On the Client Preference of LLM Fine-tuning in Federated Learning

Anonymous ACL submission

Abstract

Reinforcement learning with human feedback (RLHF) fine-tunes a pretrained large language model (LLM) using preference datasets, enabling the LLM to generate outputs that align with human preferences. Given the sensitive nature of these preference datasets held by various clients, there is a need to implement RLHF within a federated learning (FL) framework, where clients are reluctant to share their data due to privacy concerns. To address this, we introduce a feasible framework in which clients collaboratively train a binary selector with their preference datasets using our proposed FedBis. With a well-trained selector, we can further enhance the LLM that generates human-preferred completions. Meanwhile, we propose a novel algorithm, FedBiscuit, that trains multiple selectors by organizing clients into balanced and disjoint clusters based on their preferences. Compared to the FedBis, FedBiscuit demonstrates superior performance in simulating human preferences for pairwise completions. Our extensive experiments on federated human preference datasets – marking the first benchmark to address heterogeneous data partitioning among clients – demonstrate that FedBiscuit outperforms FedBis and even surpasses traditional centralized training.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capacities in responding to a wide range of open-ended instructions as professionally as human beings. This achievement is attributed to reinforcement learning with human feedback (RLHF) (Ziegler et al., 2019; Christiano et al., 2017; Ouyang et al., 2022), a method that aligns an LLM with human preferences. Specifically, it trains a reward model (a.k.a. preference model) with the help of a preference dataset, which includes the comparisons among various completions of given instructions. Then, it fine-tunes the LLM towards generating completions that closely match human preference, evaluated by the reward model.

While empirical studies have validated the effectiveness of RLHF in enhancing LLM performance, RLHF faces a challenge regarding preference data collection. There are two approaches for constructing preference datasets, namely, human efforts (Bai et al., 2022; Ganguli et al., 2022; Stiennon et al., 2020) and ChatGPT generation (Dubois et al., 2024). The former gathers the preference data from a team of labelers, who rank the completions of each instruction from best to worst. In contrast, the latter entails a set of instructions together with pairwise completions, with ChatGPT (Achiam et al., 2023) tasked with selecting the superior completion for each instruction. As LLMs are deployed to serve diverse clients, a preference gap may occur between clients and labelers/ChatGPT, impeding the LLM's ability to generate responses that satisfy real clients' tastes. Therefore, there is a demand for a preference dataset that accurately mirrors clients' preferences in order to facilitate LLM performance.

One approach to meeting the demand is to collect client preference data and build a huge preference dataset, with which an LLM can be fine-tuned on a central entity (a.k.a. server). It is noticed that this approach has been applied to a recent open project named OASST (Köpf et al., 2024). However, this approach may be infeasible because most clients refuse the disclosure of their preference data out of privacy concerns. To this end, we adopt FedAvg, a conventional federated learning (FL) algorithm (Konečnỳ et al., 2016; McMahan et al., 2017) that avoids the collection of clients' raw data. Specifically, each client trains a reward model with their preference data, and the server aggregates the reward models into a global model. While the process seems effective, we observe the following two limitations during training:

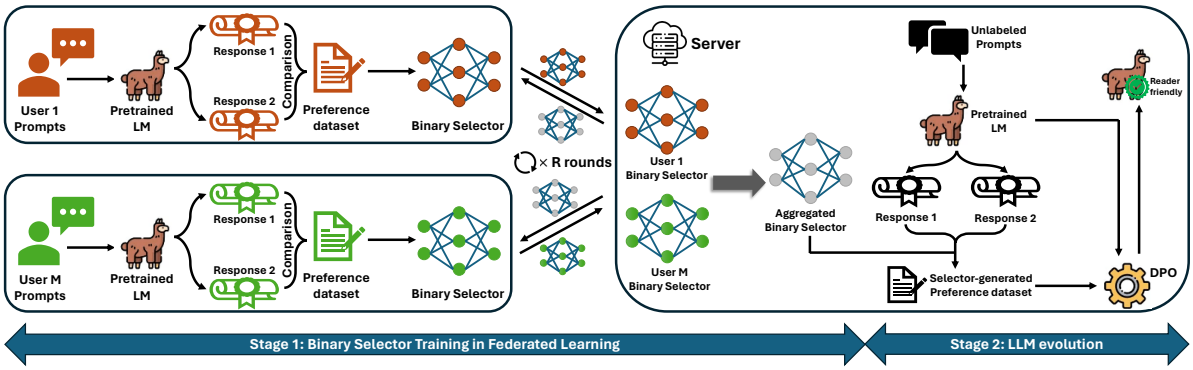

Figure 1: An outline of RLHF in federated learning.

- **Excessive Computation Overhead:** Conventionally, the reward model is designed to output a scalar value for a given prompt and completion (Stiennon et al., 2020). Its optimization is based on reward differences between preferred and dispreferred completions, i.e., the preferred completions should earn greater rewards than the dispreferred ones. As a result, when the optimization involves a data sample in training the model, it should simultaneously retain two computation graphs in the forward and backward pass, leading to significant computation overhead and intensive GPU requirements.

- **Degraded Performance:** Preference data are heterogeneous among the clients in view that the preferences vary among the clients. Consequently, each client trains a reward model towards a local minima, deviating from the global optima and resulting in a longer training time to converge when compared to the centralized training. Moreover, the method is likely to suffer from reward hacking, a phenomenon where the reward model heads to a poor performance after several training rounds (Askell et al., 2021; Michaud et al., 2020; Tien et al., 2022; Skalse et al., 2022).

In this paper, we propose to address these two limitations and propose effective and computationally efficient methods for preference collection and subsequent fine-tuning. We start with a solution that addresses the first limitation. The key idea is to train a binary selector, which chooses a superior response between two completions under a given instruction. Compared with preference ranking, the binary selector requires much less computation during training. Casting binary selector training into a federated learning setting, we thus propose the federated binary selector training (FedBis), as depicted in the first stage of Figure 1. Each client independently trains the selector with local preference dataset, and the server aggregates the selectors into a global one and broadcasts it to the clients. Afterwards, we utilize the binary selector to enhance the performance of LLM. Specifically, we assume the server holds a set of instructions, together with pairwise responses generated by an LLM. Then, we build a preference dataset with the help of the binary selector and boost the LLM by means of direct preference optimization (DPO) (Rafailov et al., 2023).

