RewardDS: Privacy-Preserving Fine-Tuning for Large Language Models via Reward Driven Data Synthesis

Anonymous ACL submission

Abstract

The success of large language models (LLMs) has attracted many individuals to fine-tune them for domain-specific tasks by uploading their data. However, in sensitive areas like healthcare and finance, privacy concerns often arise. One promising solution is to sample synthetic data with Differential Privacy (DP) guarantees to replace private data. However, these synthetic data contain significant flawed data, which are considered as noise. Existing solutions typically rely on naive filtering by comparing ROUGE-L scores or embedding similarities, which are ineffective in addressing the noise. To address this issue, we propose *RewardDS*, a novel privacy-preserving framework that fine-tunes a reward proxy model and uses reward signals to guide the synthetic data generation. Our *RewardDS* introduces two key modules, Reward Guided Filtering and Self-Optimizing Refinement, to both filter and refine the synthetic data, effectively mitigating the noise. Extensive experiments across medical, financial, and code generation domains demonstrate the effectiveness of our method.

1 Introduction

The remarkable capabilities of Large Language Models (LLMs) in general tasks have motivated many individuals and organizations to customize their own LLMs for domain-specific applications, such as medical diagnosis, financial analysis, etc. (Wu et al., 2023; Chen et al., 2023). While domain adaptation through fine-tuning is attractive, high computational costs make local fine-tuning impractical for most users. Currently, most LLM service providers (Achiam et al., 2023; Yang et al., 2024a; Doubao, 2024) offer fine-tuning services, allowing users to customize LLMs for their needs by preparing and uploading their domain-specific data. However, these data may contain sensitive information, and directly transferring it to the LLM service provider can lead to significant privacy concerns (Zeng et al., 2024; Abdelnabi et al., 2023). Under the client-server context, we consider individuals and organizations seeking to customize LLMs as the clients, the LLM service providers as servers, and the model to be fine-tuned as the target LLM. It remains a critical challenge to develop privacy-preserving fine-tuning methods in such a client-server scenario.

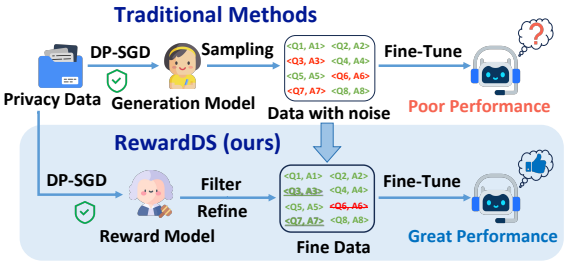


Figure 1: Illustration of how *RewardDS* overcome the dilemma of traditional synthetic data methods. The synthetic data directly sampled from the generation proxy model contain significant flaws, such as incoherent text or incomplete storylines, which are considered noise.

Prior works proposed data synthesis as a promising solution to the challenge (Yue et al., 2023; Kurakin et al., 2023; Yu et al., 2023; Mattern et al., 2022; Flemings and Annavaram, 2024). This approach generates synthetic data to replace the private data used for fine-tuning, thus ensuring privacy protection. Specifically, a generation proxy model is first trained on the private data, optimized by DP-SGD (Abadi et al., 2016) to safeguard privacy. The generation proxy model then generates synthetic data for subsequent LLM training. However, due to the inherent randomness of the sampling process, the synthetic data inevitably contains significant flawed data, including text incoherence or storyline incompleteness, which is considered as noise and leads to less effective LLM fine-tuning, as illustrated in Figure 1 We also provide an example in Figure 7.

To mitigate the noise, existing methods (Wang et al., 2022; Yu et al., 2024; Xie et al., 2024) proposed to filter out flawed data by measuring its similarity to private data. Wang et al. (2022) use ROUGE-L similarity, while Yu et al. (2024); Xie et al. (2024) compute embedding similarity. However, these metrics fail to evaluate the synthetic data's effectiveness for domain-specific tasks. Alternative methods (Li et al., 2024b; Wang et al., 2024) sample synthetic data directly from the target LLM on the server to improve quality. But the target LLM is not fine-tuned on domain-specific tasks, so the sampled synthetic data does not help improve model performance on those tasks.

To mitigate the noise in synthetic data while protecting privacy, we propose *RewardDS* (**Reward**-driven **D**ata **S**ynthesis), a novel privacy-preserving framework that improves synthetic data quality for the target LLM's privacy-preserving fine-tuning. RewardDS implements a two-stage quality control process, i.e., filtering and refinement, as illustrated in Figure 1. Specifically, we first train a reward proxy model on private data to assess data quality for domain-specific tasks, using DP-SGD to safeguard privacy. Through **Reward Guided Filtering**, we apply the reward proxy model to assess synthetic data generated by the generation proxy model and remove samples with low reward scores. Filtering alone may remove a large amount of data, leaving only a small fraction. Therefore, we aim to further refine the synthetic data to obtain more high-quality data. Our **Self-Optimizing Refinement** module generates multiple candidate responses for each synthetic query and computes their rewards. The generation proxy model analyzes the highest and lowest scoring responses and then generates improvement feedback. Based on this feedback, the target LLM refines the synthetic data following a refinement instruction. The resulting high-quality, filtered, and refined synthetic data is then used to fine-tune the target LLM for domain-specific tasks.

We conduct extensive experiments across various domain-specific generation tasks, including Medical Question Answering (QA), Legal QA, and Code Generation tasks. The results consistently demonstrate the effectiveness of our method in improving the quality of the synthetic data, achieving better performance while preserving privacy. Our main contributions are summarized as follows:

- We propose *RewardDS*, a novel privacy-preserving fine-tuning framework that improves the quality of synthetic data by training a Reward Proxy Model on the client side to guide synthetic data generation.
- We introduce the Reward Guided Filtering and Self-Optimizing Refinement modules to filter and refine the synthetic data, thereby enhancing its quality.
- We conducted extensive experiments across Medical QA, Legal QA, and Code Generation tasks to validate the effectiveness of our proposed framework.

2 Related Work

2.1 Privacy Preserving Fine-tuning Methods.

Domain-specific data, such as medical diagnoses and financial reports, often contain sensitive information, and directly fine-tuning LLMs on such data raises privacy concerns (Mokhtarabadi et al., 2024; Wang et al., 2023; Jang et al., 2023). Differentially Private Stochastic Gradient Descent (DP-SGD) injects noise into gradients during fine-tuning, ensuring the model does not memorize private data (Abadi et al., 2016; McMahan et al., 2017). Alternatively, data anonymization methods, such as k-anonymity and adversarial anonymization, detect and remove private information to prevent privacy leakage while maintaining model utility (Sweeney, 1997; Romanov et al., 2019; Staab et al., 2024). Another promising approach is generating synthetic data with Differential Privacy (DP) guarantees as a substitute for private data (Yue et al., 2023; Flemings and Annavaram, 2024). This synthetic data is protected by the DP mechanism, contains no user privacy, and can be freely used for further fine-tuning.

2.2 Privacy-Preserving Synthetic Text Generation.

Recent studies have explored synthetic data with differential privacy guarantees as a substitute for private data in LLM fine-tuning, achieving a balance between data utility and privacy protection (Yue et al., 2023; Yu et al., 2024; Kurakin et al., 2023). A series of work has been proposed to reduce computational costs and achieve high-quality synthetic data. Lin et al. (2024); Xie et al. (2024) use APIs and zero-shot learning to generate synthetic data without fine-tuning. Du et al. (2024) combine the strengths of multiple models to generate synthetic data, mitigating the risks associated

with relying on a single model. Zou et al. (2025) integrate knowledge from pre-trained language models and generate differentially private synthetic data. Wang et al. (2024) integrates differential privacy with knowledge distillation from professional models, leveraging both local and professional models to generate high-quality synthetic data. However, these methods overlook the noise introduced during the synthetic data sampling process, which can degrade performance. To mitigate this, Yu et al. (2024); Wang et al. (2022) compute the similarity between synthetic and private data to filter out those with low similarity. However, these similarity measures are too surface-level to effectively capture the quality of synthetic data for domain-specific tasks. Therefore, a more robust framework is needed to address the noise in synthetic data and enhance its quality for domain-specific tasks.

