FedMKT: Federated Mutual Knowledge Transfer for Large and Small Language Models

Anonymous ACL submission

Abstract

Recent research in federated large language models (LLMs) has primarily focused on enabling clients to fine-tune their locally deployed homogeneous LLMs collaboratively or on transferring knowledge from server-based LLMs to small language models (SLMs) at downstream clients. However, a significant gap remains in the simultaneous mutual enhancement of both the server's LLM and clients' SLMs. To bridge this gap, we propose FedMKT, a parameter-efficient federated mutual knowledge transfer framework for large and small language models. This framework is designed to adaptively transfer knowledge from the server's LLM to clients' SLMs while concurrently enriching the LLM with clients' unique domain insights. We facilitate token alignment using minimum edit distance (MinED) and then selective mutual knowledge transfer between client-side SLMs and a server-side LLM, aiming to collectively enhance their performance. Through extensive experiments across three distinct scenarios, we evaluate the effectiveness of FedMKT using various public LLMs and SLMs on a range of NLP text generation tasks. Empirical results demonstrate that FedMKT simultaneously boosts the performance of both LLMs and SLMs.

1 Introduction

Large Language Models (LLMs) have emerged as a transformative force in artificial intelligence, profoundly altering our perception of natural language processing capabilities. The advent of cutting-edge LLMs like ChatGPT (OpenAI, 2022), and LLaMA (Touvron et al., 2023) with their billions of parameters, has sparked the imagination of both researchers and practitioners, owing to their exceptional performance across diverse text generation tasks. Despite their widespread success in various general NLP tasks, LLMs face challenges that hinder their adoption in domain-specific applications (Kang et al., 2023) (Fan et al., 2023). The primary challenges include domain-specific knowledge Privacy, constrained computing resources, and mutual knowledge transfer between the LLM and SLMs. A significant challenge arises from the inherent model heterogeneity between the LLM and SLMs, particularly when aligning distributions of output logits. The mismatch between the tokenizers of different LLM and SLMs poses a notable obstacle. Furthermore, the mutual transfer of knowledge between the server's LLM and clients' SLMs remains a largely unexplored area in academic literature, warranting further investigation.

To fill these gaps, we propose FedMKT, a novel federated mutual knowledge transfer framework designed to enhance the performance of both large and small language models. By leveraging the complementary strengths of federated learning and knowledge distillation, FedMKT facilitates effective mutual knowledge transfer between clients' SLMs and the LLM owned by the server.

As illustrated in Figure 1, FedMKT deploys an LLM on the server and a set of K heterogeneous SLMs across various clients. The cornerstone of FedMKT lies in its selective mutual knowledge transfer process. During each round of federated learning, the clients transmit the output logits of their updated SLMs on the public dataset to the server. The server then selectively aggregates and distills the knowledge encoded within these SLMs output logits into the server-side LLM. This process allows the server LLM to incorporate the domain-specific knowledge learned by the clients, thereby enhancing its comprehensive capabilities. Simultaneously, the server-side LLM also selectively distills its knowledge to the clients' SLMs, which is similar to the knowledge transfer from clients to the server. By leveraging the knowledge of the server LLM, the clients' SLMs are able to improve their performance and generalize better to unseen data. To address the model heterogeneity between the LLM and SLMs, FedMKT incorporates a token alignment technique utilizing minimum edit distance (MinED) prior to knowledge transfer. This alignment ensures seamless integration and efficient knowledge transfer between LLM and SLMs.

Our contributions are summarized as follows:

- **Federated Mutual Knowledge Transfer Framework**. FedMKT introduces a novel federated mutual knowledge transfer framework that enables effective knowledge transfer between an LLM deployed on the server and SLMs residing on clients. This framework fills the gap by simultaneously enhancing both the server's LLM and the clients' SLMs.

- **Selective Knowledge Transfer and Token Alignment**. FedMKT implements a selective knowledge transfer mechanism that selectively distills knowledge from the most informative SLMs to the server's LLM and vice versa. Furthermore, it incorporates a token alignment technique using minimum edit distance (MinED) to address model heterogeneity between LLM and SLMs, ensuring efficient knowledge transfer.

- **Empirical Evaluation and Performance Enhancement**. Extensive experiments conducted based on various publicly available LLMs and SLMs demonstrate the competitive performance of FedMKT across a wide range of NLP text-generation tasks. We evaluate FedMKT with heterogeneous, Homogeneous, and One-to-One settings. The results show that the performance of SLMs can be significantly enhanced with the help of the LLM, while the LLM can deliver comparable results to fine-tuning with all clients' data centralized.

2 Related Work

2.1 Model Heterogeneous Federated Learning

Model heterogeneous federated learning (MHFL) aims to address the challenges associated with heterogeneity in federated learning. Initial research in MHFL primarily concentrated on addressing heterogeneity in model architectures. Various approaches have been proposed to accommodate clients with different model architectures participating in a federated learning task. These methods typically involve techniques such as knowledge distillation (Hinton et al., 2015), mutual learning and split learning that can handle heterogeneous models. Knowledge distillation-based MHFL methods, such as FedMD (Li and Wang, 2019) and FedET (Cho et al., 2022), involve the server aggregating the output logits of different clients' heterogeneous models on a public dataset to construct global logits. Mutual learning-based MHFL, such as Deep Mutual Learning (DML) (Zhang et al., 2018), PFML (Yang et al., 2021) and FedLoRA (Yi et al., 2023), design a small homogeneous model and a large heterogeneous model in each client. Split learning-based MHFL approaches, such as FedClassAvg (Jang et al., 2022) and CHFL (Liu et al., 2022), share a homogeneous classifier to improve model classification while personalizing the local feature extractor.

While previous works have mainly focused on computer vision scenarios, the literature has limitedly explored MHFL in LLMs. This gap motivates this study, which aims to explore MHFL in the context of LLMs.