To further address the performance deterioration due to preference heterogeneity and reward hacking, we propose a method named FedBis with cluster-wise aggregation (FedBiscuit). This method ensembles multiple binary selectors, each trained by the clients possessing similar preferences. In light of privacy concerns, which prevent explicit sharing of clients' data, the server intermittently collects client losses on all binary selectors. Subsequently, clients are organized into disjoint clusters, and when comparing two completions, the one selected by the majority of binary selectors is deemed better. The proposed method has two main advantages. Firstly, clients with similar preferences jointly train a binary selector, moderating data heterogeneity and mitigating performance deterioration. Secondly, the method alleviates reward hacking by having numerous binary selectors jointly decide on optimal completions.

Contributions. In this paper, our contributions are highlighted as follows:

- To the best of our knowledge, this is the first feasible framework to achieve RLHF in FL. In detail, the framework trains binary selector(s) with clients' local datasets, distills the selector(s) toward an LLM, and boosts LLM performance in the meantime. Under this framework, we introduce two methods, i.e., FedBis and FedBiscuit.

- Previous works offer a number of human preference datasets, but none of them address the FL setting. This is the first work to discuss the possible data partition approaches to build a heterogeneous human preference dataset. To this end, we introduce a benchmark that includes several human preference datasets suitable for FL.

- We conduct extensive experiments to demonstrate the performance of the proposed FedBis and FedBiscuit. As expected, FedBiscuit demonstrates superior performance over FedBis and even surpasses traditional centralized training. Meanwhile, we present some insights from the empirical studies.

2 Related Work

LLM Fine-tuning in FL. Recent studies have increasingly focused on fine-tuning large language models (LLMs) using federated datasets (Sun et al., 2024; Ye et al., 2024; Zhang et al., 2023a; Yi et al., 2023; Zhang et al., 2023b). However, these approaches often suffer from high computation and communication costs due to the necessity of training and synchronizing the model with clients. To mitigate these issues, lightweight methods such as black-box fine-tuning (Sun et al., 2023; Lin et al., 2023) and offsite-tuning (Wu et al., 2023b; Kuang et al., 2023) have emerged. Despite their advancements, these methods primarily focus on fine-tuning LLMs for specific downstream tasks, neglecting user preferences in the generated responses. To address this gap, our work aims to align LLMs with human preferences and introduces a feasible training framework in federated learning.

Reinforcement Learning with Human Feedback (RLHF). RLHF typically involves supervised fine-tuning, reward modeling, and reward optimization, initially popularized by Christiano et al. (2017). Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a common RLHF algorithm, yet it struggles with instability, inefficiency, and high resource demands (Choshen et al., 2019; Engstrom et al., 2020). These challenges have led to the development of alternative methods, such as Direct Preference Optimization (DPO) (Rafailov et al., 2023) and others (Dong et al., 2023; Zhao et al., 2023; Azar et al., 2024; Ethayarajh et al., 2024; Gulcehre et al., 2023), which offer more stable and efficient solutions. However, these methods typically operate within a centralized training framework, where the LLM owner retains control over the preference data. In contrast, our work seeks to expand data sources and incorporate real user preferences during the fine-tuning of the LLM.

3 FedBis: A Feasible Framework for Achieving RLHF in FL

The objective of RLHF is to align a pretrained language model with human preferences. RLHF comprises two phases: (i) preference modeling and (ii) reinforcement-learning fine-tuning. The first phase aims to develop a model that simulates human preferences to select the superior options from numerous pairwise completions. Subsequently, the second phase enhances the language model's performance by creating a preference dataset, enabling the model to generate responses preferred by humans. In the following, We describe the proposed FedBis that achieves RLHF in FL in the first two subsections, followed by a brief discussion of its limitations that motivate the proposed FedBiscuit presented in Section 4.

3.1 Preference Modeling

3.1.1 Problem Formulation.

In preference modeling, our objective is to train a binary selector using data from multiple clients. Consider an FL system with M clients, coordinated by a central server. Denote the weight of client m as p_m such that $\sum_{m\in[M]} p_m = 1$, and we aim to optimize the following objectives:

$$\min_{\phi\in\mathbb{R}^d} \quad F(\phi) \triangleq \sum_{m\in[M]} p_m F_m(\phi) \tag{1}$$

where $F_m(\phi)$ is the expected loss on client m given the binary selector ϕ. Suppose client $m \in [M]$ holds a set of pairwise data with the size of n_m, i.e., $\hat{\mathcal{D}}_m = \{(x_i, y_{i,w}, y_{i,l})\}_{i\in[n_m]}$, where x_i is the prompt, $y_{i,w}$ is the preferred completion out of the pair of $y_{i,w}$ and $y_{i,l}$. We reorganize these data and build a preference dataset $\mathcal{D}_m$ to be $\{(x_i, y_{i,w}, y_{i,l}, 0), (x_i, y_{i,l}, y_{i,w}, 1) | (x_i, y_{i,w}, y_{i,l}) \in \hat{\mathcal{D}}_m\}$ for training, in which each contains the prompt, a pair of completions and preference selection. Apparently, this dataset eliminates the position effects, and we can train the selector as a classification task. Therefore, we utilize cross-entropy (CE) loss ℓ_{CE} to optimize the selector and formulate the expected loss as

$$F_m(\phi) = \mathbb{E}_{(x,y_0,y_1,i)\sim\mathcal{D}_m} \left[\ell_{CE}(i|\phi; x, y_0, y_1)\right]. \tag{2}$$

Next, we will discuss how to optimize the selector ϕ under the FL scenario.