3 Problem Statement

We consider a scenario where the client holds domain-specific data, such as patient's medical records, which contain sensitive information. Hence, directly transmitting those data to servers for LLM fine-tuning is not allowed. This private data typically is structured as *Query-Response* pairs, with both query and response containing confidential private information (Wang et al., 2024). The server, which hosts the target LLM, offers only API access while keeping model weights confidential, preventing clients from accessing or locally fine-tuning the model. While clients can fine-tune lightweight LLMs within their computational constraints, these models have inherently weaker capabilities than the target LLMs. This creates a critical challenge: how to leverage a client's private data to improve the server-hosted LLM's performance on domain-specific tasks while preserving privacy, given that clients cannot locally fine-tune the target LLM due to inaccessibility of model weights.

Existing methods utilize a lightweight **Generation Proxy Model** on the client side to generate safe synthetic data for fine-tuning the target LLM on the server (Yue et al., 2023; Yu et al., 2024). However, the randomness of the sampling process may introduce noise in the synthetic data, potentially causing performance degradation. Therefore, our main goal is to *explore a more effective method for mitigating the noise in synthetic data, enabling better fine-tuning performance while maintaining user privacy*.

4 Method

To address the performance degradation caused by noise in synthetic data, we propose a novel framework, *RewardDS* (**Reward**-driven **D**ata **S**ynthesis), as shown in Figure 2. Our approach additionally trains a **Reward Proxy Model** on the client side. Then the reward proxy model filters and refines the synthetic data sampled from the generation proxy model through **Reward Guided Filtering** and **Self-Optimizing Refinement** modules on the server side. Both modules collaborate to enhance the quality of the synthetic data, driven by the reward signal from the reward model. We will introduce the training process of the generation proxy model and reward proxy model in § 4.1 and the details of reward guided filtering and self-optimizing refinement module are provided in § 4.2.

4.1 Client Side

Generation Proxy Model Training. The generation proxy model is responsible for generating safe synthetic data as a substitute for private data. Following (Yue et al., 2023; Yu et al., 2024, 2022; Kurakin et al., 2023), we fine-tune a generation proxy model on the client's private data using the DP-SGD algorithm (Abadi et al., 2016). The backbone of generation proxy model should be lightweight due to limited computational resources on the client side, e.g., Qwen2.5-0.5B-Instruct (Yang et al., 2024b). The DP-SGD algorithm protects the privacy of the training data by injecting noise into the gradients during model training. This noise ensures that the inclusion or exclusion of any individual training sample has minimal impact on the fine-tuned model, thereby providing privacy protection.

Reward Proxy Model Training. The reward model is responsible for evaluating the quality of the synthetic data. It should provide higher rewards for high-quality data while lower rewards for poor-quality data. Following standard reward model training practices Liu et al. (2024), we train the reward proxy model using paired comparison data. Let W_0 denote the initial backbone model, W_{gen} the fine-tuned generation proxy model, and W_{rwd} the fine-tuned reward proxy model. For each query Q from the private dataset with its gold response A_{gold}, we generate two responses: A_0 from W_0 and A_{gen} from W_{gen}. We then create preference pairs by selecting either A_{gen} or A_{gold} as the chosen response A_c, with A_0 serving as the rejected

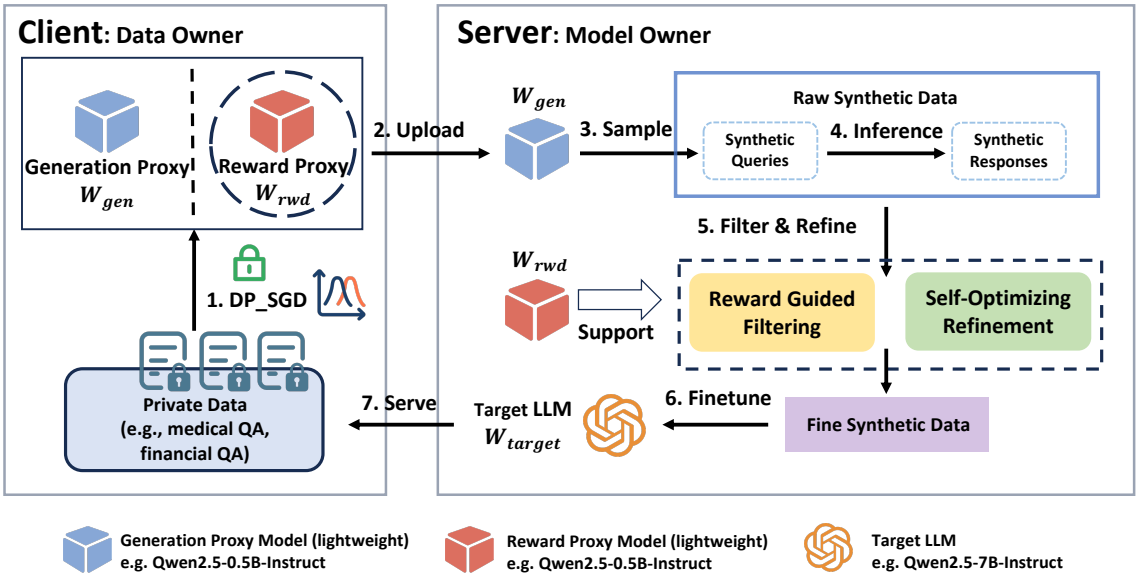


Figure 2: The overview of our *RewardDS* framework. The client uses DP-SGD to fine-tune two lightweight proxy models on privacy-sensitive data: the Generation Proxy Model W_{gen} and the Reward Proxy Model W_{rwd}. Both proxy models are then sent to the server. The Generation Proxy Model is used to sample raw synthetic data, consisting of queries and responses. The Reward Proxy Model supports the **Reward Guided Filtering** and **Self-Optimizing Refinement** modules, which filter and refine the raw synthetic data to produce fine synthetic data. Finally, the target LLM W_{target} is fine-tuned on the fine synthetic data and provides service to the client for domain-specific tasks.

response A_r. The reward proxy model maintains a lightweight architecture for client-side deployment and is fine-tuned using differential privacy (DP-SGD) to prevent privacy leakage.

Following Rafailov et al. (2023), we define the training loss as:

$$\mathcal{L} = -\log \sigma \left(f_{\text{rwd}}(Q, A_c) - f_{\text{rwd}}(Q, A_r) \right), \quad (1)$$

where $f_{\text{rwd}(\cdot)}$ represents the reward predicted by W_{rwd}. This training loss encourages the reward model to assign higher scores to responses from the generation proxy model and gold responses compared to those from the initial backbone model.

After training, both generation proxy model and reward proxy model are sent to the server.

4.2 Server Side

Synthetic Data Generation. Following Yu et al. (2024); Wang et al. (2024), we use W_{gen} to generate both synthetic queries and their corresponding responses, collectively referred to as raw synthetic data. Although the generation proxy model W_{gen} is trained on private data and learns domain-specific knowledge, the generation process of raw synthetic data is random and unstable. As a result, the raw synthetic data inevitably contains noisy samples, and fine-tuning the LLM directly on this data can lead to performance degradation.

Reward Guided Filtering. We leverage the reward proxy model W_{rwd} to evaluate each synthetic data and filter out those with low rewards. A lower reward indicates a higher likelihood of the synthetic data being noisy. We select only the top $\lfloor L/k \rfloor$ data, where L is the total number of synthetic data and k is the partition fold (Line 10 in Alg. 1). To compensate for the reduced synthetic dataset size after filtering, we replicate the high-reward data to maintain the total data volume during the target LLM fine-tuning (Line 11 in Alg. 1).