2.2 Federated Learning for LLMs

Parameter-Efficient Fine-Tuning (PEFT) methods (Houlsby et al., 2019; He et al., 2021; Lester et al., 2021; Li and Liang, 2021; Hu et al., 2021) offer a direct solution to the issues of communication overhead and fine-tuning costs in federated learning (FL) for LLMs. A number of studies have built upon PEFT methods in the context of FL for LLMs, including FedPETuning (Zhang et al., 2022b), Federated Adapter Tuning (Cai et al., 2022), Federated Prompt Tuning (Zhao et al., 2022), and FATE-LLM (Fan et al., 2023). For example, the FedPETuning (Zhang et al., 2022b) has demonstrated a significant reduction in communication overhead, reducing 1 to 2 orders of magnitude compared to full fine-tuning in the FL setting. These findings imply that FL clients, such as devices with limited storage capacity, can greatly benefit from PEFT methods. These methods enable the sharing of LLMs across different tasks while maintaining only a few parameters for each task, thereby reducing the storage requirement. By leveraging PEFT methods, FL clients can efficiently adapt LLMs to their specific needs while minimizing communication overhead and fine-tuning costs.

3 The Proposed FedMKT Method

In this section, we introduce FedMKT, an innovative and parameter-efficient federated mutual

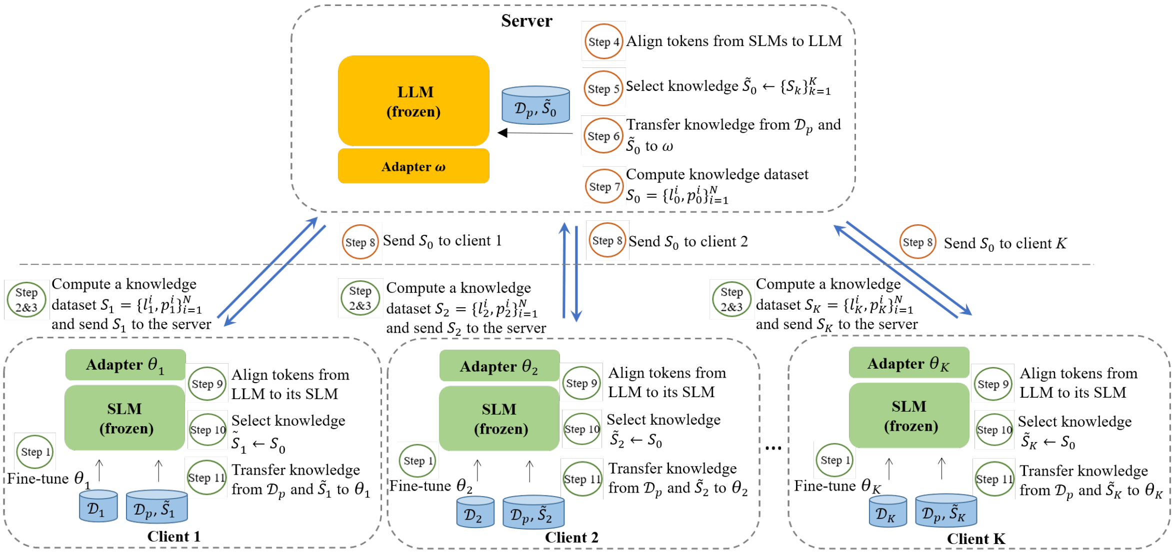

Figure 1: Overview of the proposed FedMKT workflow. Each communication round of FedMKT involves 11 steps to fine-tune the server' LLM and clients' SLMs.

knowledge transfer approach for large and small language models. The FedMKT primarily comprises two key modules: *Bidirectional Token Alignment* and *Selective Mutual Knowledge Transfer*. We will elaborate on these two modules in Section 3.2 and Section 3.3, respectively after we define the problem we try to address in Section 3.1.

3.1 Problem Definition

We consider the federated learning setting, involving one server that owns an LLM f_ψ parameterized by ψ and K clients that each client k has an SLM g_{ϕ_k} parameterized by ϕ_k. Each client owns a local private dataset $\mathcal{D}_k$ with N training samples, and all clients and the server share a public dataset $\mathcal{D}_p$.

The server and clients aim to collaboratively enhance the performance of the LLM and SLMs through federated learning without disclosing any private data. We assume that the K clients execute the same text generation task, but they may hold heterogeneous or homogeneous SLM models. The collaboration between clients and the server involves the following sub-procedures:

- Each client k trains its SLM g_{ϕ_k} using its private data $\mathcal{D}_k$. The objective is formulated as follows:

$$\min_{\phi_1,\phi_2,\ldots,\phi_K} \mathcal{L}_1(\phi_1, \phi_2, \ldots, \phi_K; \{\mathcal{D}_k\}_{k=1}^{K}) \quad (1)$$

- Each client computes the output logits on $\mathcal{D}_p$ and securely uploads them to the server. Upon receiving output logits of all clients, the server computes the distillation loss by comparing these client logits with the output logits produced by its own LLM on $\mathcal{D}_p$. The objective can be formulated as follows:

$$\min_{\psi} \mathcal{L}_2(\psi; \mathcal{D}_p, \phi_1, \phi_2, \ldots, \phi_K) \quad (2)$$

The server aims to transfer knowledge from the clients' SLMs g_{ϕ_k} to its owned LLM f_ψ.

- The server dispatches the LLM's output logits on $\mathcal{D}_p$ to all the clients. Subsequently, the clients compute the distillation loss by comparing LLM output logits with SLMs' output logits on $\mathcal{D}_p$. The objective can be formulated as follows:

$$\min_{\phi_1,\phi_2,\ldots,\phi_K} \mathcal{L}_3(\phi_1, \phi_2, \ldots, \phi_K; \mathcal{D}_p, \psi) \quad (3)$$

The clients aim to transfer knowledge from LLM f_ψ to enhance their SLMs.