3.1.2 Algorithm Design

We consider a practical and efficient FL scenario where not all clients but only a sampled subsets of clients participate in each communication round (Yang et al., 2020). Before the commencement of FL training, we initialize the binary selector with a pretrained LLM such as LLaMA-2 (Touvron et al., 2023), and set the hyperparameters.

An FL algorithm requires multiple communication rounds and consists of three phases in each round, i.e., *model broadcast*, *local training*, and *global aggregation*. Following this paradigm, we design FedBis and optimize the selector ϕ, i.e., in the communication round $r \in [R]$, as discussed as follows.

Step 1: Model Broadcast. The server uniformly samples A clients without replacement, denoted by $\mathcal{A}$. Let the selector be ϕ_r in the r-th communication round, and the server broadcasts it to the sampled clients.

Step 2: Local Training. At this step, client $m \in \mathcal{A}$ optimizes the selector based on local preference data. First, the client initializes the local selector $\phi_{r,0}^m$ with the global selector ϕ_r received from the server. Subsequently, the client trains the selector for K iterations, where the update rule between consecutive iterations follows:

$$\phi_{r,k+1}^m = \phi_{r,k}^m - \eta \nabla F_m(\phi_{r,k}^m), k \in [K] \quad (3)$$

where the gradient $\nabla F_m(\phi_{r,k}^m)$ is approximated using a data batch sampled from the local preference dataset $\mathcal{D}_m$and can incorporate optimizers such as AdamW (Loshchilov and Hutter, 2017). Finally, the client m transmits the updated local selector $\phi_{r,K}^m$ back to the server.

Step 3: Global Aggregation. After receiving the local selectors from the sampled clients $\mathcal{A}$, the server updates the global selector:

$$\phi_{r+1} = \frac{M}{A} \sum_{m \in \mathcal{A}} p_m \phi_{r,K}^m. \quad (4)$$

This aggregation method, based on Li et al. (2019) where the clients are uniformly sampled to train a global model, ensures consistency with Problem (1) in mathematical expectation.

After R communication rounds of training, FedBis outputs a binary selector ϕ_R that reflects the overall preferences of all clients. The selector can then be used to enhance the performance of the LLM, as discussed in the next section.

3.2 Reinforcement-learning Fine-tuning

3.2.1 Problem Formulation.

Traditionally, reinforcement-learning fine-tuning adopts PPO algorithm (Schulman et al., 2017) to enhance the performance of an LLM using a reward model that can rate a completion (Stiennon et al., 2020; Ouyang et al., 2022; Dai et al., 2023), which does not fit the proposed framework with a binary selector.

One practical approach to aligning the LLM with clients' preferences is to create a preference dataset with the help of the binary selector. Suppose the server holds a set of instructions $\hat{\mathcal{D}}$, and we can expand it to a preference dataset $\mathcal{D}_{gen} = \{(x, y_0, y_1, i) | x \in \hat{\mathcal{D}}\}$, where y_0 and y_1 are two completions generated by the LLM θ, and $i \in \{0, 1\}$ indicates the preferred completion as chosen by the binary selector ϕ. With this generated dataset, we apply the Direct Preference Optimization (DPO) algorithm (Rafailov et al., 2023) to optimize the LLM consistent with clients' preferences, which is formulated as

$$\min_{\theta} \mathbb{E}_{(x,y_0,y_1,i)\sim\mathcal{D}_{gen}} \mathcal{L}_{DPO}(\theta | x, y_0, y_1, i) \quad (5)$$

where the DPO loss is $\mathcal{L}_{DPO}(\theta | x, y_0, y_1, i) = -\log \sigma \left(\beta \log \frac{\pi_\theta(y_i|x)}{\pi_{\theta_0}(y_i|x)} - \beta \log \frac{\pi_\theta(y_{1-i}|x)}{\pi_{\theta_0}(y_{1-i}|x)} \right)$. Next we discuss the specifics of the preference data generation and LLM optimization.

3.2.2 Algorithm Design

The reinforcement-learning fine-tuning takes place on the server and includes two phases: 1) a preference dataset is created with a pretrained LLM θ_0 and a well-trained selector ϕ_R from FedBis. 2) LLM is optimized according to the objective defined in Equation (5) with the generated dataset.

Step 1: Preference Dataset Generation. Suppose the server holds a set of instructions $\hat{\mathcal{D}}$. With the LLM θ_0, we can generate multiple completions for an instruction $x \in \hat{\mathcal{D}}$, resulting in a set of n completions $(y_0, \ldots, y_{n-1}) \sim \pi_{\theta_0}(y|x)$. For each instruction, we can form a total of $\binom{n}{2}$ pairs of completions. We then use the binary selector ϕ_R to choose the optimal completion for each pair (y_j, y_l) where $0 \le j < l \le n-1$. The pair is labeled with $i = 0$ if the first logit output is greater than the

second, i.e., $\pi_{\phi_R}(0|x, y_j, y_l) > \pi_{\phi_R}(1|x, y_j, y_l)$, or $i = 1$ otherwise. This process builds the preference dataset $\mathcal{D}_{gen}$.

Step 2: LLM Fine-tuning. With the constructed preference dataset $\mathcal{D}_{gen}$, we evolve the LLM to align with clients' preferences. Specifically, in the t-th training round, where $t \in \{0, 1, \dots\}$, we sample a data batch (x, y_0, y_1, i) from $\mathcal{D}_{gen}$, and update the LLM using the following rule:

$$\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}_{DPO}\left(\theta_t | x, y_0, y_1, i\right), \quad (6)$$

where η is the learning rate. The gradient computation $\nabla \mathcal{L}_{DPO}$ is given by Rafailov et al. (2023). In a nutshell, we distill the binary selector's preferences into the LLM, allowing it to function as a binary selector itself implicitly.

3.3 Discussion

We discuss the limitations of FedBis which motivate us to propose FedBiscuit.

Preference Heterogeneity. A performance gap between FedBis and centralized training could arise from data heterogeneity among clients, a common issue in FL. Different from centralized training that aggregates all the clients' data and samples an i.i.d. batch in each training round, FedBis samples a subset of clients in each round, with each client independently optimizing the model based on their local data. This could result in a global aggregation that diverges from the global optimum (Karimireddy et al., 2020; Wu et al., 2023a).

