Model-based Large Language Model Customization as Service

Zhaomin Wu[1*], Jizhou Guo[2†*], Junyi Hou[1], Bingsheng He[1], Lixin Fan[3], Qiang Yang[4]
[1]National University of Singapore, [2]Zhiyuan Colleage, Shanghai Jiao Tong University
[3]WeBank, [4]The Hong Kong University of Science and Technology
zhaomin@nus.edu.sg, sjtu18640985163@sjtu.edu.cn, hhh@u.nus.edu
dcsheb@nus.edu.sg, lixinfan@webank.com, qyang@cse.ust.hk

Abstract

Prominent Large Language Model (LLM) services from providers like OpenAI and Google excel at general tasks but often underperform on domain-specific applications. Current customization services for these LLMs typically require users to upload data for fine-tuning, posing significant privacy risks. While differentially private (DP) data synthesis presents a potential alternative, its application commonly results in low effectiveness due to the introduction of excessive noise on data for DP. To overcome this, we introduce *Llamdex*, a novel framework that facilitates LLM customization as a service, where the client uploads pre-trained domain-specific *models* rather than data. This client-uploaded model, optionally protected by DP with much lower noise, is inserted into the base LLM via connection modules. Significantly, these connecting modules are trained without requiring sensitive domain data, enabling clients to customize LLM services while preserving data privacy. Experiments demonstrate that Llamdex improves domain-specific accuracy by up to 26% over state-of-the-art private data synthesis methods under identical privacy constraints and, by obviating the need for users to provide domain context within queries, maintains inference efficiency comparable to the original LLM service.

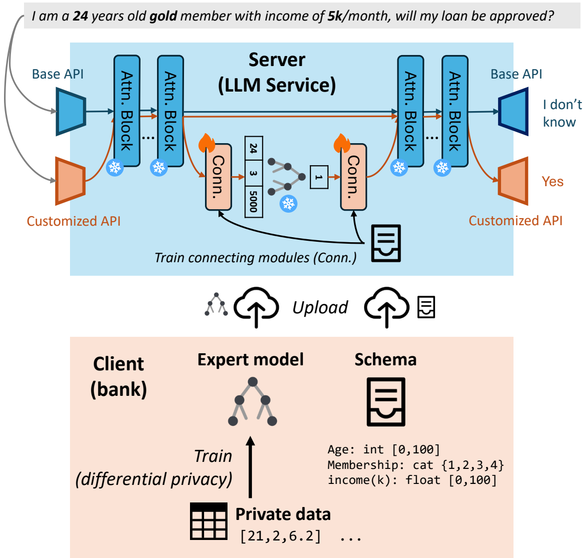


Figure 1: Overview of Llamdex customization pipeline

1 Introduction

While Large Language Model (LLM) services, such as Gemini (Team et al., 2023) and ChatGPT (OpenAI, 2023), excel at general tasks, they often exhibit limitations in domain-specific applications due to insufficient access to relevant private data. Customizing these LLM services typically involves *clients* uploading domain data to providers (*servers*) like Google or OpenAI for fine-tuning (Hu et al., 2022a), with the resulting customized LLMs often hosted on platforms such as Gemini's Gem and OpenAI's GPT Store. However, this data-upload requirement introduces significant privacy risks, deterring clients in sensitive sectors such as healthcare and finance.

Existing privacy-preserving approaches for LLM customization services often compromise on effectiveness to achieve privacy. A common method is differentially private (DP) data synthesis (Hong et al., 2024; Tian et al., 2022; Duan et al., 2024). This technique involves clients uploading synthetic data, generated from their original private datasets under DP constraints, to the service provider. While this approach theoretically provides strong DP guarantees for the client's underlying domain data, the requisite noise injection for achieving such privacy significantly degrades synthetic data quality. This degradation, in turn, diminishes the LLM customization service's effectiveness, leading to customized LLM services that often exhibit

[*]Equal contribution. [†]Work done during an internship at National University of Singapore.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 4904–4921
November 4-9, 2025

notably reduced response accuracy on domain-specific tasks, a phenomenon termed "disparate impact" (Ganev et al., 2022).

To improve this privacy-effectiveness trade-off in LLM customization services, we propose a novel framework named **L**arge **la**nguage **m**odel with **d**omain **ex**pert (*Llamdex*). This framework enables clients to customize the LLM service by uploading a domain-specific *model* instead of potentially sensitive *data*, thereby offering superior privacy compared to data-sharing methods. An overview of the Llamdex service architecture, illustrated for financial inquiry applications, is presented in Figure 1. This architecture involves the client providing a pre-trained, domain-specific model (termed the *expert model*) to the server. The server then inserts this expert model into an intermediate layer of the base LLM via learnable connecting modules. Concurrently, a data schema detailing each column's name, type, and range is also supplied to the service, enabling the base LLM to correctly interpret the inputs and outputs of the client's expert model during service operation.

This model-based customization approach for LLM services yields several advantages regarding effectiveness, privacy, and efficiency. First, **it enhances effectiveness by decoupling context understanding from task-solving processes**. The LLM can then focus on context understanding, its area of strength, while task-solving is delegated to the expert model, which can be any model (e.g., XGBoost (Chen and Guestrin, 2016)) optimized for the specific task. This separation improves overall effectiveness, as LLMs often exhibit lower accuracy on tasks like arithmetic calculations (Yuan et al., 2023) or precise search (Saparov et al., 2025) where specialized models excel. Second, concerning privacy, the client's expert model can be trained using established DP techniques (Abadi et al., 2016). This approach **yields considerably lower noise than DP data synthesis methods** (Tian et al., 2022) under an equivalent privacy budget (Li et al., 2021). Finally, the Llamdex service architecture **maintains inference efficiency comparable to the base LLM**. Integrating the expert model avoids the need for users to embed extensive contextual information within each service prompt.

Concurrently, this design presents two significant challenges. The first is the misalignment between the domain expert's operational space (e.g., 1D feature vectors from tabular data) and the LLM's token embedding representations, necessitating complex design for trainable connecting modules. The second challenge is the absence of training data for these modules, since the client's original data is inaccessible due to privacy. Overcoming these obstacles is vital for successful model-based knowledge transfer.

To overcome these challenges, we design the architecture and training algorithm of Llamdex to facilitate effective model-based knowledge transfer. Specifically, to resolve the first challenge, we design *Llamdex encoder* that maps the original tokens to feature vectors and a *Llamdex decoder* that converts the expert's output into multiple token embeddings. These embeddings are then appended to the original sequence of token embeddings. To address the second challenge, we train the mapping modules using synthetic texts generated from randomly distributed data under the public schema. This allows the mapping modules to learn to extract the required feature values from texts without relying on the real data distribution. Our source code is available at an anonymous repository[1]. The contributions of this paper are as follows:

- We propose a novel framework, Llamdex, that enhances LLM customization by effectively integrating domain-specific models into LLMs through trainable connecting modules.

- We design a training algorithm for connecting modules using only the public schema, without requiring access to domain data distributions in either original or perturbed form.

- Experiments on real-world datasets demonstrate Llamdex's superior customization effectiveness, achieving accuracy improvements of up to 14% over LoRA fine-tuning and up to 26% over PromptPATE (at an equivalent differential privacy level), while maintaining inference efficiency comparable to the base LLM.

2 Background

To provide essential background for comparing the privacy levels of different approaches, we briefly introduce the concept of differential privacy (DP).

Definition 2.1 (Differential Privacy (Dwork, 2006))**.** A randomized algorithm $\mathcal{M} : \mathcal{D} \to \mathcal{O}$ is said to be (ε, δ)-differentially private if, for any two datasets $D, D' \in \mathcal{D}$ that differ by a single

[1] https://github.com/Xtra-Computing/Llamdex

record, and for all output sets $S \subseteq \mathcal{O}$,

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta, \quad (1)$$

where ε is the privacy budget; smaller ε implies stronger privacy guarantees. δ is the probability that the privacy guarantee is breached.

DP guarantees are typically achieved by introducing Gaussian noise during computation. Applying this concept to train deep learning models, notably through the Differentially Private Stochastic Gradient Descent (DP-SGD) algorithm (Abadi et al., 2016), enables training private model.

3 Related Work

Privacy-preserving LLM customization primarily falls into two categories: data-based and API-based methods. Data-based approaches involve clients providing synthetic data (generated from private domain data under DP guarantees) to the LLM service for fine-tuning. Conversely, API-based methods enable LLM customization by having LLM query an external, domain-specific API at inference time.