Self-Optimizing Refinement. While filtering mitigates noise, it selects only a small subset of samples, potentially leading to overfitting on limited data. Building on LLMs' self-reflection capabilities (Madaan et al., 2023), we implement a dynamic data refinement strategy to improve low-reward samples, enhancing overall data quality. Initially, for each synthetic query, we generate N candidate responses rather than only one response using the generation proxy model (Line 3 in Alg. 1). The reward proxy model then selects the response with the highest reward score as the chosen response (Line 5 in Alg. 1). We directly fine-tune the target LLM W_{target} on the chosen response (Line 16 in Alg. 1).

After fine-tuning the target LLM W_{target} for each epoch, we dynamically refine the synthetic data for

Algorithm 1 *RewardDS* based LLM Fine-tuning

Input: Synthetic query set $\mathcal{Q}_{\text{query}}$, number of synthetic query L, number of candidate responses N, partition fold k, generation proxy model W_{gen}, reward proxy model W_{rwd}, target LLM W_{target}, training epoch T

Output: The fine-tuned LLM W_{target}^{T}

// Before Fine-tuning LLM

for *each query* $q \in \mathcal{Q}_{query}$ **do**

  Generate candidate response set: $\{A_j\}_{j=1}^{N} \leftarrow W_{\text{gen}}(q)$

  Predict the reward score: $\{s_j\}_{j=1}^{N} \leftarrow W_{\text{rwd}}(q, A_j)$

  Select the best and the worst response: $(A_c, A_r) \leftarrow \left(A_{\arg\max_j s_j},\ A_{\arg\min_j s_j}\right)$

  Record the best reward score: $s_c \leftarrow \max_j s_j$

Gather the initial synthetic dataset: $\mathcal{D}_0 \leftarrow \{(q_i, A_c^i, A_r^i, s_c^i)\}_{i=1}^{L}$

Sort $\mathcal{D}_0$ by reward: $\mathcal{D}_0^{\text{sorted}} \leftarrow \{(q_i, A_c^i, A_r^i, s_c^i)\}_{i=1}^{L}$, where $s_c^1 \geq \cdots \geq s_c^L$

Partition $\mathcal{D}_0^{\text{sorted}}$ into k folds: $\{\mathcal{D}_0^m\}_{m=1}^{k} \leftarrow \text{split}(\mathcal{D}_0^{\text{sorted}}, k)$

Extract top-$\lfloor L/k \rfloor$ samples: $\mathcal{D}_{\text{high}} \leftarrow \mathcal{D}_0^1$

Replicate subset to obtain the train set: $\mathcal{T}_0 \leftarrow \bigoplus_{\lceil L/|\mathcal{D}_{\text{high}}| \rceil} \mathcal{D}_{\text{high}}$

Determine score threshold τ: $\tau \leftarrow \min_{s_c \in \mathcal{D}_{\text{high}}} s_c$

// During Fine-tuning LLM

Initialize target LLM: $W_{\text{target}}^{0} \leftarrow W_{\text{target}}$

for *iteration* $t = 1$ **to** T **do**

  Fine-tune target LLM W_{target}^{t-1} on $\{(q, A_c) \in \mathcal{T}_{t-1}\}$ and get W_{target}^{t}

  for *each query* $q \in \mathcal{Q}_{query}$ **do**

    Generate feedback ϕ: $\phi \leftarrow W_{\text{gen}}(q, A_c, A_r)$

    Re-generate the candidate response set: $\{A_j\}_{j=1}^{N} \leftarrow W_{\text{target}}^{t}(q, \phi)$

    Predict the reward score, select the best and worst responses, and record the highest reward score to update $\mathcal{D}_{t-1}$, yielding $\mathcal{D}_t$.

  Filter and get new training set $\mathcal{T}_t$: $\mathcal{T}_t \leftarrow \{(q, A_c, A_r, s_c) \in \mathcal{D}_t \mid s_c \geq \tau\}$

return M_{target}^{t}

the next epoch's training. For each query's N candidate responses, we identify the lowest-reward responses and combine them with the highest-reward responses to form the rejected (A_r) and chosen (A_c) response pairs. The generation proxy model M_{gen} analyzes these responses and provides feedback, highlighting the strengths of A_c and weaknesses of A_r (Line 18). This feedback, along with the original query, guides the target LLM W_{target} to generate N refined candidate responses (Line 19). Finally, the reward proxy model selects the highest-reward response from these refined candidates for the next epoch's LLM fine-tuning (Line 20).

The collaborative process between the reward-guided filtering and self-optimizing refinement modules is presented in Alg. 1. The refinement instruction templates are provided in Appendix H. After the LLM is fine-tuned on the refined synthetic data, it can provide service to the client for those domain-specific tasks.

5 Privacy Analysis

The only transmitted content between the client and server are the generation proxy model and the reward proxy model. Both models are fine-tuned on the private dataset using the DP-SGD algorithm (Abadi et al., 2016). According to the definition of differential privacy (DP) (Dwork and Roth, 2014), adversaries cannot infer any private data from the fine-tuned proxy models. Additionally, based on the post-processing property of the DP framework (Dwork and Roth, 2014), any further operations on the two proxy models will not cause privacy leakage. All subsequent operations on the server, including synthetic data generation, reward-guided filtering, and self-optimizing refinement, are privacy-preserving.

We have fine-tuned two proxy models on the private dataset and the privacy budget of each fine-tuning is (ϵ, δ). According to the sequential composition law of DP mechanism (Dwork and Roth, 2014), the total privacy budget of our framework is $(2\epsilon, 2\delta)$.

6 Experiments

6.1 Experiments Setup

Datasets. We evaluate our method across three domain-specific generation tasks using established datasets: Medical QA using HealthCareMagic-100k (Li et al., 2023), Financial QA using fingpt-fiqa_qa (Zhang et al., 2023), and Code Generation using opc-sft-stage2 (Huang et al., 2024).

Evaluation Metrics. For the evaluation of the QA task, we employ the ROUGE-1 (R1), ROUGE-L (RL) (Lin, 2004), and Perplexity (PPL) (Hu et al., 2024) as metrics. While automated metrics focus on lexical overlap and fluency, LLM-Judge (Zheng et al., 2023) provides a more comprehensive assessment of semantic accuracy and response quality. Hence, we also use LLM-Judge as a metric. For the code generation task, we use Pass@1 and Pass@10 as evaluation metrics (Chen et al., 2021).

Implementation Details. We use the Qwen2.5-0.5B-Instruct model (Yang et al., 2024b) as the backbone for the generation/reward proxy model, and the Qwen2.5-7B-Instruct model as the target LLM on the server. During each DP-SGD fine-tuning process of both proxy models, we set the privacy budget to $(8, 1e^{-5})$. As a result, the total privacy budget for our method is $(16, 2e^{-5})$, according to the sequential composition law of the DP mechanism (Abadi et al., 2016). For a fair comparison, we set the same privacy budget for all compared methods. The size of the synthetic

Methods	Medical QA			Financial QA			Code Generation	
	R1 ↑	RL ↑	PPL ↓	R1 ↑	RL ↑	PPL ↓	Pass@1 ↑	Pass@10 ↑
Vanilla LLM	21.60	11.50	1.34	23.91	11.72	1.38	18.82	42.06
Locally Fine-tuning	<u>23.82</u>	<u>15.46</u>	1.71	13.26	10.19	1.67	<u>28.34</u>	43.99
DP-Generation (Kurakin et al., 2023)	16.22	10.94	<u>1.06</u>	14.97	11.20	1.05	25.51	42.75
DP-Instruct (Yu et al., 2024)	11.94	8.44	**1.04**	14.06	10.76	<u>1.04</u>	26.27	48.06
KnowledgeSG (Wang et al., 2024)	20.28	10.74	1.31	<u>24.14</u>	12.33	1.21	23.93	<u>49.58</u>
RewardDS	**27.78**	**17.02**	1.17	**24.42**	**14.96**	**1.02**	**32.41**	**49.99**
w/o Reward Guided Filtering	20.38	13.11	1.28	17.93	<u>12.52</u>	1.25	23.03	34.96
w/o Self-Optimizing Refinement	22.70	13.42	1.36	14.14	11.07	1.18	22.27	33.17

Table 1: Comparisons of our method with baselines across three domain-specific tasks: Medical QA, Financial QA, and Code Generation. Higher values of ROUGE-1 (R1) and ROUGE-L (RL), and lower values of Perplexity (PPL) indicate better performance on the QA generation task. Higher values of Pass@1 and Pass@10 reflect better performance in the code generation task. Numbers in **bold** and <u>underlined</u> represent the best and second-best results, respectively.

dataset is always kept to twice that of the client's private data across all baselines. These settings align with established DP deployments such as Apple's QuickType and Google's models, as noted by Lukas et al. (2023).