We consider the server *semi-honest*, meaning that the server may try to recover the private data of clients from the information it observes.

FedMKT solves the optimization problems formulated in Eq.(1), Eq.(2), and Eq.(3) in an efficient and privacy-preserving manner. We illustrate the workflow of FedMKT in Figure 1 and elaborate on the associated training algorithm in Algorithm 1.

3.2 Bidirectional Token Alignment

A significant challenge in aligning output logits distributions lies in the mismatch between tokenizers of different LLM and SLMs, exemplified by

Algorithm 1 FedMKT

Input:

K: number of clients;
T: total number of communication rounds;
R: local number of rounds in the server;
E: local number of rounds in the client;
η_ω: the learning rate of LLM $f_{\psi+\omega}$;
η_θ: the learning rate of SLM $g_{\phi_k+\theta_k}$.

Output: $f_{\psi+\omega}$, $g_{\phi_1+\theta_1}$,$g_{\phi_2+\theta_2}$,...,$g_{\phi_K+\theta_K}$.

// **Server side:**
for t in communication round T **do**
  $\{\mathcal{S}_k\}_{k=1}^K \leftarrow$ **ClientUpdate1**(t).
  Token Alignment from SLMs to LLM.
  $\tilde{\mathcal{S}}_0 \leftarrow$ **DualMinCE**$(\mathcal{D}_p, f_{\psi+\omega}, \{\mathcal{S}_k\}_{k=1}^K)$.
  // knowledge transfer based on $\mathcal{D}_p$ and $\tilde{\mathcal{S}}_0$.
  for each epoch $r \in [R]$ **do**
    $\omega^{t,r+1} \leftarrow \omega^{t,r} - \eta_\omega \bigtriangledown \mathcal{L}_2$.
  end for
  $\omega^{t+1} = \omega^{t,R}$.
  Compute $\mathcal{S}_0 = \{l_0^i, p_0^i\}_{i=1}^N$ based on $\mathcal{D}_p$.
  ClientUpdate2$(t, \mathcal{S}_0)$.
end for

ClientUpdate1(t)**:**
for each client k (in parallel) **do**
  // local fine-tuning based on $\mathcal{D}_k$.
  for each local epoch $e \in [E]$ **do**
    $\theta_k^{t,e+1} \leftarrow \theta_k^{t,e} - \eta_\theta \bigtriangledown \ell_{\text{TA}}$.
  end for
  Compute $\mathcal{S}_k = \{l_k^i, p_k^i\}_{i=1}^N$ based on $\mathcal{D}_p$.
end for
Upload $\{\mathcal{S}_k\}_{k=1}^K$ to the server

ClientUpdate2$(t, \mathcal{S}_0)$**:**
for each client k (in parallel) **do**
  Token Alignment from LLM to SLMs.
  $\tilde{\mathcal{S}}_k \leftarrow$ **DualMinCE**$(\mathcal{D}_p, g_{\phi_k+\theta_k}, \mathcal{S}_0)$.
  // knowledge transfer based on $\mathcal{D}_p$ and $\tilde{\mathcal{S}}_k$.
  for each local epoch $e \in [E, 2E]$ **do**
    $\theta_k^{t,e+1} \leftarrow \theta_k^{t,e} - \eta_\theta \bigtriangledown \mathcal{L}_3$.
  end for
  $\theta_k^{t+1} = \theta_k^{t,2E}$.
end for

Algorithm 2 DualMinCE

Input:

$\mathcal{D}_p$: the public dataset;
h: either the SLM $g_{\phi_k+\theta_k}$ of client k or the LLM $f_{\psi+\omega}$ of the server;
$\mathcal{S}_k = \{(l_k^i, p_k^i)\}_{i=1}^N, k = 0$ or $\lceil K \rceil$: loss-logit pairs passed from either the server or clients.

Output: $\mathcal{S}$.

$\tilde{\mathcal{S}} \leftarrow \{\}$ $\triangleright$ initialize an empty set of selective knowledge.
for each x^i in $\mathcal{D}_p$ **do**
  $l_{\text{local}}^i \leftarrow h(x^i)$
  $$k^* = \begin{cases} \arg\min_k (l_k^i), & \text{if } k = \lceil K \rceil \\ 0, & \text{if } k = 0 \end{cases}$$
  $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} + (x^i, p_{k^*}^i)$ if $l_{k^*}^i < l_{\text{local}}^i$
end for

Bloom and LLaMa. Consider the sentence, "we utilize the dynamic programming approach to align tokens" as an example. Utilizing the Bloom tokenizer would segment it into the following tokens: ['we', 'utilize', 'the,' 'dynamic,' 'programming,' 'approach,' 'to,' 'align,' 'tokens']. However, if the LLaMa tokenizer were used, the segmentation would be: ['we', 'util', 'ize', 'the', 'dynamic', 'programming', 'approach', 'to', 'align', 'tokens'].

To tackle this issue, we adopt dynamic programming techniques to promote robust alignment, as evidenced in studies (Wan et al., 2024; Fu et al., 2023). Utilizing LLaMa2 and Bloom as illustrative examples, we establish an optimized vocabulary mapping table based on minimum edit distance (MinED). This mapping table identifies the closest Bloom token for each LLaMa2 token (e.g., 'utilize' for 'util'). We then tokenize a sentence using both tokenizers and apply a dynamic programming algorithm to determine the optimal matching path. When multiple LLaMa2 tokens align to a single Bloom token (e.g., 'util' and 'ize' aligning to 'utilize'), we handle them according to the mapping table. Please refer to Appendix B for more details.

In FedMKT, a bidirectional token alignment process occurs before knowledge transfer between LLMs and SLMs. One the one hand, when clients transfer knowledge from their SLMs to the server's LLM, the server aligns SLM tokens to LLM tokens. On the other hand, when the server transfers knowledge from its LLM back to clients' SLMs, each client aligns LLM tokens to its SLM tokens.

