Declarative Privacy-Preserving Inference Queries

Hong Guan, Ansh Tiwari, Summer Gautier, Rajan Hari Ambrish,
Lixi Zhou, Yancheng Wang, Deepti Gupta[†],
Yingzhen Yang, Chaowei Xiao[‡], Kanchan Chowdhury, Jia Zou 🖂

Arizona State University, Texas A&M University-Central Texas[†], University of Wisconsin at Madison[‡], 🖂Jia.Zou@asu.edu

Abstract. Detecting inference queries running over personal attributes and protecting such queries from leaking individual information requires tremendous effort from practitioners. To tackle this problem, we propose an end-to-end workflow for automating privacy-preserving inference queries including the detection of subqueries that involve AI/ML model inferences on sensitive attributes. Our proposed novel declarative privacy-preserving workflow allows users to specify *"what private information to protect"* rather than *"how to protect"*. Under the hood, the system automatically chooses privacy-preserving plans and hyper-parameters. Link to our video: `https://youtu.be/nK2deY_6adM`

1 Introduction

With more database systems supporting AI/ML, queries increasingly involve AI/ML model inferences. However, limited research has addressed declarative support for differential privacy (DP) in inference queries, particularly when the underlying dataset contains sensitive information. Ideally, with declarative DP support, a data owner will declare sensitive information that needs to be protected by tainting the data attributes and tuples. Database users could then issue arbitrary inference queries without needing to define the specifics of the DP mechanism or model. Instead, the system would automatically identify sensitive subqueries and apply DP safeguards based on the user's privacy budget. We use an example to illustrate the idea as follows.

Motivating Example. Online social media posts can reveal personal details such as usage patterns and social status, making them vulnerable to linkage attacks. For instance, in the query "*SELECT count(*) FROM IMDB_ MOVIE_ REVIEW R WHERE R.date > '06/01/2015' AND R.date < '06/05/2015' AND sentiment_ classifier(R.Review) = Positive* ", instead of sharing private reviews with analysts, a social media platform can train a sentiment classification model that takes reviews within a query range as input and outputs their sentiment. The sentiment predictions are then aggregated. Differential Privacy-Stochastic Gradient Descent (DP-SGD) [1] protects private training data from being reconstructed, achieving a better privacy-utility trade-off than adding noise to aggregated results. The privacy cost of a query depends on the remaining privacy budget of both the dataset and the user. Two key challenges arise: (1) detecting which queries require protection and (2) selecting a neural network architecture that balances privacy and accuracy given the training data, validation data, and privacy budget (ϵ). For instance, one approach fine-tunes a pre-trained BERT model on private social media posts using DP-SGD. Another approach trains a bi-LSTM model from scratch with pretrained Word2Vec embeddings using DP-SGD. A third approach encodes data points into embedding vectors using a pre-trained deep model, adding noise for differential privacy, with inference performed via approximate nearest neighbor search. Using the IMDB dataset, we found that the first two approaches outperformed the third in privacy-utility trade-offs. The fine-tuned BERT model achieved higher accuracy for small privacy budgets ($\epsilon < 6$), while the bi-LSTM model performed better for larger privacy budgets.

To address these challenges, we developed a declarative system that consists of the following components: (1) Taint Analysis, which automatically identifies the sensitive subqueries; (2) Privacy-Preserving Query Transformer, which transforms a user-requested inference operator into an equivalent operator with desired privacy guarantees; (3) Differentiable Neural Architecture Search, which automatically searches for the optimal neural network architecture for training the model to be used by the transformed inference operator, if there are no existing suitable models available. In the rest of the paper, we will introduce each component in Sec. 2 and describe the demonstration proposal in Sec. 3.