More details on the datasets used and the implementation are provided in Appendix A and Appendix C, respectively.

6.2 Compared Methods.

To demonstrate the effectiveness of our method, we consider several baselines for comparison:

Vanilla LLM refers to using a general-purpose LLM for domain-specific tasks without any domain adaptation or fine-tuning. **Locally Fine-tuning** refers to training a lightweight model locally on clients' private data.

DP-Generation (Kurakin et al., 2023) fine-tunes the generation proxy model on the client side using DP-SGD. This proxy model is then used to generate synthetic data, which is subsequently utilized to fine-tune the target LLM on the server. **DP-Instruct** (Yu et al., 2024) introduces additional filtering operations based on the similarity of synthetic queries before LLM fine-tuning; **KnowledgeSG** (Wang et al., 2024) utilizes the synthetic data to enhance the generation proxy model for domain-specific tasks instead of fine-tuning the target LLM.

More details of the compared method are provided in Appendix B.

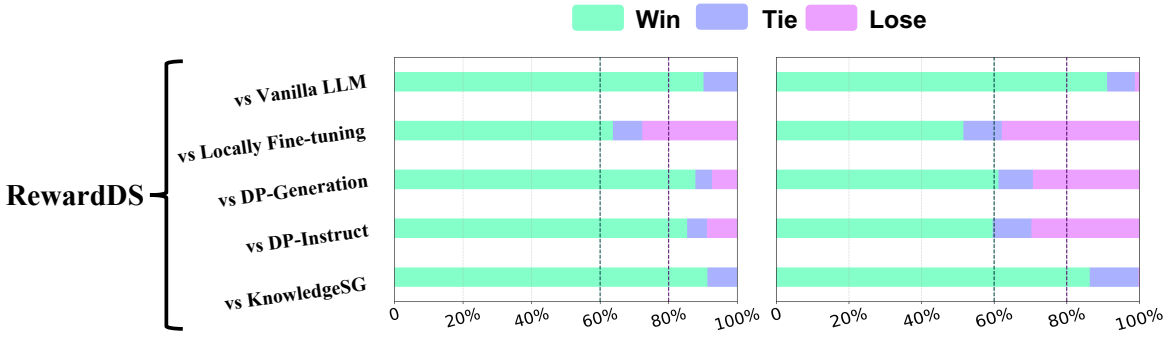
6.3 Main Results

As shown in Table 1, *RewardDS* outperforms all other baselines across the three domain-specific tasks, except for the PPL on the Medical QA task. DP-Instruct achieves marginally lower PPL in medical QA. This is possibly due to the filtering by similarity, leading the target LLM to overfit on these highly similar samples.

The Vanilla LLM exhibits suboptimal performance across medical QA, financial QA, and code generation tasks, primarily due to the lack of domain-specific fine-tuning on private data. While Locally Fine-tuning a lightweight proxy model (with only 0.5B parameters) mitigates privacy concerns, the small model's limited capacity hinders its ability to effectively learn domain-specific knowledge, leading to subpar performance.

DP-Generation samples synthetic training data to fine-tune the target LLM on the server. However, due to the randomness inherent in the sampling process, the resulting synthetic data contains significant noise, which severely impairs the fine-tuning performance of the LLM on the server. Although DP-Instruct attempts to filter the synthetic data by computing the similarity between the synthetic query and the private query, it still does not perform well. Similarity alone cannot accurately reflect the quality of synthetic data, where higher similarity does not necessarily indicate better data quality.

KnowledgeSG utilizes synthetic data to fine-tune the lightweight proxy model on the server, enhancing it with the assistance of the target LLM. However, the quality of the synthetic data is highly dependent on the target LLM's capacity for the specific domain task. If the target LLM performs poorly, the synthetic data will likely contain more



(a). Medical QA Task (b). Financial QA Task

Figure 3: Using LLM-Judge (Zheng et al., 2023) to compare the outputs generated by our method with those of other baselines. **Win** means our method outperformed the baselines, **Tie** means the results were similar, and **Lose** means our method performed worse than the baselines.

noise, which could harm the performance of the proxy model. KnowledgeSG performs relatively better on the financial QA task compared to the other two tasks, primarily because the Vanilla LLM performs well on financial QA, whereas it does not on the other tasks. Only in the financial QA task, the performance of the Vanilla LLM surpasses that of the Locally Fine-tuned model, whereas this is not the case for the other tasks.

We also present results after removing the Reward Guided Filtering and Self-Optimizing Refinement modules respectively. We observe the decline in performance across all tasks when either module is removed, which highlights the effectiveness of these modules. Without these modules, more noisy synthetic samples are included during LLM fine-tuning, resulting in performance degradation.

6.4 Evaluation using LLM-Judge

We also use LLM-Judge (Zheng et al., 2023) for a more reliable evaluation of the medical QA and financial QA tasks. While ROUGE metrics measure lexical similarity to references and PPL captures fluency, these metrics often fail to assess deeper aspects of response quality. Inspired by Zheng et al. (2023), we fine-tune an LLM judger to assess the quality of generated outputs across different baselines. We provide the judger with both the user query and the generated outputs from our method and the baselines, allowing it to determine which is better or declare a tie. Details of the judger training and evaluation process are shown in Appendix D.

As shown in Figure 3, our method outperforms other baselines in both the medical QA and financial QA tasks. DP-Generation and KnowledgeSG struggle with noisy samples from synthetic data, leading to poor performance. Although DP-Instruct filters synthetic data by comparing with private data and removing low-similarity samples, it achieves only limited performance gains compared to DP-Generation. This shows that simple similarity measures do not fully capture the quality of synthetic data. Locally Fine-tuning avoids synthetic data noise by fine-tuning a lightweight proxy model on private data locally, but it still underperforms our method due to the limited learning capacity of the lightweight model for domain-specific knowledge.

6.5 Hyperparameter Analysis

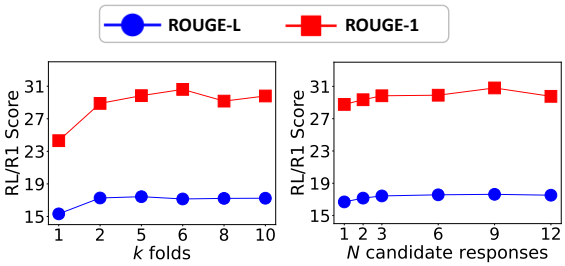
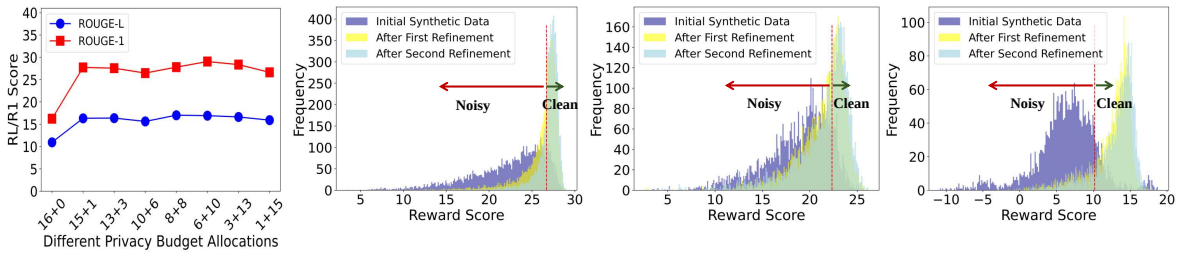


Figure 4: Analysis of hyperparameters including the number of folds k and the number of candidate responses N in Alg. 1 on the medical QA task. To analyze k, we set $N = 3$; To analyze N, we set $k = 6$.

