Discovering Frequent Patterns in Sensitive Data

Abhradeep Guha Thakurta
azg161@cse.psu.edu

Department of Computer Science and Engineering
Pennsylvania State University

Joint work with Raghav Bhaskar and Srivatsan Laxman, Microsoft Research India and Adam Smith, Pennsylvania State University
Frequent Pattern Mining (FPM)

- **Widely used** tool for exploratory data analysis
- **Application:** Recommendation systems (e.g. Amazon, Wal-Mart)

Two variants of FPM:
- **Threshold**: return all patterns with frequency above $\theta$
- **Top-$k$**: return $k$ most frequent patterns
Top-\(k\) Frequent Pattern Mining (FPM)

- **Notation.**
  - **\(U\):** Universe of patterns
  - **\(T\):** Data set of \(n\) records
  - **Frequency** of a pattern
    \[
    \text{Frequency} = \frac{\text{# of records in which it appears}}{n}
    \]
  - **Output:** The \(k\) most frequent patterns in the data set \(T\) and their frequencies

---

**EMR Data**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>427.9DX, 44140PX, 44120PX, 93503PX, 276.3DX, 518.5DX</td>
<td>12345</td>
</tr>
<tr>
<td>373.2DX, 92002PX, 427.9DX, 410.91DX, 44120PX</td>
<td>12333</td>
</tr>
<tr>
<td>573.9DX, 276.3DX</td>
<td>12222</td>
</tr>
<tr>
<td>92002PX, 155.2DX</td>
<td>9876</td>
</tr>
<tr>
<td>373.2DX, 410.91DX</td>
<td>9777</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>155.2DX, 570DX</td>
<td>7654</td>
</tr>
</tbody>
</table>

**FPM Output**
The data set $T$ may contain potentially sensitive information about an individual.
Need for privacy

- The data set $T$ may contain potentially sensitive information about an individual
- Want to protect the privacy of individual records in $T$
  - e.g., Medical records
- Caution: Releasing exact results does not preserve privacy
The data set $T$ may contain potentially sensitive information about an individual.
Want to protect the privacy of individual records in $T$
  * e.g., Medical records
Caution: Releasing exact results does not preserve privacy
  * e.g., it is known that inverse FPM is NP-hard [Mie03]
    * Thus, it is hard to recover the entire data set
    * But it might be easy to recover specific pieces of information
The data set $T$ may contain potentially sensitive information about an individual.

Want to protect the privacy of individual records in $T$:
- *e.g.*, Medical records

**Caution:** Releasing exact results does **not** preserve privacy:
- *e.g.*, it is known that inverse FPM is NP-hard [Mie03]
  - Thus, it is hard to recover the entire data set
- But it might be easy to recover specific pieces of information
Example of privacy breach for FPM

**T_1**

<table>
<thead>
<tr>
<th>Data Set</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>427.9DX, 44140PX, 44120PX, 93503PX, 276.3DX, 518.5DX</td>
<td></td>
</tr>
<tr>
<td>373.2DX, 92002PX, 427.9DX, 410.91DX, 44120PX</td>
<td></td>
</tr>
<tr>
<td>573.9DX, 155.2DX, 276.3DX, 44120PX, 570DX</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
</tr>
<tr>
<td>92002PX, 573.9DX, 427.9DX</td>
<td></td>
</tr>
</tbody>
</table>

**T_2**

<table>
<thead>
<tr>
<th>Data Set</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>373.2DX, 92002PX, 427.9DX, 410.91DX, 44120PX</td>
<td></td>
</tr>
<tr>
<td>573.9DX, 155.2DX, 276.3DX, 44120PX, 570DX</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
</tr>
<tr>
<td>92002PX, 573.9DX, 427.9DX</td>
<td></td>
</tr>
</tbody>
</table>

