+Noise Data+Noise

Trust boundary

FREQUENCY ORACLE FRAMEWORK

- $x := E(v)$

takes input value v from domain D and outputs an encoded value x

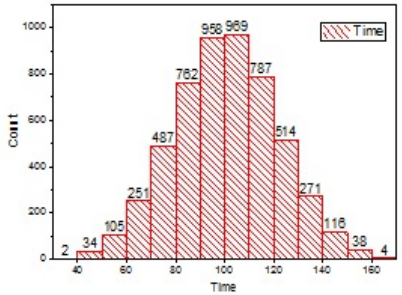
- $y := P(x)$

takes an encoded value x and outputs y.

y

- $c := Est(\{y\})$

takes reports $\{y\}$ from all users and outputs estimations $c(v)$ for any value v in domain D



P is ε -LDP iff for any v and v' from D, and any valid output y,

$$\frac{\Pr[P(E(v))=y]}{\Pr[P(E(v'))=y]} \leq e^{\varepsilon}$$

RANDOM RESPONSE (WARNER '65)

Survey technique for private questions

Survey people:

- "Do you have a disease?"

Each person:

- Flip a secret coin 
- Answer truth if head (w/p 0.5)
- Answer randomly if tail
- E.g., a patient will answer "yes" w/p 75%, and "no" w/p 25%

For any v and v' from "yes" and "no",

$$\frac{\Pr[P(v) = v]}{\Pr[P(v') = v]} \leq 3 = e^{\varepsilon}$$

Provide deniability: seeing answer, not certain about the secret.

This only handles binary attribute.
We will handle the more general setting.

To get unbiased estimation of the distribution:

- If n_v out of n people have the disease, we expect to see

  $E[\, I_v] = 0.75n_v + 0.25(n - n_v)$ "yes" answers

- $c(n_v) = \frac{I_v - 0.25n}{0.5}$ is the unbiased estimation of number of patients

CONCRETE EXAMPLE (LET'S DO MATH)

A patient will answer "yes" w/p 75%, and "no" w/p 25%

	truth	->yes	->no
yes	80	40+20	0+20
no	20	0+5	10+5

$$c(n_v) = \frac{I_v - 0.25n}{0.5}$$

observed	65	35
estimate	80	20

(SIMPLE) PROOFS

$E[c(n_v)] = n_v$

We have

- $c(n_v) = \frac{I_v - 0.25n}{0.5}$
- $E[\, I_v] = 0.75n_v + 0.25(n - n_v)$

$$E[c(n_v)] = \frac{E[I_v] - 0.25n}{0.5} = \frac{0.75n_v + 0.25(n - n_v) - 0.25n}{0.5} = n_v$$

Can be extended to other protocols

Variance can be derived similarly

PROBABILISTIC ANALYSIS

Compare the result $c(v)$ with the ground truth n_v.

$c(v)$ is a random variable

Show that $c(v)$ is unbiased: $E[c(v)] = n_v$

Compute the variance of $c(v)$: $Var[c(n_v)]$

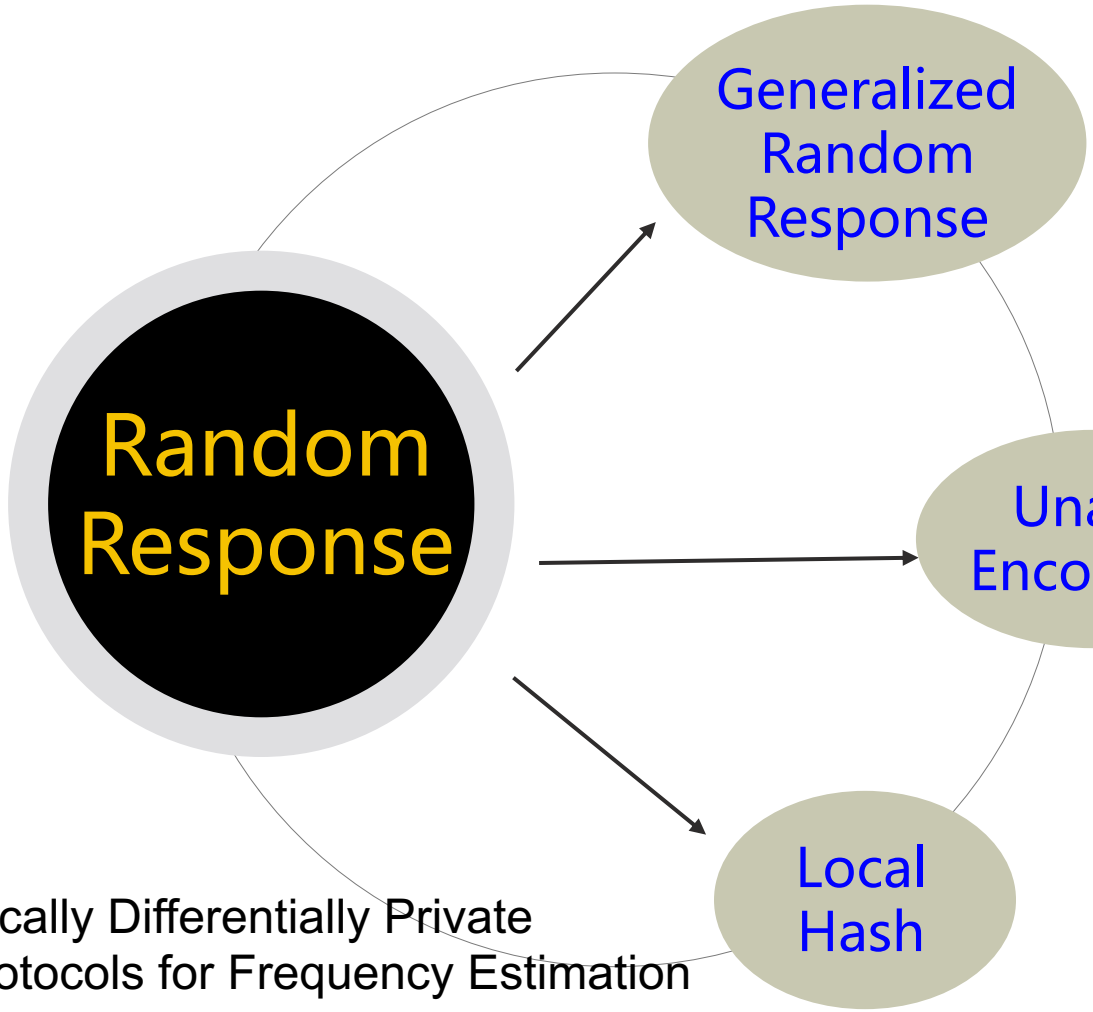
Use appropriate inequality to bound the error

