SQL Queries under Differential Privacy: the R2T Algorithm

Shen,Zixuan

August 14, 2024

Abstract

This project applied and evaluated the concept and algorithm presented in Dong et al. [1] on a novel approach for differentially private query evaluation in databases with foreign-key constraints and self-joins. The mechanism was implemented on various query types, where counting was conducted on SNAP Nets' graph edges and summation was conducted with the TPC-H benchmark. We compared and analyzed the relative error between the actual and post-application query results, providing insights of the R2T mechanism.

Github Link: **https://github.com/hareisland/differential-privacy-R2T**

1 Introduction

1.1 Background of the Project

Differential privacy stands as a robust framework ensuring the safeguarding of individual data integrity while facilitating valuable dataset analysis and querying. By maintaining the statistical properties of query results indistinguishable, regardless of individual data inclusion, it strikes a delicate balance between data utility and privacy, making it applicable across diverse domains, including SQL-based systems.

In terms of the query outcomes, its random noise follows the Laplace distribution, a fundamental component of differential privacy. The privacy parameter ϵ governs the noise level, with smaller values enhancing privacy but introducing more noise, and larger values offering improved accuracy at the cost of reduced privacy. Regarding this, the CPLEX package is involved to tackle optimization challenges stemming from the R2T mechanism application. Therefore, the R2T mechanism determines an optimal truncation threshold to achieve desired utility guarantees while upholding privacy preservation.

Through the implementation and evaluation of this mechanism on selected query types of COUNT and SUM, we aim to shed light on its utility and effectiveness in maintaining privacy integrity while delivering precise query outcomes.

1.2 Related Work

Previous works in high-frequency trading and sentiment analysis demonstrates the critical role of real-time data in enhancing trading strategies. Our project builds upon these foundational studies by combining high-frequency factor analysis with real-time sentiment insights, offering a novel framework for informed and proactive financial decision-making.

1.2.1 Instance-Optimality in Differential Privacy

Asi and Duchi [2] focus on instance-optimality in differential privacy through approximate inverse sensitivity mechanisms. They extend and approximate the inverse sensitivity mechanism to provide instance-optimal algorithms. The study presents two approximation frameworks that efficiently compute for a broad class of functions, including local sensitivities and gradient-based approximations for optimization problems. The mechanisms developed by Asi and Duchi show significant performance improvements compared to minimax bounds for well-behaved instances, particularly in unbounded-range mean estimation, principal component analysis, and linear regression.

1.2.2 Bounding User Contributions in Differential Privacy

Amin et al.[3] explore the bias-variance trade-off in differential privacy when bounding user contributions. They address the challenge of determining the maximum contribution that can be made by a single user in differentially private learning algorithms. The analysis highlights the trade-off between allowing users to contribute large amounts of data, which increases noise levels to protect outliers, and limiting contributions to keep noise levels low but potentially discarding valuable data. The study emphasizes the importance of balancing privacy protection and data utility based on measurable properties of the dataset, showcasing a concrete cost to privacy that cannot be mitigated by simply collecting more data.

2 Description

2.1 Overview of the Query

The R2T paper addresses self-join challenges by categorizing them into three query types: SJ, SJA, and SPJA (Select Project Join Aggregate). It uses specific optimization models for computing $Q(I, \tau)$ based on the query type, focusing on P, SPJ, and SPJA queries to calculate edges, lengths, and triangles. The system emulates a structure based on PostgreSQL and CPLEX, accepting SQL-formatted SPJA queries with a designated primary private relation R_p. It supports SUM and COUNT aggregations, expanding them into reporting queries to establish tuple reference relationships between I and J.

Query 1: Select Triangle in Undirected Graph

Query 1 utilizes SJP (Select-Project-Join) and it retrieves the IDs of the nodes that form a triangle in an undirected graph. It does this by joining the *edge* table with itself three times (as *e1*, *e2*, and *e3*) and applying the condition that the *to_id* of *e1* matches the *from_id* of *e2*, the *to_id* of *e2* matches the *from_id* of *e3*, and the *from_id* of *e1* matches the *from_id* of *e3*.

```
SELECT 1, e1.from_id, e2.from_id, e2.to_id
FROM edge e1
JOIN edge e2 ON e1.to_id = e2.from_id
JOIN edge e3 ON e2.to_id = e3.to_id
WHERE e1.from_id = e3.from_id;
```

Query 2: Counting

This query extends the previous one by counting the number of triangles in the undirected graph. It utilized SPJA (Select-Project-Join-Aggregate) and performs the same join operation as Query 1, but adds a *GROUP BY* clause to aggregate the results by the node IDs that form the triangles, and a *COUNT(*)* to count the number of triangles.

```
SELECT count(*), e1.from_id, e2.from_id, e2.to_id
FROM edge e1
JOIN edge e2 ON e1.to_id = e2.from_id
JOIN edge e3 ON e2.to_id = e3.to_id
WHERE e1.from_id = e3.from_id
GROUP BY e1.from_id, e2.from_id, e2.to_id;
```

Query 3-1: Primary Private Relation (single join key)

TPC-H is a widely used decision support benchmark that consists of a set of complex queries designed to evaluate the performance of database systems.