Reward Hacking. As demonstrated in experiments, FedBis's performance improves first but may later decline with the increase of training rounds. This phenomenon, known as reward hacking, is discussed by Skalse et al. (2022) as an inevitable issue in training a reward proxy model, which is used to enhance the performance of a policy model (e.g., LLM). However, we can mitigate this impact by delaying the inflection point, allowing the reward proxy model to continue improving performance for more training rounds and ultimately achieve a higher rating.

4 FedBiscuit: FedBis with Cluster-wise Aggregation

In this section, we aim to address the aforementioned limitations of FedBis. To tackle reward hacking, Eisenstein et al. (2023) and Coste et al. (2024) introduce a promising approach that trains multiple reward models at the same time because aggregation over multiple reward model outputs can provide a more robust reward estimate. Furthermore, recognizing that some clients may share similar preferences, we employ clustered FL (Sattler et al., 2020; Ghosh et al., 2020; Ma et al., 2023) to group clients with similar preferences for joint model training. Notably, these two approaches complement each other, inspiring us to combine them into a novel algorithm FedBiscuit that simultaneously combats reward hacking and preference heterogeneity.

Problem Formulation. In this work, we consider training multiple binary selectors of U. To ensure that all selectors are trained without bias towards a small specific group, we mandate that these selectors be trained using evenly disjoint clusters of clients. Additionally, a client's preference should align more closely with those within the same cluster than with those in different clusters. To this end, we can formulate the following objective:

$$\min_{\phi_{[U]} \in \mathbb{R}^{U \times d}} F(\phi_{[U]}) \triangleq \sum_{m \in [M]} p_m \left(\min_{u \in [U]} F_m(\phi_u) \right)$$
$$s.t. \quad \max\{|M_u|\}_{u \in [U]} - \min\{|M_u|\}_{u \in [U]} \leq 1 \quad (7)$$

where the function F_m follows the same definition of Equation (2).ϕ_u indicates the u-th binary selector, and M_u means a set of clients using the u-th selector. By definition, $\cup_{u \in [U]} M_u = [M]$, and $\cap_{u \in [U]} M_u = \emptyset$.

Next we explore how the proposed FedBiscuit optimizes Equation (7).

4.1 Algorithm Design

Section 3.1 mentions that a client $m \in [M]$ holds a preference dataset $\mathcal{D}_m$. Before the model training, client m splits her dataset into two disjoint sets, namely, a training set $\mathcal{D}_{m,train}$ and a validation set $\mathcal{D}_{m,val}$, where $|\mathcal{D}_{m,train}| >> |\mathcal{D}_{m,val}|$.

The proposed FedBiscuit consists of two phases: 1) We train each selector for a couple of rounds so that all U selectors have fundamental capacities in selecting the preferred completion, and 2) we divide the clients into disjoint clusters of size U and train each binary selector with a specific cluster.

Phase 1: Warm-up. In the beginning, we initialize each binary selector $\phi_u (u \in [U])$ with an

identical pretrained LLM. Subsequently, starting from $u = 0$, we train a selector ϕ_u for R_{pre} consecutive communication rounds following the steps of FedBis: In each communication round, the server samples a subset of client $\mathcal{A}$ and broadcasts the selector ϕ_u to them. Each client $m \in \mathcal{A}$ then locally trains the selector for K iterations using the dataset $\mathcal{D}_{m,train}$. At the end of the communication round, the server aggregates and updates the selector ϕ_u via Equation (4). After completing the training of ϕ_u, the server initiates the training of the next selector ϕ_{u+1} by repeating the above steps until all selectors are trained.

The selectors are trained with different data distributions because the clients participating in each training round are randomly selected. Consequently, all the selectors $\phi_{[U]}$ have distinct model parameters, leading to varied performance in terms of final logit output when given an instruction and a pair of completions.

Phase 2: Clustered FL Training. After the first phase, we obtain U different selectors, denoted as $\{\phi_{u,0}\}_{u\in[U]}$. Unlike FedBis, this phase includes an additional step called *client grouping*, which partitions the clients into multiple disjoint clusters based on their preferences. In each communication round $r \in [R]$, the proposed FedBiscuit optimizes all the selectors $\phi_{[U]}$ using the following four steps:

Step 2.1: Client Grouping. This step is executed every τ communication rounds, i.e., when r can be divided by τ, or $\tau | r$. During this step, the server broadcasts all selectors $\phi_{[U],r}$ to all clients $[M]$. Then, a client m calculates the averaged loss for each selector $\phi_{u,r}$ using local validation set via $\frac{1}{|\mathcal{D}_{m,val}|}\sum_{(x,y_0,y_1,i)\sim\mathcal{D}_{m,val}}[\ell_{CE}(i|\phi_{u,r}; x, y_0, y_1)]$. The server thereby collects all these losses and adopts a greedy clustering approach (Sattler et al., 2020; Ma et al., 2023) to assign each client to the selector where they achieve the minimum loss. However, an obvious deficiency is an imbalance where some selectors are chosen by many clients and others by few. It is noted that the selectors trained with more clients achieve remarkable performance, while some may be overfitted to a specific group of clients. Therefore, the greedy clustering approach negatively impacts the overall performance when building a global preference dataset. To tackle the limitation, we propose to balance the clusters using the following steps repeatedly until the clients are evenly distributed:

- Choose the cluster selected by the most clients.
- If the cluster can accommodate n clients, cap the cluster at n clients and reassign the rest to other clusters where they achieve suboptimal loss.

Finally, we obtain balanced and disjoint clusters. Let a client m train with the U_m-th selector ϕ_{U_m} for the next τ rounds. After client grouping step, the proposed method proceeds to the following three steps as outlined in FedBis.