We analyze the effect of hyperparameters on our method described in Alg. 1. As shown in Alg. 1, the number of folds k controls the amount of selected synthetic data. A smaller k means more synthetic data is included, but it may also introduce more noise. As illustrated in Figure 4(a), when $k = 1$ (using all the synthetic data), performance decreases. Using a larger k can help exclude noisy data, improving performance. However, setting k too large and excluding too much synthetic data can slightly degrade performance. Therefore, we set $k = 6$ for the medical QA task.



(a). Different Allocations for Two Proxy Models

(b). Data Quality Change on Medical QA Task

(c). Data Quality Change on Financial QA Task

(d). Data Quality Change on Code Generation Task

Figure 5: Effectiveness of *RewardDS* design. (a): Performance on medical QA with different privacy budget allocations for generation and reward proxy model training. The allocation of 'x + (16-x)' means the privacy budget for training the generation proxy model is set to x, while the reward proxy model is set to (16-x); (b)/(c)/(d): Quality improvement of the synthetic data, on the Medical QA/Financial QA/Code Generation tasks, after applying the Self-Optimizing Refinement module multiple times.

The hyperparameter N controls the number of candidate responses. As shown in Figure 4(b), increasing N slightly improves performance because a larger number of candidates increases the chance of selecting better synthetic data. However, generating more candidates incurs additional time and computational costs. Therefore, for the medical QA task, we set $N = 3$.

Further hyperparameter analyses for the legal QA task and code generation task can be found in Appendix E.

6.6 In-depth Analysis of *RewardDS* Design

Here, we provide more detailed analysis on the design and effectiveness of *RewardDS*.

Analysis 1: *The impact of different privacy budget allocations.*

Although the total privacy budget is controlled at $(16, 2e^{-5})$, we can allocate more privacy budget to the generation reward model or the reward proxy model. We will investigate the impact of different privacy budget allocations in Figure 5(a). The results indicate that even a small privacy budget for reward model fine-tuning (e.g., "15+1", with only 1 allocated to the reward model) outperforms the case where no privacy budget is allocated to the reward model ("16+0"). No privacy budget allocated for reward model means that we do not train the reward proxy model on the client for data filtering or refinement. This suggests that **even a marginal privacy cost for reward model training can yield substantial benefits**. Furthermore, allocating more privacy budget to the reward model will bring only marginal performance improvements.

Analysis 2: *Impact of Self-Optimizing Refinement on Synthetic Data Quality.*

As shown in Alg. 1, we use the self-optimizing refinement module to re-generate synthetic responses and improve quality during each training epoch. To assess the effectiveness of our self-optimizing refinement module (Alg. 1), we track reward scores of synthetic responses across multiple refinement iterations. A higher reward score indicates better synthetic data quality. Figures 5(b-d) demonstrate that **synthetic data quality improves gradually through iterative refinement**, explaining our method's superior performance.

Analysis 3: *Generalizability across different LLM backbones.*

We have evaluated our method with different backbones as the target LLM, including Llama-2-7B-chat-hf (MetaAI, 2023) and Qwen2.5-14B-Instruct (Yang et al., 2024b). Table 3 shows that our method maintains superior performance across various backbones, **confirming its backbone-agnostic effectiveness**. More analysis is in Appendix F.

7 Conclusion

We propose a novel privacy-preserving framework, *RewardDS*, to mitigate noise in synthetic data during LLM privacy-preserving fine-tuning. Specifically, *RewardDS* fine-tunes a reward model and leverages the reward signal to guide the synthetic data generation process. During the data synthesis process, *RewardDS* employs the collaboration of Reward Guided Filtering and Self-Optimizing Refinement modules to filter and refine synthetic data, mitigating noise. We conduct extensive experiments across medical QA, legal QA, and code generation tasks. The results consistently demonstrate the effectiveness of *RewardDS* for privacy-preserving LLM fine-tuning.

Limitations

While *RewardDS* has demonstrated its effectiveness in medical QA, legal QA, and code generation tasks, it incurs additional training costs for the reward proxy model. Although the model is lightweight, it still requires extra computational resources.

Additionally, due to computational resource constraints, we applied LoRA fine-tuning on the Qwen2.5-14B-Instruct model to validate our method, as discussed in Appendix F. Full-parameter fine-tuning may yield even better performance. Future work will explore larger LLM backbones and additional categories to further demonstrate the effectiveness of our method as computational resources allow.

In the future, we aim to expand our experiments to include more domain-specific tasks and a wider range of LLM backbones. Furthermore, we plan to optimize the local fine-tuning process of the lightweight proxy models on the client side to reduce computational burdens, enhancing the scalability and feasibility of our method.

8 Ethics Statement

We adhere to the ACL Ethics Policy and all of our research is based on publicly available repositories and datasets. In the *RewardDS* framework, we uphold strict ethical standards to protect user privacy and ensure data security. The datasets used, covering medical QA, financial QA, and code generation domains, are publicly available and free of personally identifiable information, minimizing privacy risks. Our methodology does not access or reconstruct identifiable data, safeguarding individual privacy rights.

However, as our study involves multiple LLMs, such as Llama and Qwen, the findings may be influenced by the inherent biases, linguistic patterns, and assertiveness of these models.

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*, CCS '16, page 308–318, New York, NY, USA. Association for Computing Machinery.

Sahar Abdelnabi, Kai Greshake, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, AISec 2023, Copenhagen, Denmark, 30 November 2023*, pages 79–90. ACM.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, et al. 2023. Meditron-70b: Scaling medical pretraining for large language models. *arXiv preprint arXiv:2311.16079*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nat. Mac. Intell.*, 5(3):220–235.

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, Bin Gu, Mengfei Yang, and Ge Li. 2024. Generalization or memorization: Data contamination and trustworthy

evaluation for large language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 12039–12050.

Doubao. 2024. Doubao: Ai large model platform. Accessed: 2024-12-09.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407.

James Flemings and Murali Annavaram. 2024. Differentially private knowledge distillation via synthetic text generation. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 12957–12968. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

Yutong Hu, Quzhe Huang, Mingxu Tao, Chen Zhang, and Yansong Feng. 2024. Can perplexity reflect large language model's ability in long text understanding? In *The Second Tiny Papers Track at ICLR 2024, Tiny Papers @ ICLR 2024, Vienna, Austria, May 11, 2024*.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, J. H. Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu, Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. 2024. Opencoder: The open cookbook for top-tier code large language models.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2023. Exploring the benefits of training expert language models over instruction tuning. In *International Conference on Machine Learning*.

Alexey Kurakin, Natalia Ponomareva, Umar Syed, Liam MacDermed, and Andreas Terzis. 2023. Harnessing large-language models to generate private synthetic text. *CoRR*, abs/2306.01684.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Haoran Li, Dadi Guo, Donghao Li, Wei Fan, Qi Hu, Xin Liu, Chunkit Chan, Duanyi Yao, Yuan Yao, and Yangqiu Song. 2024a. Privlm-bench: A multi-level privacy evaluation benchmark for language models. *Preprint*, arXiv:2311.04044.

Haoran Li, Xinyuan Zhao, Dadi Guo, Hanlin Gu, Ziqian Zeng, Yuxing Han, Yangqiu Song, Lixin Fan, and Qiang Yang. 2024b. Federated domain-specific knowledge transfer on large language models using synthetic data. *CoRR*, abs/2405.14212.

Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. 2023. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6).

