Federated Instruction Tuning of LLMs with Domain Coverage Augmentation

Anonymous authors
Paper under double-blind review

Abstract

Federated Domain-specific Instruction Tuning (FedDIT) utilizes limited cross-client private data together with server-side public data for instruction augmentation, ultimately boosting model performance within specific domains. To date, the factors affecting FedDIT remain unclear, and existing instruction augmentation methods primarily focus on the centralized setting without considering distributed environments. Our experiments reveal that the cross-client domain coverage, rather than data heterogeneity, drives model performance in FedDIT. In response, we propose FedDCA, which optimizes domain coverage through greedy client center selection and retrieval-based augmentation. For client-side computational efficiency and system scalability, FedDCA*, the variant of FedDCA, utilizes heterogeneous encoders with server-side feature alignment. Extensive experiments across four distinct domains (code, medical, financial, and mathematical) substantiate the effectiveness of both methods. Additionally, we investigate privacy preservation against memory extraction attacks utilizing various amounts of public data. Results show that there is no significant correlation between the volume of public data and the privacy-preserving capability. However, as the fine-tuning rounds increase, the risk of privacy leakage reduces or converges.

1 Introduction

Table 1: Performance(%) of different augmentation settings on each domain, conducted via FedAvg protocol with 10 clients. Specifically, the zero-shot directly inferences without any fine-tuning, while the base data method utilizes only the client's local data for FedDIT. Additionally, we compare FedDCA with other two augmentation strategies: random sampling and direct retrieval (described in Appendix A.3), respectively.

Metric	Zero-shot	Base Data	Random Sampling	Direct Retrieval	FedDCA
HumanEval/Pass@1	29.88	**39.03**	32.93	34.14	36.58
MMLU-Med/Acc	70.60	68.40	71.30	72.20	**74.50**
FPB/Acc	55.94	58.25	64.19	66.31	**67.24**
FiQA/Acc	18.54	14.18	13.09	19.11	**35.27**
TFNS/Acc	59.21	66.62	65.53	67.62	**73.32**
GSM8K/Exact Match	23.27	47.46	47.38	50.87	**52.46**
Avg.	42.91	48.99	49.07	51.71	**56.56**

Recently, federated instruction tuning (FedIT) has gained attention as a novel approach that leverages the principles of federated learning (FL) to facilitate collaborative training of large language models (LLM) in distributed environments while maintaining the confidentiality of private data (McMahan et al., 2017; Ye et al., 2024b). This methodology allows for the exchange of model parameters among distributed data holders, thereby achieving a careful balance between privacy preservation and efficient model optimization. Despite the establishment of various FedIT frameworks (Ye et al., 2024b; Kuang et al., 2023; Zhang et al., 2023c), existing literature has not adequately addressed the practical challenges that Federated Domain-specific Instruction Tuning (FedDIT) may encounter in real-world applications. For instance, FedIT generally necessitates a sufficient amount of instruction data for fine-tuning, which is often a shortage in domain-specific fine-tuning contexts (Zhang et al., 2024b).

In this study, we investigate FedDIT, a novel approach within the FL paradigm aimed at boosting the performance of LLMs in specific domains. Unlike general FedIT, which seeks to enhance model effectiveness across diverse tasks without accounting for local data shortage, FedDIT encounters the unique challenge of clients possessing only a limited quantity of local domain-specific data. To overcome this, FedDIT leverages a server-hosted public dataset that spans multiple domains to enrich the local data through a specific instruction augmentation strategy. This strategic enrichment is crucial for achieving effective instruction tuning and needs to be meticulously designed to avoid performance degradation. Except for the code domain, which primarily adheres to a standardized paradigm, our results reveal that when clients rely solely on their local data for FedDIT, even the presence of high-quality in-domain local data can be insufficient due to limited scale, leading to a decline in performance, as reflected in the underlined values in Table 1. In summary, the goal of FedDIT is to develop a domain-specific LLM that employs collaborative training and instruction augmentation with public data while safeguarding client privacy, thereby ensuring the model's proficiency in executing tasks pertinent to its designated domain.

Additionally, the factors affecting FedDIT are still unclear. Compounding this uncertainty, introducing augmented instructions may further complicate results, making it difficult to ascertain effective improvement strategies. Shepherd (Zhang et al., 2023c) approaches this problem from the perspective of heterogeneity, constructing heterogeneity based on the topic of general instruction datasets. It demonstrates that, unlike the consensus of traditional FL, for general FedIT, heterogeneity has a positive effect. By aggregating diverse instructions from clients, the diversity increases, thereby enhancing the model's adaptability to various tasks. However, it just scratches the surface and does not explore issues in FedDIT.

Going one step further, we conduct experiments to unveil a significant finding (Appendix A.2): there is no linear correlation between the degree of non-independent and identically distributed (non-iid) and LLM's performance in the context of FedDIT. Inspired by Explore-Instruct (Wan et al., 2023), which shows the potential of domain coverage in domain-specific instruction tuning. The cross-client domain coverage metric is initially defined, followed by an investigation into its impact on FedDIT. Results demonstrate that domain coverage significantly influences model performance in the corresponding domain.

To maximize the cross-client domain coverage without compromising client data privacy, we propose a novel FedDIT algorithm, **Fed**erated Instruction Tuning of LLMs with **D**omain **C**overage **A**ugmentation, termed **FedDCA**. This algorithm employs a greedy client center selection process and implements instruction augmentation through dense retrieval on the server side. The fundamental idea of FedDCA is to select client centers to expand the diversity and coverage of augmented instruction datasets within a specific domain. By strategically optimizing domain coverage at each step, FedDCA efficiently constructs the augmented train set that enhances both the learning and generalization capabilities of the model, leading to superior performance on domain-specific tasks. Furthermore, to mitigate computational overhead on the client side and enhance the system scalability, we propose FedDCA*, which employs heterogeneous encoders of different sizes and capacities. To achieve feature alignment, we train a projector on the server side using public data and employ contrastive learning techniques.

We demonstrate the effectiveness of FedDCA through comprehensive experiments conducted across four domains: code, medical, financial, and mathematical. These are compared against a range of baselines, which can be categorized into unaugmented and augmented methods. In the unaugmented setting, our method is compared with `FedIT`, which includes four orthodox FL techniques: FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020), SCAFFOLD (Karimireddy et al., 2020), and FedAvgM (Hsu et al., 2019). In the augmented setting, we compare FedDCA against methods such as random sampling, direct retrieval, LESS (Xia et al., 2024), and Self-Instruction (Wang et al., 2022). Additionally, we present the performance outcomes of FedDCA when applied under various FL strategies. We also compare the computational efficiency on the client side between FedDCA and its variant, FedDCA*. For privacy analysis, experiments against memory extraction attacks are conducted to evaluate how different quantities of retrieved public data affect the privacy of client local data. Results indicate that while reliance on local data increases memorization of sensitive information, the risk of privacy leakage diminishes or converges in the augmented setting as the training rounds progress.

The main contribution is as follows:

- We reveal a critical finding: in the context of FedDIT, there exists no linear correlation between the degree of heterogeneity and model performance. Instead, cross-client domain coverage substantially impacts LLM's effectiveness, as elaborated in Appendix A.2.
- We propose a novel FedDIT algorithm (Section 4), termed FedDCA, aimed at maximizing cross-client domain coverage through greedy client center selection followed by retrieval-based instruction augmentation executed on the server. Additionally, we introduce FedDCA* to further lessen the client-side computational overhead while enhancing the system scalability. This variant utilizes a heterogeneous encoder structure, paired with a projector on the server side for feature alignment.
- Through extensive experiments (Section 5), we demonstrate the effectiveness of FedDCA and FedDCA*. We also investigate privacy preservation against memory extraction attacks, conducting experiments based on various amounts of public data. Results suggest that the capacity for privacy preservation does not correlate significantly with the quantity of public data. In contrast, the risk of privacy leakage tends to decrease or converge as the fine-tuning rounds increase.