3.3 Selective Mutual Knowledge Transfer Between LLM and SLMs

To transfer knowledge between the server and clients efficiently, we leverage LoRA to fine-tune the server's LLM and clients' SLMs. Specifically, each client k inserts a small low-rank adapter parameterized by θ_k into its local SLM. We denote client k local SLM with the added θ_k as $g_{\phi_k+\theta_k}$. Likewise, the server inserts a small low-rank adapter parameterized by ω into its LLM f_ψ. We denote the server's LLM f_ψ with the added ω as $f_{\psi+\omega}$. During the whole federated learning training process, $\theta_k, k=1,...,K$ and ω are trained, while $\phi_k, k=1,...,K$ and ψ are frozen.

Before transferring knowledge to the server, each client k trains its LoRA adapter θ_k using its private dataset $\mathcal{D}_k$. Consequently, Eq.(1) can be reformulated as follows:

$$\mathcal{L}_1(\theta_1,\theta_2,...,\theta_K;\{\mathcal{D}_k\}_{k=1}^K) = \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{(x,y)\sim\mathcal{D}_k}\ell_{\text{TA}}(g_{\phi_k+\theta_k}(x),y) \tag{4}$$

where ℓ_{TA} is the task loss for training θ_k of each client k. The original model parameter ϕ_k of client k's SLM is frozen during training.

Then, both the server and clients fine-tune their LoRA adapters based on a shared public dataset $\mathcal{D}_p$. We formulate the losses of fine-tuning $f_{\psi+\omega}$ and $g_{\phi_k+\theta_k}$ (denoted as $\mathcal{L}_{\text{FT}}^f$ and $\mathcal{L}_{\text{FT}}^g$) as follows:

$$\mathcal{L}_{\text{FT}}^f(\omega;\mathcal{D}_p) = \mathbb{E}_{(x,y)\sim\mathcal{D}_p}\ell_{\text{CE}}(f_{\psi+\omega}(x),y) \tag{5}$$

$$\mathcal{L}_{\text{FT}}^g(\theta_k;\mathcal{D}_p) = \mathbb{E}_{(x,y)\sim\mathcal{D}_p}\ell_{\text{CE}}(g_{\phi_k+\theta_k}(x),y) \tag{6}$$

where ℓ_{CE} is the cross-entropy loss; the model parameters ψ and ϕ_k are frozen during fine-tuning.

Next, the server and clients conduct selective knowledge transfer to each other. The motivation for applying selective knowledge transfer is that some clients' knowledge may adversely affect the performance of LLM on the server and vice versa in a heterogeneous environment. Therefore, it is critical to guarantee that the knowledge transferred between the server and clients is positive to the performance of LLM and SLMs. To this end, we propose a selective knowledge transfer strategy on both the server and client sides, termed *DualMinCE*.

DualMinCE aims to select knowledge that is positive to the performance of the server's LLM from clients and vice versa. Specifically, when knowledge needs to be transferred from SLMs to the LLM, each client k computes a knowledge set $\mathcal{S}_k=\{l_k^i,p_k^i\}_{i=1}^N$ consisting of loss-logit pairs through its local model based on the public dataset $\mathcal{D}_p$. Then, all K clients send their $\{\mathcal{S}_k\}_{k=1}^K$ to the server. By leveraging DualMinCE (see Algorithm 2 for detail), the server picks a logit p_{k*}^i with the smallest loss from $\{l_k^i,p_k^i\}_{k=1}^K$ and adds p_{k*}^i to a selective knowledge set $\tilde{\mathcal{S}}_0$ if the loss l_{k*}^i of p_{k*}^i is smaller than the loss l_{local}^i computed through the server's local LLM based on x^i for each x^i in $\mathcal{D}_p$.

Next, the server leverages the knowledge distillation loss, denoted as $\mathcal{L}_{\text{KD}}^f$, to fine-tune $f_{\psi+\omega}$:

$$\mathcal{L}_{\text{KD}}^f(\omega;\tilde{\mathcal{S}}_0) = \mathbb{E}_{(x,p)\sim\tilde{\mathcal{S}}_0}\ell_{\text{CE}}(f_{\psi+\omega}(x),p) \tag{7}$$

Likewise, each client k leverages DualMinCE to form its selective knowledge set $\tilde{\mathcal{S}}_k$ from the knowledge $\mathcal{S}_0$ sent from the server. Each client k leverages the following knowledge distillation loss to fine-tune its local model $g_{\phi_k+\theta_k}$:

$$\mathcal{L}_{\text{KD}}^g(\theta_k;\tilde{\mathcal{S}}_k) = \mathbb{E}_{(x,p)\sim\tilde{\mathcal{S}}_k}\ell_{\text{CE}}(g_{\phi_k+\theta_k}(x),p) \tag{8}$$

Combining Eq.(5) and Eq.(7), we reformulate the knowledge transfer from SLMs to LLM conducted on the server to enhance LLM as follows:

$$\mathcal{L}_2 = \lambda\mathcal{L}_{\text{FT}}^f + (1-\lambda)\mathcal{L}_{\text{KD}}^f \tag{9}$$

Combining Eq.(6) and Eq.(8), we reformulate the knowledge transfer from LLM to SLMs conducted on the clients to enhance SLMs as follows:

$$\mathcal{L}_3 = \frac{1}{K}\sum_{k=1}^{K}(\lambda\mathcal{L}_{\text{FT}}^g + (1-\lambda)\mathcal{L}_{\text{KD}}^g) \tag{10}$$

where λ is the hyperparameter that controls the weight of mutual knowledge transfer.

4 Experiments

4.1 Setup

We set up a federated learning scenario involving four clients and one server to evaluate the FedMKT using various publicly available LLMs and SLMs.