Data-based Customization. Data-based approach is broadly applicable to a diverse range of clients, from individuals to large corporations, primarily because it typically does not necessitate significant client-side computational resources or require the client to maintain constant online connectivity with the LLM service. Typically, data-based approaches employ differentially private data synthesis. For instance, some methods, such as PATE (Papernot et al., 2017) and SeqPATE (Tian et al., 2022), involve adding noise to aggregated predictions from an ensemble of teacher models (trained on private domain data) to create synthetic data. Other variants, including d_χ-DP (Feyisetan et al., 2020), selective-DP (Shi et al., 2021), Table Diffusion (Truda, 2023), PromptPATE (Duan et al., 2024), and DP-OPT (Hong et al., 2024), aim to further enhance the quality of the generated synthetic data. This synthesized data is subsequently used to fine-tune the base LLM. A notable drawback of such approaches that train models from synthetic data (a form of *input perturbation*) is the substantial degradation of model utility (Jayaraman et al., 2018) when compared to *gradient perturbation* methods that add noise during the training. Llamdex, which customizes LLM with DP-trained model, aligns with gradient perturbation methods, offering higher effectiveness of customization.

API-based Customization. This method is generally more suitable for large organizations that possess sufficient resources to develop, host, and maintain these APIs—ensuring they are consistently online and responsive—and is typically not well-suited for individual users due to these requirements. The mechanisms behind such APIs vary; for example, Yao et al. (2022) require users to provide API documentation to the LLM at inference time, while Schick et al. (2024) and Qin et al. (2024) propose fine-tuning an auxiliary LLM on the client-side to process API calls. These approaches can demand significant computational resources on the client side, may incur notable communication latency, and require the client's API infrastructure to be perpetually online. In contrast, Llamdex is designed for more general applicability, catering to both individual clients and large companies. It obviates the need for constant client online presence; clients only need to train a relatively small domain-specific model (e.g., XGBoost (Chen and Guestrin, 2016)) on their data and then upload this model to the server once for customization.

4 Problem Definition

Consider a collaboration between a *server* and a *client*. The server possesses an LLM $\mathcal{M}$, parameterized by $\theta_{\mathcal{M}}$, pre-trained on a large public text dataset $\mathbf{X}^s \in \mathbb{A}^{N_s \times l}$, where $\mathbb{A}$ denotes the vocabulary (token set) and l is the sequence length. The client holds a private, domain-specific dataset $\mathbf{X}^c \in \mathbb{R}^{N_c \times m}$, where m is the number of features. We assume the client's data is not contained within the server's training data ($\mathbf{X}^c \not\subseteq \mathbf{X}^s$) and exhibits a distinct distribution. Associated with $\mathbf{X}^c$ is a public *schema* $\mathbf{S}^c$, detailing feature names, types, and ranges. An illustrative example of $\mathbf{S}^c$ from the `titanic` dataset is provided below:

Example Features (from $\mathbf{S}^c$ for $\mathbf{X}^c$):
```
Age: int [0,100];
Pclass: category {"1","2","3"}
```
Example Target (from $\mathbf{S}^c$ for y):
```
Survived: bool {False, True}
```

The server aims to adapt its LLM $\mathcal{M}$ (assumed to be a standard decoder-only transformer (Jiang et al., 2023; Touvron et al., 2023)) into a domain-customized model $\mathcal{M}^c$ for the client's domain $\mathbf{X}^c$ without directly accessing $\mathbf{X}^c$. Specifically, we focus on a two-stage, model-based customization framework. **(Client-side)** The client trains a domain-specific expert model $\mathcal{E}^c$, parameterized by $\theta_{\mathcal{E}^c}$, on their private structured data $\mathbf{X}^c$ (feature vectors) to predict a target variable $\mathbf{y}$. Optionally, for enhanced privacy, $\mathcal{E}^c$ can be trained via DP-SGD (Abadi et al., 2016) to ensure $\theta_{\mathcal{E}^c}$ satisfies (ε, δ)-differential privacy. The client shares only these parameters $\theta_{\mathcal{E}^c}$ with the server as a component for service enhancement. **(Server-side)** The server inserts the client-provided expert model $\mathcal{E}^c$ into its frozen LLM $\mathcal{M}$ by training lightweight *connector* parameters θ_{conn}. This integration aims to enable the resulting customized LLM service, $\mathcal{M}^c$, to accurately answer domain-specific natural language questions. Formally, given a dataset $\mathbf{Z}^c = \{(\mathbf{z}_i, y_i)\}$, where each $\mathbf{z}_i$ is a natural language question related to $\mathbf{X}^c$ and y_i is the target answer, we aim to optimize θ_{conn} by minimizing a generative loss $\mathcal{L}_{\text{gen}}$ over $\mathbf{Z}^c$:

$$\min_{\theta_{\text{conn}}} \mathbb{E}_{(\mathbf{z}_i, y_i) \sim \mathbf{Z}^c} \left[\mathcal{L}_{\text{gen}}(\mathcal{M}^c(\theta_{\text{conn}}; \mathbf{z}_i, \theta_{\mathcal{M}}, \theta_{\mathcal{E}^c}), y_i) \right]. \tag{2}$$

Crucially, both the base LLM parameters $\theta_{\mathcal{M}}$ and the expert model parameters $\theta_{\mathcal{E}^c}$ remain frozen during this stage; only θ_{conn} is trained by the server.

Given that tabular data is prevalent in relational databases and readily translatable to/from text via schemas (Jatana et al., 2012), we focus primarily on tabular $\mathbf{X}^c$. Extensions of this customization service to other modalities (e.g., images, video) are discussed in Section F as future work.

Threat Model. We focus on the privacy of client data $\mathbf{X}^c$ against a semi-honest server that, while adhering to the protocol, may attempt to infer $\mathbf{X}^c$. The server accesses the LLM parameters $\theta_{\mathcal{M}}$, the client's schema $\mathbf{S}^c$, and the expert model parameters $\theta_{\mathcal{E}^c}$. The primary privacy risk involves the server inferring $\mathbf{X}^c$. If schema $\mathbf{S}^c$ is private, masking techniques (Ranganathan et al., 2023) can offer protection, an aspect orthogonal to this study.

5 Approach

This section details the Llamdex design for LLM service customization. Our approach uses two types of client-provided information: client's data distributions (via a client-trained expert model $\theta_{\mathcal{E}^c}$, optionally trained with DP) and client's data schema $\mathbf{S}^c$. The server uses $\mathbf{S}^c$ to train the connecting modules that bridge the frozen base LLM and the expert model, enabling the customized service. Section 5.1 describes this model architecture, focusing on the Llamdex encoder/decoder design. Section 5.2 details the schema-guided training and the service's inference procedure. Finally, Section 5.3 introduces service extensions for generating explanations and enabling iterative reasoning.

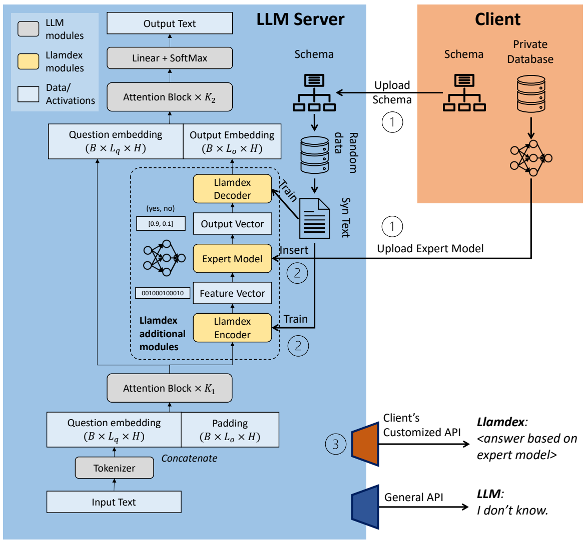


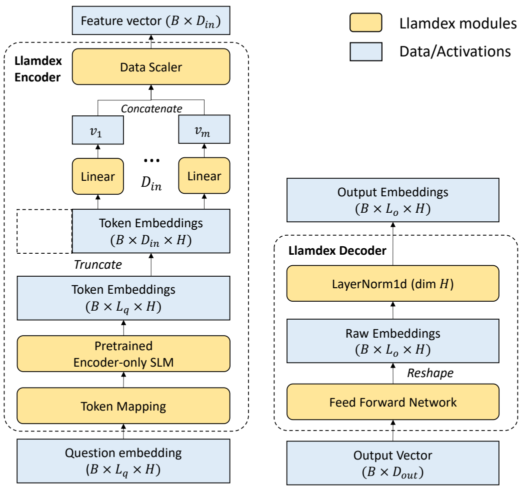
Figure 2: Llamdex structure and customization pipeline

5.1 Model Architecture

Llamdex facilitates LLM service customization by inserting a module between two attention blocks (assumed to be the k-th block) of the frozen base LLM $\mathcal{M}$, as depicted in Figure 2. This module processes LLM embeddings $\mathbf{h}_i^k$ derived from the input question $\mathbf{z}_i$ and generates domain-informed *output embeddings* $\mathbf{O}_i^{\text{emb}}$, which are then passed to subsequent LLM layers. This module comprises three core components: an expert model $\mathcal{E}^c$, a *Llamdex encoder*, and a *Llamdex decoder*. To maintain sequence length consistency required by some LLM architectures (e.g., those using RoPE (Su et al., 2024)), Gaussian noise paddings is appended to $\mathbf{h}_i^k$ before reaching the expert model, which are subsequently replaced by $\mathbf{O}_i^{\text{emb}}$.