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, Harsha Nori, and Sergey Yekhanin. 2024. Differentially private synthetic data via foundation model apis 1: Images. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella Béguelin. 2023. Analyzing leakage of personally identifiable information in language models. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 346–363.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Justus Mattern, Zhijing Jin, Benjamin Weggenmann, Bernhard Schoelkopf, and Mrinmaya Sachan. 2022. Differentially private language models for secure data sharing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4860–4873, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

H. B. McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private language models without losing accuracy. *ArXiv*, abs/1710.06963.

MetaAI. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Hojjat Mokhtarabadi, Ziba Zamani, Abbas Maazallahi, and Mohammad Hossein Manshaei. 2024. Empowering persian llms for instruction following: A novel dataset and training approach.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2022a. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1864–1874.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2022b. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1864–1874. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Aleksandr Romanov, Anna Kurtukova, Anastasia Fedotova, and Roman Meshcheryakov. 2019. Natural text anonymization using universal transformer with a self-attention. In *Proceedings of the III International Conference on Language Engineering and Applied Linguistics (PRLEAL-2019), Saint Petersburg, Russia*, pages 22–37.

Robin Staab, Mark Vero, Mislav Balunovic, and Martin T. Vechev. 2024. Large language models are advanced anonymizers. *CoRR*, abs/2402.13846.

Latanya Sweeney. 1997. Guaranteeing anonymity when sharing medical data, the datafly system. *Proceedings : a conference of the American Medical Informatics Association. AMIA Fall Symposium*, pages 51–5.

Wenhao Wang, Xiaoyu Liang, Rui Ye, Jingyi Chai, Siheng Chen, and Yanfeng Wang. 2024. Knowledgesg: Privacy-preserving synthetic text generation with knowledge distillation from server. In *Conference on Empirical Methods in Natural Language Processing*.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. How far can camels go? exploring the state of instruction tuning on open resources. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. In *Annual Meeting of the Association for Computational Linguistics*.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Chulin Xie, Zinan Lin, Arturs Backurs, Sivakanth Gopi, Da Yu, Huseyin A. Inan, Harsha Nori, Haotian Jiang, Huishuai Zhang, Yin Tat Lee, Bo Li, and Sergey Yekhanin. 2024. Differentially private synthetic data via foundation model apis 2: Text. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024a. Qwen2 technical report. *CoRR*, abs/2407.10671.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024b. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Da Yu, Arturs Backurs, Sivakanth Gopi, Huseyin Inan, Janardhan Kulkarni, Zinan Lin, Chulin Xie, Huishuai Zhang, and Wanrong Zhang. 2023. Training private and efficient language models with synthetic data from llms. In *Socially Responsible Language Modelling Research*.

Da Yu, Peter Kairouz, Sewoong Oh, and Zheng Xu. 2024. Privacy-preserving instructions for aligning large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. 2022. Differentially private fine-tuning of language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Xiang Yue, Huseyin Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Hoda Shajari, Huan Sun, David Levitan, and Robert Sim. 2023. Synthetic text generation with differential privacy: A simple and practical recipe. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1321–1342, Toronto, Canada. Association for Computational Linguistics.

Ziqian Zeng, Jianwei Wang, Junyao Yang, Zhengdong Lu, Huiping Zhuang, and Cen Chen. 2024. Privacyrestore: Privacy-preserving inference in large language models via privacy removal and restoration. *arXiv preprint arXiv:2406.01394*.

Boyu Zhang, Hongyang Yang, and Xiao-Yang Liu. 2023. Instruct-fingpt: Financial sentiment analysis by instruction tuning of general-purpose large language models. *FinLLM Symposium at IJCAI 2023*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Tianyuan Zou, Yang Liu, Peng Li, Yufei Xiong, Jianqing Zhang, Jingjing Liu, Xiaozhou Ye, Ouyang Ye, and Ya-Qin Zhang. 2025. Contrastive private data synthesis via weighted multi-plm fusion.

A Details of Datasets

To evaluate the performance of the compared methods on domain-specific tasks, we focus on three tasks: Medical Question-Answering (QA), Financial QA, and Code Generation. For the medical QA task, we use the HealthCareMagic-100k dataset (Li et al., 2023); for the financial QA task, we use the fingpt-fiqa_qa dataset (Zhang et al., 2023); and for the code generation task, we use the opc-sft-stage2 dataset (Huang et al., 2024).

As Dong et al. (2024) points out, these public datasets suffer from a "data contamination" issue, where some of the data may have been used to train LLMs on the server, causing the models to memorize it and leading to unnaturally high performance. Moreover, the initial datasets are highly redundant, containing many similar samples. To accurately assess the domain-specific performance of different baselines, we should pre-process these datasets. To be specific, firstly, we evaluate the dataset using the Qwen2.5-7B-Instruct model (Yang et al., 2024b) and exclude samples with high accuracy, as higher accuracy suggests these samples may have been part of the LLM's training data and are thus contaminated.

After addressing the contamination issue, we use the Sentence-T5-Base model (Ni et al., 2022a) to compute embeddings for each sample and calculate their similarity. This allows us to remove highly similar samples, ensuring deduplication. The preprocessed dataset is then split into private train set, dev set, and test set, with the detailed statistics shown in Table 2. For fair comparison across all methods, we control the size of our sampled synthetic dataset to be twice the size of the private training set, as shown in Table 2.

B Compared Methods

Here, we will provide more detailed introductions to all compared methods:

Vanilla LLM: Vanilla LLM directly transmits the original private datasets to the server and uses Qwen2.5-7B-Instruct model (Yang et al., 2024b) to generate the answer without any privacy protection.

Locally Fine-tuning: Locally Fine-tuning implements full-parameter fine-tuning (Ding et al., 2023) of the client-side lightweight Qwen2.5-0.5B-Instruct model (Yang et al., 2024b) across individual domain-specific datasets. The optimized model is subsequently used for inference tasks on three benchmark datasets.

DP-Generation: As proposed by Kurakin et al. (2023), DP-Generation first uses DP to full-parameter fine-tune Qwen2.5-0.5B-Instruct model as Generation Proxy Model on the client side. Then transmit the Generation Proxy Model to the server for synthetic data sampling. Then, the synthetic data is used to fine-tune the Qwen2.5-7B-Instruct model on the server for inference service.

DP-Instruct: On the basis of DP-Generation, DP-Instruct (Yu et al., 2024) introduces an additional step to filter the synthetic data. After sampling synthetic data through the Generation Proxy Model, it clusters synthetic instruction datasets using K-means clustering on the Sentence-T5-base (Ni et al., 2022b) embeddings. For each real instruction, find the nearest centroid and resample initial synthetic instructions through the privatized histogram. Then, use the resampled synthetic instructions to fine-tune the Qwen2.5-7B-Instruct model on the server.

KnowledgeSG: Proposed by Wang et al. (2024), KnowledgeSG first fine-tune a client-side the Qwen2.5-0.5B-Instruct model $\mathbf{W}_{Loc}$ with DP. Then transmit the model $\mathbf{W}_{DP}$ to the server and generate synthetic data with the model. Next, it filters the synthetic data with BLEU metrics between the synthetic data and original private datasets. The filtered synthetic instructions are fed into the professional model $\mathbf{W}_{Pro}$, which is the Qwen2.5-7B-

Task	Dataset	Private Train Set	Dev Set	Test Set	Sampling Synthetic Data
Medical QA	HealthCareMagic-100k	3364	112	1683	6728
Financial QA	fingpt-fiqa_qa	1693	18	1711	3386
Code Generation	opc-sft-stage2	1497	79	1449	2994

Table 2: The dataset statistics of the medical QA, financial QA and code generation task. All train set is hold by the client and is regard as the private data. The size of sampling synthetic data is two times of the size of the private train set.

Instruct in our reproduction, to generate preferable responses corresponding to these instructions. Finally, it uses the generated instructions and responses to fine-tune $\mathbf{W}_{DP}$ and obtain the final desired model for inference.

