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 generate 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

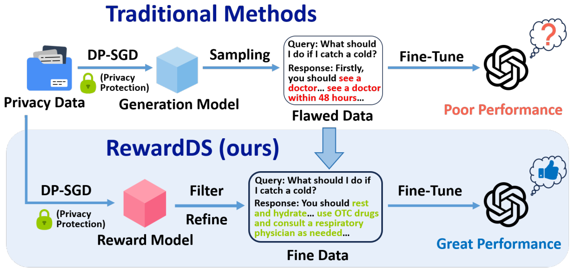


Figure 1: Illustration of how *RewardDS* overcomes the dilemma of traditional synthetic data methods. The synthetic data directly sampled from the generation proxy model contain significant flaws, such as incoherent text or incomplete storylines, which are considered noise.

service provider can lead to significant privacy concerns (Zeng et al., 2024; Abdelnabi et al., 2023). We denote the individuals or organizations which aim to customize LLMs as **client**, the LLM service providers as **server**, and the model being fine-tuned as the **target LLM**. It remains a critical challenge to develop privacy-preserving fine-tuning methods in such a client-server scenario.

Prior works proposed data synthesis as a promising solution (Kurakin et al., 2023; Yue et al., 2023; Yu et al., 2023; Mattern et al., 2022; Flemings and Annavaram, 2024). These approaches generate synthetic data to replace the private data and can be 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 generation process, the synthetic data inevitably contain significant flawed one, including text incoherence or storyline incompleteness, which is considered as noise and leads to less effective LLM fine-tuning, as illustrated in Figure 1.

To mitigate the noise, existing methods (Yu et al., 2024; Wang et al., 2022; 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 (Wang et al., 2024; Li et al., 2024b) attempt to distill the capabilities from the LLM on the server side into the generation proxy model to support domain-specific tasks. However, since the LLM is not fine-tuned for these specific domains, the distillation provides limited benefit and does not effectively improve task performance. We also present an illustrative example in Figure 7.

To more effectively mitigate noise in synthetic data, 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 are 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

In this section, we will introduce the related work on LLM privacy-preserving fine-tuning methods, which are currently divided into three categories: Anonymity-based methods, Encryption-based methods and Synthesis-based methods.

Anonymity-based methods. Techniques like k-anonymity and adversarial anonymization can identify and anonymize private data. But they will significantly harm data quality for domain-specific LLM fine-tuning on the server side. (Staab et al., 2024; Sweeney, 1997; Romanov et al., 2019)

Encryption-based methods. Some approaches employ encryption techniques, such as Homomorphic Encryption (HE) or Secure Multi-Party Computation (SMPC), to protect private data. However, encrypting data and maintaining secure communication between server and client incur significant computational and time overhead, making these methods impractical in real-world scenarios. (Frery et al., 2025; Lou et al., 2020; You et al., 2025)

Synthesis-based methods. Recent studies have explored using synthetic data with differential privacy (DP) guarantees as a substitute for private data in LLM fine-tuning. While this offers a practical and efficient solution, the synthetic data often contain noisy or flawed samples that significantly hinder LLM fine-tuning. Simple filtering based on text similarity is insufficient to effectively eliminate such noise. (Kurakin et al., 2023; Yue et al., 2023; Yu et al., 2024; Hou et al., 2024; Wang et al., 2024, 2022)

Due to the limited space, a detailed introduction of the above works can be found in Appendix A.

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 some lightweight LLMs within their computational constraints, these models have inherently weaker capabilities than the target LLM. 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.

Existing synthesis-based methods utilize a lightweight **Generation Proxy Model** 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 generation process introduces significant 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. 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 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