Expert Model. The expert model $\mathcal{E}^c$, parameterized by the client-provided $\theta_{\mathcal{E}^c}$, encapsulates information about the client's domain-specific data distribution. It accepts a feature vector $\mathbf{x}_i^c$ and outputs a prediction $\hat{y}_i = \mathcal{E}^c(\theta_{\mathcal{E}^c}; \mathbf{x}_i^c)$. As $\theta_{\mathcal{E}^c}$ is frozen on the server-side, $\mathcal{E}^c$ can be any suitable model (e.g., multi-layer perceptron (MLP) or XGBoost), optionally trained with DP guarantees by the client.

Llamdex Encoder. The Llamdex encoder translates intermediate question embeddings from the LLM $\mathcal{M}$ into the structured feature vector $\mathbf{x}_i^c$ required by the expert model $\mathcal{E}^c$ (see Figure 3). The primary challenge stems from the auto-regressive



(a) Llamdex Encoder (b) Llamdex Decoder

Figure 3: Llamdex encoder and decoder architecture

nature of decoder-only LLMs; these models are optimized to predict the next token one at a time based on prior context, which complicates the direct, single-step extraction of a complete, structured feature set from their hidden states. To address this inherent limitation, we employ a pre-trained encoder-only small language model (SLM), e.g. RoBERTa (Liu, 2019), for feature extraction.

A significant hurdle when using an auxiliary SLM is the potential misalignment between the tokenizers of the main LLM ($\mathcal{M}$) and the SLM. To overcome this, we introduce a *token mapping* module, inspired by the logit lens (nostalgebraist, 2020) which was originally proposed for token interpretation. Specifically, given intermediate LLM question embeddings $\mathbf{h}_i^k$ at k-th attention block, the token mapping module computes logits $\mathbf{\Lambda}_i$ using the RMSNorm layer and LLM head $\mathcal{H}_{\text{LLM}}$ of $\mathcal{M}$:

$$\mathbf{\Lambda}_i = \mathcal{H}_{\text{LLM}}(\text{RMSNorm}_{\text{LLM}}(\mathbf{h}_i^k)). \quad (3)$$

For each token, the most probable token ID $\boldsymbol{\tau}_i^{\text{LLM}}$ according to the LLM are extracted:

$$\boldsymbol{\tau}_i^{\text{LLM}} = \arg\max(\mathbf{\Lambda}_i). \quad (4)$$

These LLM token IDs are then decoded into text tokens using the LLM's tokenizer $\mathcal{T}_{\text{LLM}}$, formally, $\mathbf{t}_i^{\text{LLM}} = \mathcal{T}_{\text{LLM}}.\text{Decode}(\boldsymbol{\tau}_i^{\text{LLM}})$. Finally, these text tokens are re-encoded using the SLM's tokenizer $\mathcal{T}_{\text{SLM}}$ to obtain token IDs compatible with the SLM, i.e., $\boldsymbol{\tau}_i^{\text{SLM}} = \mathcal{T}_{\text{SLM}}.\text{Encode}(\mathbf{t}_i^{\text{LLM}})$. These aligned SLM token IDs $\boldsymbol{\tau}_i^{\text{SLM}}$ are embedded into $\mathbf{H}_i^{\text{SLM}}$ with the SLM. Then, a truncated sequence $\tilde{\mathbf{H}}_i^{\text{SLM}}$ is obtained by keeping the last D_{in} tokens embeddings; these token embeddings are projected into D_{in} feature values of $\mathbf{x}_i^c \in \mathbb{R}^{D_{\text{in}}}$ via linear layers, followed by min-max scaling activations based on the ranges specified in $\mathbf{S}^c$, ensuring features adhere to valid ranges consistent with $\mathcal{E}^c$'s training.

Llamdex Decoder. The Llamdex decoder maps the expert model's prediction $\hat{y}_i$ back into the LLM's embedding space, generating the output embeddings $\mathbf{O}_i^{\text{emb}}$. The mapping is achieved using a simple feed-forward network (FFN) with SwiGLU activation (Shazeer, 2020). A key design challenge is the scale mismatch between these generated $\mathbf{O}_i^{\text{emb}}$ and the LLM's internal hidden states (e.g., $\mathbf{h}_i^k$ from preceding layers). A large discrepancy can destabilize subsequent computations, particularly skip-connections and attention mechanisms. To mitigate this, we apply an independent Layer Normalization (LayerNorm) specifically to the decoder's output before it is appended to the LLM's representations. This LayerNorm allows the model to adaptively learn the appropriate scale for $\mathbf{O}_i^{\text{emb}}$ that aligns with $\mathbf{h}_i^k$. The output embeddings $\mathbf{O}_i^{\text{emb}}$ are then appended to the question embeddings $\mathbf{h}_i^k$ and passed to the subsequent LLM layers.

5.2 Training and Inference

Llamdex service customization follows a two-stage protocol separating client and server responsibilities. **Client-side:** The client trains an expert model $\mathcal{E}^c$ on their private data $\mathbf{X}^c$, optionally protected by DP, and shares $\mathcal{E}^c$ and the public schema $\mathbf{S}^c$ with the server. **Server-side:** The server trains the connector parameters θ_{conn} (which comprise the Llamdex encoder and Llamdex decoder) using only the client's schema $\mathbf{S}^c$ and the received expert model parameters $\theta_{\mathcal{E}^c}$. Both the base LLM $\mathcal{M}$ and the client's expert model $\mathcal{E}^c$ remain frozen during this server-side training phase.

Training Llamdex Encoder. Since the server does not access the client's actual private data $\mathbf{X}^c$, the Llamdex encoder is trained using synthetically generated data. Based solely on the public schema $\mathbf{S}^c$ provided by the client, the server creates synthetic tabular feature vectors $\bar{\mathbf{x}}_i^c$ (e.g., via `sklearn`). Crucially, $\bar{\mathbf{x}}_i^c$ adheres to the schema's structure (types, ranges) but is generated from a **completely random distribution**, independent of the client's true data distribution underlying $\mathbf{X}^c$. For each $\bar{\mathbf{x}}_i^c$, a corresponding natural language questions $\bar{\mathbf{z}}_i$ is generated using an auxiliary LLM (e.g., Mistral-7B (Jiang et al., 2023)). The Llamdex encoder is then trained to map $\bar{\mathbf{z}}_i$ to $\bar{\mathbf{x}}_i^c$ with these synthetic data. This optimization uses a Mean Squared Error (MSE) loss, teaching the encoder the mapping from textual descriptions (e.g., "`He is ten years old`") to structured features (e.g., "`10`"). Our experiments show that the Llamdex encoder's learned mapping $\bar{\mathbf{z}}_i \to \bar{\mathbf{x}}_i^c$ generalizes to $\mathbf{z}_i \to \mathbf{x}_i^c$ without knowing the distribution of $\mathbf{z}_i$ or $\mathbf{x}_i^c$.