**PATTERN** | **FREQUENCY**

| 427.9DX, 518.5DX | 12345 |
| 427.9DX, 44120PX | 12333 |
| 573.9DX, 276.3DX | 12222 |
| 92002PX, 155.2DX | 9876  |
| 373.2DX, 410.91DX| 9777  |
| :               |      |
| 155.2DX, 570DX  | 7654  |

**PATTERN** | **FREQUENCY**

| 427.9DX, 518.5DX | 12344 |
| 427.9DX, 44120PX | 12332 |
| 573.9DX, 276.3DX | 12222 |
| 92002PX, 155.2DX | 9876  |
| 373.2DX, 410.91DX| 9777  |
| :               |      |
| 155.2DX, 570DX  | 7654  |

First row of T_1 must contain 427.9DX, 518.5DX, 44120PX.
This work

- Provides algorithms for releasing high-frequency patterns while providing a rigorous privacy guarantee
  - We use differential privacy [DMNS06]
This work

- Provides algorithms for releasing high-frequency patterns while providing a rigorous privacy guarantee
  - We use differential privacy [DMNS06]
- Two algorithms:
  - Score perturbation-based algorithm (adapting [DMNS06])
  - Exponential sampling-based algorithm (adapting [MT07])
This work

- Provides algorithms for releasing high-frequency patterns while providing a rigorous privacy guarantee
  - We use differential privacy [DMNS06]
- Two algorithms:
  - Score perturbation-based algorithm (adapting [DMNS06])
  - Exponential sampling-based algorithm (adapting [MT07])
- Rigorous privacy and utility guarantees
- The experimental results support theoretical predictions
Differential Privacy

- Output should not reveal information about any individual record
- Informally, the output of FPM should not change by much by changing one record of $T$
Differential Privacy

- Output should not reveal information about any individual record
- Informally, the output of FPM should not change by much by changing one record of $T$

[DMNS06] A randomized algorithm $A$ is $\epsilon$-differentially private if for all data sets $T, T' \in \mathcal{D}^n$ differing in at most one record and for all events $\mathcal{O} \subseteq \text{Range}(A)$:

$$\Pr[A(T) \in \mathcal{O}] \leq e^\epsilon \Pr[A(T') \in \mathcal{O}]$$
Why differential privacy?

- Protects against arbitrary side information
  - Adversary learns the same thing whether or not Alice’s record was there in the data set

Differentially private algorithms exist for learning [BDMN05, KLNRS08], statistical inference [DL09], recommendation systems [MM09].
Why differential privacy?

- Protects against arbitrary side information
  - Adversary learns the same thing whether or not Alice’s record was there in the data set
- Protects against attacks like re-identification, attribute linkage etc
Why differential privacy?

- Protects against arbitrary side information
  - Adversary learns the same thing whether or not Alice’s record was there in the data set
- Protects against attacks like re-identification, attribute linkage etc
- Widely studied since 2006

Differentially private algorithms exist for learning [BDMN05, KLNRS08], statistical inference [DL09], recommendation systems [MM09].
Why differential privacy?

- Protects against arbitrary side information
  - Adversary learns the same thing whether or not Alice’s record was there in the data set

- Protects against attacks like re-identification, attribute linkage etc

- Widely studied since 2006

- Differentially private algorithms exist for
  - learning [BDMN05,KLNRS08], statistical inference [DL09], recommendation systems [MM09]
Randomized response: Each entry in the data set $T$ is independently randomized before allowing data mining algorithm to access it.
Randomized response: Each entry in the data set $T$ is independently randomized before allowing data mining algorithm to access it.

[AH05],[EGS03] considered randomized response in the context of FPM.
Related work [AH05],[EGS03]

- **Randomized response:** Each entry in the data set $T$ is independently randomized before allowing data mining algorithm to access it.

- [AH05],[EGS03] considered randomized response in the context of FPM:
  - Work of [AH05] is a generalization of [EGS03]
  - Privacy guarantees are equivalent to differential privacy.
Randomized response: Each entry in the data set $T$ is independently randomized before allowing data mining algorithm to access it

[AH05],[EGS03] considered randomized response in the context of FPM
- Work of [AH05] is a generalization of [EGS03]
- Privacy guarantees are equivalent to differential privacy
- No formal utility guarantees
Randomized response: Each entry in the data set $T$ is independently randomized before allowing data mining algorithm to access it.