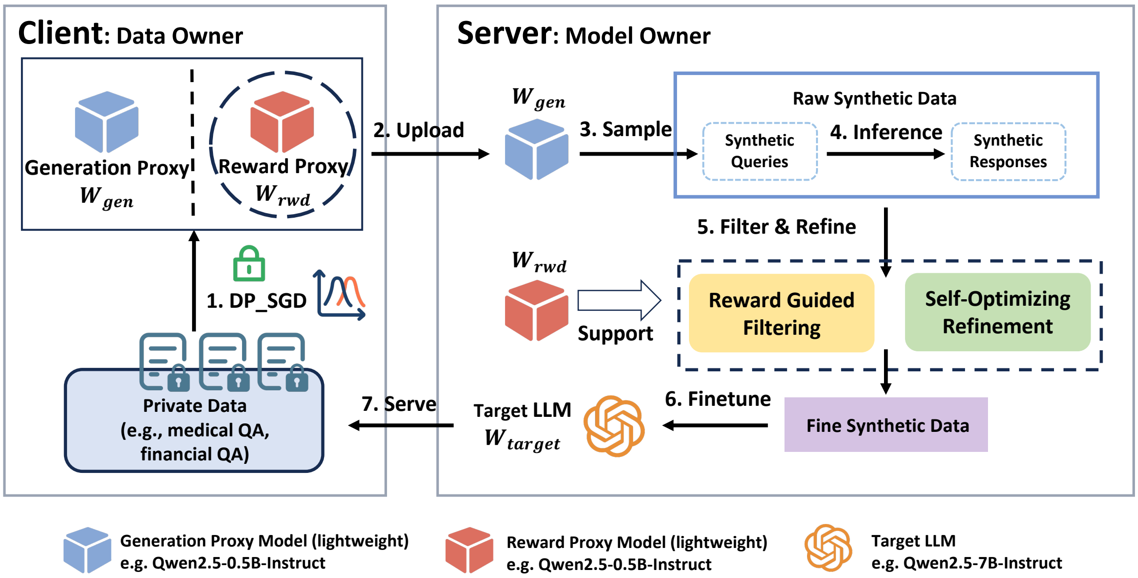


Figure 2: The overview of our *RewardDS* framework. The client uses DP-SGD to fine-tune two lightweight proxy models on privacy-sensitive data: the Generation Proxy Model W_{gen} and the Reward Proxy Model W_{rwd}. Both proxy models are then sent to the server. The Generation Proxy Model is used to sample 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.

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 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 J. After the LLM is fine-tuned on the refined synthetic data, it can provide service to the client for those domain-specific tasks.