Training Llamdex Decoder. Similarly, the Llamdex decoder is trained by the server using synthetically generated data. It learns to map the prediction $\hat{y}_i$ of $\mathcal{E}^c$ to LLM's embedding embeddings that finally produces the correct textual answer a_i (e.g., "`Yes`", "`No`"). To achieve this without accessing real label distribution, the server generate synthetic target labels $\bar{y}_i$ and corresponding textual answers $\bar{a}_i$, consistent with the target definition in the client's schema $\mathbf{S}^c$ (e.g., type and range). The Llamdex decoder is trained using a cross-entropy loss between its predicted logits and the ground truth logits of these synthetic textual answers $\bar{a}_i$. Our experiments find the $\bar{y}_i \to \bar{a}_i$ mapping learned by the decoder generalizes well to $\hat{y}_i \to a_i$ without being aware of the distribution of y_i or a_i.

Inference for Customized Service. The Llamdex system provides an end-to-end customized LLM inference service. Upon receiving a user's natural language question $\mathbf{z}_i$, the base LLM $\mathcal{M}$ processes it up to the k-th attention block, which is the insertion point for the integrated module. Here, the trained Llamdex encoder extracts the relevant feature vector $\mathbf{x}_i^c$ from the LLM's intermediate hidden states $\mathbf{h}_i^k$. This vector $\mathbf{x}_i^c$ is then fed to the client's expert model $\mathcal{E}^c$, which produces a domain-specific prediction $\hat{y}_i$. Subsequently, the trained Llamdex decoder transforms this prediction $\hat{y}_i$ into output embeddings $\mathbf{O}_i^{\text{emb}}$. These embeddings are appended to $\mathbf{h}_i^k$ and propagated to the LLM's subsequent layers to generate the final textual response. An illustrative use case is provided in Table 4 (Appendix A). Notably, Llamdex does not require either contextual information or the schema $\mathbf{S}^c$ at inference time, resulting in better efficiency than API-based services (Yao et al., 2022).

5.3 Iterative Reasoning Mechanism

While the primary objective of Llamdex is to enable effective LLM customization, allowing the customized LLM to generate accurate answers based on the client's model, enhancing these answers with explanations or reasoning offers an additional service utility. To facilitate this, Llamdex incorporates a straightforward *iterative feedback* mechanism, providing users with a method to obtain basic explanations for the service's outputs.

This mechanism allows users to request clarification by augmenting their initial query. Specifically, the original answer generated by the Llamdex service is concatenated with a predefined prefix prompt (e.g., "`The expert's answer is`") and then appended to the original question $\mathbf{z}_i$. Users have the option to further refine this augmented query with custom prompts. The resulting revised input is subsequently resubmitted to the base LLM $\mathcal{M}$ component of the service. This base LLM processes the augmented context to produce a new response, aiming to deliver the requested reasoning or elaboration beyond the answer. An illustrative example of this iterative feedback mechanism is presented in Table 6 in Appendix C.

6 Experiment

This section presents our experimental evaluation. We begin by detailing the experimental settings (Section 6.1), followed by an assessment of the proposed method's accuracy (Section 6.2), inference efficiency (Section 6.3), and privacy guarantees (Section 6.4). Supplementary analyses, including a comparison of training efficiency (Appendix B), additional experimental results such as those for iterative reasoning (Appendix C), comprehensive ablation studies (Appendix D), and detailed hyperparameter configurations of baselines (Appendix E), are provided in the appendices.

6.1 Experiment Setting

This subsection outlines experimental setups, including datasets, model configuration, evaluation methods, baselines, and the environment.

Dataset. Our experiments incorporate four public real-world datasets: `titanic` (H., 2021), `wine` (Cortez et al., 2009), `bank` (Moro et al., 2014), and `nursery` (Rajkovic, 1989). For evaluation, all datasets are in tabular format with well-defined schema and meaningful column names. We split each dataset into training and test sets by 8:2. The details of each dataset are included in Appendix A.

Expert Model. The expert models $\mathcal{E}_c$ are trained directly on the respective tabular datasets, utilizing MLPs (default) and XGBoost (Chen and Guestrin, 2016). MLPs, implemented in `PyTorch` (Paszke et al., 2019), feature two hidden layers (400/200 neurons, ReLU activation (Nair and Hinton, 2010)), trained with AdamW (Loshchilov, 2017) (learning rate 10^{-4}, batch 64, max 30 epochs or convergence). For DP expert model training, we adopt the DP-SGD implementation in `Opacus` (Yousefpour et al., 2021) library, which clips gradients by norm 1 and injects noise, with varying ε and $\delta = 1/N_c$, where N_c is the number of samples. XGBoost models are configured with the following core hyperparameters: a learning rate of 0.1, `max_depth` set to 50, `subsample` of 0.8, `colsample_bytree` of 0.8, and `n_estimators` set to 50.

LLM and SLM. In our experiments, we employ the pretrained `Mistral-7B` (Jiang et al., 2023) as the base LLM, and the pretrained `Roberta-large` (Liu et al., 2019), with 0.355 billion parameters, as the SLM within the Llamdex encoder. In Llamdex training, the LLM remains frozen while the SLM undergoes full-parameter fine-tuning. The Llamdex encoder and decoder are trained with a batch size of 128 and a learning rate of 5×10^{-5}, using the AdamW optimizer (Loshchilov, 2017). The learning rate is adjusted by a cosine scheduler with 500 steps of warmup. The Llamdex encoder and decoder are trained for 30 and 10 epochs, respectively.

Evaluation. The effectiveness of customization is evaluated by the accuracy of answering domain-specific questions derived from client's tabular datasets. For question generation, each dataset row with m columns, with 10% of its values randomly masked to simulate missing data, is converted to text (format: $\#c_1 : v_1, \ldots, \#c_m : v_m$, where c_i is column name, v_i is value). This textual representation is then fed to `Mistral-7B` with a system prompt to generate a corresponding question. During accuracy evaluation, the LLM provides a single-word answer: "Yes"/"No" for binary classification or an uppercase letter (e.g., "A"-"Z") for multiclass tasks. The predicted class with the highest probability is compared against the ground truth to compute accuracy. we report the mean accuracy of five independent runs.

Baselines. To evaluate the accuracy and efficiency of Llamdex, we compare it against baselines without DP guarantees, including:

- **Original LLM**: Unmodified base LLM.
- **Real Data LoRA** (Hu et al., 2022a): LLM parameter-efficiently fine-tuned (LoRA) on questions/labels derived from real domain data.
- **Expert API**: A simplified API-based approach (Schick et al., 2024; Qin et al., 2024) where the LLM is prompted to extract feature values from questions; values are regex-extracted and fed to a domain expert model that predicts the final answer.

To evaluate Llamdex's accuracy under privacy guarantees, we compare it with baselines employing DP data synthesis. The synthetic data generated by these methods is subsequently used for LoRA fine-tuning of the LLM. These baselines include:

- **PATE-GAN** (Jordon et al., 2018): Traditional DP tabular data synthesis.
- **SeqPATE** (Tian et al., 2022): DP data synthesis method using knowledge distillation.
- **PromptPATE** (Duan et al., 2024): State-of-the-art DP data synthesis using private prompts.
- **Table Diffusion** (Truda, 2023): Diffusion-based DP tabular data generation.
- **DP-OPT** (Hong et al., 2024): DP prompt generation using an ensemble of 205 LLMs.

Environment. Evaluations utilize a system with 4x NVIDIA H100 GPUs (80GB each) and an AMD EPYC 9654 96-Core processor with 1.11TB of CPU memory (large CPU memory is unnecessary).

6.2 Effectiveness

Evaluation results without differential privacy noise (Table 2) reveal two key findings. First, **Llamdex with an MLP expert model significantly outperforms all baselines across all datasets.** Notably, on `nursery`, it surpassed the next best baseline (LoRA fine-tuning on real data) by 14%, despite no direct access to real domain data. This superiority arises from Llamdex's Llamdex encoder, specifically trained for accurate and comprehensive feature extraction from noisy natural language - a capability often underdeveloped in fine-tuned LLMs. This demonstrates the strong generalization of Llamdex's synthetic data-based training to real-world domain queries. Second, **Llamdex with an XGBoost expert also shows competitive effectiveness,** generally exceeding other baselines, though typically with slightly lower accuracy than its MLP counterpart. We attribute this to the hard

Table 1: Inference Time (minutes) and Peak Inference Memory Cost (GB)

Method	Inference Time (min)					Inference Memory (GB)				
	titanic	wine	bank	nursery	Relative[1]	titanic	wine	bank	nursery	Relative[1]
Original LLM	0.03	0.40	2.58	0.72	×0.49	18.26	19.76	20.29	20.45	×0.96
Syn. Data LoRA[2]	0.03	0.42	2.75	0.72	×0.52	18.06	19.73	20.26	21.67	×0.97
Expert API	0.87	21.73	165.0	33.13	×29.08	25.42	29.02	37.89	30.27	×1.49
Llamdex	0.07	0.80	5.27	1.45	×1.00	17.96	19.29	21.82	23.29	×1.00

[1]Mean relative value compared with Llamdex across all datasets.
[2]This category encompasses LoRA with PATE-GAN, SeqPATE, PromptPATE, Table Diffusion, and DP-OPT.