[AH05],[EGS03] considered randomized response in the context of FPM.

- Work of [AH05] is a generalization of [EGS03].
- Privacy guarantees are equivalent to differential privacy.
- No formal utility guarantees.
- Our algorithms perform consistently better (in experiments).
By definition, any non-trivial differentially private algorithm has to introduce error in the output.
By definition, any non-trivial differentially private algorithm has to introduce error in the output.

Differentially private FPM will:

- insert low frequency patterns in the output
- remove high frequency patterns from the output
- perturb the frequencies of the patterns being output

An “useful” FPM output should have small error.
Need for approximate utility

- By definition, any non-trivial differentially private algorithm has to introduce error in the output.
- Differentially private FPM will:
  - insert low frequency patterns in the output
  - remove high frequency patterns from the output
  - perturb the frequencies of the patterns being output.
- An “useful” FPM output should have small error.
- To quantify utility, we:
  - introduce a notion of “approximate” top frequent patterns
  - evaluate our algorithms both theoretically and empirically with respect to this notion.
Let $q_k$ be the $k^{th}$ highest frequency based on data set $T$.

An FPM output is $(\gamma, \eta)$-useful if:

- **(Soundness)** No pattern in the output has frequency less than $(q_k - \gamma)$
- **(Completeness)** Every pattern with frequency greater than $(q_k + \gamma)$ is in the output
- **(Precision)** The reported frequency for every pattern in the output is within $\eta$ of its true frequency
Let $q_k$ be the $k^{th}$ highest frequency based on data set $T$.

An FPM output is $(\gamma, \eta)$-useful if:

- **(Soundness)** No pattern in the output has frequency less than $(q_k - \gamma)$.
- **(Completeness)** Every pattern with frequency greater than $(q_k + \gamma)$ is in the output.
- **(Precision)** The reported frequency for every pattern in the output is within $\eta$ of its true frequency.
Approximate utility for FPM

Let $q_k$ be the $k^{th}$ highest frequency based on data set $T$. An FPM output is $(\gamma, \eta)$-useful if:

- (Soundness) No pattern in the output has frequency less than $(q_k - \gamma)$
- (Completeness) Every pattern with frequency greater than $(q_k + \gamma)$ is in the output
- (Precision) The reported frequency for every pattern in the output is within $\eta$ of its true frequency
Score perturbation-based algorithm
Score perturbation-based algorithm

Frequency pairs of attributes, sorted by true frequency

{B,C} {A,B} {E,F} {B,D} {C,F} {A,G} {D,F} {B,G}
Score perturbation-based algorithm

Top 5 frequencies

Pairs of attributes, sorted by true frequency

{B,C} {A,B} {E,F} {B,D} {C,F} {A,G} {D,F} {B,G}
Score perturbation-based algorithm

Frequency

Pairs of attributes, sorted by true frequency:

- {B, C}
- {A, B}
- {E, F}
- {B, D}
- {C, F}
- {A, G}
- {D, F}
- {B, G}

- Blue circles represent true frequency.
- Red circles represent noisy frequency.

Pairs of attributes, sorted by true frequency:
Score perturbation-based algorithm

Frequency
pairs of attributes, sorted by true frequency
{B,C} {A,B} {E,F} {B,D} {C,F} {A,G} {D,F} {B,G} = noisy frequency

= true frequency

pairs of attributes, sorted by true frequency

Frequency

{B,C} {A,B} {E,F} {B,D} {C,F} {A,G} {D,F} {B,G}
Score perturbation-based algorithm

Output:
List of patterns and noisy frequencies (with fresh noise)
Details of the algorithm

How much noise?