- Bernstein or Hoeffding inequalities

Transform from variance to error bound

- Since $c(v)$ is a binomial variable (sum of iid Bernoulli variables)

FROM TWO TO ANY CATEGORIES



RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. Ú. Erlingsson, V. Pihur, A. Korolova, CCS 2014

Local, Private, Efficient Protocols for Succinct Histograms R. Bassily, A. Smith. STOC 2015.

Locally Differentially Private Protocols for Frequency Estimation T. Wang, J. Blocki, N. Li, S. Jha: USENIX Security 2017

GENERALIZED RR (DIRECT ENCODING)

User:

- Encode $x = v$ (suppose v from $D = \{1,2, \dots, d\}$)
- Toss a coin with bias p
- If it is head, report the true value $y = x$
- Otherwise, report any other value with probability $q = \frac{1-p}{d-1}$ (uniformly at random)
  - $p = \frac{e^{\varepsilon}}{e^{\varepsilon}+d-1}, q = \frac{1}{e^{\varepsilon}+d-1} \Rightarrow \frac{\Pr[P(\boldsymbol{v})=\boldsymbol{v}]}{\Pr[P(\boldsymbol{v}')=\boldsymbol{v}]} = \frac{p}{q} = e^{\varepsilon}$

Aggregator:

- Suppose n_v users possess value v, I_v is the number of reports on v.
- $E[I_v] = n_v \cdot p + (n - n_v) \cdot q$
- Unbiased Estimation: $c(v) = \frac{I_v - n \cdot q}{p-q}$

GENERALIZED RR (DIRECT ENCODING)

User:

- Encode x =
- Toss a coin
- If it is head,
-

Intuitively, the higher p, the more accurate

However, when d is large, p becomes small (for the same ε)

ε	$p(d=2)$	$p(d=8)$	$p(d=128)$	$p(d=1024)$
0.1	0.52	0.13	0.016	0.001
1	0.73	0.27	0.027	0.002
2	0.88	0.51	0.057	0.007
4	0.98	0.88	0.307	0.05

Ag

- v.
- $E[I_v] = n_v \cdot$
- Unbiased E

To get rid of dependency on domain size, we move to the unary encoding protocols.

UNARY ENCODING (BASIC RAPPOR)

Encode the value v into a bit string $\boldsymbol{x} \coloneqq \vec{0}, \boldsymbol{x}[v] \coloneqq 1$