2 PRELIMINARIES

FedDIT aims to leverage cross-client private domain-specific instruction data and utilize the multi-domain public data on the server to achieve instruction augmentation, collaboratively enhancing the model's performance in specific domains. Consider N distributed clients, each with local private data D_k^l. The server maintains a public dataset D^p that encompasses multiple domains and is responsible for implementing data augmentation strategies and aggregating model parameters received from clients. The training process follows the standard FL protocol (McMahan et al., 2017). For computational efficiency, we adopt Low-Rank Adaption (LoRA) (Hu et al., 2022) as the fine-tuning method, which involves tuning additional parameters $\Delta\phi$ while keeping the pre-trained LLM's parameters ϕ frozen. In the initial training phase, the server dispatches ϕ and $\Delta\phi$ to each client for $\mathcal{R}$ training rounds. In the t-th round, the server sends the aggregated $\Delta\phi^t$ to clients. Clients use $\Delta\phi^t$ to update their local LoRA parameters $\Delta\phi_k^t$ and conduct instruction tuning based on augmented instruction datasets D_k. Subsequently, the clients return $\Delta\phi_k^{t+1}$ to the server. The server then aggregates $\{\Delta\phi_k^{t+1} \mid k = 1, \ldots, N\}$ to obtain $\Delta\phi^{t+1}$ for the next round.

3 PROBLEM FORMULATION

For better understanding, we list the frequently used notation in Appendix A.11. The objective of FedDIT is to enhance the domain-specific performance of LLMs through FL without sharing private data (Ye et al., 2024b; Zhang et al., 2024b). Under the FL framework, suppose we have N clients, where each client c_k has a local dataset D_k^l with its size N_k^l, and an augmented dataset D_k^g from the server public dataset D^p, respectively. Due to constraints such as memory, computational overhead, and maximum tolerated training time, client c_k can accept at most N_k^p public instructions. Denote the augmentation strategy as Λ, through which the server performs instruction augmentation on the public dataset. If Λ is `null`, it indicates that clients conduct FedDIT solely based on their local private data, which may lead to performance degradation. Conversely, Λ may be classified into two categories: (1) focusing exclusively on the client's own local data distribution or (2) considering the cross-client data distribution. In conclusion, the global objective of FedDIT is defined as follows:

$$\arg\min_{\Delta\phi} \left\{ F(\phi, \Delta\phi) \triangleq \sum_{k=1}^{N} p_k \left((1-\alpha_k) F_k\left(\phi, \Delta\phi_k; D_k^l\right) + \alpha_k F_k\left(\phi, \Delta\phi_k; D_k^g\right) \right) \right\}, \tag{1}$$

where the first term and second term denote the accumulated fine-tuning loss computed on the client c_k's local instructions D_k^l and the augmented public instructions D_k^g from the server, respectively. p_k is the weight of the k-th client, which is determined by the ratio of k-th client's data size to all clients' total data size. α_k is the ratio of the public data amount of its augmented instruction dataset D_k, which is computed as $\frac{N_k^p}{N_k^l+N_k^p}$. Eq.1 minimizes the summed empirical loss across clients' augmented instructions to pursue the in-domain utility of the obtained global model. Denote P as the model parameters and $\mathcal{D}$ as a specific dataset, then the client c_k's empirical loss $F_k(\phi, \Delta\phi_k; \mathcal{D})$ base on $\mathcal{D}$ is calculated as:

$$F_k(\phi, \Delta\phi_k; \mathcal{D}) \triangleq \frac{1}{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} l(P_{\phi+\Delta\phi_k}; x_j), \tag{2}$$

where $x_j \in D, \forall j \in \{1, 2, \ldots, |D|\}$. The instruction tuning loss $l(\cdot;\cdot)$ on a sample (x, y) is defined as $-\sum_{t=1}^{|y|} \log\left(w\left(y_t|x, y_{<t}\right)\right)$, where x is the formated instruction with Alpaca instruction template[1] and y is the coressponding response.

In contrast to the problem formulations in many prior works on FedIT, the primary distinction in the formulation of FedDIT lies in acknowledging the lack of in-domain instructions across clients. FedDIT establishes a public multi-domain dataset on the server for domain-specific instruction augmentation by a specific sampling strategy Λ. Therefore, each client only needs to hold a few high-quality domain-specific private data to collaborate with other clients to obtain a strong domain-specific LLM.

4 METHOD

We propose Federated Instruction Tuning of LLMs with Domain Coverage Augmentation (FedDCA), which enhances domain coverage to obtain a LLM that performs well on domain-specific tasks. FedDCA follows the standard FedIT protocol(Ye et al., 2024b) to perform federated instruction tuning. The novel design of FedDCA lies in the phase before training, which consists of two key modules: greedy client center selection and domain data retrieval, which are elaborated in the following; also see details in Algorithm 1 and Figure 1. For computational efficiency and system scalability, we introduce a variant of FedDCA with heterogeneous encoder setting, named FedDCA*. We first formulate the optimization problem to solve for FedDCA.

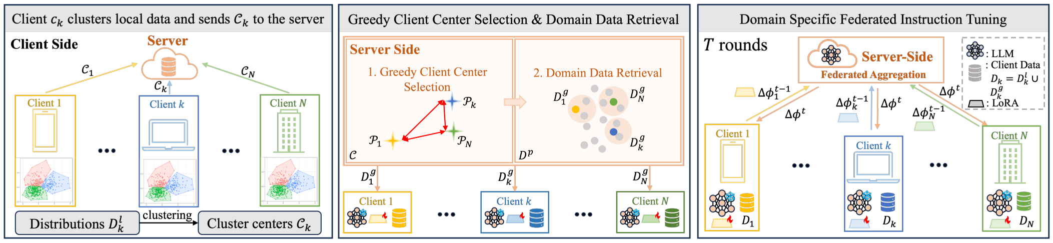


Figure 1: Overview of FedDCA, which consists of three stages: 1) The client c_k performs local instructions clustering and sends cluster centers $\mathcal{C}_k$ to the server. 2) The server first does greedy client center selection to maximize domain coverage and performs client-center-based domain data retrieval, then sends the augmented instructions D_k^g to the client c_k. 3) Clients fine-tune the LLM collaboratively, and the LoRA parameters $\Delta\phi$ are exchanged between clients and the server.

4.1 OPTIMIZATION PROBLEM

As domain coverage directly affects the in-domain performance of the LLM (Appendix A.2), FedDCA aims to maximize the domain coverage of the cross-client augmented data $\cup_{i=1}^{N} D_i$ respect to the in-domain data distribution D^d, which is defined in Eq. 5. However, as clients can not send the local data to the server for privacy, directly finding a cross-client dataset that maximizes the domain coverage in Eq. 5 is unrealistic. To find a proper client center set $\mathcal{P}$ that maximizes domain coverage and uses it to perform instruction retrieval on the public data is an approximation problem.