Models. We evaluate FedMKT on one LLM (LLaMa2-7B (Touvron et al., 2023)) in the server, four SLMs in the clients including GPT-2-xlarge (1.5B) (Radford et al., 2019), OPT-1.3B (Zhang et al., 2022a), Bloom-1.1B (Scao et al., 2022) and LLaMa2-1.3B (Xia et al., 2023). In our experiments, we evaluate our framework in three distinct scenarios: **Heterogeneous**, **Homogeneous** and **One-to-One**. Table 1 details the setup for the LLM and SLMs in different settings.

Datasets. We evaluate FedMKT comprehensively on 6 QA datasets and 2 instruction-following datasets. Specifically, for QA tasks, we use RTE (Wang et al., 2019), WTC (Wang et al., 2019), BoolQ (Clark et al., 2019), CommonsenseQA(CQA) (Talmor et al., 2018), ARC-E

Setting	Server	Client-1	Client-2	Client-3	Client-4
Heterogeneous	LLaMa2- 7B	GPT-2-xlarge(1.5B)	OPT-1.3B	Bloom-1.1B	LLaMa2-1.3B
Homogeneous	LLaMa2- 7B	LLaMa2-1.3B	LLaMa2-1.3B	LLaMa2-1.3B	LLaMa2-1.3B
Homogeneous	LLaMa2- 7B	OPT-1.3B	OPT-1.3B	OPT-1.3B	OPT-1.3B
One-to-One	LLaMa2- 7B	-	-	-	LLaMa2-1.3B
One-to-One	LLaMa2- 7B	-	OPT-1.3B	-	-

Table 1: The five different settings we utilize to evaluate FedMKT.

Task	Method	GPT-2-xlarge	OPT-1.3B	Bloom-1.1B	LLaMa2-1.3B	LLaMa2-7B
RTE	Centralized	-	-	-	-	85.9
	Zero-Shot	52.4	52.7	52.7	49.8	63.2
	Standalone	65.7	62.5	58.1	55.6	-
	FedMKT	**70.4**	**65.7**	**61.7**	**58.8**	**82.3**
WIC	Centralized	-	-	-	-	70.4
	Zero-Shot	49.8	50.8	50	50	50.3
	Standalone	59.3	52.2	59.1	50.6	-
	FedMKT	**63.2**	**62.2**	**61.1**	**51.9**	**61.3**
BoolQ	Centralized	-	-	-	-	87.6
	Zero-Shot	61.3	58.4	59.0	61.0	70.1
	Standalone	71.1	74.1	69.7	69.9	-
	FedMKT	**75.1**	**76.8**	**71.4**	**75.1**	**85.0**
CQA	Centralized	-	-	-	-	69.5
	Zero-Shot	36.7	41.9	33.8	30.1	39.5
	Standalone	56.0	58.6	44.7	56.7	-
	FedMKT	**58.3**	**60.5**	**50.8**	**57.0**	**71.8**
ARC-E	Centralized	-	-	-	-	76.9
	Zero-Shot	58.3	57.0	51.5	53.1	69.3
	Standalone	59.3	57.9	56.9	60.4	-
	FedMKT	**59.8**	**59.6**	**57.5**	**60.8**	**76.1**
ARC-C	Centralized	-	-	-	-	48.9
	Zero-Shot	25.0	23.4	23.6	26.7	40.0
	Standalone	28.2	28.4	24.9	28.5	-
	FedMKT	**30.2**	**29.4**	**26.6**	**30.0**	**44.7**
S-NI	Centralized	-	-	-	-	49.3
	Zero-Shot	5.0	5.2	5.1	5.8	12.0
	Standalone	27.9	26.1	10.6	33.4	-
	FedMKT	**34.2**	**36.0**	**15.1**	**37.3**	**41.4**
DialogSum	Centralized	-	-	-	-	27.7
	Zero-Shot	5.4	6.4	4.9	5.7	8.5
	Standalone	22.3	19.8	13.2	21.4	-
	FedMKT	**23.2**	**20.9**	**14.9**	**21.6**	**24.2**

Table 2: Method Performance Comparison in **the Heterogeneous setting**. We evaluate FedMKT with 8 different tasks. In all the 8 tasks, the server is deployed with a LLaMa2-7B model, and the 4 clients are deployed with a GPT-2-xlarge, a OPT-1.3B, a Bloom-1.1B, and a LLaMa2-1.3B, respectively. The '-' indicates a method does not apply to the corresponding participant (either the server or the client).

(Clark et al., 2018), ARC-C (Clark et al., 2018) to evaluate FedMKT. As for instruction-following tasks, we evaluate FedMKT on S-NI (Wang et al., 2022), DialogSum (Chen et al., 2021).

Baselines. We conduct a comparative analysis of FedMKT against the following baselines:

- Centralized, in which the server's LLM is fine-tuned locally using the datasets combining private datasets of involved clients and the public dataset. In the One-to-One setting, the data of one client and the public data are used to fine-tune the server's LLM, whereas in other settings, the data of all four clients and the public data are used to fine-tune the LLM;
- Zero-Shot, representing the zero-shot capabilities of LLM or SLMs (without fine-tuning);
- Standalone, in which each client independently fine-tunes its local SLM using its private dataset;
- FedAvg, representing the standard federated averaging algorithm. FedAvg is only used in homogeneous settings because it requires all clients' models have the same architecture.
- LLM2SLM, representing FedMKT involving one server with an LLM and one client with an SLM. The LLM is not updated and is used to transfer knowledge to SLM. LLM2SLM is only used in the One-to-One setting.

Evaluation Metrics. For the QA datasets, we primarily use **Accuracy** as the evaluation metric, whereas for the instruction-following datasets, we primarily rely on **Rouge-L**.

4.2 Evaluation on Heterogeneous Setting

In the Heterogeneous setting, the server is deployed with a LLaMa2-7B model, and the 4 clients are deployed with a GPT-2-xlarge, a OPT-1.3B, a Bloom-1.1B, and a LLaMa2-1.3B, respectively. Table 2 reports the performance comparisons of FedMKT against baselines on 8 tasks.