- e.g., $D = \{1,2,3,4\}, v = 3,$ then $\boldsymbol{x} = [0,0,1,0]$

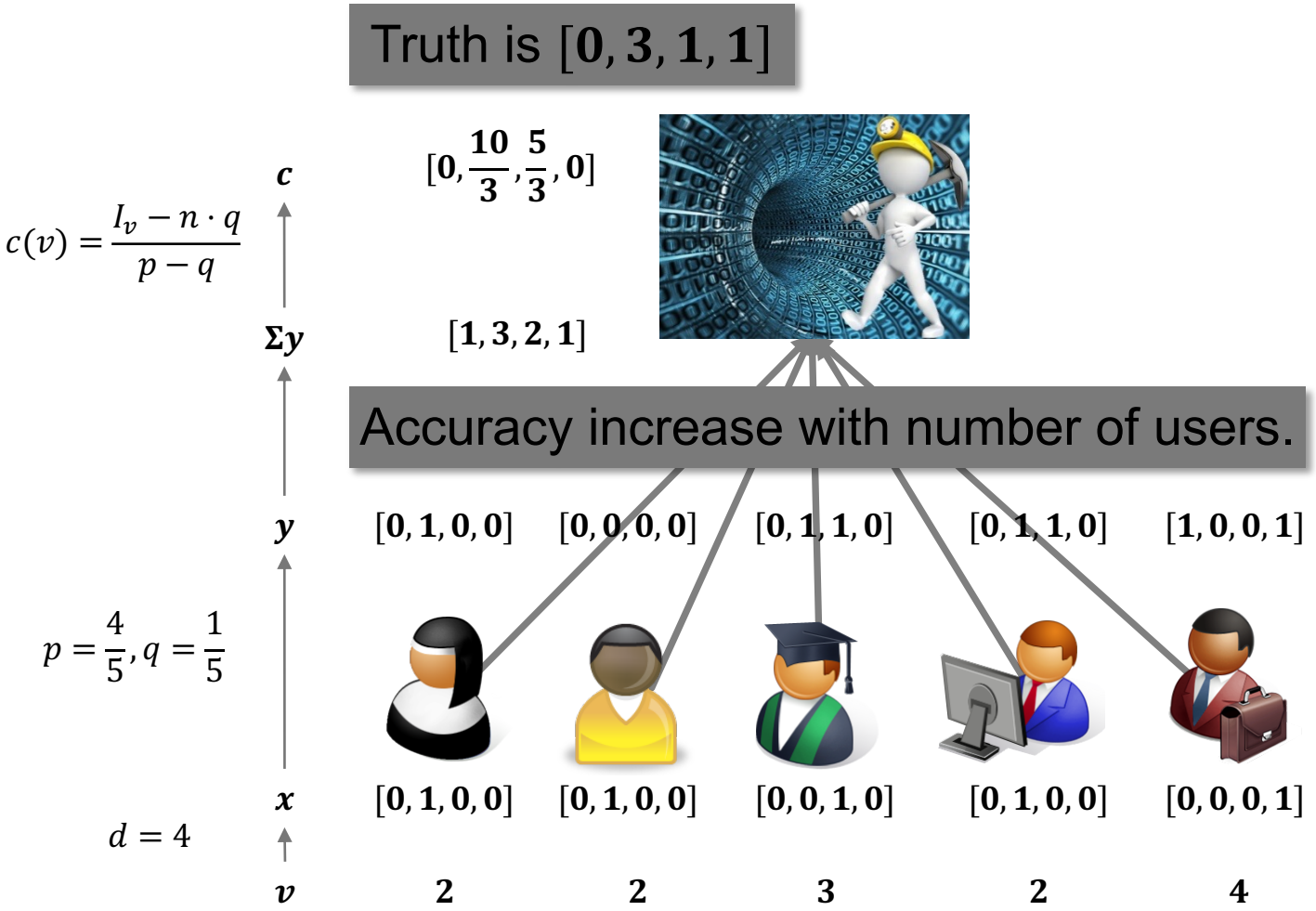
Perturb each bit, preserving it with probability p

- $p_{1\to1} = p_{0\to0} = p = \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}$ $\qquad$ $p_{1\to0} = p_{0\to1} = q = \frac{1}{e^{\varepsilon/2}+1}$
- $\Rightarrow \frac{\Pr[P(E(v))=\boldsymbol{x}]}{\Pr[P(E(v'))=\boldsymbol{x}]} \leq \frac{p_{1\to1}}{p_{0\to1}} \times \frac{p_{0\to0}}{p_{1\to0}} = e^{\varepsilon}$
  - Since $\boldsymbol{x}$ is unary encoding of $v, \boldsymbol{x}$ and $\boldsymbol{x}'$ differ in two locations

Intuition:

- By unary encoding, each location can only be 0 or 1, effectively reducing d in each location to 2. (But privacy budget is halved.)
- When d is large, UE is better than DE.

To estimate frequency of each value, do it for each bit.


Truth is [0, 3, 1, 1]
$c(v) = \frac{I_v - n \cdot q}{p - q}$
c
$[0, \frac{10}{3}, \frac{5}{3}, 0]$
Σy
[1, 3, 2, 1]
Accuracy increase with number of users.
y
[0, 1, 0, 0]
[0, 0, 0, 0]
[0, 1, 1, 0]
[0, 1, 1, 0]
[1, 0, 0, 1]
$p = \frac{4}{5}, q = \frac{1}{5}$
x
[0, 1, 0, 0]
[0, 1, 0, 0]
[0, 0, 1, 0]
[0, 1, 0, 0]
[0, 0, 0, 1]
$d = 4$
v
2
2
3
2
4

LAPLACIAN (GAUSSIAN)

Instead of using randomized response for each bit, add Laplacian (Gaussian) noise to each bit.

- Sensitivity is 2, because two vectors differ in two bits.

It is equivalent to the centralized setting, but the number of record is only 1.

The server aggregates the results.

This is worse than UE.