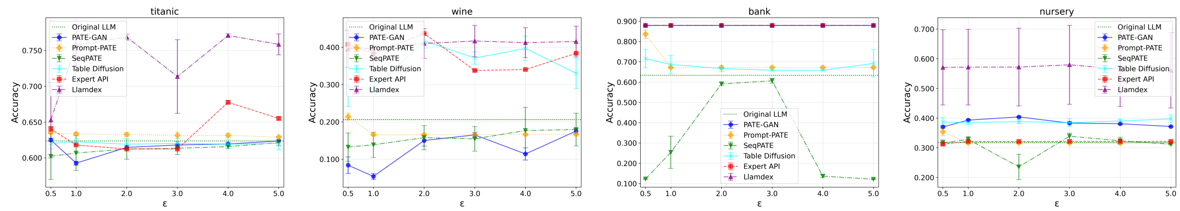


Figure 4: Accuracy of Llamdex and the baselines under different privacy budget ε

thresholding in tree-based models like XGBoost, which can be more sensitive to minor inaccuracies in extracted input features, potentially leading to incorrect predictions more readily than an MLP's smoother decision boundaries. Overall, these results validate the Llamdex framework's effectiveness in achieving high accuracy for domain-specific question answering without direct access to sensitive domain data.

Table 2: Customization Effectiveness without additional privacy noise (**bold**: best, underlined: second best).

Method	Accuracy (%)			
	titanic	wine	bank	nursery
Real Data LoRA	62.14	31.38	81.24	37.40
Original LLM	62.36	20.59	63.35	31.71
Expert API	51.46	40.63	85.32	32.10
Llamdex-XGBoost	72.81	33.75	87.94	40.92
Llamdex-MLP	**75.51**	**41.42**	**87.94**	**51.69**

6.3 Efficiency

This subsection evaluates the inference efficiency (time and memory consumption) of Llamdex against baselines, with results in Table 1 (training efficiency is in Appendix B). Two key observations emerge. First, **Llamdex achieves significantly faster inference than the Expert API**, offering an average 29× speedup while maintaining inference times similar to lower-performing baselines like LoRA fine-tuned LLMs and the original LLM. Second, Llamdex's memory consumption is similar to these less accurate baselines and 1.49× lower than the Expert API. This implies Llamdex's superior accuracy-efficiency trade-off over Expert API.

6.4 Privacy

In this subsection, we evaluate the privacy-accuracy trade-off by comparing accuracy under differential privacy with varying ε. The results, shown in Figure 4, reveal two key observations. First, **Llamdex consistently outperforms baselines even with increased noise** (smaller ε). For instance, on the `wine` dataset with $\varepsilon = 2$, Llamdex outperforms PATE-GAN and PromptPATE by 26%. Second, we observe that DP data synthesis provides accuracy close to that of the original LLM on most datasets. This is because the synthetic data usually contains too much noise, limiting the useful information available to the LLM and leading to poorer accuracy. In summary, Llamdex strikes a balance between privacy and utility, maintaining high accuracy while ensuring a strong privacy guarantee.

Membership Inference Attack. To assess our model's robustness against advanced membership inference attacks (MIA), we conducted an experiment on the `titanic` dataset using a state-of-the-art adversary. For a rigorous evaluation, we employed MICO, the winning solution from the official Microsoft Membership Inference Competition, a featured event at the premier IEEE Conference on Secure and Trustworthy Machine Learning (SaTML) 2023. We specifically utilized the champion from the `Purchase-100` track, designed for tabular data, ensuring a highly relevant and challenging adversary. We evaluated MICO's effectiveness against our expert model trained with varying levels of DP, quantified by the privacy budget ε. The results, presented in Figure 5, are evaluated using attack AUC and the competition's official metric: TPR10% FPR (True Positive Rate at a 10% False Positive Rate). The findings demonstrate that our DP-protected models are highly effective at mitigating this advanced attack. For instance, at a strong privacy level of $\varepsilon = 1.0$, the attack's TPR drops to 0.059, a substantial reduction from the 0.130 achieved against the non-private model. Crucially, this strong privacy protection is achieved while maintaining a high main task accuracy of 0.757. This indicates that our application of DP successfully protects data privacy with minimal impact on model utility, even when tested against a competition-winning attack method.

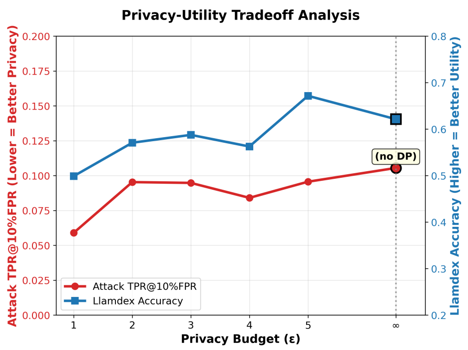


Figure 5: Tradeoff between response accuracy and attack success rate (True Positive Rate at a 10% False Positive Rate—TPR@10%FPR)

6.5 Ablation Study

This subsection presents ablation studies on the effect of expert weight and the insert layer. Additional ablation studies are detailed in Appendix D.

Effect of Expert Weight. To demonstrate that the LLM utilizes the expert's output, we scale the expert's output by a weight α and evaluate Llamdex's performance as α varies. The results, shown in Figure 6, reveal a positive correlation between Llamdex's performance and the weight α, confirming that the LLM leverages the expert's output to enhance prediction accuracy. Notably, when $\alpha = 0$, Llamdex's performance drops significantly, underscoring the importance of the expert's output for its effectiveness.

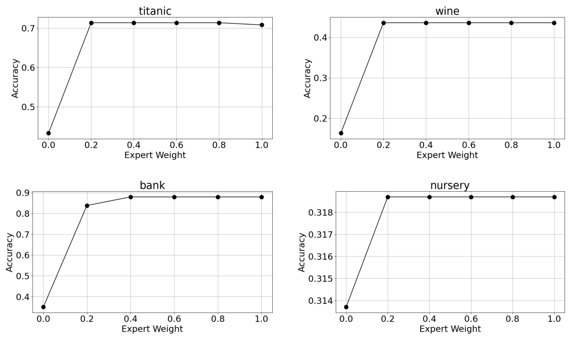


Figure 6: Effect of expert weight on Llamdex accuracy

Effect of the Insert Layer. We explore the accuracy of Llamdex when the expert model is inserted at different layers of the LLM. The effect of the depth of the insertion layer is shown in Figure 7. From the figure, we observe that the best accuracy occurs when the expert model is inserted in either the first few layers or the last few layers. The probable reason is that the initial and final layers are more closely aligned with natural language tokens, making the information easier to interpret, whereas the intermediate layers are more abstract and harder to map directly to natural language.

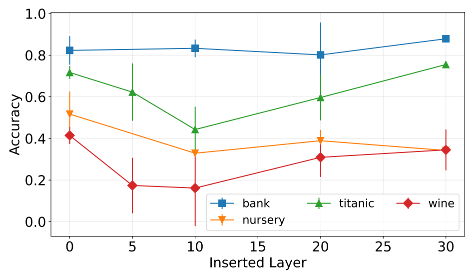


Figure 7: The effect of the depth of the inserted layer on the accuracy of Llamdex

7 Conclusion

We introduced Llamdex, a novel framework for privacy-preserving LLM service customization. Llamdex allows clients to customize LLM service with domain-specific models, thereby preserving the privacy of their private data. Experiments show that Llamdex significantly improves customization effectiveness over baselines without accessing sensitive domain data. By effectively balancing effectiveness, privacy, and inference efficiency, Llamdex provides a robust solution for deploying customized LLM services in sensitive domains.

8 Limitation

While Llamdex demonstrates a novel approach to privacy-preserving LLM service customization, the current work possesses certain limitations that also highlight avenues for future development. Firstly, the framework is primarily designed for interactions involving a single client-provided expert model. Future work could explore mechanisms for dynamic routing and complex architecture to effectively integrate multiple, diverse expert models for multi-task scenarios. Secondly, the present study focuses on customization using tabular data. Extending Llamdex to robustly support multi-modal client data, such as images or unstructured text, which would necessitate distinct Llamdex encoder and decoder architectures, remains a significant area for future research and development.