Tables 2 show that FedMKT performs superior over Zero-Shot and Standalone on all clients' SLMs. Take the RTE dataset as an example, FedMKT outperforms Zero-Shot by 34% and Standalone by 7% on the GPT-2-xlarge SLM; FedMKT surpasses Zero-Shot by 25% and Standalone by 5% on the OPT-1.3B SLM; FedMKT-SLM achieves a 17% improvement over Zero-Shot and a 6% improvement over Standalone on the Bloom-1.1B SLM; FedMKT-SLM outperforms Zero-Shot by 18% and Standalone by 6% on the LLaMa2-1.3B SLM. These empirical results demonstrate that, by leveraging FedMKT, SLMs are able to effectively leverage the knowledge transferred from the LLM, leading to enhanced model capabilities.

Table 2 also shows that FedMKT outperforms Zero-Shot and Centralized on the LLaMa2-7B of the server. For instance, on the RTE QA dataset, FedMKT outperforms Zero-Shot by 30% and achieves a performance level that is nearly on par with Centralized, reaching approximately 96% of its fine-tuning performance. This significant achievement signifies that FedMKT effectively facilitates the acquisition of knowledge from all clients by the server.

4.3 Evaluation on Homogeneous Setting

We conduct experiments with two Homogeneous settings, as shown in Table 1. The first setting (denoted as S1) involves one server-side LLaMa2-7B and four client-side LLaMa2-1.3B. The second setting (denoted as S2) involves one server-side LLaMa2-7B and four client-side OPT-1.3B.

Table 3 reports the performance comparisons of FedMKT against baselines in the two Homogeneous settings. The top sub-table and the bottom sub-table compare the performance of FedMKT against baselines on the server's LLM and clients' SLMs, respectively.

The top sub-table of Table 3 shows that FedMKT significantly outperforms Zero-Shot on the server's LLM (i.e., LLaMa2-7B) in the two Homogeneous settings. It also shows that FedMKT achieves comparable performance of the Centralized scenario, in which the server' LLM is fine-tuned using all clients' data and the public data combined.

The bottom sub-table of Table 3 shows that FedMTK performs better than the Zero-Shot, Standalone, and FedAvg due to the assistance of the server's LLM. For example, in the CQA dataset, FedMKT outperforms FedAvg by 4% on the LLaMa2-1.3 SLM and by 5% on the OPT-1.3B SLM, respectively.

4.4 Evaluation on One-to-One Setting

We evaluate FedMKT using two One-to-One settings. The first setting (denoted as S1) involves one server-side LLaMa2-7B LLM and one client-side LLaMa2-1.3B SLM, while the second setting (denoted as S2) involves one server-side LLaMa2-7B LLM and one client-side OPT-1.3B SLM.

Table 4 reports the performance comparisons of FedMKT against baselines in the two One-to-One settings. The top and bottom sub-tables compare the performance of FedMKT against baselines on the server's LLM and clients' SLMs, respectively.

Task	Method	S1: Server **LLaMa2-7B**	S2: Server **LLaMa2-7B**
CQA	Zero-Shot	39.5	39.5
	Centralized	69.5	69.5
	FedMKT	**68.8**	**71.3**
ARC-C	Zero-Shot	40.0	40.0
	Centralized	49.4	49.4
	FedMKT	**46.2**	**46.2**
ARC-E	Zero-Shot	69.3	69.3
	Centralized	75.5	75.5
	FedMKT	**74.9**	**74.8**

Task	Method	S1: Clients **LLaMa2-1.3B**	S2: Clients **OPT-1.3B**
CQA	Zero-Shot	30.1	41.9
	Standalone	56.4	58.1
	FedAvg	56.4	58.6
	FedMKT	**58.6**	**61.5**
ARC-C	Zero-Shot	26.7	23.4
	Standalone	30.4	28.5
	FedAvg	29.7	28.6
	FedMKT	**31.7**	**29.9**
ARC-E	Zero-Shot	53.1	57.0
	Standalone	60.3	57.9
	FedAvg	60.6	58.8
	FedMKT	**61.7**	**60.1**

Table 3: Method Performance Comparison in **Homogeneous settings**. We evaluate FedMKT using two homogeneous settings. The first setting (denoted as S1) involves one server-side LLaMa2-7B LLM and four client-side LLaMa2-1.3B SLMs, while the second setting (denoted as S2) involves one server-side LLaMa2-7B LLM and four client-side OPT-1.3B SLMs. *The top and bottom sub-tables compare the performance of FedMKT against baselines on the server's LLM and clients' SLMs, respectively*. The results reported in the bottom sub-table are the average of all clients.

Task	Method	S1: Server **LLaMa2-7B**	S2: Server **LLaMa2-7B**
CQA	Zero-Shot	39.5	39.5
	Centralized	69.0	68.3
	FedMKT	**69.0**	**71.0**
ARC-C	Zero-Shot	40.0	40.0
	Centralized	45.9	48.6
	FedMKT	**45.9**	**45.8**
ARC-E	Zero-Shot	69.3	69.3
	Centralized	74.4	73.6
	FedMKT	**74.8**	**74.8**

Task	Method	S1: Clients **LLaMa2-1.3B**	S2: Clients **OPT-1.3B**
CQA	Zero-Shot	30.1	41.9
	Standalone	56.7	58.6
	LLM2SLM	56.76	59.1
	FedMKT	**56.84**	**60.7**
ARC-C	Zero-Shot	26.7	23.4
	Standalone	30.3	28.8
	LLM2SLM	30.1	29.6
	FedMKT	**30.8**	**30.4**
ARC-E	Zero-Shot	53.1	57.0
	Standalone	57.0	57.9
	LLM2SLM	60.7	58.4
	FedMKT	**60.8**	**58.5**

Table 4: Method Performance Comparison in **One-to-One settings**. We evaluate FedMKT using two one-to-one settings. The first setting (denoted as S1) involves one server-side LLaMa2-7B LLM and one client-side LLaMa2-1.3B SLM, while the second setting (denoted as S2) involves one server-side LLaMa2-7B LLM and one client-side OPT-1.3B SLM. *The top and bottom sub-tables compare the performance of FedMKT against baselines on the server's LLM and a client's SLM, respectively*.