Step 2.2: Model Broadcast. Similar to FedBis, the server samples A clients from all clients $[M]$, denoted by $\mathcal{A}$. For each selected client $m \in \mathcal{A}$, the server transmits the selector $\phi_{U_m,r}$. This process can be characterized by defining $\mathcal{A}_u$ as the group of clients chosen to train the selector ϕ_u. This ensures that $\cup_{u\in[U]}\mathcal{A}_u = \mathcal{A}$ and $\cap_{u\in[U]}\mathcal{A}_u = \emptyset$.

Step 2.3: Local Training. The client $m \in \mathcal{A}$ receives a binary selector $\phi_{U_m,r}$ from the server and trains the selector for K iterations following the update rule of Equation (3). Finally, let the updated local selector be $\phi^m_{U_m,r,K}$, and the client pushes it to the server.

Step 2.4: Global Aggregation. The server collects updated selectors from all participants $\mathcal{A}$. Since there are several binary selectors, the server updates each one with a designated group of clients intended to train on that specific selector. For instance, the aggregation rule for the selector $u \in [U]$ follows

$$\phi_{u,r+1} = \left(1 - \sum_{m\in\mathcal{A}_u} p_m\right)\phi_{u,r} + \sum_{m\in\mathcal{A}_u} p_m \phi^m_{u,r,K} \tag{8}$$

It is noted that performance degradation occurs when a model is trained by clients with time-varying sizes in FedAvg (Gu et al., 2021; Wang and Ji, 2023). In other words, Equation (4) is no longer suitable for multi-selector aggregation due to the fluctuation in the number of clients training a selector in each communication round. Therefore, FedBiscuit adopts a new aggregation rule as formulated in Equation (8).

FedBiscuit finally produces a set of well-trained selectors $\phi_{[U],R}$ and he subsequent objective is to enhance LLM performance with the help of these selectors, as explored below.

Reinforcement-learning Fine-tuning with Multiple Selectors. We can leverage the methodology

mentioned in Section 3.2, and one of the key steps involves constructing a preference dataset incorporating multiple selectors. For this, we employ a strategy of majority voting. Given an instruction $x \in \hat{\mathcal{D}}$ and a pair of generated completions (y_0, y_1), we assume a selector $u \in [U]$ prefers y_{i_u}, where $i_u \in \{0, 1\}$. Therefore, the pair is assigned a label $i = \arg\max\{i_u\}_{u\in[U]}$, meaning that the completion y_i is favored by the most clients.

4.2 Discussion: Integration with LoRA

As all binary selectors are LLM, training them may consume significant communication and computation overheads. Besides, multiple LLMs lead to considerable storage burdens shouldered by the server. To reduce the costs, we adopt a parameter-efficient fine-tuning approach LoRA (Hu et al., 2021), where all binary selectors share the same base model while using different adapters.

In comparison with FedBis, FedBiscuit requires extra costs, i.e., $O(MU\lfloor R/\tau \rfloor \cdot C)$, where C is the communication cost of a selector. This is because FedBiscuit involves client grouping periodically, unilaterally transferring all selectors from the server to the clients. Despite the extra costs, extensive experiments demonstrate non-trivial improvement by comparing FedBiscuit with FedBis.

5 Federated Human Preference Benchmark

This section mainly focuses on how we prepare federated human preference datasets, while the next section introduces the experimental setup and analyzes numerical results. Specifically, we cover two of the most common NLP tasks, i.e., summarization and question-answering. All two datasets are partitioned based on the public datasets, and the following subsections will include the details. We will release these datasets on HuggingFace soon.

Summarization. Stiennon et al. (2020) introduces a summarization dataset that consists of Reddit posts with human-written tl;dr (Völske et al., 2017). This dataset consists of two parts, one is a pretrained dataset, while the other is a dataset with human preference. As suggested by Ouyang et al. (2022), we ensure a post does not appear in both datasets. We assume the pretrained dataset is stored on the server side, and 60% of data are served for model pertaining such that the model can perform well on summarization. The remaining 40% are used for the RLHF process to improve the LLM performance and generate human-preferred content. Since the human-preference dataset contains the worker ID, we partition the dataset based on the worker ID so that the dataset can be partitioned into 53 workers.

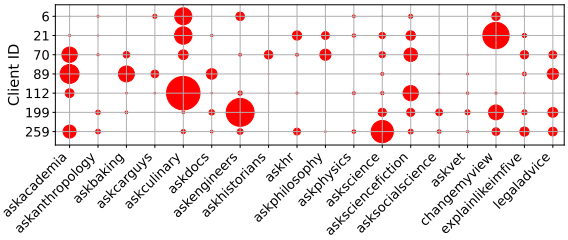

Figure 2: Data distribution across different question domains on the selected clients.

Question-Answering (QA). We reconstruct the public dataset SHP, which comprises numerous questions from Reddit posts and their corresponding user answers (Ethayarajh et al., 2022). The preference indicator is based on the number of likes an answer receives. Given that the dataset spans 18 domains, we partition the dataset using a Dirichlet distribution with a parameter of 0.3, ensuring that no questions overlap between clients. In our experiment, we prepare 300 clients, and Figure 2 visualizes the data distribution on the selected clients. For the RLHF process, we use a set of 2.6K Reddit questions.

6 Experiments

6.1 Experimental Setup

Model and computation environment. We initialize the binary selector(s) using the pretrained LLaMA-2-7B (Touvron et al., 2023), configuring the final layer to produce binary outputs "A" and "B" only. The LLM chosen for content generation depends on the tasks: (i) For the summarization task, we start with LLaMA-2-7B and fine-tune it using a pretrained dataset; (ii) For the QA task, we initialize the LLM with Alpaca-7B (Taori et al., 2023). To reduce computation efforts, we employ LoRA to fine-tune the models. Our implementation, built upon FederatedScope (Xie et al., 2023; Kuang et al., 2023), will soon be available on GitHub. The experiments are conducted on machines equipped with two Nvidia A100 GPU cards, Intel Xeon Platinum 8369B CPUs, and 256GB RAM.