---

[1] `https://crfm.stanford.edu/2023/03/13/alpaca.html`

Algorithm 1 FedDCA: greedy client center selection & domain data retrieval

```
Parameters: Number of clusters $\xi$; Encoder model $w_{enc}$; Client local datasets $D = \{D_1, D_2, \ldots, D_N\}$;
Public dataset $D^p$; Similarity score threshold $\alpha$; Number of public data samples retrieved per client $N_k^p$;
Client centers $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$; Pre-encoded public instruction embeddings $\mathcal{E}$.
for $i \in \{1, 2, \ldots, N\}$ do
    $\mathcal{E}' \leftarrow \{w_{enc}(x) \mid x \in D^l_{i,instruction}\}$
    $\mathcal{C}_i, S_i \leftarrow$ k-means($\xi$).fit($\mathcal{E}'$)    ▷ Cluster local instructions, return cluster centers $\mathcal{C}_i$ and sizes $S_i$
end for
Send $\{\mathcal{C}_i, S_i \mid i \in \{1, 2, \ldots, N\}\}$ to the server for the greedy client center selection
$\mathcal{P} \leftarrow \{\mathcal{C}_{0,k} \mid k = \arg\max(S_0)\}$    ▷ Initialize the client center set
for $i \in \{1, 2, \ldots, N-1\}$ do    ▷ Greedy client center selection
    $\mathcal{S} \leftarrow \sum(\mathcal{C}_i \cdot \mathcal{P}^T, \text{dim} = -1)$    ▷ Compute summed similarity score of each cluster center
    $\mathcal{I} \leftarrow (N-i)\text{-}\arg\text{sort}(\mathcal{S})$    ▷ Filter $i$ cluster centers close to the client center set $\mathcal{P}$
    $\mathcal{I}' \leftarrow \arg\text{sort}(-S_i)$    ▷ Sort cluster center by cluster size in descending order
    $j \leftarrow$ first element of $\mathcal{I} \cap \mathcal{I}'$    ▷ Selected cluster center $\mathcal{C}_{i,j}$
    $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{C}_{i,j}$    ▷ Update the client center set
end for
for $i \in \{1, 2, \ldots, N\}$ do    ▷ Client center based domain data retrieval
    $\mathcal{S} \leftarrow \mathcal{P}_i \cdot \mathcal{E}^T$    ▷ Compute similarity score between $\mathcal{P}_i$ and $\mathcal{E}$
    $\mathcal{S}' \leftarrow \{s \mid s \in \mathcal{S}, s < \alpha\}$    ▷ Filter instructions with similarity score larger than $\alpha$
    $\mathcal{I} \leftarrow$ indices of $N_k^p$-top in $\mathcal{S}'$
    $D_i^g \leftarrow \{D_j^p \mid j \in \mathcal{I}\}$
    $D_i \leftarrow D_i^l \cup D_i^g$    ▷ Obtain the augmented instruction dataset $D_i$
end for
```

Suppose there exists a metric space $\mathcal{X}$, a set of cross-client local instruction embeddings $\mathcal{E} \in \mathcal{X}$, and a set of cluster centers $\mathcal{P} \in \mathcal{X}$. Each client c_k has maximum ξ clusters obtained by k-means algorithm (Wu, 2012). In the federated setting, communication cost is always a critical factor. Consequently, we formulate the optimization problem to include communication costs as follows:

$$\arg\min_{\mathcal{P}} \left\{ \sum_{i=1}^{N} |\mathcal{P}_i| + \sum_{d \in D^d} \left(\min_{p \in \mathcal{P}} sim(d, p) \right) \right\}, \tag{3}$$

s.t. $|\mathcal{P}_i| \leq \xi, \forall i \in \{1, 2, \ldots, N\}$. The second term is the domain coverage of the selected client center set $\mathcal{P}$, and $sim(\cdot, \cdot)$ is the cosine similarity function. However, this optimization problem is NP-hard. Specifically, to select N client center for retrieval, the time complexity is $\mathcal{O}\left(C_{\xi N}^{N} (\mathcal{N} N)\right)$, where $\mathcal{N}$ is the size of the public data D^p. As it is a factorial equation, the computational cost explodes with N. What's worse, usually, there are enormous amounts of public data on the server, which makes a huge $\mathcal{N}$. For computational efficiency, we propose a greedy algorithm to solve this problem in $\mathcal{O}(N(N\xi + \xi \log \xi))$, which is described below.

4.2 GREEDY CLIENT CENTER SELECTION & DOMAIN DATA RETRIEVAL

As described in Section 4.1, the goal of FedDCA is to find a proper client center set $\mathcal{P}$ that maximizes the domain coverage $d(D^d, \mathcal{P})$, as defined in Eq. 3. For computational efficiency, we propose a greedy algorithm as shown in Algorithm 1 and Figure 2 to solve this problem in polynomial time and obtain a sub-optimal solution. Given the cluster centers $\{\mathcal{C}_i, i = 1, 2, \ldots, N\}$ received from each client, which are obtained by clustering local instructions. FedDCA on the server consists of two main steps: greedy client center selection and client-center-based domain data retrieval.

Greedy client center selection. We will consider this problem from two aspects: 1) Select a client center that can represent the distribution of the local data. 2) To optimize the cross-client domain coverage, we filter client centers that are close to the previously selected client centers. First, we randomly choose a client and select the largest cluster center as its center and be the initial of the client center set $\mathcal{P}$. Then, we iteratively select the client center $\mathcal{P}_k$ of the top cluster size while avoiding selecting the cluster center close to the previously selected client center to maximize the domain coverage of $\mathcal{P}$. Specifically, for the i-th iteration, we filter i-top cluster centers that have the largest summed similarity with previously selected client centers and select the center of the rest largest cluster as the i-th client center. This procedure is repeated until we have selected N client centers.

Domain data retrieval. For each client center $\mathcal{P}_k$, the server performs dense retrieval (detailed in Appendix A.3) on public dataset D^p to get the top-N_k^p similar public instructions, then sends retrieved public datasets $\{D_1^g, \ldots, D_N^g\}$ to each clients. Specifically, to avoid the overlap between public data and local private data, we set a threshold α to filter the public instructions that have a similarity score larger than α with the client center.

In summary, FedDCA establishes a comprehensive training set that captures the essence and distribution of the domain by iteratively selecting client centers with unique coverage increments. This ensures that selected centers are representative and achieve broad domain coverage. Additionally, domain data retrieval exposes the model to a wider range of in-domain features, enhancing its understanding of domain-specific tasks. This not only enriches the training data but also reduces the risks of overfitting to the limited local data available to each client, ultimately improving performance across various tasks within the domain.

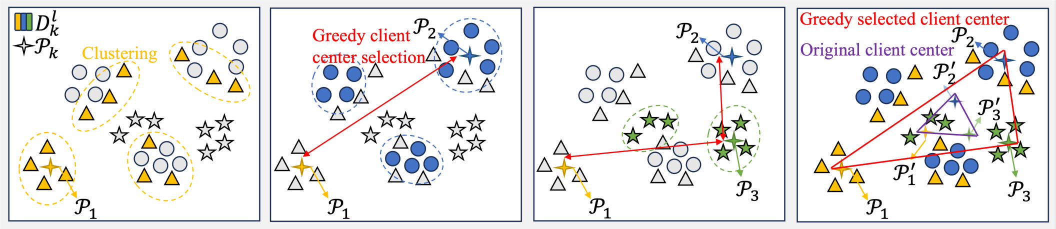


Figure 2: Greedy client center selection iteratively selects the client center in each step, which synthetically considers both the representativeness of the cluster center and its distance from the previously selected client center set $\mathcal{P}$ to maximize the cross-client domain coverage.