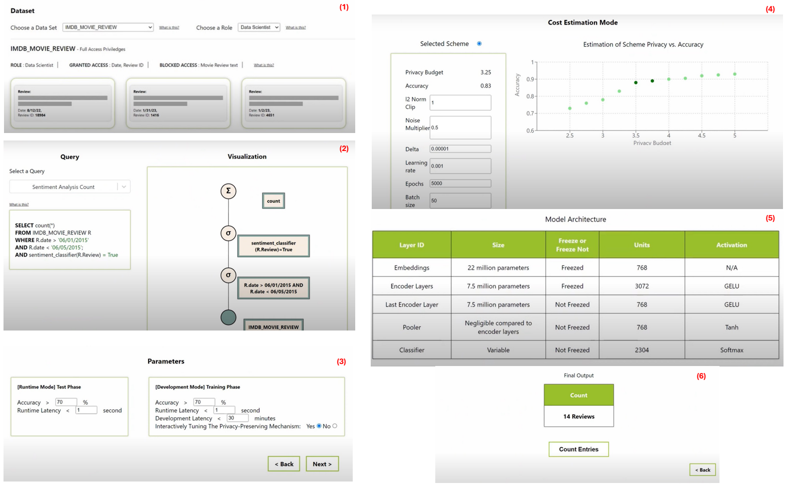
2 System Architecture

Our proposed system consists of the following components:

Query Taint Analysis We enhance the data catalog to enable the tainting of private attributes based on users' privacy requirements [4]. Attributes to be tainted are directly specified by the data owners or extracted from the view-based access control policy specified by the data owners. Then, once a data scientist issues a query, the query will be lowered to a graph-based Intermediate Representation (IR) based on the nested relational algebra, where each node represents a relational operator, and each edge represents a dataset, which could be a relation/view or a collection of objects, such as images, video, and text files. The tainted sources will propagate the taint through the IR graph, so that the sensitive sub-queries that access private information are identified.

Privacy-Preserving Transformation We abstract the process of providing DP support for a sensitive query as a special type of query transformation rules, described as follows (using the classical relational algebra notations): (1) $\lambda_f R \overset{\Delta acc,\epsilon}{=} \lambda_{f'} R$. λ_f represents the prediction operator that takes each tuple in the relation (or a collection of arbitrary objects) R as input features, and outputs prediction results. The DP-SGD-trained model, denoted as $\lambda_{f'} R$, outputs the prediction results with privacy guarantee ϵ. In addition, such transformation may results in an accuracy drop, represented as Δacc. (2) $\lambda_f(\pi_A(\sigma_p(R))) \overset{R,\Delta acc,\epsilon}{=} \lambda_{f'}(A,p)$. λ_f represents the prediction operator taking the query output $\pi_A(\sigma_p(R))$ as input features. The model trained with DP-SGD, denoted as $\lambda_{f'}(A,p)$, which takes the projection attributes A and selection predicate p as inputs, and outputs

the final prediction result for $\lambda_f(\pi_A(\sigma_p(R)))$. We have similar rules extended for aggregation queries, which are omitted due to space limitation. A query optimizer searches for the most promising transformation plan using a learned cost model.
Differentiable Neural Architecture Search To efficiently determine the optimal neural network architecture with minimal human effort, we use Differentiable Neural Architecture Search (DNAS) [3]. If a relevant foundation model exists, DNAS identifies existing layers or new adaptive layers for fine-tuning; otherwise, it finetunes a new foundation model based on data modality. DNAS optimizes architecture and weights via gradient descent, balancing prediction accuracy and computational cost. To mitigate privacy concerns, we reset the searched model's weights to random initialization and train it on private data using DP-SGD, ensuring no additional privacy budget is consumed [2].