Evaluation. We evaluate two models produced by our proposed FedBis and FedBiscuit: a binary selector and an LLM. We employ different strategies to assess each model:

	Selector		LLM	
	Agreement	Best-of-n	Rating	Win Rate
SFT	-	-	5.028	29.71%
Centralized	**73.10%**	5.302	5.688	78.89%
FedBis	70.44%	5.274	5.661	71.35%
FedBiscuit	70.52%	**5.305**	**5.703**	**80.65%**

Table 1: Performce under summarization task. All values here indicate their best performance within 500 communication rounds of training.

- **Binary selector:** The evaluation includes two metrics: agreement and best-of-n. Agreement measures the hit rate of a selector against a preference dataset annotated by humans or ChatGPT. Additionally, we use the best-of-n approach by selecting the best completion from n generated by a task-specific LLM. We then evaluate the average rating for the selector's choices using Auto-J (Li et al., 2023a).

- **LLM:** After reinforcement-learning fine-tuning, the LLM is evaluated on its ability to generate human-preferred content. This means we can assess the quality of the generated texts. For example, given an instruction set, the LLM produces one completion per instruction, and Auto-J evaluates the average rating of these completions. Furthermore, we compare the generated completions with a reference set of responses, annotated by humans or ChatGPT, and calculate a win rate based on how often the generated response is superior to the reference one.

Due to the space limit, the hyperparameter settings are presented in Appendix A. Moreover, Appendix B provides real cases to demonstrate the performance of the proposed FedBis and FedBiscuit. In particular, Appendix B.1 discusses the results under the QA task.

6.2 Numerical Results on Summarization

In this section, the evaluation data originates from the TL;DR dataset, as mentioned in Section 5. The dataset comprises two disjoint parts: one ranked by a group of labelers for model-generated responses, and the other written by users to summarize the key content of a post. We use the former to compute the consistency between the selector and the human annotator. For the latter, we apply various metrics, including best-of-n, rating, and win rate. The results are presented in Table 4 and Figure 3.

The table shows that conventional centralized training outperforms the proposed FedBiscuit in terms of agreement. This is because the agreement evaluation data have a similar distribution to the training dataset, as their outputs are generated from the same language models and labeled by the same group of labelers (Stiennon et al., 2020). Consequently, centralized training performs better than the proposed FedBis and FedBiscuit, which are affected by data heterogeneity.

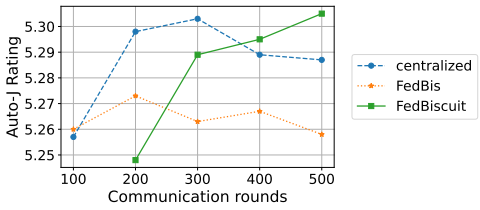

Figure 3: Auto-J rating of best-of-n against communication rounds under summarization tasks.

However, when evaluating the selectors with datasets generated by a supervised fine-tuning model, the proposed FedBiscuit slightly outperforms centralized training. These results suggest that a centrally trained selector performs poorly in terms of generalization and is prone to overfitting to a specific dataset distribution. In contrast, comparing FedBiscuit with FedBis, we find that FedBiscuit mitigates data heterogeneity and produces a more robust selection of completion pairs.

Figure 3 illustrates the performance trend across communication rounds. As discussed in Section 3.3, training a binary selector can lead to reward hacking. For both centralized training and FedBis, which train a single selector, we observe an inflection point where the selector's performance begins to decline. However, this inflection point has not yet appeared in FedBiscuit, allowing it to continuously improve and eventually surpass the best performance of centralized training. It is important to note that the warmup rounds are included in the communication rounds, which explains FedBiscuit's initial poor performance.

7 Conclusion

In this work, we explore a feasible framework to achieve RLHF in FL. Specifically, we train a binary selector across different clients using their local preference datasets, and then use the well-trained selector to align an LLM with human preferences. We propose two approaches to enable selector training: FedBis and FedBiscuit. FedBis provides a framework to train a single selector, while FedBiscuit ensembles multiple selectors to more robustly simulate human preferences. With the proposed federated human preference datasets, we conduct empirical studies to validate our statements and demonstrate the superiority of FedBiscuit.

Ethics Statement

This paper investigates clients' preferences using a publicly available dataset, ensuring that all data sources are appropriately cited to maintain academic integrity and transparency. By leveraging this public dataset, we avoid using private or sensitive client data, thus upholding ethical standards in data usage and research practices. Furthermore, this work prioritizes the protection of clients' privacy and strictly avoids any disclosure of local data. When clients utilize their own data to fine-tune the model, robust privacy measures are in place to ensure that no other clients can access or infer any information related to their data. This approach not only safeguards individual privacy but also fosters trust and security in the application of the model.

Limitations

One notable limitation of our work lies in the construction of the preference dataset, which relies solely on publicly available data rather than gathering information directly from real clients. By doing so, we miss out on the nuances and intricacies of individual preferences that can only be captured through firsthand data collection. As a result, our dataset may lack the depth and breadth necessary to fully comprehend the true heterogeneity of preferences among clients. Without access to authentic client data, we may inadvertently overlook important variations in preferences, potentially limiting the applicability and robustness of our findings.

Another limitation pertains to the use of a task-specific dataset rather than a more generalized one encompassing a broader spectrum of tasks. While task-specific datasets offer advantages such as focused analysis and tailored insights, they may also restrict the scope of our research and hinder its generalizability. By incorporating a more diverse range of tasks into our dataset, we could gain a more comprehensive understanding of clients' preferences across various domains, thereby enhancing the versatility and validity of our findings.

Additionally, our work employs a binary selector that implicitly assumes one response is superior to another, overlooking scenarios where responses may exhibit similar levels of quality. This oversimplified approach fails to leverage valuable data that could provide valuable insights into subtle differences and nuances in preferences. By adopting a more nuanced and inclusive framework that acknowledges and incorporates variations in response quality, we could extract richer insights and make more informed decisions regarding client preferences. Addressing these limitations could bolster the robustness and validity of our research, ultimately enhancing its relevance and impact in real-world applications.