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 ∈ Q_query do
    Generate candidate response set: {A_j}_{j=1}^N ← W_gen(q)
    Predict the reward score: {s_j}_{j=1}^N ← W_rwd(q, A_j)
    Select the best and the worst response:
        (A_c, A_r) ← (A_{arg max_j s_j}, A_{arg min_j s_j})
    Record the best reward score: s_c ← max_j s_j
Gather the initial synthetic dataset: D_0 ← {(q_i, A_c^i, A_r^i, s_c^i)}_{i=1}^L
Sort D_0 by reward: D_0^sorted ← {(q_i, A_c^i, A_r^i, s_c^i)}_{i=1}^L, where
    s_c^1 ≥ ··· ≥ s_c^L
Partition D_0^sorted into k folds: {D_0^m}_{m=1}^k ← split(D_0^sorted, k)
Extract top-⌊L/k⌋ samples: D_high ← D_0^1
Replicate subset to obtain the train set: T_0 ← ⊕_{⌈L/|D_high|⌉} D_high
Determine score threshold τ: τ ← min_{s_c ∈ D_high} s_c
// During Fine-tuning LLM
Initialize target LLM: W_target^0 ← W_target
for iteration t = 1 to T do
    Fine-tune target LLM W_target^{t-1} on {(q, A_c) ∈ T_{t-1}} and get
        W_target^t
    for each query q ∈ Q_query do
        Generate feedback φ: φ ← W_gen(q, A_c, A_r)
        Re-generate the candidate response set:
            {A_j}_{j=1}^N ← W_target^t(q, φ)
        Predict the reward score, select the best and worst responses, and record the highest reward score to update
            D_{t-1}, yielding D_t.
    Filter and get new training set T_t:
        T_t ← {(q, A_c, A_r, s_c) ∈ D_t | s_c ≥ τ}
return M_target^t
```

5 Privacy Analysis

The only transmitted contents 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. Moreover, we conduct Data Extraction Attack (Carlini et al., 2021) and Membership Inference Attack (Yeom et al., 2018; Choquette-Choo et al., 2021) on our method to empirically demonstrate its privacy protection capability in Section 6.4.

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, we also use LLM-Judge (Zheng et al., 2023) to provide a more comprehensive assessment of semantic accuracy and response quality. 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 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 B and Appendix D, 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**

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.

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 are subsequently utilized to fine-tune target LLM on the server. **DP-Instruct** (Yu et al., 2024) additionally filters synthetic data based on text similarity before LLM fine-tuning; **KnowledgeSG** (Wang et al., 2024) distills the capacity from LLM into the generation proxy model to enhance its performance.

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

6.3 Main Results

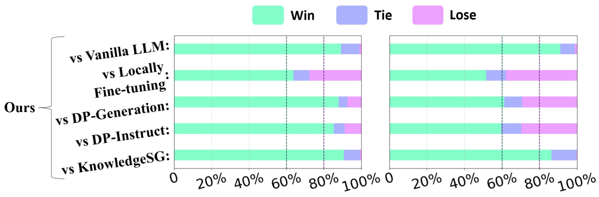
As shown in Table 1, *RewardDS* outperforms all other baselines across the three domain-specific tasks, except for the PPL on the Medical QA task. DP-Instruct achieves marginally lower PPL in medical QA. This may be attributed to the filtering strategy based on similarity, which could lead the target 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. DP-Instruct attempts to filter the synthetic data by computing the similarity between the synthetic query and the private query. But, similarity alone cannot accurately reflect the quality of synthetic data, where higher similarity does not necessarily indicate better data quality. KnowledgeSG distills the capabilities of the target LLM on the server into the generation proxy model for domain-specific tasks. However, since the target LLM is not specifically fine-tuned for these tasks, the improvement through distillation is limited.

We observe consistent performance declines across all tasks when either the Reward Guided Filtering or Self-Optimizing Refinement module is removed, highlighting the importance of both components. Without these components, more noisy synthetic samples are used during LLM fine-tuning, leading to degraded performance.

In addition, following Zheng et al. (2023), we employ an LLM-Judger to assess the generated response from our method and those baseline approaches for QA tasks. Specifically, we provide the LLM-Judger with responses from our method and those from baseline methods, prompting it to judge which response is better. As shown in Figure 3, our method consistently outperforms all baselines on both medical and financial QA tasks. More implementation details are provided in Appendix E. We also provide a case study in Figure 7 and Appendix F to further demonstrate the effectiveness of our method.



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

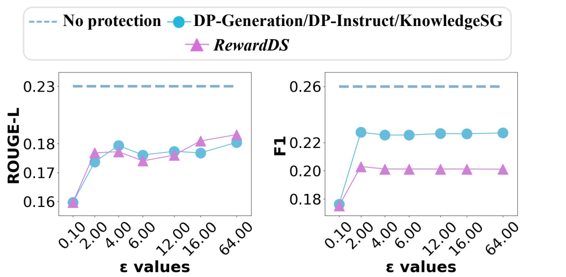
Figure 3: Using LLM-Judge to compare the outputs generated by our method with those of other baselines. **Win** means our method outperformed the baselines, **Tie** means the results were similar, and **Lose** means our method performed worse than the baselines.

6.4 Empirical Privacy Protection Results

In this section, we implement Data Extraction Attack (Carlini et al., 2021) and Membership Inference Attack (Yeom et al., 2018; Choquette-Choo et al., 2021) on our method and baselines to empirically evaluate the privacy protection capability. For those baselines, including DP-Generation/DP-Instruct/KnowledgeSG, the client only transfers the generation proxy model to the server. In contrast, *RewardDS* transfers both the generation proxy model and the reward proxy model. Accordingly, the attack targets for the baselines are limited to the generation proxy model, while for *RewardDS*, both models can be attacked. We also provide the attack results of No Protection, which serves as the upper bound.