OUTLINE

Local Differential Privacy for Distributed Data

1. Local Differential Privacy in Practice
2. LDP Guarantees
3. LDP Applications

THE HEAVY HITTER PROBLEM

Goal: Find the k most frequent values from a large D

Scenario (Application): Find the most popular

- url
- hashtag
- new phrase

Assumption:

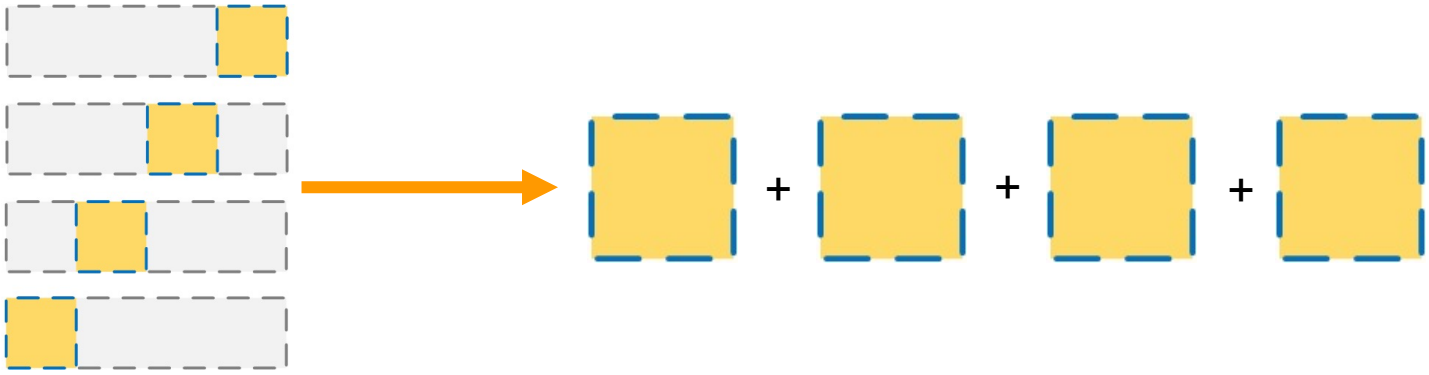
- each user has a single value x and it is represented in bits
- D is large (when D is small, frequency oracle suffices)

A FIRST SOLUTION

Simpler Goal: Find one most frequent value from D

Idea:

- Users are partitioned into four groups
- Each user reports one portion of its string (segment)
- Server queries FO to one find frequent pattern in each segment
- Concatenate the four frequent patterns



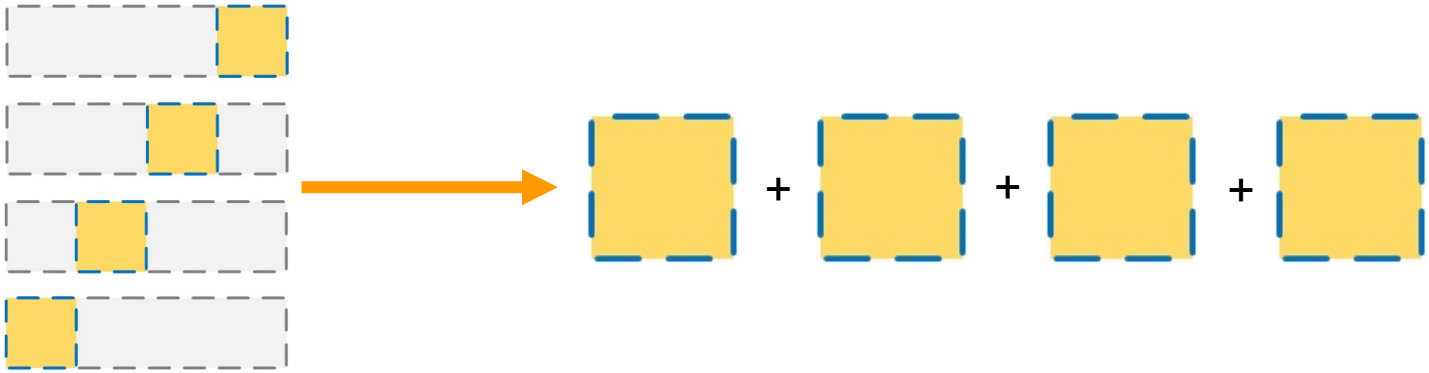
A FIRST SOLUTION

Simpler Goal: Find one most frequent value from D

Idea:

- Users are partitioned into four groups
- Each user reports one portion of its string (segment)
- Server que segment
- Concatena

Drawback:
Composing the four segment candidate sets gives a very large set of results.

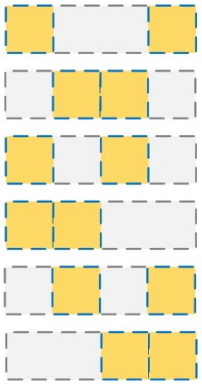


SEGMENT PAIR METHOD

Each user reports a pair of two randomly chosen segments.

A-priori principle:

- A pair of segments is frequent iff both segments are frequent
- A string is frequent iff any pair of segments is frequent



Step 1: For each location, test all possible segment

Step 2: For each pair, test all frequent segment pairs

Step 3: Build a k-partite graph where

- each node represents a frequent segment
- each edge represents a frequent segment pair

Step 4: Find cliques in the graph (heavy hitter candidates)

Step 5: Estimate frequencies of the heavy hitters

Drawback:
There are many possible pairs, accuracy for each group is limited ($Var[c(v)/n] = \frac{4e^{\varepsilon}}{n\cdot(e^{\varepsilon}-1)^2}$)

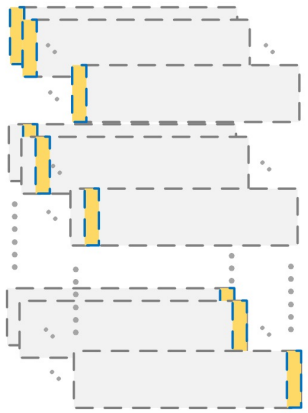
MULTIPLE CHANNEL METHOD

Suppose there is only one heavy hitter, we can afford the Cartesian product, which contains only one element.