Acknowledgement

This research is supported in part by WeBank Scholars Program Innovative Research 2024. This research is also supported in part by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Paulo Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. 2009. Wine Quality. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56S3T.

Haonan Duan, Adam Dziedzic, Nicolas Papernot, and Franziska Boenisch. 2024. Flocks of stochastic parrots: Differentially private prompt learning for large language models. *Advances in Neural Information Processing Systems*, 36.

Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer.

Oluwaseyi Feyisetan, Borja Balle, Thomas Drake, and Tom Diethe. 2020. Privacy-and utility-preserving textual analysis via calibrated multivariate perturbations. In *Proceedings of the 13th international conference on web search and data mining*, pages 178–186.

Georgi Ganev, Bristena Oprisanu, and Emiliano De Cristofaro. 2022. Robin hood and matthew effects: Differential privacy has disparate impact on synthetic data. In *International Conference on Machine Learning*, pages 6944–6959. PMLR.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

M. Yasser H. 2021. Titanic. Kaggle. Https://www.kaggle.com/datasets/yasserh/titanic-dataset.

Junyuan Hong, Jiachen T Wang, Chenhui Zhang, Zhangheng Li, Bo Li, and Zhangyang Wang. 2024. DP-OPT: Make large language model your privacy-preserving prompt engineer. In *The Twelfth International Conference on Learning Representations*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022a. Lora: Low-rank adaptation of large language models. *ICLR*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022b. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Nishtha Jatana, Sahil Puri, Mehak Ahuja, Ishita Kathuria, and Dishant Gosain. 2012. A survey and comparison of relational and non-relational database. *International Journal of Engineering Research & Technology*, 1(6):1–5.

Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. 2018. Distributed learning without distress: Privacy-preserving empirical risk minimization. *Advances in Neural Information Processing Systems*, 31.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. 2018. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. 2021. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

S. Moro, P. Rita, and P. Cortez. 2014. Bank Marketing. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5K306.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

nostalgebraist. 2020. Interpreting GPT: The logit lens. *AI Alignment Forum*.

R OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2017. Semi-supervised knowledge transfer for deep learning from private training data. *ICLR*.

Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Ulfar Erlingsson. 2018. Scalable private learning with PATE. In *International Conference on Learning Representations*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. *ICLR*.

Vladislav Rajkovic. 1989. Nursery. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5P88W.

Roehit Ranganathan, G Senthil Kumar, and TS Shiny Angel. 2023. A tool for database masking and anonymization of postgresql. In *2023 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*, pages 1–6. IEEE.

C van Rijsbergen. 1979. Information retrieval 2nd ed buttersworth. *London [Google Scholar]*, 115.

Abulhair Saparov, Srushti Ajay Pawar, Shreyas Pimpalgaonkar, Nitish Joshi, Richard Yuanzhe Pang, Vishakh Padmakumar, Mehran Kazemi, Najoung Kim, and He He. 2025. Transformers struggle to learn to search without in-context exploration. In *ICLR*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Weiyan Shi, Aiqi Cui, Evan Li, Ruoxi Jia, and Zhou Yu. 2021. Selective differential privacy for language modeling. *ACL*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Zhiliang Tian, Yingxiu Zhao, Ziyue Huang, Yu-Xiang Wang, Nevin L Zhang, and He He. 2022. Seqpate: Differentially private text generation via knowledge distillation. *Advances in Neural Information Processing Systems*, 35:11117–11130.

Hugo Touvron, Louis Martin, Kevin Stone, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Gianluca Truda. 2023. Generating tabular datasets under differential privacy. *arXiv preprint arXiv:2308.14784*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, and 1 others. 2021. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint arXiv:2109.12298*.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*.

Appendix

Table of Contents

A Experimental Details

In this section, we provide additional details on the datasets used in our experiments and an example of evaluation question.

Dataset Details. Table 3 presents detailed information (including licenses) about the tabular datasets employed in the experiments, including the additional dataset `adult` in Appendix C.

Dataset and Library Licensing, and Data Considerations. The datasets utilized in our experiments are publicly available and are generally provided under permissive licenses. Specifically, the 'titanic' dataset is available under a CC0 license. The 'wine', 'adult', 'bank', and 'nursery' datasets, commonly sourced from the UCI Machine Learning Repository, are typically distributed under the Creative Commons Attribution 4.0 International (CC BY 4.0) license, which allows for sharing and adaptation with appropriate attribution. For the implementation of differentially private expert model training, we utilize the Opacus library (Yousefpour et al., 2021), which is open-source and licensed under the Apache License 2.0. This permissive licensing for both datasets and key software components facilitates reproducibility and further research. Our work relies on these pre-existing public versions and does not involve re-collection or further direct processing of raw sensitive data containing personally identifying information.

Example of Evaluation. An example of the tabular data and questions generated from the `titanic` dataset is shown in Table 4.

B Training Efficiency

The training time and memory consumption for Llamdex and the baselines are presented in Table 5. From the results, we can make one key observation: training Llamdex requires a similar order of magnitude of resources as LoRA fine-tuning, with memory consumption also comparable to LoRA. The slightly higher memory usage and increased training time in Llamdex are due to the additional parameters introduced by the Llamdex encoder and decoder. This indicates a trade-off between efficiency and accuracy, with Llamdex providing a significant improvement in accuracy over LoRA. Expert API and original LLM is not included in the training time comparison as they do not require training.

C Additional Results

In this section, we present additional experimental results. We first present the results of iterative reasoning, followed by an evaluation of F1 scores. Finally, we compare performance on the `adult` dataset.

Results of Iterative Reasoning. We demonstrate the results of iterative reasoning using a specific example. In this example, a connection prompt is used: `"The expert's answer is { }. Regard the expert's answer as a fact. Based on the expert's answer, directly answer the following question:"`. During iterative reasoning, the expert's answer is inserted into the blank and fed back into the original LLM, along with a follow-up prompt provided by the user. The results are shown in Table 6. The observations reveal that

Table 3: Detailed information of tabular datasets, including their licenses.

Dataset	#Instances	#Features	#Classes	License
titanic (H., 2021)	887	7	2	CC0
wine (Cortez et al., 2009)	4,898	11	11	CC BY 4.0
adult (Becker and Kohavi, 1996)	48,842	14	2	CC BY 4.0
bank (Moro et al., 2014)	45,211	16	2	CC BY 4.0
nursery (Rajkovic, 1989)	12,960	8	4	CC BY 4.0

Table 4: The used prompt and example of tabular data and questions from the `titanic` dataset in the evaluation

	Example
Column Names	Age, Fare[1], Parents/Children Aboard, Pclass, Sex, Siblings/Spouses Aboard, Survived
Tabular Row	18.0, 9.35, 1, 3, female, 0, 1
Question Generation Prompt	Convert the following information about a Titanic passenger into natural language. Ensure and double-check that you do not miss any information, add some irrelevant context, and ask if the passenger survived or not at the end without answering, # please: #Sex: **female** #num_parents_and_children_aboard: **1** #Fare: **9.35** #Age: **18.0** #num_siblings_and_spouses_aboard: **0** #ticket_class: **Third class**
Generated Question	This information pertains to a **female** passenger aboard the Titanic. She was **18 years old** and traveled in the **Third class**. She was accompanied by **one parent or child**. It is also noteworthy that she did **not have any siblings or spouses** aboard the ship. Her fare for the journey was **9.35** dollars. Could you please confirm if this passenger survived the tragic sinking of the Titanic or not?
System Prompt	Respond the user's question in only one word: Yes or No.
Answer of Llamdex	Yes

[1]As the unit of "fare" is not defined in the dataset, we interpret it as being in dollars without specifying the currency. Given that `titanic` serves as an external knowledge base, the actual unit does not affect the evaluation outcome.

Llamdex **not only accurately predicts the result, consistent with the real data, but also identifies related features such as age and sex.** In contrast, the original Mistral model fails to make an accurate prediction and provides vague answers.

Performance under F1 score. We evaluate Llamdex's performance on additional metrics - F1 score (Rijsbergen, 1979) - for binary classification tasks. The results, presented in Table 7, demonstrate that Llamdex achieves significant improvements in F1 scores on `titanic` and shows competitive performance on `bank`. The relatively low F1 scores across all methods on `bank` are attributed to the dataset's significant class imbalance.