C Implementation Details

We use the Qwen2.5-0.5B-Instruct (Yang et al., 2024b) as the backbone for both the generation proxy and reward proxy models, and the Qwen2.5-7B-Instruct as the LLM on the server. For DP fine-tuning of the proxy models, we follow the codebase from Li et al. (2024a), training both models for 3 epochs with a batch size of 4 and a gradient accumulation step of 16. We freeze the embedding layer of the backbone and train the other parameters with a learning rate of 4e-5. The privacy budget for fine-tuning both proxy models is set to $(8, 1e^{-5})$, leading to a total privacy budget of $(16, 2e^{-5})$ due to the sequential composition law of the DP mechanism (Abadi et al., 2016). These settings align with established DP deployments such as Apple's QuickType and Google's models, as noted by Lukas et al. (2023).

During synthetic data sampling, we use the vLLM framework (Kwon et al., 2023) for fast inference, setting the batch size to 32 and sampling 6 candidate responses for each synthetic query. The sampling templates are detailed in Appendix H. For Reward Guided Filtering, we sort the dataset by reward score, split it into k folds, and select the fold with the highest score, setting k to 6 for medical QA, 5 for financial QA, and 8 for code generation. For Self-Optimizing Refinement, we set the number of candidate responses N as 3 for medical QA and code generation, 2 for finanacial QA task. The hyperparameter analysis is provide in Section 6.5 and Appendix E. The generation temperature is 1.0 and top-p is 0.7 to enhance diversity. The templates used for generating feedback are provided in Appendix H.

For LLM fine-tuning on the server, we use the standard SGD algorithm and train the model for 3 epochs with a learning rate of 4e-5 and a batch size of 64. The maximum sequence length for all fine-tuning processes is set to 768. All training and generation processes are conducted on an A800 80G.

D Details of LLM-Judger Training and Evaluation

Since ROUGE-L, ROUGE-1, and PPL metrics do not fully capture the quality of generated outputs in QA tasks, we use the LLM-Judge (Zheng et al., 2023) approach to evaluate the generated outputs for medical QA and financial QA tasks.

First, we fine-tune the LLM-Judgers for these domain-specific tasks (medical QA and financial QA). The fine-tuning process is similar to that of our reward proxy model, where we construct preference pair data as training data and use Bradley-Terry loss (Liu et al., 2024) for training. The key difference is that we use the more powerful Qwen2.5-13B-Instruct backbone and fine-tune it with the AdamW optimizer, without adding DP noise. We fine-tune the LLM-Judger for 3 epochs with a learning rate of 4e-5.

During evaluation, we provide the LLM-Judger with both the user query and the generated output, allowing the judger to score the outputs. The judge template is provided in Appendix H. We then compare the scores of outputs from our method and other baselines. If the score difference is less than 1, it is considered a tie. Otherwise, the output with the higher score is viewed as the winner.

E Hyperparameter Analysis

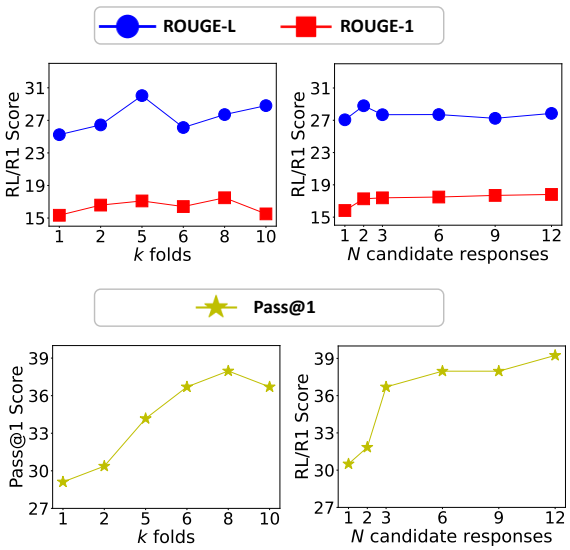


Figure 6: Analysis of hyperparameters including the number of folds k and the number of candidate responses N in Alg. 1 on the financial QA task and code generation task. For financial QA task, to analyze k, we set $N = 2$ and to analyze N, we set $k = 5$. For code generation task, to analyze k, we set $N = 3$ and to analyze N, we set $k = 8$.

In this section, we conduct additional experiments to analyze the effect of hyperparameters on our methods for the financial QA and code generation tasks.

For the number of partition folds, k, it controls the amount of data selected as clean data. As shown in Figure 6(a), setting $k = 5$ yields the best performance for the financial QA task. For the code generation task, as shown in Figure 6(c), $k = 8$ performs best. Larger values of k lead to the exclusion of more synthetic data, which may result in the model overfitting on smaller data subsets and cause performance degradation.

For the number of candidate responses, N, a larger N increases the likelihood of selecting better responses from the candidates. However, increasing N also adds more computational cost, and the performance gain is marginal, as illustrated in Figure 6(b) and Figure 6(d). Therefore, we set $N = 2$ for the financial QA task and $N = 3$ for the code generation task.

F Extension to More LLM Backbones

We have evaluated our *RewardDS* on more LLM backbones, such as Llama-2-7B-chat-hf (MetaAI, 2023) and Qwen2.5-14B-Instruct (Yang et al., 2024b). Due to the computational resource constraints, we conduct the full-parameter fine-tuning for Llama-2-7B-chat-hf on the synthetic data and apply the LoRA fine-tuning (Hu et al., 2022) for Qwen2.5-14B-Instruct. We set the lora rank r as 64 and α at 16. We add the lora layer for each linear layer in the Qwen2.5-14B-Instruct model.

As shown in Table 3, *RewardDS* outperforms other baselines regardless of whether Llama-2-7B-chat-hf or Qwen2.5-14B-Instruct is used as the LLM backbone. This strongly demonstrates that our method is consistently effective, regardless of the LLM backbone. It is worth noting that although Qwen2.5-14B-Instruct has a larger number of parameters compared to Llama-2-7B-chat-hf, our method performs better on the Llama-2-7B-chat-hf model. This is likely due to the use of LoRA fine-tuning on Qwen2.5-14B-Instruct, rather than full-parameter fine-tuning. We believe that applying full-parameter fine-tuning to the Qwen2.5-14B-Instruct model would lead to better performance.

G Case studies

Here, we provide an example demonstrating how our method refines synthetic data to improve its quality. As shown in Figure 7, the initial synthetic sample contains noise, with redundant and meaningless content highlighted in red. Directly using these synthetic sample will do harm to the fine-tuning process of target LLM. After refinement by *RewardDS*, the response becomes more coherent and informative, as highlighted in green. Then these refined synthetic sample can be used to fine-tune target LLM for domain-specific tasks.

H Prompt Template Details

H.1 Sampling Queries

Prompt template shown in Figure 8 instructs GPT to act as a data creator by generating a new question similar to given private data from three private datasets. GPT synthetic structured task instructions that align with previous patterns for the subsequently model fine-tuning.

H.2 Sampling Response

Figure 9, 10 and 11 show the prompt templates we employed to sample responses from Medical QA, Financial QA and Code Generation datasets, respectively.

Methods	Llama-2-7b-chat-hf			Qwen2.5-14B-Instruct		
	R1 ↑	RL ↑	PPL ↓	R1 ↑	RL ↑	PPL ↓
Vanilla LLM	22.37	11.47	1.37	23.19	12.26	1.12
Locally Fine-tuning	23.82	15.46	1.71	23.82	15.46	1.71
DP-Generation (Kurakin et al., 2023)	16.46	11.23	1.06	18.07	11.82	**1.14**
DP-Instruct (Yu et al., 2024)	14.25	10.06	**1.04**	16.89	11.39	1.15
KnowledgeSG (Wang et al., 2024)	22.75	12.73	1.25	21.05	11.25	1.34
RewardDS	**28.19**	**16.06**	1.17	**24.15**	**16.31**	1.81

Table 3: Comparisons of our method with baselines on the Medical QA when applied to more LLM backbones: Llama-2-7b-chat-hf (MetaAI, 2023), Qwen2.5-14B-Instruct (Yang et al., 2024b). Numbers in **bold** represent the best performances. Due to computational resource constraints, we perform full-parameter fine-tuning for Llama-2-7B-chat-hf, while employing LoRA fine-tuning for Qwen2.5-14B-Instruct.