Use multiple channels and isolate heavy hitters.

Each user reports a bit in each channel:

- In channel $H(v)$, report $v[l]$;
- In other channels, report a uniformly random bit.



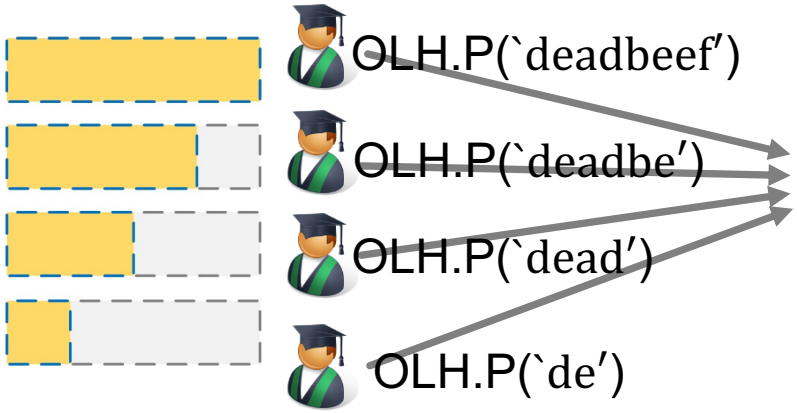
Aggregator identifies the dominant bits in each channel

Estimate frequencies of the heavy hitters

Drawback:
To avoid collision, many ($n^{1.5}$) channels are used.
Number of users in each channel is limited.
Computational cost is high.

PREFIX EXTENDING METHOD

- Start from a prefix, and gradually extend this prefix.
- Identify the frequent patterns for a small prefix first, and then extend to a larger prefix.
- Result for the last group can be used for frequency estimation



OLH.Q(`deadbe ∗∗ ′)->`deadbeef'
OLH.Q(`dead ∗∗ ′)->`deadbe'
OLH.Q(`de ∗∗ ′)->`dead'
OLH.Q(` ∗∗ ′)->`de'

$v =$ `deadbeef′

FREQUENT ITEMSET MINING

- Can be used for association rule mining .etc

Report under LDP

Each user has a set of values

{a, c, e} {b, e} {a, b, e} {a, d, e} {a, b, c, d, e, f}

- The goal is to find the frequent *singletons* and *itemsets*
- Top-3 singletons: e(5), a(4), b(3)
- Top-3 itemsets: {e}(5), {a}(4), {a, e}(4)

FREQUENT ITEMSET MINING

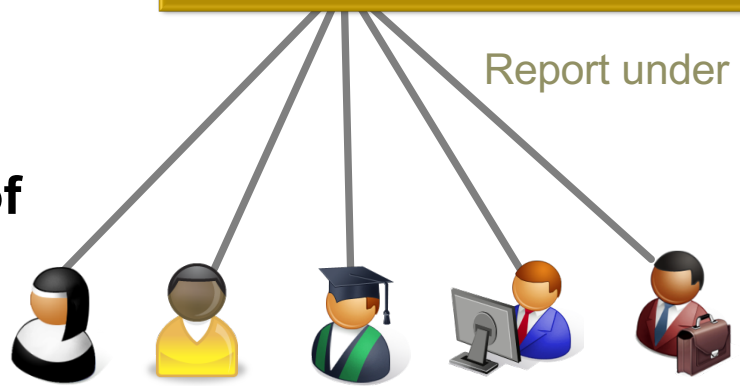
- Can be used for association rule mining etc

Strawman Method:
- Encode the itemset as a value in a bigger domain (of size 2^d).

Disadvantage:
- Cannot scale.
- If an item is contained in many infrequent itemsets, it will not be captured

The goal is to find the frequent *singletons* and *itemsets*

Each user has a set of values



{a, c, e} {b, e} {a, b, e} {a, d, e} {a, b, c, d, e, f}

- Top-3 singletons: e(5), a(4), b(3)
- Top-3 itemsets: {e}(5), {a}(4), {a, e}(4)

Challenges:
1. Each user has multiple items
2. Each user's itemset size is different

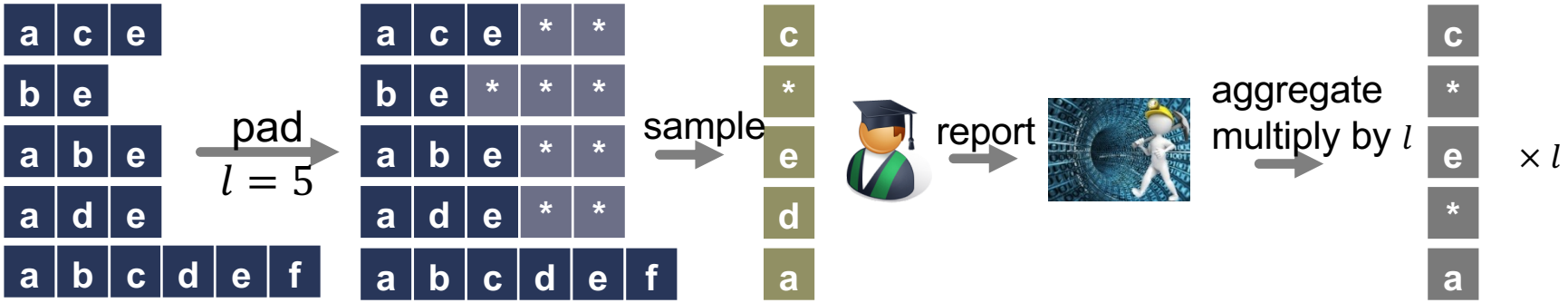
PAD AND SAMPLE FREQUENCY ORACLE

Each user's itemset size is different

- Pad it to a fixed length l

Each user now has l items (or more)