As shown in Figure 4, *RewardDS* demonstrates superior privacy protection capacity compared to those baselines, as indicated by its comparable ROUGE-L scores and significantly lower F1 scores. This is possibly due to *RewardDS* allocating the privacy budget for both generation proxy model and reward proxy model, thereby reducing the privacy budget of each individual model and making them more difficult to be attacked.

Implementation details of these attack methods can be found in Appendix G.



(a). Data Extraction Attack (b). Membership Inference Attack

Figure 4: Results of Data Extraction Attack and Membership Inference Attack for *RewardDS* and all baselines under different privacy budgets ϵ on Medical QA task.

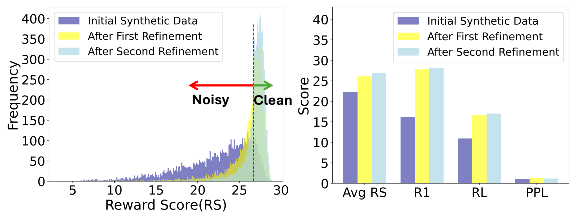
6.5 In-depth Analysis of *RewardDS* Design

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

Analysis 1: *Impact of RewardDS on Synthetic Data Quality and Downstream Performance.*

According to Alg. 1, *RewardDS* iteratively refines the synthetic data during each training epoch. As shown in Figure 5(a), the reward score of synthetic data gradually increases with iterative refinement, indicating improved data quality.

We also evaluate the downstream performance of the target LLM on the Medical QA task after being fine-tuned on the synthetic data from different refinement stages. The results in Figure 5(b) show that the downstream performance of the fine-tuned LLM also improves significantly with the improvement of the synthetic data quality, strongly highlighting the effectiveness of *RewardDS*.



(a). Reward Score Change on Medical QA Task

(b). Downstream Performance Change on Medical QA Task

Figure 5: Changes of Reward Score and Downstream Performance in *RewardDS* for the Medical QA Task. **Avg RS** indicates the average reward score of the synthetic data, while **R1**, **RL**, and **PPL** represent the downstream performance metrics of the target LLM fine-tuned using these synthetic data.

Analysis 2: *Training Cost Analysis of RewardDS.*

As shown in Figure 2, *RewardDS* introduces additional modules, including Reward Proxy Model Training, Reward-Guided Filtering, and Self-Optimizing Refinement, into the privacy-preserving fine-tuning process of the server-side LLM. We measure the additional time cost of these modules and compare it with that of the original modules: Generation Proxy Model Training and LLM fine-tuning. As shown in Table 2, the additional time cost from our method accounts for **only 29.69%** of the total time cost, with most of the time consumed by LLM fine-tuning modules. This is primarily due to the use of a lightweight reward proxy model, making the associated modules highly efficient. Overall, **the additional time cost introduced by *RewardDS* is minimal**, strongly demonstrating the practicality of our method.

	Time (min)	Percentage
Initial Modules:	315	70.31%
Generation Proxy Model Training	45	10.04%
LLM fine-tuning	**270**	**60.26%**
Additional Modules From *RewardDS*:	133	29.69%
Reward Proxy Model Training	49	10.94%
Reward Guided Filtering	12	2.67%
Self-Optimizing Refinement	72	16.07%

Table 2: Runtime of different modules in *RewardDS* for privacy-preserving fine-tuning on medical QA task. The most time-consuming module is marked in **bold**.

Analysis 3: *RewardDS vs Filtering according to Reward Score.*

According to Figure 5(a), the initial synthetic data contains substantial samples with low reward scores. One straightforward strategy is to filter out these low-quality samples to reduce noise. As shown in Table 3, simply selecting Top 50% synthetic data for fine-tuning can improve the overall data quality and slightly enhance downstream performance. However, filtering also reduces the size of the training set. As more data is discarded, the downstream performance begins to drop due to the limited training data.

In contrast, *RewardDS* applies Self-Optimizing Refinement to improve the quality of low reward samples instead of discarding them. **It can significantly enhance data quality while maintaining a stable dataset size**. Consequently, *RewardDS* achieves superior downstream performance, as demonstrated in Table 3.

	Data Count ↑	Avg RS ↑	Downstream RL ↑
Raw Synthetic Data	**6420**	22.30	10.94
– Select Top 80%	<u>5136</u>	24.05	11.09
– Select Top 50%	3210	25.66	<u>12.43</u>
– Select Top 30%	1926	26.53	11.19
– Select Top 10%	642	**27.43**	6.82
Refinement by *RewardDS*	**6420**	<u>26.82</u>	**17.02**

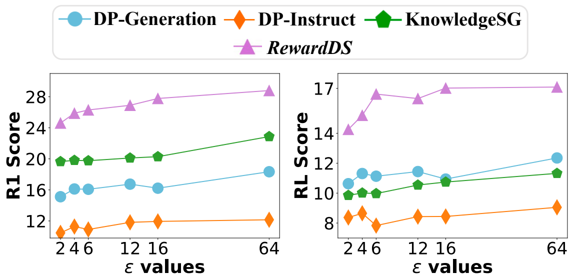
Table 3: Comparison between filtering synthetic data only based on Reward Score (RS) and iterative refinement using *RewardDS* on Medical QA task. Higher **Avg RS** indicates better overall data quality. The best results are shown in **bold**, and the second-best results are <u>underlined</u>.

Other Analysis: Furthermore, we evaluate the performance of *RewardDS* with **alternative LLM backbones** (Llama-2-7B-chat-hf and Qwen2.5-14B-Instruct) and analyze **the impact of different privacy budget allocations** in Appendix H. The results in Table 5 and Figure 8 also clearly demonstrate the effectiveness and robustness of our method.

6.6 Hyperparameter Analysis

In this section, we conduct hyperparameter analysis to further prove the effectiveness of our method.

Privacy Budget ϵ. We evaluate the performance of our method and baseline approaches under varying privacy budgets ϵ. As shown in Figure 6, our method consistently outperforms all baselines, not only under the commonly used setting of $\epsilon = 16$, but also under stricter privacy budgets (e.g., $\epsilon = 2$, 4, and 6). These results demonstrate the superior performance and broad applicability of our method.