Dataset

Choose a Data Set: IMDB_MOVIE_REVIEW
Choose a Role: Data Scientist

IMDB_MOVIE_REVIEW - Full Access Priviledges

ROLE : Data Scientist | GRANTED ACCESS : Date, Review ID | BLOCKED ACCESS : Movie Review text |

Review: [redacted]
Date: 8/12/22,
Review ID: 18984

Review: [redacted]
Date: 1/31/23,
Review ID: 1416

Review: [redacted]
Date: 1/2/23,
Review ID: 4651

Select a Query: Sentiment Analysis Count

```
SELECT count(*)
FROM IMDB_MOVIE_REVIEW R
WHERE R.date > '06/01/2015'
AND R.date < '06/05/2015';
AND sentiment_classifier(R.Review) = True
```

Σ
count
σ
sentiment_classifier (R.Review)=True
σ
R.date > 06/01/2015 AND R.date < 06/05/2015
IMDB_MOVIE_REVIEW
Parameters
(3)

[Runtime Mode] Test Phase
Accuracy > 70 %
Runtime Latency < 1 second

[Development Mode] Training Phase
Accuracy > 70 %
Runtime Latency < 1 second
Development Latency < 30 minutes
Interactively Tuning The Privacy-Preserving Mechanism: Yes ● No ○

Selected Scheme ●

Privacy Budget 3.25
Accuracy 0.83
l2 Norm Clip: 1
Noise Multiplier: 0.5
Delta: 0.00001
Learning rate: 0.001
Epochs: 5000
Batch size: 50

Model Architecture

Layer ID	Size	Freeze or Freeze Not	Units	Activation
Embeddings	22 million parameters	Freezed	768	N/A
Encoder Layers	7.5 million parameters	Freezed	3072	GELU
Last Encoder Layer	7.5 million parameters	Not Freezed	768	GELU
Pooler	Negligible compared to encoder layers	Not Freezed	768	Tanh
Classifier	Variable	Not Freezed	2304	Softmax

Final Output

Count
14 Reviews

Fig. 1: Illustration of the Demo and User Interfaces

3 Demonstration Proposal

We will demonstrate the proposed declarative privacy-preserving workflow. The demo audience can select a database, and then run the following steps:
Step 1. Data Owners Annotate Sensitive Information. Data owners label the attributes and tuples containing private information and specify the corresponding privacy parameters for different user roles. Such information can also be automatically extracted from the traditional RDBMS view-based access control policies that are defined by the data owners. As illustrated in Figure 1 (1), From the perspective of the data scientist, sensitive information is redacted while only insensitive information is listed.

Step 2. Query Request and Analysis. As illustrated in Figure 1 (2), after the data scientist issues queries over private datasets, an intermediate representation of the query is visualized at the right side of the panel, and the sensitive subqueries that involve private attributes will be highlighted in green boxes.
Step 3. Parameter Specification. The system recommends training and test parameters to the data scientist, such as accuracy and latency. The system administrator can change these default settings and choose to interactively tune the parameters, as illustrated in Figure 1 (3).
Step 4. Privacy-Preserving Recommendation. As illustrated in Figure 1 (4), the system will visualize the top k privacy-preserving schemes that are recommended by the system. These recommended schemes are sampled from the Pareto-optimal plans. For each privacy-preserving scheme, it will list the path to the pre-trained model, or the model architecture and hyper-parameters for training a new model, as illustrated in Figure 1 (5). The system administrator or a system program can select a configuration according to the task requirement.
Step 5. Query Execution. The system proceeds to run the query with visualized query results as shown in Figure 1 (6).

Acknowledgement

This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 17STQAC00001-07-00.

Disclaimer

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. pp. 308–318 (2016)
2. Cheng, A., Wang, J., Zhang, X.S., Chen, Q., Wang, P., Cheng, J.: Dpnas: Neural architecture search for deep learning with differential privacy. In: Proceedings of the AAAI conference on artificial intelligence. vol. 36, pp. 6358–6366 (2022)
3. Wan, A., Dai, X., Zhang, P., He, Z., Tian, Y., Xie, S., Wu, B., Yu, M., Xu, T., Chen, K., et al.: Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In: CVPR (2020)
4. Zhou, L., Yu, L., Zou, J., Min, H.: Privacy-preserving redaction of diagnosis data through source code analysis. In: Proceedings of the 35th International Conference on Scientific and Statistical Database Management. pp. 1–4 (2023)