References

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*.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Leshem Choshen, Lior Fox, Zohar Aizenbud, and Omri Abend. 2019. On the weaknesses of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1907.01752*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. 2024. Reward model ensembles help mitigate overoptimization. In *The Twelfth International Conference on Learning Representations*.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2023. Safe rlhf: Safe reinforcement learning from human feedback. *arXiv preprint arXiv:2310.12773*.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin,

Percy S Liang, and Tatsunori B Hashimoto. 2024. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, et al. 2023. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. *arXiv preprint arXiv:2312.09244*.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. 2020. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*.

Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. 2022. Understanding dataset difficulty with $\mathcal{V}$-usable information. In *International Conference on Machine Learning*, pages 5988–6008. PMLR.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597.

Xinran Gu, Kaixuan Huang, Jingzhao Zhang, and Longbo Huang. 2021. Fast federated learning in the presence of arbitrary device unavailability. *Advances in Neural Information Processing Systems*, 34:12052–12064.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

Weirui Kuang, Bingchen Qian, Zitao Li, Daoyuan Chen, Dawei Gao, Xuchen Pan, Yuexiang Xie, Yaliang Li, Bolin Ding, and Jingren Zhou. 2023. Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. *arXiv preprint arXiv:2309.00363*.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. 2023a. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023b. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Zihao Lin, Yan Sun, Yifan Shi, Xueqian Wang, Lifu Huang, Li Shen, and Dacheng Tao. 2023. Efficient federated prompt tuning for black-box large pre-trained models. *arXiv preprint arXiv:2310.03123*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Jie Ma, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2023. Structured federated learning through clustered additive modeling. *Advances in Neural Information Processing Systems*, 36.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Eric J Michaud, Adam Gleave, and Stuart Russell. 2020. Understanding learned reward functions. *arXiv preprint arXiv:2012.05862*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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*, volume 36, pages 53728–53741. Curran Associates, Inc.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2020. Learning to summarize from human feedback. In *NeurIPS*.

Jingwei Sun, Ziyue Xu, Hongxu Yin, Dong Yang, Daguang Xu, Yiran Chen, and Holger R Roth. 2023. Fedbpt: Efficient federated black-box prompt tuning for large language models. *arXiv preprint arXiv:2310.01467*.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. 2024. Improving lora in privacy-preserving federated learning. *arXiv preprint arXiv:2403.12313*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Jeremy Tien, Jerry Zhi-Yang He, Zackory Erickson, Anca D Dragan, and Daniel S Brown. 2022. Causal confusion and reward misidentification in preference-based reward learning. *arXiv preprint arXiv:2204.06601*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017. Tl; dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63.

Shiqiang Wang and Mingyue Ji. 2023. A lightweight method for tackling unknown participation probabilities in federated averaging. *arXiv preprint arXiv:2306.03401*.

Feijie Wu, Song Guo, Zhihao Qu, Shiqi He, Ziming Liu, and Jing Gao. 2023a. Anchor sampling for federated learning with partial client participation. In *International Conference on Machine Learning*, pages 37379–37416. PMLR.

Feijie Wu, Zitao Li, Yaliang Li, Bolin Ding, and Jing Gao. 2023b. Fedbiot: a solution for federated large language model fine-tuning with intellectual property protection.

Yuexiang Xie, Zhen Wang, Dawei Gao, Daoyuan Chen, Liuyi Yao, Weirui Kuang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2023. Federatedscope: A flexible federated learning platform for heterogeneity. *Proceedings of the VLDB Endowment*, 16(5):1059–1072.

Haibo Yang, Minghong Fang, and Jia Liu. 2020. Achieving linear speedup with partial worker participation in non-iid federated learning. In *International Conference on Learning Representations*.

Rui Ye, Wenhao Wang, Jingyi Chai, Dihan Li, Zexi Li, Yinda Xu, Yaxin Du, Yanfeng Wang, and Siheng Chen. 2024. Openfedllm: Training large language models on decentralized private data via federated learning. *arXiv preprint arXiv:2402.06954*.

Liping Yi, Han Yu, Gang Wang, and Xiaoguang Liu. 2023. Fedlora: Model-heterogeneous personalized federated learning with lora tuning. *arXiv preprint arXiv:2310.13283*.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Guoyin Wang, and Yiran Chen. 2023a. Towards building the federated gpt: Federated instruction tuning. *arXiv preprint arXiv:2305.05644*.

Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. 2023b. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Annual Meeting of the Association of Computational Linguistics 2023*, pages 9963–9977. Association for Computational Linguistics (ACL).

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. 2023. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A More Implementation Details

In this section, we include various settings, such as the prompt and the hyperparameters.

A.1 Hyperparameter Settings

In our work, we fine-tune all models using LoRA, which is consistently set to rank 8, $\alpha = 16$, and the dropout rate 0.0. For the generation, we apply with these parameters:

- If it is required to generate multiple completions, then we set the temperature to 1.0.
- If it is required to generate a single completion, then we adopt greedy search by setting the temperature to 0.0.

In the following part, we show the hyperparameter setting for different tasks:

	SFT	Selector Training	RLFT
Participation Rate	-	5/53	-
Local Iterations	30	30	30
Batch Size	32	16	32
Rounds	1000	500	500
Optimizer	AdamW	AdamW	RMSprop
Hyperparameters	(0.9, 0.95)	(0.9, 0.95)	–
Learning rate	$1e-4$	$1e-5$	$1e-6$

Table 2: Hyperparameter Settings for the Summarization Task

	Selector Training	RLFT
Participation Rate	10/300	-
Local Iterations	10	10
Batch Size	16	16
Rounds	200	200
Optimizer	AdamW	RMSprop
Hyperparameters	(0.9, 0.95)	–
Learning rate	$1e-5$	$1e-6$

Table 3: Hyperparameter Settings for the QA Task

Special Setting for FedBiscuit For the above two tasks, we ensemble three binary selectors (i.e., LoRAs). In the warmup round, we train the selector for 50 rounds under an FL framework. FedBiscuit performs regrouping every 50 rounds in the summarization task, while regrouping every 100 rounds in the QA task.