Query 3-1 (Q12 of TPC-H) counts the number of line items, using the primary key of dataset, orderkey, as join key.

```
SELECT COUNT(*)
FROM orders, lineitem
WHERE o_orderkey = l_orderkey.
```

2.1.1 Query 3-2: Primary Private Relation (multiple join key)

Under the same TPC-H dataset, Query 3-2 (Q5) counts the number of line items with more join keys, such that the customer and supplier are from the same nation and region.

```
SELECT COUNT(*)
FROM supplier, lineitem, orders, customer, nation, region
WHERE supplier.S_SUPPKEY = lineitem.L_SUPPKEY
AND orders.O_CUSTKEY = customer.C_CUSTKEY
AND customer.C_NATIONKEY = nation.N_NATIONKEY
AND nation.N_NATIONKEY = supplier.S_NATIONKEY
AND region.R_REGIONKEY = nation.N_REGIONKEY.
```

Query 4: Summation

This query (Q18 of TPC-H) is selected for the SUM aggregation operation. It returns the total quantity of line items purchased by all customers, where the primary private relation is customer, CUSTKEY.

```
SELECT SUM(l_quantity)
FROM customer, orders, lineitem
WHERE c_custkey = o_custkey
AND o_orderkey = l_orderkey.
```

2.2 Algorithm Description

The R2T mechanism introduces the first dynamic programming solution for arbitrary SPJA queries in databases with foreign-key constraints, achieving strong optimality. The algorithm

is straightforward to implement on any RDBMS and LP solver. Experimental results show significant improvements over existing techniques, including those designed for graph pattern counting.

1. **Categorize Queries:** Determines the query type from SJ, SJA and SPJA.
2. **Model Optimization:** Uses specific optimization models for each query type and adds the Laplace noise L to compute $Q(I,\tau)$:

$$\widetilde{Q}(I,\tau^{(j)}) = Q(I,\tau^{(j)}) + \mathrm{L}\left(\frac{\log(GS_Q)\tau^{(j)}}{\epsilon}\right) - \log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\cdot\frac{\tau^{(j)}}{\epsilon}$$

   where truncation thresholds $\tau^{(j)} = 2^j, j = 1\ldots\log(GS_Q)$
3. **SQL Query Processing:** Accepts SPJA queries in SQL format with primary relation R_p.
4. **Aggregation Support:** Expands SUM and COUNT aggregations into reporting queries.
5. **Differential Privacy Mechanism:** Implements the R2T algorithm, and selects the final differentially private query result:

$$\widetilde{Q}(I) = \max\left\{\max_j \widetilde{Q}(I,\tau^{(j)}), Q(I,0)\right\}$$

3 Implementation Details

In the implementation and application of the R2T mechanism, a truncation function $Q(I,\tau)$ is calculated in different optimization models based on the type of query.

In particular, the implementation involves a series of functions and classes designed to apply the R2T mechanism to ensure differential privacy in data analysis tasks.

The `QueryHandler` class encapsulates query details, facilitating data retrieval from a PostgreSQL database. The `get_data` method establishes a database connection, executes the query, and fetches the resulting data. Then, `linear_problem` function formulates and solves a linear problem using the CPLEX optimization package. And `generate_laplace_noise` generates Laplace noise with a given scale parameter, essential for introducing privacy-preserving noise to query results.

The core `apply_r2t` function applies the R2T mechanism iteratively, calculating optimal truncation thresholds and evaluating relative errors between real and differentially private query results.

`configure_logging` sets up logging to a specific file, recording essential information and results during mechanism execution. `execute_query` orchestrates the execution of the R2T mechanism for a given query and parameters, logging running times and relative errors across multiple runs. It ensures result integrity by sorting and filtering outliers from the recorded data.

Finally, the SQL queries, categorized by primary private relation types, are defined within the script to evaluate the R2T mechanism's performance across different query complexities.

The input-output flow is as follows:

Input:

- Load dataset.

- Input ϵ for differential privacy, β for probability and the truncation function $Q(I, \tau)$. Initialize τ increase geometrically.
- For each τ value, calculate adjusted $\tilde{Q}$ and select the maximum

$$\hat{Q}(I) = \max_j \left(\max(\tilde{Q}(I, \tau(j)), Q(I, 0)) \right)$$

Output:

- Record $\hat{Q}(I)$ as the result, ensuring differential privacy.
- Record the solve time.

4 Experiment

4.1 setup

Prior to conducting the experiments, we prepared the necessary data and setup the required software components. This included setting up a database management system, such as MySQL or PostgreSQL, and ensuring the TPC-H dataset was properly loaded into the database. Additionally, we ensured that any necessary APIs, such as CPLEX, were properly configured and accessible for the experiments.