Performance on `adult` Dataset. We evaluate the performance of Llamdex (without DP noise) against the original LLM, Real Data LoRA, and Expert API on the `adult` dataset, as shown in Table 8. The results indicate that Llamdex outperforms the original LLM and is competitive with the Expert API. The smaller performance improvement of Llamdex on this dataset can be attributed to the simplicity of the schema of `adult`, which makes it easier for the Expert API to extract accurate values from natural language descriptions.

D Ablation Study

D.1 Effect of Base LLM - Llama-based Llamdex

We conduct experiments similar to those in Table 2 to evaluate performance without additional privacy noise. Llamdex's performance on another base LLM (Llama-2-7B (Touvron et al., 2023)) is presented in Table 9. Notably, Llamdex with Llama-2-7B demonstrates a significant improvement in accuracy compared to the baselines, indicating that Llamdex is robust to the choice of the base LLM.

D.2 Effect of Number of Tokens

We fix the inserted layer to 0 and vary the number of tokens used to store the expert's output embeddings to evaluate Llamdex's performance. The results, shown in Figure 8, indicate that increasing the number of tokens generally improves accuracy across datasets.

Table 5: Average Training Time (minutes per epoch) and Peak Training Memory Cost (GB)

Method	Training Time[1] (min)				Training Memory (GB)			
	titanic	wine	bank	nursery	titanic	wine	bank	nursery
Real Data LoRA / PATE-GAN LoRA	3.94	4.76	4.28	3.98	16.05	17.89	16.90	16.07
Llamdex[2]	7.42 / 4.73	12.15 / 7.47	12.17 / 7.42	7.77 / 4.85	29.96	29.47	29.28	34.46

[1]For a fair comparison of efficiency, we fix the number of instances per epoch at 10,000 for all methods.
[2]The per-epoch training time of Llamdex is reported in the format of (training time of Llamdex encoder)/(training time of Llamdex decoder).

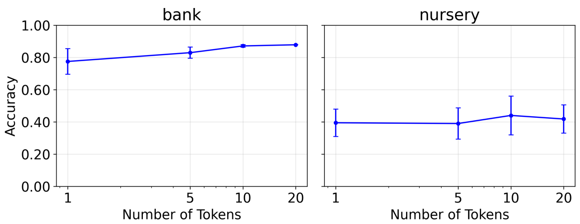


Figure 8: The effect of the number of tokens on the accuracy of Llamdex

D.3 Effect of Token Mapping

We evaluate the performance of Llamdex with and without token mapping, as shown in Table 10. The results indicate that removing token mapping significantly reduces Llamdex's accuracy, highlighting the importance of the token mapping layer. This is because the token embeddings of one LLM cannot be directly used by another LLM with a different token vocabulary. Token mapping enables the SLM to interpret the LLM's token embeddings, resulting in more accurate predictions.

D.4 Effect of Gaussian Padding.

We evaluate the performance of Llamdex with zero padding and Gaussian padding, as shown in Table 11. The results show that Gaussian padding significantly improves Llamdex's accuracy compared to zero padding. This improvement occurs because Gaussian padding breaks the symmetry of parameters, facilitating more effective learning, similar to the model's parameter initialization.

D.5 Effect of Incomplete Schemas

This subsection evaluates the robustness of Llamdex to inaccurate feature ranges in the schema. We conduct an experiment using the `titanic` dataset, focusing on the common `age` feature. In this experiment, we vary the declared range for `age` in the schema while keeping the underlying data unchanged. As demonstrated in Table 12, the resulting AUC-ROC score exhibits minimal variance, remaining stable even when the declared range is substantially mis-specified. This resilience is by design, as Llamdex utilizes broad, semantic estimates (e.g., an age range of $[0, 100]$) rather than requiring precise statistical ones.

E Baseline Hyperparams

In this section, we detail the hyperparameter settings used for each baseline in our experiments.

E.1 PATE-GAN

For PATE-GAN (Jordon et al., 2018), we follow the standard hyperparameter configurations as provided in the original GitHub implementation[2]. For models employing noisy Stochastic Gradient Descent (SGD), this involves setting the Gaussian noise variance multiplier to 2 and the gradient clipping coefficient to 0.1. Additionally, a micro-batch size of 8 is used, with training carried out over 500 epochs and a general batch size of 64. In the case of the PATE-GAN model, the default configuration specifies an inverse Laplace noise scale multiplier of 0.0001 and a batch size of 64. The model architecture includes 10 teacher discriminators. During each generator iteration in the training process, there are 5 iterations for the teachers and 5 iterations for the student.

E.2 Prompt-PATE

For the Prompt-PATE (Duan et al., 2024) baseline, we employ a configuration of 50 teacher models,

[2]https://github.com/BorealisAI/private-data-generation

Table 6: Example of Llamdex with iterative reasoning on Titanic survival prediction

Question	This information pertains to a **female** passenger aboard the Titanic. She was **18 years old** and traveled in the third class. She was accompanied by one parent or child. It is also noteworthy that she did not have any siblings or spouses aboard the ship. Her fare for the journey was 9.35 dollars. Could you please confirm if this passenger survived the tragic sinking of the Titanic or not?
Follow-up Prompt	Do you know what is the probable result and the reason what factors affect the result?
Mistral's Answer	Based on the information provided, it is not possible to definitively confirm whether this passenger survived the sinking of the Titanic. Third-class passengers were generally less likely to be rescued due to the limited lifeboats available and the priority given to first- and second-class passengers. However, there were some exceptions, and it is always possible that this individual managed to find a place on a lifeboat. Factors that affected the survival rate on the Titanic included class, gender, age, and physical ability. Women and children were given priority boarding on lifeboats, and first-class passengers were more likely to be rescued due to their ability to pay for additional seats on lifeboats. Physical ability also played a role, as those who were stronger and more agile were more likely to survive the frigid waters. It is important to note that the survival rate for third-class passengers was still relatively low, with only about **20% surviving** the sinking. However, without more specific information about this passenger, it is impossible to say for certain whether she survived or not.
Llamdex's Answer	Based on the expert's answer that the 18-year-old third-class female passenger **survived** the Titanic sinking, the probable result is that she was among the survivors. The factors that likely affected the result include: 1. Class: Third-class passengers were less likely to have access to lifeboats, though some secured spots due to **being women/children** or through other means 2. Age: **Younger** passengers, especially women and children, had boarding priority 3. Physical condition: Better condition improved survival chances in freezing water 4. Swimming ability: Swimming skills increased survival chances 5. Life jacket: Improved flotation and survival odds 6. Being in a lifeboat: Direct path to survival 7. Being rescued by another ship: Some survivors were picked up by other vessels 8. Proximity to lifeboats at launch: Better chances of boarding 9. Visibility to passing lifeboats: Increased rescue chances 10. Access to floating debris: Improved survival time in water
Survived in Dataset?	**Yes**

Table 7: F1 scores of different methods

Method	F1-Score	
	bank	titanic
Real Data LoRA	0.7988	0.5721
Original LLM	0.6936	<u>0.5738</u>
Expert API	**0.8355**	0.4933
Llamdex-MLP	<u>0.8228</u>	**0.7578**

Table 8: Accuracy of Llamdex-MLP on adult dataset

Dataset	Real Data LoRA	Original LLM	Expert API	Llamdex
adult	69.84	60.86	**76.55**	<u>76.08</u>

each consisting of Mistral-7B, which is consistent with the base LLM utilized in our primary experiments. The Confident-GNMax algorithm (Papernot et al., 2018) utilizes a confidence threshold of 10.0. Differentially private synthetic data generated by these teacher models is subsequently employed to fine-tune the target Mistral-7B LLM. This fine-

Table 9: Accuracy of Llamdex (Llama-2) without additional privacy noise

Method	Accuracy (%)			
	titanic	wine	bank	nursery
Real Data LoRA	<u>59.55</u>	<u>1.03</u>	<u>85.80</u>	**35.09**
Original LLM	39.33	0.00	12.14	30.98
Llamdex-MLP	**75.17**	**25.91**	**87.91**	<u>31.89</u>

Table 10: Performance Comparison between Llamdex w/ and w/o Token Mapping

Dataset	Accuracy (%)	
	w/o token mapping	w/ token mapping
titanic	44.38	**75.51**
wine	12.52	**41.42**
bank	45.57	**87.94**
nursery	32.20	**51.69**

tuning process is conducted using Low-Rank Adaptation (LoRA) (Hu et al., 2022b) with a rank $r = 16$ and $\alpha = 32$. Training proceeds for 10 epochs, with a learning rate of 5×10^{-5} and a batch size of 32.