The top sub-table of Table 4 shows that FedMKT notably surpasses Zero-Shot and rivals Centralized on the performance of the server's LLM. The bottom sub-table of Table 4 shows that FedMTK achieves superior SLM performance over Zero-Shot, Standalone, and LLM2SLM due to the assistance of LLM. These empirical results demonstrate the effectiveness of FedMKT in transferring knowledge between the LLM and SLMs.

5 Conclusions

In this study, we have presented FedMKT, a parameter-efficient federated mutual knowledge transfer framework tailored for large and small language models. FedMKT bridges the gap between the server-side LLM and clients' SLM, enabling selective mutual knowledge transfer while preserving data privacy. Through extensive experiments across three distinct scenarios, we have demonstrated that FedMKT simultaneously boosts the performance of both LLMs and SLMs.

Limitations

In this study, we transfer knowledge between the server and clients using logits of a public dataset, motivated by efficiency and privacy considerations. Although empirical evidence suggests that sharing logits of public datasets between the server and clients is more privacy-preserving than sharing model gradients or parameters (Li and Wang, 2019; Cho et al., 2022), there is no theoretical guarantee that this approach does not compromise the privacy of clients' sensitive data. This issue warrants further investigation. Furthermore, our study is limited by computational and storage constraints, which preclude the exploration of larger language models. This highlights the inherent trade-off between utility and efficiency. Our future research aims to investigate and optimize this trade-off.

References

Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. 2022. Autofednlp: An efficient fednlp framework. *arXiv preprint arXiv:2205.10162*.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. Dialogsum: A real-life scenario dialogue summarization dataset. *arXiv preprint arXiv:2105.06762*.

Yae Jee Cho, Andre Manoel, Gauri Joshi, Robert Sim, and Dimitrios Dimitriadis. 2022. Heterogeneous ensemble knowledge transfer for training large models in federated learning. *arXiv preprint arXiv:2204.12703*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Tao Fan, Yan Kang, Guoqiang Ma, Weijing Chen, Wenbin Wei, Lixin Fan, and Qiang Yang. 2023. Fate-llm: A industrial grade federated learning framework for large language models. *arXiv preprint arXiv:2310.10049*.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

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*.

Jaehee Jang, Heoneok Ha, Dahuin Jung, and Sungroh Yoon. 2022. Fedclassavg: Local representation learning for personalized federated learning on heterogeneous neural networks. In *Proceedings of the 51st International Conference on Parallel Processing*, pages 1–10.

Yan Kang, Tao Fan, Hanlin Gu, Lixin Fan, and Qiang Yang. 2023. Grounding foundation models through federated transfer learning: A general framework. *arXiv preprint arXiv:2311.17431*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Daliang Li and Junpu Wang. 2019. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Chang Liu, Yuwen Yang, Xun Cai, Yue Ding, and Hongtao Lu. 2022. Completely heterogeneous federated learning. *arXiv preprint arXiv:2210.15865*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan.

2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

OpenAI. 2022. Chatgpt.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. *arXiv preprint arXiv:2401.10491*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint 1905.00537*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*, 2.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Ruihong Yang, Junchao Tian, and Yu Zhang. 2021. Regularized mutual learning for personalized federated learning. In *Asian Conference on Machine Learning*, pages 1521–1536. PMLR.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022a. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. 2018. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4320–4328.

Zhuo Zhang, Yuanhang Yang, Yong Dai, Lizhen Qu, and Zenglin Xu. 2022b. When federated learning meets pre-trained language models' parameter-efficient tuning methods. *arXiv preprint arXiv:2212.10025*.

Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. 2022. Reduce communication costs and preserve privacy: Prompt tuning method in federated learning. *arXiv preprint arXiv:2208.12268*.

A FedMKT Workflow

The workflow of FedMKT is outlined as follows:

1. In the t-th communication round, the K clients train their respective LoRA adapters using their private data. This step allows the clients to adapt their models to their specific data distributions.

2. After local training, each client k computes a knowledge set $\mathcal{S}_k = \{l_k^i, p_k^i\}_{i=1}^N$ consisting of loss-logit pairs through its local model based on the public dataset.

3. Each client k upload $\mathcal{S}_k$ to the server.

4. On the server side, token alignment is performed from the SLMs to the LLM, guaranteeing compatibility between the SLMs and the LLM.

5. On the server side, knowledge is selected from the SLMs to the LLM according to Algorithm 2.

6. On the server side, knowledge is transferred from the SLMs to the LLM based on the selected knowledge.

7. Once the knowledge transfer from SLMs to LLM is completed on the server, the server then computes a knowledge set $\mathcal{S}_0 = \{l_0^i, p_0^i\}_{i=1}^N$ consisting of loss-logit pairs through LLM based on the public dataset.

8. The server disseminates $\mathcal{S}_0$ to all the clients.

9. On the client side, the token alignment flow reverses, and token alignment is performed from the LLM to SLMs.

10. On the client side, knowledge is selected from the LLM to each client SLM according to Algorithm 2.

11. On the client side, knowledge is transferred from the LLM to each client SLM based on the selected knowledge.

B Implementation Details of Token Alignment

In our work, we engage in a bidirectional token alignment procedure, encompassing the alignment of SLM tokens with their corresponding LLM tokens, and vice versa. Both alignments adhere to a similar methodology. Presently, we shall elaborate on the process of aligning LLM tokens with their matching SLM tokens. To map the predicted token logits from the LLaMa2-7B (LLM) model to the Bloom-1.1B (SLM) model, several steps must be undertaken. The detailed process is as follows:

1. Building an Optimal Vocabulary Mapping Table:
   (a) For each token in the LLaMa2 vocabulary, iterate through the Bloom vocabulary.
   (b) Use edit distance as a similarity measure to find the closest token in the Bloom vocabulary to the token in the LLaMa2 vocabulary.
   (c) If there are multiple token with the same minimum edit distance, choose the one with the lexicographically smallest order.
   (d) Save this mapping relationship in the optimal vocabulary mapping table.