A.2 Instruction Tuning Prompt

In this section, we highlight the prompts used to fine-tune the summarization tasks and the QA task:

Summarization. For pertaining (SFT) and the later reinforcement-learning fine-tuning (RLFT), it follows the prompt below

> Below is a forum post. Write a precise and concise summary that includes the most important points of the post.
>
> ### SUBREDDIT: r/{subreddit}
> ### TITLE: {title}
> ### POST: {post}
> ### TL;DR:

For comparison:

> Below is a forum post followed by two summaries. Pick a more precise and concise one that summarizes the most important points in the given forum post, without including unimportant or irrelevant details. State your choice with a single capital letter, i.e., Äïf SUMMARY A is better, Bïf SUMMARY B is better.
>
> ### SUBREDDIT: r/{subreddit}
> ### TITLE: {title}
> ### POST: {post}
> ### SUMMARY A: {output_A}
> ### SUMMARY B: {output_B}
> ### YOUR CHOICE:

QA. As the QA utilizes a pretrained model named Alpaca-7B, we follow its pretrained format

> Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.
>
> ### Instruction:
> {instruction}
>
> ### Input:
> {input}
>
> ### Response:

For comparison between the two responses:

Below is a query followed by two responses. Pick a helpful response that is precise, concise, and casual. State your choice with a single capital letter, i.e., Äïf RESPONSE A is better, Bïf RESPONSE B is better.

QUERY: {instruction}
RESPONSE A: {output_A}
RESPONSE B: {output_B}
YOUR CHOICE:

B More Numerical Results and Analysis

B.1 Numerical Results on QA

In this section, the test dataset comes from AlpacaFarm (Dubois et al., 2024; Li et al., 2023b). The results are as follows:

	2-completion Rating	4-completion Rating
Alpaca-7B	3.752	-
FedBis	4.140	4.113
FedBiscuit	4.094	3.830

Table 4: Performce under QA.

As presented in Section 4, the LLM owner will generate a set of responses to a given instruction before building a preference dataset. Therefore, the column "2-completion" means the owner prepares 2 completions for each instruction, while "4-completion" means 4 completions for each instruction and forms 6 pairs. The row "Alpaca-7B" acts as a baseline to help us understand the performance of the proposed FedBiscuit and FedBis. All the rating comes from Auto-J (Li et al., 2023a), which would be different from the ratings reported by (Li et al., 2023b) because it evaluates with GPT-4 (Achiam et al., 2023).

The table above may lead to conclusions different from those drawn from the summarization task. First, FedBis achieves better performance than FedBiscuit. This is within our expectations. First, these selectors are trained for a total of 200 rounds. As presented in Figure 3, FedBiscuit surpasses FedBis after 300 communication rounds. This is because the selectors of FedBiscuit are trained for 100 rounds only, while the selector of FedBis has been fully trained for 200 rounds. When the inflection point appears in FedBis, we can hypothesize that the dominance of FedBiscuit still exists.

Another comparison arises between different numbers of generations to a given prompt. From the table, we notice that "2-completion" can achieve better performance than "4-completion," meaning that the performance may not be relevant to the size of the RLFT training set. Instead, it may rely on the quality of the training data. As we can see, Alpaca-7B hardly generates high-quality data, leading to limited improvement with training with these generated data. In other words, if we generate more low-quality completions, the improvement of the model would be more limited. At the same time, we can hypothesize that if the generated data are of high quality, "4-completion" may outperform "2-completion" in terms of the final rating.

B.2 Examples on the Final RLHF Results

In this section, we prepare two examples to illustrate the performance of our proposed FedBis and FedBiscuit and compare them with other baselines. Refer to Table 5 and Table 6 for the demonstration of the examples.

Subreddit: r/relationships Title: Me [25F] with my SO [27 M] of 3.5 years, I went through his texts and can't stop thinking about something I saw. Post: I admit that going through his text messages was wrong and I do feel incredibly guilty about it now but the bigger issue for me is that I found something that I can't stop thinking about. He was talking to his good female friend and I saw a text that said "am I being an idiot for leaving [my name]"? . I stopped reading after that because I started having a panic attack. The idea of breaking up was not even on my radar and I thought he was more than happy. Again, I know I shouldn't have looked but whats done is done and now I can't stop worrying about when it's going to happen. And now i'm being clingy just because of my worry which i'm sure doesn't help. I love him and don't want to lose him.
Human Summary: I saw a text from my SO to his friend about leaving me, can't stop freaking out. Need serious advice.
SFT: I went through my SO's texts and found something I wasn't supposed to. Now I'm being clingy and I don't want to lose him.
Centralized: went through my boyfriends texts and found something that i'm now worrying about when/if he's going to break up with me. I love him and don't want to lose him. Help.
FedBis: went through SO's texts and found a text that said "am I being an idot for leaving [my name]"? and now I'm worried he's going to break up with me and can't stop thinking about it.
FedBiscuit: went through SO's texts, found something that i'm now worrying about when the inevitable breakup is going to happen. Now being clingy and can't stop thinking about it. Help!

Table 5: An example for summarization task. Caution: The example data that may be offensive or harmful.

List the layers of the TCP/IP model and for each layer identify one protocol.
Alpaca-7B: The TCP/IP model consists of four layers: the application layer, transport layer, internet layer and network access layer. Application Layer: HTTP Transport Layer: TCP Internet Layer: IP Network Access Layer: Ethernet
FedBis: The TCP/IP model has four layers: the application layer, transport layer, internet layer and network access layer. At the application layer, the protocol is TCP. At the transport layer, the protocol is UDP. At the internet layer, the protocol is IP. At the network access layer, the protocol is Ethernet. 1. Application layer: TCP 2. Transport layer: UDP 3. Internet layer: IP 4. Network access layer: Ethernet
FedBiscuit: The TCP/IP model has four layers: the application layer, transport layer, internet layer and network access layer. At the application layer, the HTTP protocol is used. The transport layer uses the TCP protocol. At the internet layer, the IP protocol is used. Finally, the network access layer uses the Ethernet protocol.

Table 6: An example for QA task. Caution: The example data that may be offensive or harmful.