- Sample one at uniform random
- Report via LDP (e.g., using Random Response)



Idea: when many users have the item, it is sampled frequently; and the server can capture that.

LDPMINER

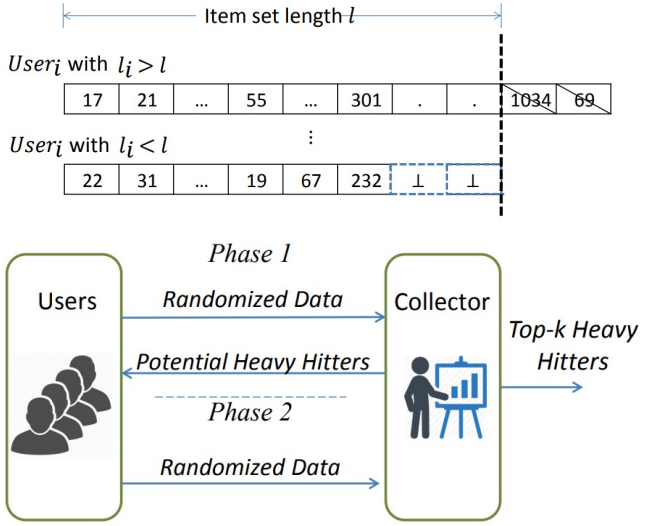
Observations
1. The value of l affects error in two ways.
2. Sampling may have a privacy amplification effect.

Phase 1 (identify candidates)

- Pad to l
  - l is the 90 percentile of the size distribution
- Randomly select one
- Report
- Potential $2k$ frequent items returned

Phase 2 (estimate frequency)

- Intersects v with the $2k$ items
- Pad to $2k$
  - Ensures no missed item
- Randomly select one
- Report

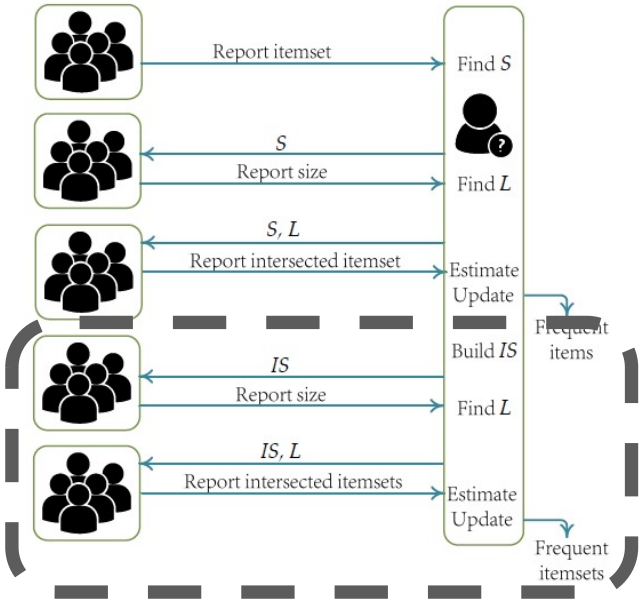


Could find frequent items only. Left finding frequent itemsets as an open problem.

SET-VALUE ITEM MINING

- Phase 1 (identify candidates)
  - Randomly select one ($l = 1$)
  - Report
  - Potential $2k$ frequent items returned
- Phase 2 (estimate length distribution)
  - Intersects $\boldsymbol{v}$ with the $2k$ identified ones
  - Report the size
  - The 90-percentile l is returned
- Phase 3 (estimate frequency)
  - Pad to l
  - Randomly select one
  - Report via Adaptively chosen FO

SVIM (Set Value Item Mining)



Itemset Mining (more in paper)

With the new design, SVIM can identify 3× more frequent items, with 3 magnitudes less errors.

MORE LDP APPLICATIONS

Beyond heavy hitters and simple statistics, many LDP applications:

- Text and language modelling
- Marginal distributions and correlation
- Spatial data
- Graphs and social networks
- Machine learning
- Across all these LDP applications, some common features emerge