4.3 HETEROGENEOUS ENCODER WITH FEATURE ALIGNMENT

For the client-side computational efficiency and system scalability, we propose a heterogeneous encoder method FedDCA* to reduce the client's computational overhead by using a small encoder ω on the client side and a larger encoder ω' on the server side. However, the output dimension of heterogeneous encoders may not be consistent. Therefore, we introduce a projector w_p on the server for feature alignment, as shown in Appendix A.4. We use contrastive learning (Khosla et al., 2020) and train w_p on the public instructions. Specifically, w_p comprises two fully connected layers and a ReLU activation layer in between, projecting ω's dimension to the output dimension of ω'. For $\forall x_i \in D^p$, where $i \in \{1, 2, \ldots, \mathcal{N}\}$, ω outputs embedding h_i, ω' outputs embedding h_i' and w_p outputs embedding φ_i. For input x_i, the positive sample pair is (h_i', φ_i) and negative sample pairs are $\{\varphi_j, j \neq i\}$. Let the batch size be B, then the training objective is defined as follows:

$$\mathcal{L} = -\sum_{i=1}^{B} \log \frac{\exp\left(\mathrm{sim}(\varphi_i, h_i')/\tau\right)}{\sum_{j=1}^{B} \mathbb{I}_{[j \neq i]} \exp\left(\mathrm{sim}(\varphi_i, h_j')/\tau\right)}, \tag{4}$$

where $\mathrm{sim}(\cdot, \cdot)$ denotes the cosine similarity and τ denotes the temperature parameter.

The heterogeneous encoder setting, by employing a projector w_p for feature alignment, proves effective in FL contexts where clients and the server use different encoder sizes or capacities. This approach facilitates mapping lower-dimensional client features $\mathcal{H}$ to the higher-dimensional server space $\mathcal{H}'$ through a strategic training of w_p with both positive and negative pairs. It adeptly aligns similar feature representations closely while positioning dissimilar ones further apart. As a result, the heterogeneous encoder setting not only enhances system scalability and flexibility but also facilitates the creation of a unified feature space that effectively integrates and utilizes diverse representations collaboratively.

4.4 DISCUSSIONS

Computation. Through greedy client center selection, FedDCA solves the optimization problem defined in Eq. 3 in the polynomial time and is quite efficient. On the client side, the k-means algorithm (Wu, 2012) is $\mathcal{O}(|D_k^l|)$. On the server side, as mentioned in Section 4.1, the time complexity of greedy client center selection is $\mathcal{O}\left(N\left(N\xi+\xi\log\xi\right)\right)$. Specifically, for each client, the similarity computation between selected client center set $\mathcal{P}$ and client c_k's cluster $\mathcal{C}_k$ is $\mathcal{O}\left(N\xi\right)$, and the sorting process is $\mathcal{O}\left(\xi\log\xi\right)$. For domain data retrieval, given that the sorting algorithm is $\mathcal{O}\left(\mathcal{N}\log\mathcal{N}\right)$, where $\mathcal{N}$ is the size of the public dataset. Thus the time complexity is $\mathcal{O}\left(N(\mathcal{N}\log\mathcal{N})\right)$. In addition, the heterogeneous encoder setting FedDCA* further reduces the computation overhead on the client side.

Communication. In the domain instruction augmentation stage, each client sends ξ cluster centers to the server. Next, the server performs greedy client center selection and domain data retrieval, then sends retrieved public data D_k^g to the client c_k, whose size is N_k^p. In addition, in the fine-tuning stage, FedDCA follows the standard FedIT procedure, which exchanges the LoRA parameters $\Delta\phi_k$ between clients and the server. Specifically, for the Llama3-8B model, the number of trainable LoRA parameters is just 13.6 million. Compared with the number of frozen pre-trained LLM parameters, the communication for LoRA tuning is quite efficient.

Privacy. The proposed method, FedDCA, does not violate the client's privacy. Compared with other FedIT methods (Zhang et al., 2024b; Ye et al., 2024b), the difference lies in the greedy client center selection. In this stage, the client only uploads the cluster center to the server, which is the average of embeddings to its cluster. In addition, the potential privacy leakage can be further avoided through homomorphic encryption (Acar et al., 2018), which allows the server to directly compute on ciphertext for matrix multiplication for dense retrieval.

5 EXPERIMENTS

To demonstrate the effectiveness of FedDCA and its variant FedDCA*, we conduct extensive experiments across various domains and with several baselines. For additional details and results, please refer to Appendix A.

5.1 EXPERIMENTAL SETUP

Dataset and Evaluation Metrics. To evaluate the performance of FedDCA, we conduct experiments on four domains: code, medical, financial, and mathematical. The detail of constructing the public dataset is described in Appendix A.5. For evaluation, we select a range of datasets, including HumanEval (H-Eval for short) for coding (Chen et al., 2021), MMLU-Med (abbreviated as M-Med) for medical (Hendrycks et al., 2021), and GSM8K for mathematics (Cobbe et al., 2021). Additionally, financial datasets such as FPB, FiQA, and TFNS (Yang et al., 2023a) are utilized. HumanEval is evaluated using Pass@1. For MMLU-Med, FPB, FiQA, and TFNS, we use accuracy as the evaluation metric. While GSM8K is evaluated using exact match (EM). Please see Appendix A.5 for more dataset details.

Baselines. We compare FedDCA and FedDCA* with the following baselines: 1) Unaugmented methods, including zero-shot inference and `FedIT`, which are composed of four widely used FL methods, including FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020), SCAFFOLD (Karimireddy et al., 2020) and FedAvgM (Hsu et al., 2019). 2) Augmented methods, which include random sampling, direct retrieval, LESS (Xia et al., 2024), and Self-Instruct (Wang et al., 2022). Additionally, we report FedDCA's performance with different FL strategies in Table 2. Specifically, zero-shot inference shows the performance of the pre-trained LLM without FedDIT, which gives the lower performance bound. Direct retrieval is described in Appendix A.3. Self-Instruct augments instruction in a generative way through GPT-3.5-turbo (Sun et al., 2023). Based on the prompt provided by Self-Instruct, we define the prompts for generating instructions and responses as shown in Appendix A.7.

5.2 PERFORMANCE ANALYSIS

Performance. We evaluate the performance of FedDCA and compare it with several baselines on four domains (see training details in Appendix A.6). As shown in Table 2, FedDCA outperforms the other nine baselines in all domains. In particular, FedDCA+FedProx achieves the best performance in most domains, with a substantial improvement from at least 5.61% to the maximum of 14.63% over other baselines on the average performance. In addition, to reduce the computation overhead of clients, FedDCA* attempts to utilize heterogeneous encoders. We see that although FedDCA* has a performance drop of 1.85% than FedDCA, it still outperforms other baselines in the average performance. Furthermore, we report FedDCA's performance with different FL strategies, and we can see that no FL method can keep the leading position in all domains. In specific, FedProx and SCAFFOLD FL strategies perform better in the average performance. Overall, the result shows the effectiveness of FedDCA and FedDCA*.

Table 2: Performance(%) of FedDCA, FedDCA* and other nine baselines on various domains. We report FedDCA with different FL strategies in the last four rows, and FedDCA+FedProx achieves the best average performance across four domains.