- Laplace noise with $\lambda = \Theta \left( \frac{k}{\epsilon n} \right)$
- $\text{Lap}(\lambda) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}}$
Details of the algorithm

How much noise?
- Laplace noise with $\lambda = \Theta \left( \frac{k}{\epsilon n} \right)$
- $Lap(\lambda) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}}$

Straightforward implementation needs time $O(|U|)$
- Might be exponentially large
- e.g., Frequent Itemset Mining: $m$ items $\rightarrow 2^m$ itemsets
Details of the algorithm

- **How much noise?**
  - Laplace noise with \( \lambda = \Theta \left( \frac{k}{\epsilon n} \right) \)
  - \( \text{Lap}(\lambda) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}} \)

- **Straightforward implementation needs time** \( O(|U|) \)
  - Might be exponentially large
  - *e.g.*, Frequent Itemset Mining: \( m \) items \( \rightarrow 2^m \) itemsets

- **Our implementation takes time** “roughly” \( \propto k \)
**Theorem:** The algorithm is $\epsilon$-differentially private
Theorem: The algorithm is $\epsilon$-differentially private

- **Naive analysis:**
  - Consider the frequencies of $|U|$ patterns as a vector of length $|U|$.
  - Assure privacy for each element of the vector individually using [DMNS06] style analysis.
  - Requires $\Theta \left( \frac{|U|}{\epsilon n} \right)$ noise for $\epsilon$-differential privacy.

- **Our analysis:** $\Theta \left( \frac{k}{\epsilon n} \right)$ noise suffices.
Theorem (Utility): For all $\rho > 0$: with probability at least $1 - \rho$, the output is $(\gamma, \eta)$-useful, where
\[
\gamma = \frac{8k}{\epsilon n} \left( \log \left| \frac{U}{\rho} \right| \right)
\]
and
\[
\eta = \frac{2k}{n\epsilon} \ln \left( \frac{k}{\rho} \right)
\]

Take away: Privacy does not degrade the utility by too much.
Experimental results (Frequent Itemset Mining)

- All the data sets from the FIMI repository ([http://fimi.cs.helsinki.fi/](http://fimi.cs.helsinki.fi/))
- Accurate results for a wide range of parameters \((k, \epsilon, \gamma, \rho)\)
- Error rates match theoretical predictions
- This talk: variation of FNR (False Negative Rate) with \(\epsilon\)
  - Note that False Positive Rate is not an effective measure of utility because the # of true negatives is inherently high
Score perturbation-based algorithm: Variation of FNR vs $\epsilon$

**Parameters:** $\rho = 0.1$, $k = 10$ and the size of the itemsets mined = 3

(N.B. Av. transaction length: **Connect**: 44, **Kosarak**: 8.09)
Randomized response [AH05]

- [AH05] introduces the FRAPP framework
- DET-GD and RAN-GD are two algorithms under the FRAPP framework
- Use the CENSUS data set used by [AH05]
Conclusion

This work:

- First work towards providing both formal privacy and utility guarantees for FPM
Conclusion

- This work:
  - First work towards providing both formal privacy and utility guarantees for FPM
  - Two algorithms which provide a strong notion privacy and are accurate on a wide range of data sets
  - Far more accurate than previous, randomized-response algorithms
  - Our algorithms are also useful for the more general problem of private ranking [KKMN09, GMW+09]

- In the paper:
  - Another algorithm: Exponential sampling-based
  - Implementation details for both the algorithms
  - Comprehensive experimental results

- Open Problem: Can we have differentially private algorithms for other high dimensional problems?
Conclusion

This work:

- First work towards providing both formal privacy and utility guarantees for FPM
- Two algorithms which provide a strong notion privacy and are accurate on a wide range of data sets
- Far more accurate than previous, randomized-response algorithms

Another algorithm: Exponential sampling-based
Implementation details for both the algorithms
Comprehensive experimental results

Open Problem: Can we have differentially private algorithms for other high dimensional problems?
This work:

- First work towards providing both formal privacy and utility guarantees for FPM
- Two algorithms which provide a strong notion privacy and are accurate on a wide range of data sets
- Far more accurate than previous, randomized-response algorithms
- Our algorithms are also useful for the more general problem of private ranking [KKMN09, GMW^+09]
This work:
- First work towards providing both formal privacy and utility guarantees for FPM
- Two algorithms which provide a strong notion privacy and are accurate on a wide range of data sets
- Far more accurate than previous, randomized-response algorithms
- Our algorithms are also useful for the more general problem of private ranking [KKMN09, GMW+09]

In the paper:
- Another algorithm: Exponential sampling-based
- Implementation details for both the algorithms
- Comprehensive experimental results
This work:
- First work towards providing both formal privacy and utility guarantees for FPM
- Two algorithms which provide a strong notion privacy and are accurate on a wide range of data sets
- Far more accurate than previous, randomized-response algorithms
- Our algorithms are also useful for the more general problem of private ranking [KKMN09, GMW⁺09]

In the paper:
- Another algorithm: Exponential sampling-based
- Implementation details for both the algorithms
- Comprehensive experimental results

Open Problem: Can we have differentially private algorithms for other high dimensional problems?
Shipra Agrawal and Jayant R. Haritsa.  
A framework for high-accuracy privacy-preserving mining.  

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith.  
Calibrating noise to sensitivity in private data analysis.  

Michaela Götz, Ashwin Machanavajjhala, Guozhang Wang, Xiaokui Xiao, and Johannes Gehrke.  
Privacy in search logs.  

Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas.  
Releasing search queries and clicks privately.  

Taneli Mielikäinen.  
On inverse frequent set mining.  

Frank McSherry and Kunal Talwar.  
Mechanism design via differential privacy.  
Data set $T$

Given universe of patterns $U$

Frequencies of all the patterns in $U$

Sample $k$ patterns without replacement s.t.
Pr[selecting pattern $i$] $\propto \exp(q_T(i)\epsilon/2k)$

Add Lap($2k/\epsilon n$) noise to the frequencies of the patterns picked

Output the patterns picked and their noisy frequencies
Analysis

- The privacy guarantee is same as score perturbation-based algorithm
- The utility guarantee is better by a small constant factor
- The algorithm runs in $O(|U| \log^* |U|)$
### Exponential sampling-based algorithm: Running time on various data sets

<table>
<thead>
<tr>
<th>Data sets</th>
<th>FIM (ms)</th>
<th>$\epsilon/2 = 0.06$</th>
<th>$\epsilon/2 = 0.7$</th>
<th>$\epsilon/2 = 1.3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>accidents</td>
<td>897</td>
<td>878 (1.0)</td>
<td>875 (1.0)</td>
<td>895 (1.0)</td>
</tr>
<tr>
<td>chess</td>
<td>61</td>
<td>-</td>
<td>77 (1.3)</td>
<td>89 (1.4)</td>
</tr>
<tr>
<td>connect</td>
<td>273</td>
<td>364 (1.3)</td>
<td>284 (1.0)</td>
<td>300 (1.1)</td>
</tr>
<tr>
<td>kosarak</td>
<td>1077</td>
<td>1073 (1.0)</td>
<td>1084 (1.0)</td>
<td>1058 (0.98)</td>
</tr>
<tr>
<td>mush</td>
<td>105</td>
<td>10542 (100.1)</td>
<td>78 (0.8)</td>
<td>125 (1.2)</td>
</tr>
<tr>
<td>pumsb</td>
<td>386</td>
<td>834 (2.2)</td>
<td>393 (1.0)</td>
<td>389 (1.0)</td>
</tr>
<tr>
<td>pumsb*</td>
<td>288</td>
<td>317 (1.1)</td>
<td>288 (1.0)</td>
<td>289 (1.0)</td>
</tr>
<tr>
<td>retail</td>
<td>150</td>
<td>-</td>
<td>183 (1.2)</td>
<td>172 (1.2)</td>
</tr>
<tr>
<td>T10</td>
<td>530</td>
<td>-</td>
<td>6912 (13.1)</td>
<td>1339 (2.5)</td>
</tr>
<tr>
<td>T40</td>
<td>6191</td>
<td>-</td>
<td>33006 (5.3)</td>
<td>14190 (2.3)</td>
</tr>
</tbody>
</table>