TEXT AND LANGUAGE MODELLING

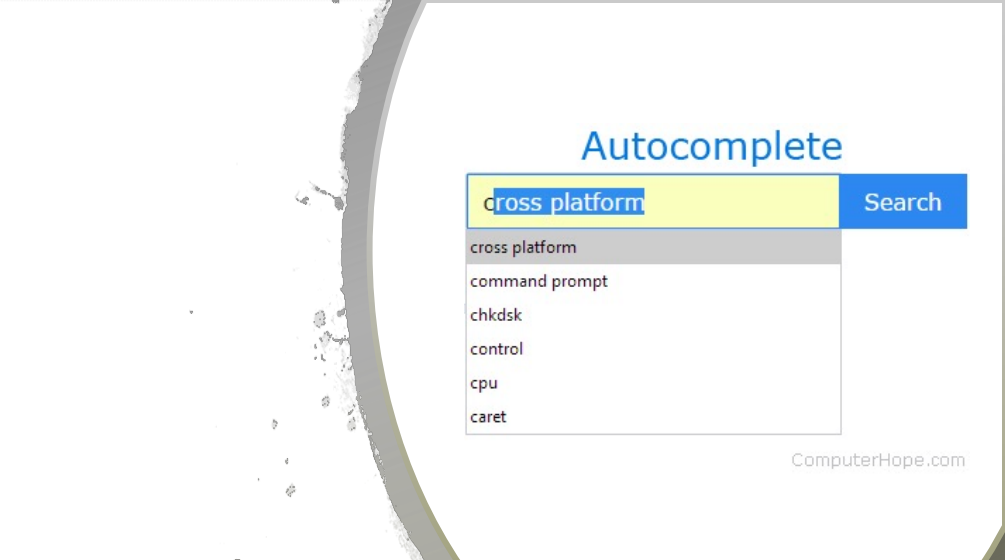
Text is among the most sensitive datasets.

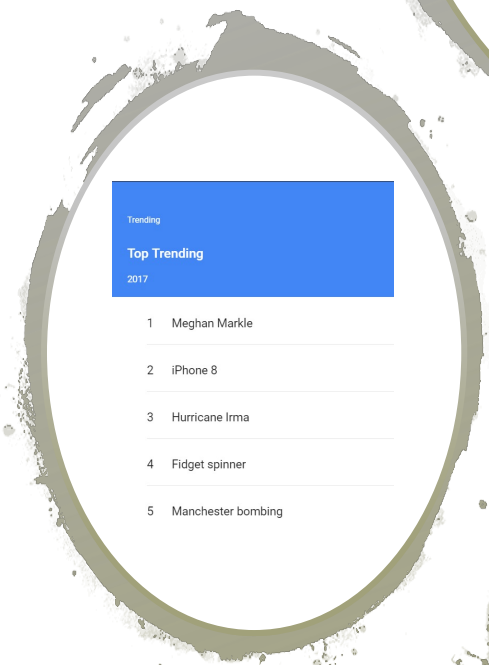
Many reasons to analyze text.

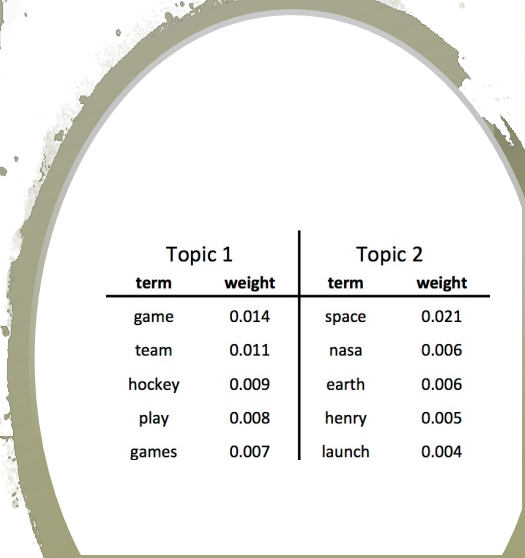
- Understand evolving use of language
- Understand service usage
- Improve mobile users input experience

Data comes from a large (hierarchical / unbounded) domain

- Google tracking popular URLs, which are long text strings
- Browsers locally caching the most popular search queries and their URLs reducing number of server round trips.







Topic 1		Topic 2	
term	weight	term	weight
game	0.014	space	0.021
team	0.011	nasa	0.006
hockey	0.009	earth	0.006
play	0.008	henry	0.005
games	0.007	launch	0.004

SPATIAL DATA

People's geographic locations can be very privacy sensitive

- Canonical example: presence at a health clinic
- But we want to capture population location distribution
- Applications: Traffic, demographic analysis

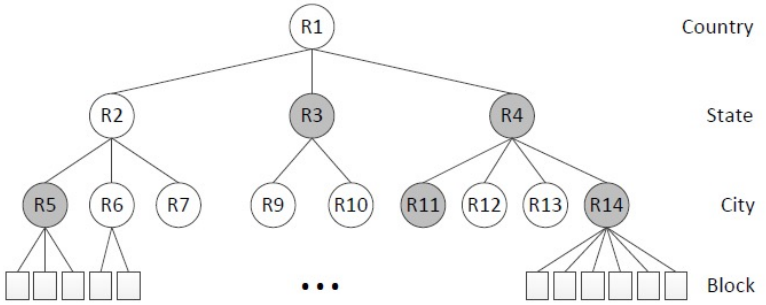
First attempt: impose a hierarchical grid structure and count

- Similar ideas successful in the centralized model
- Identify heavy regions: a heavy hitter problem
- Could be solved via frequency oracle (again)!

Limitations:

- Doesn't account for varying spatial density (the difference between central London vs outskirts)
- Unable to account user preferences: "I don't mind sharing my country but not my post code!"