Method	H-Eval	M-Med	FPB	FiQA	TFNS	GSM8K	Avg.
Zero-shot	29.88	70.60	55.94	18.54	59.21	23.27	42.91
FedAvg	39.03	68.40	58.25	14.18	66.62	47.46	48.99
FedProx	37.20	69.10	56.51	14.90	66.45	47.15	48.55
SCAFFOLD	37.80	70.20	62.71	15.27	66.49	49.27	50.29
FedAvgM	32.32	64.70	68.14	29.27	70.32	46.85	51.93
Random Sampling	32.93	71.30	64.19	13.09	65.53	47.38	49.07
Direct Retrieval	34.14	72.20	66.31	19.11	67.62	50.87	51.71
LESS	28.04	71.00	60.56	16.00	61.14	43.13	46.65
Self-Instruct	32.92	71.90	59.73	20.67	66.54	50.79	50.43
FedDCA*	34.75	73.30	67.10	30.54	71.01	51.55	54.71
FedDCA+FedAvg	36.58	**74.50**	67.24	35.27	73.32	**52.46**	56.56
FedDCA+FedProx	32.92	72.40	**72.93**	**38.18**	**77.55**	51.25	**57.54**
FedDCA+SCAFFOLD	**39.87**	73.20	72.68	33.09	75.50	50.26	57.29
FedDCA+FedAvgM	33.53	68.90	71.45	31.45	72.52	49.76	54.60

Additionally, Table 2 can be divided into two parts: unaugmented methods and augmented methods. We can observe that `FedIT` methods perform well in the code domain, even better than most augmented methods. This could be attributed to two reasons: 1) The code domain is more about following a certain paradigm. 2) As the local data is few, so compared with the same epoch and batch size, the unaugmented methods learn the same data more times, which is kind of not a fair setting. However, FedDCA+SCAFFOLD still surpasses the best baseline in the code domain, further demonstrating the effectiveness of FedDCA.

Efficiency. We see that direct retrieval is the best baseline in the average performance. However, it does not consider cross-client domain coverage, resulting in an overlapping retrieved data distribution. Self-Instruct (Wang et al., 2022) represents the upper limit performance of the generation-based method, which is restricted to the high expense of cost, limited API call frequency, and heavy quality screening process. On the other hand, FewFedPIT (Zhang et al., 2024b) attempts to augment local data by leveraging the pre-trained Llama2 model to perform self-instructed generation during training. However, two concerns exist: 1) The pre-trained LLM cannot provide effective, stable, and high-quality instruction generation. 2) Generating instructions locally is very costly regarding both time and computational resources. Overall, generating instructions self-instructively is not a satisfactory method for the FedDIT scenario. LESS (Xia et al., 2024) represents the augmentation methods through gradient feature retrieval. However, its performance is underwhelming. Compounding this issue, LESS presupposes access to the validation set, which is not always available, and necessitates an initial warmup on the public dataset before calculating gradients for both the public and validation data. This process becomes computationally burdensome with large public datasets. In conclusion, FedDCA achieves better performance while requiring lower computational resources and a more relaxed training condition, which shows its efficiency.

Table 3: Domain coverage of FedDCA, FedDCA* and other baselines on four domains, where `FedIT` methods only use local data without domain augmentation.

Method	Code	Med.	Fin.	Math.	Avg.
`FedIT`	0.8126	0.6990	0.8529	0.7871	0.7879
Random Sampling	0.8512	0.7940	0.9196	0.8651	0.8575
Direct Retrieval	0.9396	0.8830	0.9293	0.8967	0.9122
LESS	0.8509	0.7737	0.8917	0.8352	0.8379
Self-Instruct	0.8966	0.8586	0.9015	0.8811	0.8845
FedDCA*	0.9532	0.8972	0.9538	0.9096	0.9285
FedDCA	**0.9766**	**0.9348**	**0.9815**	**0.9320**	**0.9562**

Domain coverage. Each method's domain coverage is shown in Table 3. Additionally, visualization of domain coverage across different strategies can be found in Appendix A.10. For augmented methods, a correlation is evident between higher domain coverage in a specific domain and improved performance of the method within that domain. Given that FedDCA aims to maximize the domain coverage through greedy client center selection, therefore it achieves the highest domain coverage in all domains, which surpasses other baselines from 4.82% to 21.36% average relative improvement. Furthermore, while FedDCA* employs a smaller encoder on the client side to enhance computational efficiency, this leads to a slight compromise in semantic precision, observed as a relative drop of 2.89% in average domain coverage. Nevertheless, FedDCA* still outperforms the best baseline by an average of 2.78%. In brief, the result proves the effectiveness of FedDCA and FedDCA* in domain coverage augmentation.

5.3 COMPUTATION ANALYSIS

Heterogenous encoder setting FedDCA* allows the system to be more scalable and flexible based on different client capacities. To show the computational efficiency of FedDCA* compared with FedDCA, we compare these two methods from the following aspects: 1) Encoder model size. 2) Encoding time overhead. We first give the evaluation setup of computation analysis.

Evaluation setup. The encoders used for FedDCA and FedDCA* on the client side are `bge-large-en-v1.5` and `all-MiniLM-L6-v2` respectively (detailed in Appendix A.6). Then, we show the model size and the time overhead of encoding the public instructions in Table 4

Table 4: Computational cost comparison between FedDCA and FedDCA*. We report the encoding time on the public dataset and the model size of each encoder.

Method	Encoding Time	Model Size
FedDCA	15 min 46 s	335 M
FedDCA*	5 min 02 s	22.7 M

Compared to FedDCA, FedDCA* has significant computational and time advantages while maintaining acceptable performance (shown in Table 2). This is accomplished by employing heterogeneous encoders and configuring a projector on the server to achieve dimensional mapping.

5.4 PRIVACY ANALYSIS

Next, we evaluate the privacy-preserving capability of different ratios of public data against memory extraction attacks (Carlini et al., 2021; Zhang et al., 2024a), which utilizes the autoregression nature of LLM to extract information from the memory (Xu et al., 2024).

Evaluation setup. We focus on one client's instruction tuning in FedDIT, using FedDCA for instruction augmentation with 1000 to 5000 public instructions. We set up 10 clients with full participation for 10 rounds. Specifically, we record the average ROUGE-L score (Lin, 2004) of client c_0 for each round. The designed prompt to extract instructions memorized by the LLM is detailed in Appendix A.8. For each setting, we repeat memory extraction 100 times and report the average ROUGE-L score based on each generated instruction and all local data instructions to evaluate the privacy-preserving capability of different public data amounts. Specifically, denoting the

Table 5: Privacy-preserving capability against memory extraction attack with different amounts of augmented public data. We report the average ROUGE-L score of each setting in four domains.

Retrieval Num.	Code	Med.	Fin.	Math.
Base Data	0.2338	0.1579	0.1527	0.1718
1000	0.2228	0.1505	0.1451	0.1635
2000	0.2226	0.1531	0.1445	0.1627
3000	0.2195	0.1513	0.1454	0.1555
4000	0.2162	0.1434	0.1438	0.1631
5000	0.2119	0.1501	0.1445	0.1647

generated $\mathcal{N}$ instructions as $\mathcal{I}$ and the client's local instructions $\mathcal{I}^l$. The calculation is defined as $\frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \texttt{ROUGE-L}(\mathcal{I}_i, \mathcal{I}^l)$.

Results. We report the average ROUGE-L scores for base-data-only and various public data fine-tuning in Table 5. The results show no significant correlation between the public data ratio and privacy-preserving capability in the same training round. After 10 rounds of FedDIT, base-data-only fine-tuning outperforms the instruction-augmented settings in all four domains by 6.69%, 5.55%, 5.55%, and 6.16% on average.