With the data and setup in place, we proceeded to load the relevant data from the database based on the designated queries. This resulted in a set of tuples that were then processed using the algorithms under evaluation. The calculated noisy results were then compared against the true results to assess the correctness and error levels, ensuring they were consistent with the findings reported in the related research papers.

4.2 Datasets

4.2.1 SNAP Nets

To evaluate the proposed model, we utilized two graph datasets obtained from the Stanford Network Analysis Project (SNAP) website.[4]

The Amazon Undirected dataset was gathered from the Amazon website and focuses on the "re-order" feature of its E-commerce. The edges of the undirected graph are formed between products that are frequently co-purchased. The ground-truth communities are defined by Amazon's product categories, with each connected component representing a distinct community. In this experiment, we retained the top 5,000 communities and thereby the network encompasses the largest connected component.

4.2.2 Dataset Generation using TPC-H

For the purposes of this study, we involved the TPC-H benchmark. The TPC-H is a decision support benchmark.[5] It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. It provides a set of database schemas, data, queries, and performance metrics to evaluate the performance of database management systems.

To generate the TPC-H dataset, we utilized the provided *dbgen* tool. This tool is responsible for creating TPC-H compliant datasets according to the specified scale factor. We compiled the *dbgen* by modifying the *makefile.suite* file with the following parameters:

$$\begin{aligned} \text{CC} &= \text{gcc} \\ \text{DATABASE} &= \text{INFORMIX} \\ \text{MACHINE} &= \text{LINUX} \\ \text{WORKLOAD} &= \text{TPCH} \end{aligned}$$

We then ran the compiled *dbgen* tool to generate the TPC-H dataset with a scale factor of 1, which resulted in approximately 1 GB of data. The generated dataset included eight tables, such as LINEITEM, ORDERS, and PART, which were used in the subsequent experiments.

The TPC-H benchmark allows to create a realistic and standardized dataset to evaluate the performance of our proposed R2T techniques in a well-established and widely used scenario.

5 Results and Discussion

The experiment results are as follows:

Query ID	Query 1	Query 2	Query 3-1	Query 3-2	Query 4
R2T Error	3.3062%	3.1256%	0.1764%	0.8685%	0.0413%
Runtime (s)	31.47	39.36	13.14	1.15	28.21

Table 1: Results of R2T average relative error and compile time per iteration.

The first two queries attempted for triangle selection and counting in the Amazon Undirected Graph faced challenges due to the dataset's size and the complexity of three-node matching. As a result, the counting result did not show a seemingly discernible difference from the base query, even with the R2T algorithm. In Query 3-1, 3-2, and 4, the relative errors are all very small (0.18%, 0.87%, 0.04%), demonstrating the accuracy under the R2T mechanism. Notably, the single join key primary private relation query (3-1) exhibits a smaller error compared to the multiple join key query.

Regarding compilation time, the triangle queries require a similar lengthy execution period. Among the three TPC-H queries, the multiple join key query (3-2) is the fastest, followed by the single join key query (3-1), while the summation operation is the slowest.

It is worth mentioning that the running time in this experiment is less than that reported in Dong et al.[1], while their relative tendency remains the same. This gap might be attributed to the data size involved in the experiment, where we set the scale of data to 0.125 and the sensitivity to 10^5 as hyper-parameters.

6 Future Directions

As the field of differential privacy continues to evolve, several promising avenues for future research and development emerge. One key area is the need for enhanced differential privacy mechanisms that can provide stronger privacy guarantees while maintaining high utility in various applications. This could involve the development of novel techniques that offer improved trade-offs between privacy and utility, enabling more effective deployment in real-world scenarios.

Another important direction could be the exploration of privacy-preserving machine learning. The intersection of machine learning and privacy preservation presents an exciting opportunity to develop robust techniques for training models on sensitive data without compromising individual privacy. Addressing this challenge could significantly impact a wide range of applications where machine learning is utilized on private or confidential information.

Novel approaches for ensuring privacy in data sharing and collaborative environments, particularly in the context of multi-party computation and secure data exchange, warrant further investigation. By exploring these future directions and embracing innovative solutions, we can advance the field of differential privacy and contribute to the development of privacy-preserving technologies for a wide range of applications.

References

[1] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. R2t: Instance-optimal truncation for differentially private query evaluation with foreign keys. In *Proceedings of the 2022 International Conference on Management of Data*, SIGMOD '22, page 759–772, New York, NY, USA, 2022. Association for Computing Machinery.

[2] Hilal Asi and John C Duchi. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14106–14117. Curran Associates, Inc., 2020.

[3] Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. Bounding user contributions: A bias-variance trade-off in differential privacy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 263–271. PMLR, 09–15 Jun 2019.

[4] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[5] Peter Boncz, Thomas Neumann, and Orri Erling. Tpc-h analyzed: Hidden messages and lessons learned from an influential benchmark. In Raghunath Nambiar and Meikel Poess, editors, *Proceedings of the TPCTC 2013*, volume 8391 of *Lecture Notes in Computer Science*, pages 61–76, Heidelberg, 2014. Springer.