2. Tokenization and Alignment:
   (a) Tokenize the sentence "we utilize the dynamic programming approach to align tokens" using both the LLaMa2 and Bloom tokenizers.
   (b) To align the two tokenization results and determine the optimal matching path, we utilize a dynamic programming algorithm. As an illustration, consider the tokenization outputs from LLaMa2 and Bloom. LLaMa2's tokenization yields: ['we', 'util', 'ize', 'the', 'dynamic', 'programming', 'approach', 'to', 'align', 'tokens']. In contrast, Bloom's tokenization produces: ['we', 'utilize', 'the', 'dynamic', 'programming', 'approach', 'to', 'align', 'tokens']. In this instance, seven terms from LLaMa2 align perfectly with those from Bloom, such as "we" and "dynamic". Notably, the LLaMa2 tokens 'util' and 'ize' collectively map to the single Bloom token 'utilize'. In scenarios where multiple tokens align to one, like the 2-to-1 case of 'util' and 'ize' mapping to 'utilize', we consider 'utilize' as a match for 'util' based on an optimal vocabulary mapping.

3. Logits Mapping:
   (a) Iterate through each token t_t in the Bloom tokenization result.
   (b) For each t_t, check if it uniquely matches a token t_s in the LLaMa2 tokenization result.
   (c) If t_t uniquely matches t_s, then for each token t_p in the Top-K predicted token of t_s from LLaMa2 and its corresponding logit $logit_p$: Find the position pos in the Bloom vocabulary that corresponds to t_p using the optimal vocabulary mapping table. If pos has not been assigned a value before, copy $logit_p$ to the corresponding position in the Bloom logits distribution matrix $logit_t$.
   (d) If t_t does not have a unique match, generate one-hot logits for t_t.

4. Processing the Results:
   (a) Ultimately, each token t_t in Bloom will have a corresponding logits distribution matrix $logit_t$.
   (b) These logits can be directly used for subsequent training in the Bloom model.

C Computation and Communication Complexity

One of the key advantages of FedMKT is its computational efficiency. By leveraging PEFT, the framework significantly reduces the number of parameters that need to be updated during fine-tuning. For instance, it consumes just 0.12% of the computational cost associated with fine-tuning all parameters in OPT-1.3B when using FedMKT. This leads to faster training times and reduced computational requirements, making it more feasible to fine-tune LLM and SLMs in a federated learning setting.

In terms of communication complexity, FedMKT minimizes the amount of data exchanged between clients and the server. Instead of transmitting entire models(For example, OPT-1.3B is about 1.3B floating-point numbers), clients only share the output logits and corresponding cross-entropy losses of the public dataset with the server. Suppose there are $N = 1000$ public text samples with a text sequence length of $S = 512$ and a top token size of $K = 16$. The communication cost, denoted as $Cost_{com}$, would be calculated as follows: $Cost_{com} = N * S * K = 1000 * 512 * 16 = 8M$ floating-point numbers. This approach reduces communication overhead, allowing for more efficient data transmission and enhancing scalability in federated learning scenarios.

D More on Experimental Details

D.1 Hyperparameter Settings

LoRA Parameters. We utilized the PEFT(Mangrulkar et al., 2022) library with the following configurations: r=8, lora_alpha=16, lora_dropout=0.05.

Common Parameters for LLM and SLMs. We set batch_size=4, used the AdamW optimizer with adam_beta1=0.9 and adam_beta2=0.95. The warmup_ratio was set to 0.008, the weight_decay was 0.1, max_grad_norm was 1.0. The λ was 0.9. The number of training rounds for all data is within 10 and the number of training rounds for different datasets may be different.

LLM Parameters. During distillation, the local epoch R was set to 1. The learning rates η_ω were specified as 3e-5 for the datasets RTE/WIC/BoolQ/CQA/ARC-C/DialogSum/S-NI, and 2e-5 for ARC-E.

SLM Parameters. During training for the four clients, the local epoch E was set to 1. The learning rates η_θ were as follows: for "OPT-1.3b", η_θ=3e-5; for "GPT-2-xlarge", η_θ=3e-4; for "Bloom-1b1", η_θ=3e-5; and for "LLaMa-2-1.3b", the same learning rates as for the LLM were used.

D.2 Data Splitting

For the datasets RTE/WIC/BoolQ/CQA/ARC-E/ARC-C/DialogSum, we randomly split the training data into five equal parts, with one part serving as the public dataset and the remaining four parts as private dataset for the four clients. All these datasets(including train, validate, test) were downloaded from HuggingFace(Lhoest et al., 2021). For the S-NI dataset, we first processed the data using minillm(Gu et al., 2023) to retain samples with an output length greater than or equal to 11. From this processed data, we randomly selected 300 samples as the evaluation dataset. The remaining data was then split into five equal parts, with one part serving as the public dataset and the other four parts as private data for the four clients.

D.3 Dataset Licenses

For the datasets RTE/WIC/BoolQ/CQA/ARC-E/ARC-C/DialogSum were downloaded from HuggingFace(Lhoest et al., 2021) and under Apache License, Version 2.0. For the S-NI dataset, it was from minillm(Gu et al., 2023) and under MIT License.

D.4 Machine Configuration

The experiments were conducted on machines equipped with either 4 Nvidia V100 32G or 8 Nvidia V100 32G GPUs.