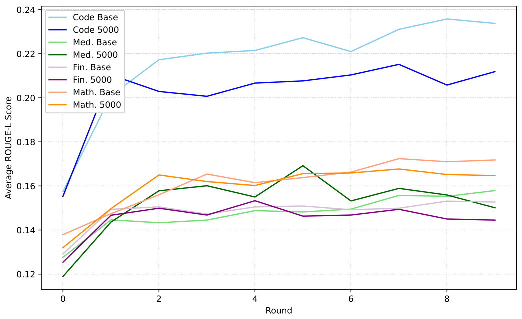


Figure 3: Average ROUGE-L score of each pair of base-data-only and augmented fine-tuning in four domains per round, where different monochromatic represents each domain.

Figure 3 shows the average ROUGE-L score trends per round for each domain, where augmented fine-tuning uses 5000 public data. Initially, the ROUGE-L scores for augmented settings increase, then decrease or converge, while the base-data-only scores continue to rise, especially in the code domain. This indicates that with more training rounds, base-data-only fine-tuning captures more privacy information, while the privacy leakage risk in augmented fine-tuning decreases or converges.

6 CONCLUSION

We reveal that in the context of FedDIT, the model performance is not linearly correlated with data heterogeneity but is significantly impacted by the cross-client domain coverage. In response, we propose a novel FedDIT method called FedDCA, which optimizes the domain coverage through greedy client center selection and retrieval-based instruction augmentation. Additionally, FedDCA* leverages heterogeneous encoders to reduce the client-side computation overhead and improve system scalability. Experiments across four domains demonstrate the effectiveness of FedDCA and the efficiency of FedDCA*. Further privacy analysis indicates that as fine-tuning advances, the risk of private data leakage diminishes or converges in FedDIT with instruction augmentation.

REFERENCES

Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *ACM Comput. Surv.*, 51(4):79:1–79:35, July 2018. ISSN 0360-0300. doi: 10.1145/3214303.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

G. Cornuejols, G. Nemhauser, and L. Wolsey. The Uncapicitated Facility Location Problem. Technical report, Cornell University Operations Research and Industrial Engineering, August 1983.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=d7KBjmI3GmQ`.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

Wenxiang Jiao, Jen-tse Huang, Wenxuan Wang, Zhiwei He, Tian Liang, Xing Wang, Shuming Shi, and Zhaopeng Tu. ParroT: Translating during chat using large language models tuned with human translation and feedback. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 15009–15020, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.1001. URL `https://aclanthology.org/2023.findings-emnlp.1001`.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5132–5143. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/karimireddy20a.html`.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised Contrastive Learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18661–18673. Curran Associates, Inc., 2020.

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

Changho Lee, Janghoon Han, Seonghyeon Ye, Stanley Jungkyu Choi, Honglak Lee, and Kyunghoon Bae. Instruction matters, a simple yet effective task selection approach in instruction tuning for specific tasks. *arXiv preprint arXiv:2404.16418*, 2024.

Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated Optimization in Heterogeneous Networks. *Proceedings of Machine Learning and Systems*, 2(arXiv:1812.06127):429–450, March 2020. doi: 10.48550/arXiv.1812.06127.

Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=UnUwSIgK5W`.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017. ISBN 2640-3498.

Erik Nijkamp, Hiroaki Hayashi, Caiming Xiong, Silvio Savarese, and Yingbo Zhou. Codegen2: Lessons for training llms on programming and natural languages. *arXiv preprint arXiv:2305.02309*, 2023.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatGPT good at search? investigating large language models as re-ranking agents. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL `https://openreview.net/forum?id=3Q6LON8y2I`.

Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. ISSN 1533-7928.

Fanqi Wan, Xinting Huang, Tao Yang, Xiaojun Quan, Wei Bi, and Shuming Shi. Explore-Instruct: Enhancing Domain-Specific Instruction Coverage through Active Exploration. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9435–9454, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.587.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BkluqlSFDS`.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, et al. Instructuie: Multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085*, 2023a.

Yiming Wang, Yu Lin, Xiaodong Zeng, and Guannan Zhang. Privatelora: For efficient privacy preserving llm. *arXiv preprint arXiv:2311.14030*, 2023b.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=gEZrGCozdqR`.

Junjie Wu. *Advances in K-means clustering: a data mining thinking*. Springer Science & Business Media, 2012.

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

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *International Conference on Machine Learning (ICML)*, 2024.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *ArXiv*, abs/2406.08464, 2024. URL `https://api.semanticscholar.org/CorpusID:270391432`.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. Fingpt: Open-source financial large language models. *FinLLM Symposium at IJCAI 2023*, 2023a.

Yi Yang, Yixuan Tang, and Kar Yan Tam. Investlm: A large language model for investment using financial domain instruction tuning. *arXiv preprint arXiv:2309.13064*, 2023b.

Rui Ye, Zhenyang Ni, Fangzhao Wu, Siheng Chen, and Yanfeng Wang. Personalized Federated Learning with Inferred Collaboration Graphs. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 39801–39817. PMLR, July 2023.

Rui Ye, Rui Ge, Xinyu Zhu, Jingyi Chai, Yaxin Du, Yang Liu, Yanfeng Wang, and Siheng Chen. Fedllm-bench: Realistic benchmarks for federated learning of large language models. *arXiv preprint arXiv:2406.04845*, 2024a.

Rui Ye, Wenhao Wang, Jingyi Chai, Dihan Li, Zexi Li, Yinda Xu, Yaxin Du, Yanfeng Wang, and Siheng Chen. Openfedllm: Training large language models on decentralized private data via federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 6137–6147, New York, NY, USA, 2024b. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3671582. URL `https://doi.org/10.1145/3637528.3671582`.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. Mammoth2: Scaling instructions from the web. *arXiv preprint arXiv:2405.03548*, 2024.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7252–7261. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/yurochkin19a.html`.

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

Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. FedALA: Adaptive Local Aggregation for Personalized Federated Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11237–11244, June 2023b. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v37i9.26330.

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

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023d.

Xinlu Zhang, Chenxin Tian, Xianjun Yang, Lichang Chen, Zekun Li, and Linda Ruth Petzold. Alpacare: Instruction-tuned large language models for medical application. *arXiv preprint arXiv:2310.14558*, 2023e.

Yiming Zhang, Nicholas Carlini, and Daphne Ippolito. Effective prompt extraction from language models. In *First Conference on Language Modeling*, 2024a. URL `https://openreview.net/forum?id=0o95CVdNuz`.

Yue Zhang, Leyang Cui, Deng Cai, Xinting Huang, Tao Fang, and Wei Bi. Multi-task instruction tuning of llama for specific scenarios: A preliminary study on writing assistance. *arXiv preprint arXiv:2305.13225*, 2023f.

Zhuo Zhang, Jingyuan Zhang, Jintao Huang, Lizhen Qu, Hongzhi Zhang, Qifan Wang, Xun Zhou, and Zenglin Xu. FewFedPIT: Towards Privacy-preserving and Few-shot Federated Instruction Tuning. *arXiv preprint arXiv:2403.06131*, 2024b.

Kun Zhou, Beichen Zhang, Jiapeng Wang, Zhipeng Chen, Wayne Xin Zhao, Jing Sha, Zhichao Sheng, Shijin Wang, and Ji-Rong Wen. Jiuzhang3.0: Efficiently improving mathematical reasoning by training small data synthesis models. *arXiv preprint arXiv:2405.14365*, 2024.