- mush=mushroom, pumsb*=pumsb-star, T10=T10I4D100K, T40=T40I10D100K

**Table:** Run-time overhead due to privacy step
Score perturbation-based algorithm

\[ \gamma = \frac{8k}{\varepsilon n} \ln \left( \frac{|U|}{\rho} \right) \]

Also get their frequencies.

Assume frequency of all other patterns = \( \psi \)
Score perturbation-based algorithm

\[ \gamma = \frac{8k}{\epsilon n} \ln \left( \frac{|U|}{\rho} \right) \]

Assume frequency of all other patterns = \( \psi \)

\[ \text{Lap}(\lambda) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}} \]

Sample “noise” i.i.d from \( \text{Lap}(4k/\epsilon n) \). Add to the frequencies of the patterns in \( U \).

\( S_0 = \) patterns with frequency > \( \psi = q_{k^T - \gamma} \)

Also get their frequencies.
Score perturbation-based algorithm

\[ \gamma = \frac{8k}{\varepsilon n} \ln \left( \frac{|U|}{\rho} \right) \]

Assume frequency of all other patterns = \( \psi \)

Sample “noise” i.i.d from \( Lap(4k/\varepsilon n) \)
Add to the frequencies of the patterns in \( U \).

Pick the top \( k \) according to noisy frequencies.
Score perturbation-based algorithm

\[ \gamma = \frac{8k}{\varepsilon n} \ln \left( \frac{|U|}{\rho} \right) \]

\[ S_0 = \text{patterns with frequency > } \psi = q_k^T - \gamma \]

Also get their frequencies.

Assume frequency of all other patterns = \( \psi \)

Sample “noise” i.i.d from \( \text{Lap} \left( \frac{4k}{\varepsilon n} \right) \)

Add to the frequencies of the patterns in \( U \).

Pick the top \( k \) according to noisy frequencies.

Can be performed efficiently with additional tricks!

\[ \text{Lap}(\lambda) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}} \]
Score perturbation-based algorithm

\[ \gamma = \frac{8k}{\varepsilon n} \ln \left( \frac{|U|}{\rho} \right) \]

\( S_0 = \text{patterns with frequency} > \psi = q_{k^T - \gamma} \)

Also get their frequencies.

Assume frequency of all other patterns = \( \psi \)

Sample "noise" i.i.d from \( \text{Lap}(4k/\varepsilon n) \)

Add to the frequencies of the patterns in \( U \).

Pick the top \( k \) according to noisy frequencies.

Add \( \text{Lap}(2k/\varepsilon n) \) noise to the "actual" frequencies of the patterns picked.

Can be performed efficiently with additional tricks!

\( \text{Lap}(\lambda) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}} \)
Score perturbation-based algorithm

\[ \gamma = \frac{8k}{\varepsilon n} \ln \left( \frac{|U|}{\rho} \right) \]

\( S_0 = \) patterns with frequency \( > \psi = q \)

Assume frequency of all other patterns = \( \psi \)

Sample “noise” i.i.d from \( \text{Lap}(4k/\varepsilon n) \)

Add to the frequencies of the patterns in \( U \).

Pick the top \( k \) according to noisy frequencies.

Output the patterns picked and their noisy frequencies.

Add \( \text{Lap}(2k/\varepsilon n) \) noise to the “actual” frequencies of the patterns picked.

\( \text{Lap}(\lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda} \)
Parameters: $\rho = 0.1$, $k = 10$ and the size of the itemsets mined $= 3$

(g) Score perturbation-based

(h) Exponential sampling-based