(a). ROUGE-1 Score (b). ROUGE-L Score

Figure 6: Performance of *RewardDS* and baselines on the Medical QA task under different privacy budgets ϵ, evaluated using ROUGE-1 (**R1**) and ROUGE-L (**RL**) scores.

More Hyperparameters. We also analyze more hyperparameters in our method described in Alg. 1, including the number of folds k and the number of candidate responses N. As shown in Figure 9, 10 and 11, our method remains effective and robust across different hyperparameter settings on three domain-specific tasks. More detailed analyses can be found in Appendix I.

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

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 H.1. Full-parameter fine-tuning may yield even better performance.

Future work will investigate larger LLM backbones to further validate the effectiveness of our method on models with greater parameter scales.

Moreover, although our method is already time-efficient, we plan to further improve efficiency by exploring lightweight training approaches, such as Low-Rank Adaptation (LoRA) and prefix tuning, during the fine-tuning of both the generation proxy model and the reward proxy model.

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*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650.

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*.

Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In *International conference on machine learning*, pages 1964–1974. PMLR.

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.

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.

Jordan Frery, Roman Bredehoft, Jakub Klemsa, Arthur Meyre, and Andrei Stoian. 2025. Private lora finetuning of open-source llms with homomorphic encryption. *arXiv preprint arXiv:2505.07329*.

Charlie Hou, Akshat Shrivastava, Hongyuan Zhan, Rylan Conway, Trang Le, Adithya Sagar, Giulia Fanti, and Daniel Lazar. 2024. Pre-text: Training language models on private federated data in the age of llms. *arXiv preprint arXiv:2406.02958*.

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.

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.

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*.

Qian Lou, Bo Feng, Geoffrey Charles Fox, and Lei Jiang. 2020. Glyph: Fast and accurately training deep neural networks on encrypted data. *Advances in neural information processing systems*, 33:9193–9202.

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.

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

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2022. 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.

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, 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*.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE.

Zhichao You, Xuewen Dong, Ke Cheng, Xutong Mu, Jiaxuan Fu, Shiyang Ma, Qiang Qu, and Yulong Shen. 2025. Prifft: Privacy-preserving federated fine-tuning of large language models via function secret sharing. *arXiv preprint arXiv:2503.03146*.

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.

Appendix Overview

The appendix is organized into two parts: Appendix A–D provide related work and the main experimental setup of *RewardDS*, while Appendix J–X present additional experimental results to further demonstrate the effectiveness of *RewardDS*.

A Related Work

In this section, we provide a more detailed introduction to the related works on LLM privacy-preserving fine-tuning methods, including Anonymity-based methods, Encryption-based methods and Synthesis-based methods.

A.1 Anonymity-based methods

Anonymity-based methods aim to identify and remove user-specific private information from private data to enable privacy-preserving LLM fine-tuning. Sweeney (1997) achieves k-anonymity by dynamically identifying user-specific private information and applying substitution or removal to protect it. Romanov et al. (2019) employs a transformer framework with attention mechanisms to enhance anonymization performance. Staab et al. (2024) proposes an adversarial anonymization approach that leverages one LLM to anonymize user privacy while using another LLM to detect privacy information, iteratively improving the anonymization performance.

All of the above anonymity-based methods require detecting and removing user privacy, which will make the data incoherent and incomplete, thereby reducing its quality for downstream LLM fine-tuning.

A.2 Encryption-based methods

Encryption-based methods focus on applying encryption to the private data and maintaining secure communication between client and server to transmit the private data. Lou et al. (2020) applies fully Homomorphic encryption to protect private data, enabling privacy security while maintaining comparable model performance after fine-tuning on the encrypted data. Frery et al. (2025) combines the Low-Rank Adaptation technique and homomorphic encryption to improve the efficiency of LLM privacy-preserving fine-tuning. You et al. (2025) introduces the hybrid secret sharing algorithm by combining arithmetic secret sharing (ASS) and function secret sharing (FSS) to achieve secure computation during LLM privacy-preserving fine-tuning.

However, current encryption-based methods still require numerous time and resources for encrypting private data and ensuring secure communication, making them impractical for real-world applications.