A APPENDIX

A.1 RELATED WORK

Federated Instruction Tuning. Instruction tuning has been widely applied across various application areas of large language models (LLM), serving as a key technique to enhance the capabilities and controllability of LLM (Zhang et al., 2023d; Wei et al., 2022). Recently, federated instruction tuning (FedIT) has emerged as an effective strategy for the distributed optimization of LLMs, leveraging federated learning (FL) protocols to improve the handling of privacy-sensitive tasks in real-world scenarios. So far, several FedIT frameworks (Ye et al., 2024b;a) have been established to evaluate the effectiveness of FedIT across multiple datasets, tasks, and FL methods. While these platforms provide a foundation for research, they have not yet introduced more complex federated algorithms and deeply investigate the challenging problems and factors affecting FedIT, which are crucial for advancing this field.

PrivateLoRA (Wang et al., 2023b) addresses privacy and efficiency issues by exploiting the low-rank properties of residual activations to reduce communication costs, significantly lowering the communication overhead through collaborative computation between the server and clients while effectively maintaining the privacy of local data. While FewFedPIT (Zhang et al., 2024b) focuses on the few-shot learning setting in FedIT, using self-generated data by pre-trained LLMs locally to mitigate the paucity of data and first discussing memory extraction attacks within FedIT.

Domain Instruction Augmentation. In the real world, there is an urgent need for training LLMs with specific functionalities (e.g., reasoning capabilities) or domain-specific LLMs (e.g., code (Nijkamp et al., 2023; Luo et al., 2024), medical (Zhang et al., 2023e), financial (Yang et al., 2023b;a; Zhang et al., 2023a; Wu et al., 2023), mathematical (Yue et al., 2024; Luo et al., 2023)). Existing works tend to use open-source domain-specific instruction tuning datasets for training. However, the target domain may not always have corresponding ready-made domain-specific instruction datasets. Even if they exist, these datasets are often limited in scale.

Several studies investigate domain-specific instruction augmentation, which can be categorized into three aspects: 1) Reusing human-curated public datasets (Wang et al., 2023a; Zhang et al., 2023f; Jiao et al., 2023; Lee et al., 2024; Xia et al., 2024). For instance, Parrot (Jiao et al., 2023) enhances translation capabilities of LLMs in chat by converting bilingual sentences into instruction-following datasets. Furthermore, works like INSTA (Lee et al., 2024) and LESS (Xia et al., 2024) attempt efficient domain-specific instruction augmentation via dense retrieval. INSTA (Lee et al., 2024) uses instructions without responses for effective retrieval. LESS (Xia et al., 2024) assumes access to a validation set and uses the warmup LLM's gradients of the train and validation sets for retrieval. 2) Using seed tasks for self-instruct (Luo et al., 2024; Wan et al., 2023): Explore-Instruct (Wan et al., 2023), for example, employs activate exploration to tree-model domain tasks from both depth and breadth, increasing seed instructions' coverage of the domain, and subsequently generating broader in-domain instructions through self-instruct. 3) Scaling instructions from the web: Recent works (Yue et al., 2024; Zhou et al., 2024) highlight the immense potential of mining naturally occurring instructions from the internet. Compared to generated data, web-mined instructions exhibit less bias and greater diversity. MAmmoTH2 (Yue et al., 2024) firstly retrieves domain-relevant texts and employs a LLM to extract Q-A pairs, further refining them into instruction-response pairs. Jiuzhang3.0 (Zhou et al., 2024) distills GPT's instruction generation capabilities into a smaller model and then uses it to generate instructions from the internet. In conclusion, models fine-tuned with augmented instructions have shown promising domain-specific capabilities.

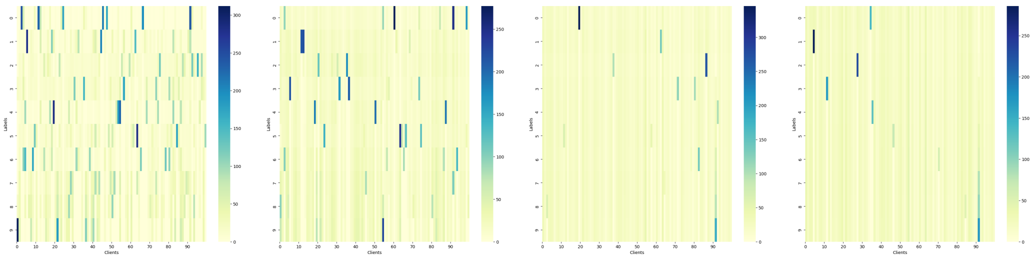
To obtain a well-performing LLM in the specific domain within the distributed environment, we utilize a multi-domain dataset as public data on the server side and perform domain-specific instruction augmentation based on the client's local instructions.

A.2 WHAT TRULY COUNTS IN FEDDIT

This section will first analyze the correlation between different non-iid levels and the model's performance with separate experiments for both single and multiple domains in Section A.2.1. Further, to demonstrate the impact of non-iid on FedDIT, we compare the performance of the global model trained on augmented data based on iid and non-iid cross-client data distribution. Additionally, we suggest that domain coverage is a key factor for FedDIT in Section A.2.2.

A.2.1 DATA HETEROGENEITY IS NOT MATTER IN FEDDIT

Following the traditional approach of constructing different degrees of non-iid, which are widely used in federated learning (Wang et al., 2020; Yurochkin et al., 2019), we adopt Dirichlet distribution to construct various heterogeneity and use k-means with the cluster num $\xi = 100$ to pseudo labeling instructions. Dirichlet distribution is affected by the hyperparameter α, which enhances heterogeneity with a smaller α and decreases heterogeneity with a larger α. We choose four widely used heterogeneity, which are $\alpha = [0.01, 0.1, 1, 10]$ (Ye et al., 2023; Zhang et al., 2023b; Li et al., 2021). Figure 4 shows a visualization of the data distribution with different heterogeneity on the code dataset.



(a) $\alpha = 10$ (b) $\alpha = 1$ (c) $\alpha = 0.1$ (d) $\alpha = 0.01$

Figure 4: Visualization of client data distribution with $\alpha = [10, 1, 0.1, 0.01]$ in the code domain.

We perform instruction tuning both on single-domain and multi-domain with different heterogeneity. Specifically, for the single domain, we only use the in-domain data and then perform FedIT. For multi-domain, we use the mixed dataset, which consists of four domain-specific instruction datasets and a general instruction dataset, which are described in Appendix A.5. The training details are shown in Appendix A.6. Results for single domain and multi-domain are shown in Table 6. Both single-domain and multi-domain clearly show that the performance of LLM does not decrease due to the increase of data heterogeneity but shows a nonlinear correlation, which indicates that the performance of LLM does not directly depend on data heterogeneity and other factors that play a key role.

Table 6: Performance(%) of different heterogeneity on single domain and multi-domain.