SPATIAL DATA



Private Spatial Data Aggregation used a relaxation of LDP

Impose a (semantic) hierarchy over locations
E.g. Country / State / City / Block

- Each user specifies which level they can be located within
- Restrict user's LDP definition to locations within the same level
- Can apply a modified frequency oracle (FO):
  locations outside a user's 'safe zone' are known to be empty

More details:

- Can easily allow a 'personalized' privacy parameter ε_i
- Group (cluster) users together to run fewer FO invocations

SOCIAL NETWORK DATA

Social graphs may have a lot of sensitive valuable information about a person's ties.

As for central DP, there are alternative LDP definitions for graphs:

- Node LDP: LDP definition applied to whole adjacency list of a **user**
- Edge LDP: LDP restricted to adjacency lists that differ in one **edge**

Goal: release social network data as graph models

- Hard enough in the central DP model, so only harder with LDP
- Aim to release parameters of a statistical graph model

Results so far only for the (more relaxed) edge LDP definition

RECOMMENDER SYSTEMS

Given a collection of user ratings, we want to recommend items

- Generalizing 'people who liked that, also liked this…'

$n \times m$ matrix of ratings R, where $r_{i,j}$ is rating of user i for item j

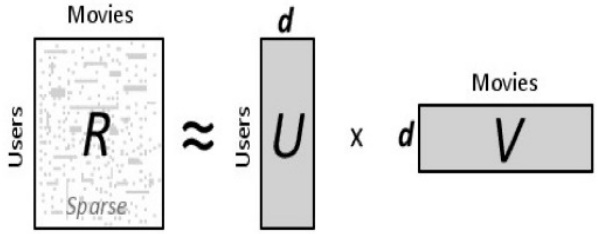
- Typically, **R** is large and sparse
- Write $\mathbf{y_{ij}=1}$ when user i has rated item j, 0 otherwise
- Aim to (approximately) factorize **R = UV** over d-dimensional space

Loss function here is $\sum_{ij} y_{ij}(r_{ij} - u_i^T v_j)^2$ (+ regularization terms)

- $\mathbf{u_i}$ is user's (d-dimensional) row, **v** is item's column in factor space
- Can solve (non-privately) by **gradient descent** over **U** and **V**

LDP version: aim for privacy at user granularity

- Similar approaches have been tried in centralized case



COMMON INGREDIENTS

Finding a good (statistical) model is often the main challenge

Seek models where each parameter is supported by many users

Often we can map onto one of a few basic problems

- Most methods use a frequency oracle, or vector generalizations

A few other tricks are seen multiple times:

- Sample/split users into batches.
- “Clip” data to ranges to reduce noise that must be added .
- Use multiple rounds to adaptively ask queries.
- Transform data to deal spread the signal.
- Reduce dimensionality.
- Enforce consistency constraints when possible.

IMMEDIATE OPEN PROBLEMS FOR LDP

Take any data type or analysis goal and ask "Can we handle this under the LDP model?":

- Sentiment analysis for (private) reviews: LDP-TripAdvisor?
- Trajectory analysis of GPS movements: LDP-RunKeeper?
- Social network data analysis: LDP-Facebook?

More challenging to apply to richer, unstructured data

- E.g. Free text in written medical notes
- E.g. Multimedia – images and videos
- The semantics of privacy are not yet well defined here!

Algorithm engineering: make LDP widely available

- RAPPOR is open source, but there's no easy toolkit for LDP yet

GENERALIZING LDP

Additive error probability

- Centralized DP allows an additional parameter δ, which relaxes the requirement to $\Pr[X \in S|D] \leq e^{\varepsilon} \Pr[X \in S|D'] + \delta$
- LDP can be relaxed similarly, but it makes no difference (in theory)
- What are the practical consequences of this relaxation?

Multiple rounds of data collection

- Some LDP protocols collect data once, others interact with users
  - Queries in subsequent rounds depend on prior rounds
- What is the tradeoff between rounds and accuracy/communication?

REFLECTING ON LDP

Local Differential Privacy is a big success for privacy research

- Adopted by Google, Apple, Microsoft and more for deployment
- Deployments affecting (hundreds of) millions of users
- In contrast, centralized DP has smaller success (but, see US Census)

However, there are reasons to pause and reflect:

- LDP only works when you can rely on millions of active participants
- Companies using LDP also gather vast amounts of raw data directly!
- Privacy settings are not very tight: deployed ε ranges from 0.5 to 8+

OPENING UP PRIVATE DATA

Deployments of LDP are still tightly controlled by aggregator

- Google/MSR/Apple controls the code on your device
- User reports typically sent over secure channel to aggregator

Could there be a more 'open' implementation of LDP?

- Users could publish their LDP outputs
- Multiple aggregators could perform independent analysis
- Would require a big change in architecture and adoption!

LDP data

ACKNOWLEDGMENTS

ILLINOIS INSTITUTE OF TECHNOLOGY

Note: the slides in this lecture are adapted based on materials created by

- **Graham Cormode** (University of Warwick/Alan Turing Institute, UK)
- **Somesh Jha** (University of Wisconsin Madison, USA)
- **Tejas Kulkarni** (University of Warwick, UK)
- **Ninghui Li** (Purdue University, USA)
- **Divesh Srivastava** (AT&T Labs-Research, USA)
- **Tianhao Wang** (Purdue University, USA)