A.3 Synthesis-based methods

Synthesis-based methods have recently emerged as a more practical and reliable approach, which leverages synthetic data as a substitute for private data in LLM fine-tuning to balance utility and privacy.

Kurakin et al. (2023); Yue et al. (2023) propose using DP-SGD to locally fine-tune a lightweight model on the client as a generation proxy. This proxy model is used to generate synthetic data without privacy risks. The server utilizes the synthetic data to fine-tune the LLM, achieving privacy-preserving training. Considering that those synthetic data often contain numerous incoherent and flawed samples, Yu et al. (2024); Hou et al. (2024) filter out low-quality data by measuring similarity between synthetic and private data. Alternatively, Wang et al. (2024, 2022) avoid sampling synthetic data from the generation proxy model, instead using LLM on the server to improve the proxy model by distillation.

Nevertheless, only text similarity is too surface-level to accurately assess the quality of synthetic data for domain-specific tasks. Moreover, since the server-side LLM is not fine-tuned for domain-specific tasks, its ability to enhance the generation proxy model through distillation is limited.

B 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., 2022) to compute embeddings for each sample and calculate their similarity. This allows us to remove highly similar samples, ensuring deduplication. The pre-processed dataset is then split into private train set, dev set, and test set, with the detailed statistics shown in Table 4. 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 4.

C Compared Methods

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

Vanilla LLM: Vanilla LLM directly uses the LLM (Qwen2.5-7B-Instruct) on the server for those domain-specific tasks.

Locally Fine-tuning: Locally Fine-tuning fine-tunes a lightweight model (Qwen2.5-0.5B-Instruct) using the private data on the client for those domain-specific tasks.

DP-Generation: As proposed by Kurakin et al. (2023), DP-Generation first uses DP to fine-tune a lightweight model (Qwen2.5-0.5B-Instruct) as the Generation Proxy Model on the client side. Then, it transmits the Generation Proxy Model to the server for synthetic data sampling. Then, the synthetic data is used to fine-tune the LLM (Qwen2.5-7B-Instruct) on the server for those domain-specific tasks.

DP-Instruct: Compared to DP-Generation, DP-Instruct (Yu et al., 2024) introduces a filtering step to improve the quality of synthetic data. After sampling synthetic data from the Generation Proxy Model, it computes the text similarity between the synthetic data and those private data. It filters out those low-similar data to improve data quality. Then the filtered synthetic data is used to fine-tune the LLM (Qwen2.5-7B-Instruct) on the server for those domain-specific tasks.

KnowledgeSG: Considering the high noise in synthetic data, KnowledgeSG Wang et al. (2024) avoids directly sampling synthetic data from the Generation Proxy Model. Instead, it distills knowledge from the LLM to enhance the Generation Proxy Model for domain-specific tasks. Specifically, KnowledgeSG first uses DP to fine-tune a lightweight model (Qwen2.5-0.5B-Instruct) as the Generation Proxy Model on the client. Then, it transmits the Generation Proxy Model to the server and generates synthetic instructions. The synthetic instructions are fed into the professional LLM (Qwen2.5-7B-Instruct) to generate high quality responses. By using the high quality responses to fine-tune the Generation Proxy Model, it can distill the capacity of LLM into the Generation Proxy Model. Finally, the Generation Proxy Model serves for those domain-specific tasks.

D 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 $(8, 1e^{-5})$,

Task	Dataset	Private Train Set	Dev Set	Test Set	Sampling Synthetic Data
Medical QA	HealthCareMagic-100k	3210	112	1683	6420
Financial QA	fingpt-fiqa_qa	1693	18	1711	3386
Code Generation	opc-sft-stage2	1497	79	1449	2994

Table 4: 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.

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). More comparisons between our method and baselines under different privacy budgets are presented in Section 6.6.

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 J. 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 financial QA task. The hyperparameter analysis is provided in Section 6.6 and Appendix I. 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 J.

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.

E Details of LLM-Judge Evaluation

Considering ROUGE-L/ROUGE-1 metrics only measure lexical similarity to references and PPL only captures fluency, they often fail to assess deeper aspects of response quality. To ensure more reliable evaluation on the generated outputs for the medical QA and financial QA tasks, we adopt the LLM-Judge approach (Zheng et al., 2023) for assessment.

First, we fine-tune the LLM-Judger 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 J. 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.

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 noise from synthetic data 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.

F Case studies

Here, we present a representative example to demonstrate the effectiveness of our method by comparing its generated response with those from baseline methods, including DP-Generation, DP-Instruct, and KnowledgeSG.