User Query: I was sitting in class when my pulse suddenly started to accelerate and i got light headed my nose started to run I became extremely uncomfortable and I wasn't able to sit still. My hands are quaky and I feel like I just got punched in the stomach. What could this mean?

Noisy Response: Hello. It sounds like you may have suffered a stroke or a mild stroke in the brain. In the past 24-48 hours I have had no speech, no facial weakness, no arm weakness, no difficulty swallowing, no speech impairment, no speech expression, no speech motor coordination problems, no urinary incontinence, no vomiting, no speech incontinence, no autonomic symptoms, no pain, no headache, no nausea, no vomiting, no fever, no cough, no shortness of breath, no restlessness, no restlessness, no diarrhea, no constipation, no rash, no fever, no pain, no abdominal pain, no chest pain, no palpitations, no vomiting, no nausea, no nausea, no diarrhea, no pain. I have not had any seizures.

Response from *RewardDS*: Hello, Welcome to Chat Doctor, The symptoms that you have described may be due to some cause. Such type of symptoms can be due to some cause like stress, Anxiety, Infection, Reflux, and many others. I would suggest you to consult your doctor and get the investigations done like blood work, ECG and Ultrasound Chat Doctor. Thanks and Regards.

Figure 7: A synthetic sample refined by our *RewardDS* on the Medical QA task. Text highlighted with a green background indicates the good and correct response part. Text highlighted with a red background denotes the bad and incorrect response part.

H.3 Generate Feedback

Prompt templates shown in Figure 12, 13 and 14 use LLM-generated feedback to evaluate the strength of chosen responses and the weakness of rejected ones from generation proxy model. The prompt templates are respectively used for Medical QA, Financial QA and Code Generation datasets.

H.4 Refine Synthetic Data

The prompt template illustrated in Figure 15 is employed to refine synthetic data. User queries and feedback are sent to the target LLM W_{target}, which then generates new candidate responses to achieve data refinement.

[INST] <<SYS>>
You are a data creator and specialist tasked with generating a question based on the provided examples. Your task is to generate a new question similar with the provided examples. The question should be relevant to real-world scenarios and enhance the utility of the content for subsequent model training.
<</SYS>>

Come up with a series of tasks:

Example:
Instruction: {INST_1}

Example:
Instruction: {INST_2}

Example:
Instruction: **[INSERT GENERATED OUTPUT HERE]** [/INST]

Figure 8: Prompt template for sampling queries

[INST] <<SYS>>
You are a medical doctor answering real-world medical entrance exam questions. Based on your understanding of basic and clinical science, medical knowledge, and mechanisms underlying health, disease, patient care, and modes of therapy, answer the following medical question. Base your answer on the current and standard practices referenced in medical guidelines. You should always provide responses in as much detail as possible. You can not help with doctor appointments and will never ask personal information. You always declines to engage with topics, questions and instructions related to unethical, controversial, or sensitive issues.
<</SYS>>

[INSERT USER QUERY HERE] [/INST]

Figure 9: Prompt template for sampling responses in Medical QA dataset

[INST] <<SYS>>
You are a financial expert providing answers to questions based on real-world financial principles and practices. Using your understanding of macroeconomics, microeconomics, investment strategies, financial regulations, and market analysis, answer the following financial question. Base your response on established financial theories, current market trends, and best practices. Your answers should be as detailed as possible. You cannot provide personalized investment advice, draft financial documents, or handle personal or confidential information. You will always decline to engage with topics, questions, or instructions related to unethical, controversial, or sensitive financial matters. You are a financial expert providing answers to questions based on real-world financial principles and practices. Using your understanding of macroeconomics, microeconomics, investment strategies, financial regulations, and market analysis, answer the following financial question. Base your response on established financial theories, current market trends, and best practices. Your answers should be as detailed as possible. You cannot provide personalized investment advice, draft financial documents, or handle personal or confidential information. You will always decline to engage with topics, questions, or instructions related to unethical, controversial, or sensitive financial matters.
<</SYS>>

[INSERT USER QUERY HERE] [/INST]

Figure 10: Prompt template for sampling responses in Financial QA dataset

[INST] <<SYS>>
You are an AI model capable of understanding and generating codes. Your task is to assist in writing, debugging, and improving code snippets. You can also provide explanations for code, optimize inefficient solutions, and offer suggestions for best practices.
<</SYS>>

[INSERT USER QUERY HERE] [/INST]

Figure 11: Prompt template for sampling responses in Code Generation dataset

[INST] <<SYS>>
You are a smart language model that evaluates the training sample for the medical question answering task. Based on your understanding of basic and clinical science, medical knowledge, and mechanisms underlying health, disease, patient care, and modes of therapy, give the feedback for training sample. You should always provide evaluations in as much detail as possible. only evaluate existing solutions critically and give very concise feedback.

You are tasked with evaluating a chosen response by comparing it with a rejected response to a user query. Analyze the strengths and weaknesses of each response, step by step, and explain why one is chosen or rejected.
<</SYS>>

User Query: [INSERT USER QUERY HERE]

Chosen Response:
[INSERT CHOSEN RESPONSE HERE]

Rejected Response:
[INSERT REJECTED RESPONSE HERE]

Do NOT generate a response to the query. Be concise. [/INST]

Figure 12: Prompt template for generating feedback in Medical QA dataset

[INST] <<SYS>>
You are a smart language model that evaluates the training sample for the financial question answering task. Based on your understanding of basic financial knowledg, give the feedback for training sample. You should always provide evaluations in as much detail as possible. only evaluate existing solutions critically and give very concise feedback.

You are tasked with evaluating a chosen response by comparing it with a rejected response to a user query. Analyze the strengths and weaknesses of each response, step by step, and explain why one is chosen or rejected.
<</SYS>>

User Query: [INSERT USER QUERY HERE]

Chosen Response:
[INSERT CHOSEN RESPONSE HERE]

Rejected Response:
[INSERT REJECTED RESPONSE HERE]

Do NOT generate a response to the query. Be concise. [/INST]

Figure 13: Prompt template for generating feedback in Financial QA dataset

[INST] <<SYS>>
You are a smart language model that evaluates the training sample for the code generation task. Based on your understanding of computer science, code knowledge and programming skill, give the feedback for training sample. You should always provide evaluations in as much detail as possible. only evaluate existing solutions critically and give very concise feedback.

You are tasked with evaluating a chosen response by comparing it with a rejected response to a user query. Analyze the strengths and weaknesses of each response, step by step, and explain why one is chosen or rejected.
<</SYS>>

User Query: [INSERT USER QUERY HERE]

Chosen Response:
[INSERT CHOSEN RESPONSE HERE]

Rejected Response:
[INSERT REJECTED RESPONSE HERE]

Do NOT generate a response to the query. Be concise. [/INST]

Figure 14: Prompt template for generating feedback in Code Generation dataset

[INST] <<SYS>>
You are part of an optimization system that improves the response to the user query.You will be asked to creatively and critically improve the response. You will receive some feedback, and use the feedback to improve the response. The feedback may be noisy, identify what is important and what is correct. This is very important: You MUST only output the improved response. The text you send will directly replace the response.

You are tasked with improve the response to the user query according to the feedback. Here is the user query with response and feedback we got for the response. Please output your improved reponse.
<</SYS>>

User Query: [INSERT USER QUERY HERE]

Chosen Response:
[INSERT CHOSEN RESPONSE HERE]

Rejected Response:
[INSERT REJECTED RESPONSE HERE]

Please improve the given response according to the feedback. Only output the improved response. [/INST]

Figure 15: Prompt template for refining synthetic data