Table 11: Performance of Llamdex w/ Zero Padding and Gaussian Padding

Dataset	Accuracy (%)	
	Zero Padding	Gaussian Padding
titanic	64.83	**75.51**
wine	13.62	**41.42**
bank	85.94	**87.94**
nursery	32.23	**51.69**

Table 12: AUC-ROC of Llamdex on the `titanic` dataset under different ranges for the `age` feature in the schema

Range of `age` in Schema	AUC-ROC
$[0, 40]$	0.7940
$[0, 100]$	0.8103
$[0, 1000]$	0.8141
$[0, 5000]$	0.8119

E.3 SeqPATE

For SeqPATE (Tian et al., 2022), we replicate the settings from the original paper (Tian et al., 2022). This configuration uses 10 teacher models (Mistral-7B, consistent with the base LLM in our experimental setup) with a top-k value of 200. The teacher supervision loss weight (λ) is set to 20, following the original paper. Both teacher and student models utilize Mistral-7B. Their training employs LoRA fine-tuning with a rank $r = 16$ and $\alpha = 32$. Training for these models spans 10 epochs, using a learning rate of 5×10^{-5} and a batch size of 4.

E.4 TableDiffusion

TableDiffusion (Truda, 2023) is trained for 5 epochs with a batch size of 1024 and a learning rate of 0.005. The number of diffusion steps is set to 3. The data generated by this method is then used to fine-tune the base LLM. The fine-tuning parameters are identical to those employed for Prompt-PATE (LoRA $r = 16$, $\alpha = 32$, learning rate 5×10^{-5}, batch size 32, training for 10 epochs).

E.5 DP-OPT

For DP-OPT (Hong et al., 2024), we adhere to the default parameter settings from its original code repository. This involves generating 5 prompts per data point. The LLM is configured to read a maximum of 5 demonstrations at a time, and generated prompts possess a maximum token length of 128. An ensemble of 205 LLMs (`ensemble_num`) is utilized. The generation temperature is set to 0.7, with only one generation round performed. Poisson sampling is employed with $q = 0.1$. To mitigate the generation of repetitive, low-quality text, a repetition penalty of 1.2 is applied. 10% of the training data is reserved for validation. Considering the potential for the LimitedDomain mechanism in DP-OPT to fail, a maximum of 20 failures per prompt (with retries) are permitted before prompt generation is terminated. The local model is Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and the server model is Mistral-7B-Instruct-v0.3, aligning with the base LLM configuration in our experiments. The initial prompts for each dataset are as follows:

- **titanic & bank_marketing:** `Answer the following question. Your answer MUST be either 0 (No) or 1 (Yes). Enclose ONLY the integer in \boxed{...}.`
- **wine_quality:** `Answer the following question. Your answer MUST be an integer between 0 and 10 (inclusive), where a larger integer indicates better wine quality. Enclose ONLY the integer in \boxed{...}.`
- **nursery:** `Answer the following question. Your answer MUST be an integer (0, 1, 2, or 3) corresponding to one of these categories: 0 - special priority (e.g. veterans, siblings), 1 - priority (e.g. staff children, local), 2 - very recommended (strong applicants), 3 - not recommended (weak applicants). Enclose ONLY the integer in \boxed{...}.`

During evaluation, the generated output is considered correct if the text within the `\boxed{}` environment matches the expected answer (after removing any leading or trailing whitespace). If the `\boxed{}` environment is not found, the prediction is treated as a chance probability.

E.6 Expert API

The Expert API baseline evaluates the LLM's capacity to interact with an external expert model by generating structured API calls. The LLM is provided with a detailed prompt that first describes the interface of the expert model. This description includes the precise JSON format specifying the structure of the expert model's inputs (features it accepts) and outputs (the predictions it returns). Subsequently, the prompt instructs the LLM on the specific string format required to construct a query for this expert model, based on the user's question. This query string is expected to be a sequence of feature-value pairs, clearly delineated. The template of the prompt provided to the LLM is as follows:

```
You are a data analysis assistant who strictly
adheres to instructions. You have access to a
model. It receives input in the following json
format:
<JSON input/output format for the expert model is
detailed here.>
Now you should generate a query to answer the
user's question. Use the following format to
generate queries:
<Specific query string format (e.g., "feature1:
[value1] feature2: [value2]...") is detailed
here.>
```

The generated text from the LLM, which should contain the formatted API call, is then parsed. This extracted call is used to query the external expert model. A prediction is considered correct if the response from the expert model aligns with the expected answer.

F Future Directions

In this section, we discuss the potential future extensions of the Llamdex service and the major challenges involved.

Multi-Task Llamdex. In real-world applications, more complex scenarios may arise where a single user question requires input from multiple client-provided expert models. For instance, in the medical domain, a question about a patient's symptoms might necessitate inferences from various diagnostic models, such as a radiology model, a pathology model, and a clinical model. The primary challenge in extending the Llamdex service to support multiple client-defined tasks lies in token routing. Similar to Mixture of Experts (MoE) in LLMs, a gating module would be required to determine which tokens (representing parts of the user query or intermediate states) should be routed to which client expert model. To support such multi-task customization, beyond the existing Llamdex encoder and decoder design, the Llamdex framework must also incorporate a carefully designed gating module for efficient token routing. This is left for future work.

Complex Questions. In practice, user questions are often more complex and may require multiple processing steps to arrive at a comprehensive answer. For example, a question might first require inferring a diagnosis from a radiology image (using one client expert model) before using that diagnosis to formulate a response (potentially involving another expert model or the base LLM's reasoning capabilities). While API-based approaches often utilize chain-of-thought reasoning or Depth First Search-based Decision Trees (DFSDT) to handle such complex queries, these methods, as demonstrated in our experiments, face significant efficiency challenges. A potentially more efficient approach for the Llamdex service could involve integrating client expert models at different layers of the base LLM, enabling the customized service to handle complex questions more effectively. We leave this extension for future work.

Multi-Modal Llamdex. In this paper, our focus is on tabular data, which is commonly found in relational databases. Potentially, the Llamdex service can be extended to support client customization with multi-modal data, such as images and text. Integrating these multi-modal data sources would necessitate different interfaces for the client's expert models, which in turn may require distinct designs for the Llamdex encoder and decoder components. For example, for image data, the Llamdex encoder might incorporate a convolutional neural network to process visual features for the client's image-based expert model, while for text data, it could utilize attention layers tailored to textual expert models. We leave this extension for future work.

G Discussion

This section discusses the use of AI assistants in this study and the potential risk of Llamdex.

Use of AI Assistants in Research and Development. The development of this research and the preparation of this manuscript were aided by AI-powered assistants. Specifically, Google's Gemini was utilized to assist with aspects of writing to ensure clarity in explanations. For coding tasks related to the implementation and experimentation of the Llamdex framework, Cursor, an AI-assisted code editor, was employed to enhance productivity and aid in code generation and debugging.

Potential Risks. The optional use of Differential Privacy (DP), particularly DP-SGD, for training the client-side expert model within the Llamdex framework prompts a broader discussion of such privacy-enhancing technologies. It is important to recognize that DP, as a general mechanism, has inherent limitations applicable to any system employing it. DP does not eliminate all privacy risks entirely; rather, it quantifies and reduces them to an acceptable level defined by the privacy budget (typically ε and δ). The selection of these parameters is critical: a very small ε provides stronger privacy but often leads to greater noise injection and, consequently, lower model utility. Conversely, a larger ε may yield better utility but offers weaker privacy guarantees. Therefore, the practical deployment of DP in any context, including its optional use with Llamdex, necessitates a careful and context-dependent calibration of the privacy budget. Implementers must consider the sensitivity of the data, the specific threat model, and the acceptable trade-off between privacy and model performance. Misconfiguration of DP parameters could lead to either inadequate privacy protection or a significantly degraded user experience. Llamdex's modular design for integrating client-provided models is advantageous in this regard, as it can be readily adapted to incorporate alternative or future privacy-enhancing technologies beyond DP as they become available and mature. Future work in the broader field could explore adaptive or automated methods for selecting optimal DP parameters based on data characteristics and specific application requirements to mitigate these challenges.