As shown in Figure 7, **DP-Generation** includes repetitive and irrelevant symptoms, such as no facial weakness and no difficulty swallowing, which are not directly related to the user's query. **DP-Instruct** avoids repeating unrelated symptoms but still offers unhelpful advice, only suggesting that the user see a doctor without providing any meaningful medical analysis. Similarly, **KnowledgeSG** offers some advice, like conducting a physical examination, but also fails to provide any professional analysis of the user's symptoms or potential underlying causes.

In contrast, ***RewardDS*** provides a more detailed analysis of the user's symptoms, offers some possible causes, and suggests feasible advice, such as scheduling a cardiologist appointment and undergoing an ECG test, strongly demonstrating its effectiveness.

User Query: I was sitting in class when my pulse suddenly started to accelerate for no apparent reason, accompanied by cold sweats. My hands are quaky and I feel like I just got punched in the stomach. What could this mean?

[DP-Generation (Kurakin et al., 2023)] No filtration
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 sudden severe headache with no known cause. If these symptoms happen frequently or last for a long time, it would be a good idea to talk to your doctor.
Disadvantages: Meaningless duplications and irrelevant symptoms.

[DP-Instruct (Yu et al., 2024)] Filtration based on similarity
Response: Hi, I'm sorry to hear about your injury. I'm not a doctor, but I can give you some general advice. 1. If you have a headache, it's important to see a doctor. 2. If you feel like you just got punched in the stomach, it's important to see a doctor. 3. If you have complications such as inflammation or nerve irritation, it's important to see a doctor.
Disadvantages: Unhelpful advice.

[KnowledgeSG (Wang et al., 2024)] Distillation from LLM to generation proxy model
Response: Given the characteristics you've described, it is crucial to have this evaluated by a healthcare professional. You should see a healthcare provider. A thorough physical examination should be conducted to assess your physiological status. Depending on the findings, the healthcare provider might recommend imaging studies to monitor your vital signs and evaluate your physiological functions.
Disadvantages: Unprofessional analyses.

[*RewardDS*]
Response : Hello. Welcome to Chat Doctor. The symptoms you described could be caused by several possible factors, such as stress, anxiety, infection, acid reflux, or other conditions. To determine the exact cause, I recommend scheduling an appointment with a cardiologist and undergoing diagnostic tests like blood work, an ECG, and an ultrasound. Thank you and best regards, Chat Doctor.
Advantages: Concise and helpful suggestions.

Figure 7: A representative example from the medical QA task illustrates the high quality of the response generated by *RewardDS*, compared to baseline methods. Text highlighted in red indicates meaningless or flawed parts of the answer, while text in green marks meaningful and helpful content. We also provide short analyses explaining the disadvantages or advantages of the generated responses.

G Details of Privacy Protection Evaluation

In this section, we provide further details about our attack methods, including the Data Extraction Attack (Carlini et al., 2021) and Membership Inference Attack (Yeom et al., 2018; Choquette-Choo et al., 2021). We implement both attacks for our method and the baseline methods and lower attack performance indicates stronger privacy protection capacity.

Data Extraction Attack. According to Carlini et al. (2021), the Data Extraction Attack aims to recover private fine-tuned data from the fine-tuned model. Specifically, the attackers provide the fine-tuned model with partial prefixes of private data and attempt to reconstruct the corresponding complete private data. We implement this attack on our method and the baselines to evaluate their privacy protection capabilities. The implementation details are as follows:

In our scenario, the private data consists of two components: the user query and corresponding answer, both of which may contain sensitive information. We apply the data extraction attack twice to recover the user query and the answer separately. For the user query, we provide the generation proxy model with the first 10 tokens of the private query and prompt it to reconstruct the complete query. Then, to extract the answer, we provide the model with the previously recovered user query and the first 10 tokens of the private answer, prompting it to generate the full private response. We use greedy decoding during generation and set the maximum output length to 256.

To evaluate the attack's performance, we utilize the ROUGE-L score between the recovered data (user query and answer) and ground true private data. A higher ROUGE-L score indicates better attack performance.

Membership Inference Attack: As proposed by Yeom et al. (2018); Choquette-Choo et al. (2021), the membership inference attack aims to determine whether a specific data point was included in the private dataset used for fine-tuning. Specifically, the attackers collect numerous mixed data, which may contain some private data, and utilize the fine-tuned model to judge which one is included in the private data. We implement this attack on both our method and the baselines to assess their privacy protection capabilities. The implementation details are as follows:

To construct the mixed dataset, we apply data augmentation techniques, such as synonym replacement and content rewriting, to the private data and generate synthetic samples that are similar in content but not identical to the original private data. **For *RewardDS***, we input the mixed data into the reward proxy model and obtain the corresponding reward scores. Samples with higher reward scores are considered more likely to be part of the private training data. **For the baseline methods** (DP-Generation, DP-Instruct, and KnowledgeSG), only the generation proxy model is transferred to the server. We then use this model to compute the Perplexity (PPL) of each sample in the mixed dataset. Samples with lower PPL values are considered as the private data.

To evaluate the effectiveness of the attack, we calculate the F1 score of private data identification. A higher F1 score indicates stronger attack performance and thus weaker privacy protection.

H More Analysis of *RewardDS* Design

H.1 Generalizability across 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 5, *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. Overall, **our method consistently achieves superior performance across various LLM backbones, which strongly demonstrates its generalizability**.

H.2 The impact of different privacy budget allocations.

As described in Section 6.1, we allocate an equal privacy budget to generation proxy model training and reward proxy model training. To explore the impact of different privacy budget allocations, we vary the privacy budget allocation while keeping the total privacy budget fixed at $(16, 2e^{-5})$.

As shown in Figure 8, our method performs consistently well across various allocations, except in the extreme case where no budget is allocated to reward model training (i.e., "16+0"). No budget for reward model means that we do not train the reward proxy model on the client for data filtering or refinement. This phenomenon demonstrates the critical role of the reward model. Notably, even allocating a small budget to the reward model (e.g., "15+1") leads to a significant performance boost over the "16+0" case, suggesting that **even a minimal privacy cost for reward model training yields substantial benefits**.

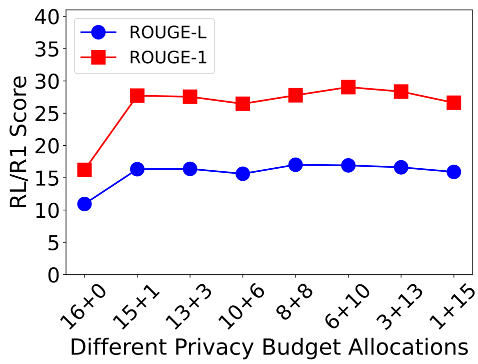


Figure 8: Performance on medical QA with different privacy budget allocations for generation proxy model and reward proxy model training. The allocation of ‘x + (16-x)’ means the privacy budget for training the generation proxy model is set to x, while the reward proxy model is set to (16-x);

I More Hyperparameter Analysis

In this section, we analyze the other hyperparameters of our method, including the number of folds k and the number of candidate responses N, for the medical QA, financial QA and code generation tasks.

As described in Alg. 1, **the number of folds** k controls how much of the synthetic data is considered clean. As shown in Figure 9, $k = 6$ yields the best performance on the medical QA task. For the financial QA and code generation tasks, the

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 5: 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.

optimal values are $k = 5$ (Figure 10) and $k = 8$ (Figure 11), respectively. Larger k values lead to stricter filtering, excluding more data, which may cause overfitting on smaller subsets and degrade performance.

As for **the number of candidate responses** N, a larger N increases the likelihood of selecting higher-quality responses but also incurs greater computational cost. As shown in Figure 9, increasing N from 1 to 3 leads to significant performance gains, while further increments yield only marginal improvements. Therefore, we set $N = 3$ for the medical QA task. For the financial QA and code generation tasks, we choose $N = 2$ (Figure 10) and $N = 3$ (Figure 11), respectively.

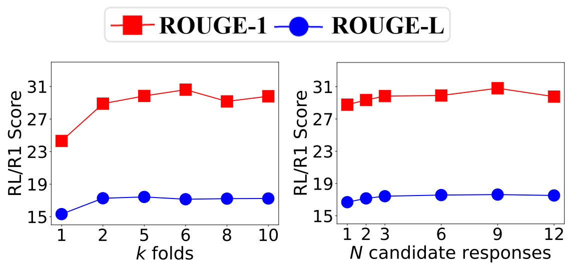


Figure 9: Performance of *RewardDS* with different numbers of folds (k) and candidate responses (N) on the dev set for the medical QA task.

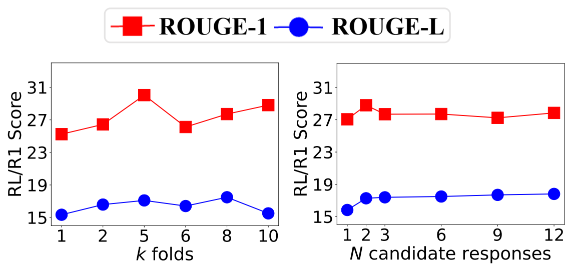


Figure 10: Performance of *RewardDS* with different numbers of folds (k) and candidate responses (N) on the dev set for the financial QA task.

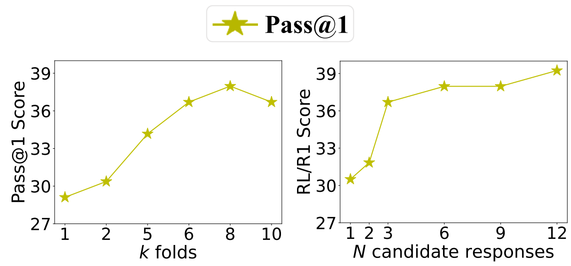


Figure 11: Performance of *RewardDS* with different numbers of folds (k) and candidate responses (N) on the dev set for the code generation task.

J Prompt Template Details

J.1 Sampling Queries

Prompt template shown in Figure 12 instructs GPT to act as a data creator by generating a new question similar to given private data from three private datasets. GPT synthesizes structured task instructions that align with previous patterns for the subsequent model fine-tuning.

J.2 Sampling Response

Figure 13, 14 and 15 show the prompt templates we employed to sample responses from Medical QA, Financial QA and Code Generation datasets, respectively.

J.3 Generate Feedback

Prompt templates shown in Figure 16, 17 and 18 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.

J.4 Refine Synthetic Data

The prompt template illustrated in Figure 19 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 12: 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 13: 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 14: 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 15: 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 16: 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 17: 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 18: 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 19: Prompt template for refining synthetic data