α	Single Domain				Multi-Domain			
	10	1	0.1	0.01	10	1	0.1	0.01
H-Eval	32.92	34.14	31.09	**34.75**	34.14	31.70	**34.75**	**34.75**
M-Med	70.6	69.9	**71.0**	71.3	**71.7**	71.3	69.9	71.3
FPB	67.90	72.11	**73.76**	68.89	60.06	61.63	60.97	**62.04**
FiQA	43.63	**45.45**	41.81	32.00	10.90	12.72	**17.81**	9.09
TFNS	74.37	**76.00**	75.71	72.69	**65.36**	65.24	64.69	64.61
GSM8K	**53.75**	51.40	51.93	52.76	**46.5**	43.5	41.5	45.5
Avg.	57.20	**58.17**	57.55	55.40	48.11	47.68	**48.27**	47.88

A.2.2 DOMAIN COVERAGE: A KEY FACTOR IN FEDDIT

Explore-Instruct (Wan et al., 2023) enhances the coverage of domain-specific seed tasks through active exploration, then uses the self-instruct method for instruction data augmentation. This approach highlights the impact of domain coverage on domain-specific instruction tuning. Inspired by Explore-Instruct, we attempt to conduct more in-depth and extensive experiments to study the effect of domain coverage on FedDIT. Firstly, we define the domain coverage of cross-client data in the FL setting. Assume the dataset of in-domain data D^d represents the latent data distribution of this domain and the cross-client data is defined as $D^c = \cup_{k=1}^{N} \left(D_k^l \cup D_k^g\right)$. Inspired by the facility location function (Cornuejols et al., 1983), we define the domain coverage of D^c respect to D^d as follows:

$$d(D^d, D^c) = \frac{1}{|D^c|} \sum_{d \in D^d} \max_{v \in (D^c \cap D^d)} sim(d, v), \tag{5}$$

where $sim(d, v)$ is the similarity between d and v. Specifically, we use the cosine similarity function in FedDCA. Note that we only use the in-domain data in D^c to calculate the domain coverage because the out-of-domain data would mislead the domain coverage evaluation, as shown in Figure 5.

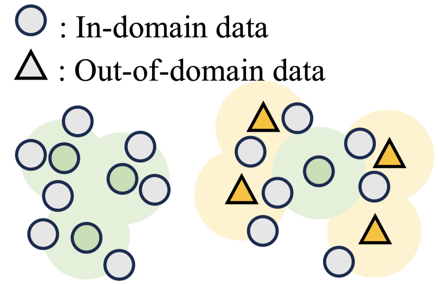


Figure 5: Misleadning of out-of-domain data on domain coverage calculation.

To better align with the real-world scenarios, we explore instruction augmentation based on both iid and non-iid cross-client data distribution and adopt direct data retrieval as described in Appendix A.3 for FedDIT. We set the number of clients to 10, while each client has 100 local instructions and obtains 5000 augmented public instructions from the server.

To construct iid data distribution, we randomly sample 1,000 from multi-domain datasets (code, medical, financial, mathematical, and general), which is detailed in Table 8 and divide them into 10 shards as each client's local data. To construct non-iid data distribution, we perform k-means clustering with $\xi = 100$. Each client randomly samples 100 instructions from different randomly selected clusters. Furthermore, for both iid and non-iid settings, direct data retrieval is performed based on the client's local data.

Table 7: Performance(%) and domain coverage of iid and non-iid settings on different domains. The higher domain coverage correlates with better performance.

Test Set	Performance (%)		Domain Coverage	
	iid	non-iid	iid	non-iid
H-Eval	**35.36**	33.53	**0.8538**	0.7994
M-Med	70.20	**71.00**	0.7800	**0.8027**
FPB	58.58	**64.19**		
FiQA	17.09	**19.27**	0.8523	**0.9327**
TFNS	66.16	**69.09**		
GSM8K	**40.50**	38.50	**0.9137**	0.8448

Table 7 presents the performance of FedDIT on different domains with iid and non-iid settings and shows the domain coverage in four domains. We can observe that both iid and non-iid settings outperform in some domains, but both collectively indicate that higher domain coverage correlates with better performance.

A.3 DIRECT DOMAIN DATA RETRIEVAL

In this section, we first show the detail of instruction-based dense retrieval in Appendix A.3.1, which is both used in direct retrieval and FedDCA. Then, we explain the direct retrieval algorithm in Appendix A.3.2.

A.3.1 INSTRUCTION BASED DENSE RETRIEVAL

Suppose an instruction dataset $\mathcal{D}$ consists of several instances. Each instance is a (*Instruction*, *Response*) pair. For instructions that have *Input*, we concatenate the *Instruction* and

Input as *Instruction*, which is consistent with OpenFedLLM (Ye et al., 2024b). Then we use only the instruction for encoding and dense retrieval. Denote $\mathcal{E}$ as a cluster center and I as an instruction of the public dataset D^p, then we measure the instruction-based similarity score for dense retrieval as follows:

$$\textbf{Score}(\mathcal{E}, I) = sim(\mathcal{E}, w_{enc}(I)), \tag{6}$$

where $sim(\cdot, \cdot)$ is the cosine similarity function. Based on the computed similarity between $\mathcal{E}$ and each I in the public data, we then select the top-N_k^p instructions as the retrieved public data for domain data augmentation.

A.3.2 DIRECT RETRIEVAL

The direct domain data retrieval only utilizes instructions of the client's local data without responses to perform the retrieval-based domain data augmentation. The detailed algorithm is described in Algorithm 2. For each client, we start by encoding the local instructions using the encoder model w_{enc}. Next, we apply the k-means algorithm to cluster the embeddings into ξ clusters. Then ξ cluster centers are sent to the server for retrieval-based domain data augmentation. Subsequently, the retrieved public data is sent to the client for instruction tuning.

Algorithm 2 Direct domain data retrieval

```
Parameters:
Clients' local datasets $D = \{D_1, D_2, \dots, D_N\}$; Public dataset $D^p$; Local datasets $D^l = \{D_1^l, D_2^l, \dots, D_N^l\}$; Number of clusters $\xi$; Encoder model $w_{enc}$; Pre-encoded public instruction embeddings $\mathcal{E}$.
$D^g \leftarrow \emptyset$
for $i \in \{1, 2, \dots, N\}$ do
    $\mathcal{E}' \leftarrow \{w_{enc}(x) \mid x \in D_{i,instruction}^l\}$    ▷ Encode the local instructions
    $\mathcal{C} \leftarrow$ k-means$(\xi).fit(\mathcal{E}')$    ▷ Cluster local instructions, return cluster centers $\mathcal{C}$
    $\mathcal{S} \leftarrow \mathcal{C} \cdot \mathcal{E}^T$    ▷ Compute the similarity score between $\mathcal{C}$ and $\mathcal{E}$
    $\mathcal{I} \leftarrow \emptyset$
    for $j \in \{1, 2, \dots, \xi\}$ do
        $\mathcal{S}' \leftarrow \{s \mid s \in S_j, \text{index}(s) \notin \mathcal{I}\}$    ▷ Filter the selected indices
        $\mathcal{I}' \leftarrow$ indices of the top-$\frac{N_k^p}{\xi}$ elements in $\mathcal{S}'$    ▷ Retrieve the top $\frac{N_k^p}{\xi}$ indices
        $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}'$    ▷ Update the selected indices
    end for
    $D_i^g \leftarrow \{D_j^p \mid j \in \mathcal{I}\}$    ▷ Selected public instructions
    $D_i \leftarrow D_i^g \cup D_i^l$    ▷ Obtain the augmented instruction dataset $D_i$
end for
```

A.4 PROJECTOR

To reduce the computational overhead of clients, we propose a heterogeneous encoder method called FedDCA*. We utilize a projector to perform the dimension alignment between the small encoder on the client side and the large decoder on the server side. As shown in Figure 6, where d_l is the dimension of the client-side encoder, d_g is the dimension of the server-side encoder.

A.5 TRAIN AND TEST DATASET INFORMATION

Here we provide the train and test dataset details of code (CodeAlpaca[2]), medical (MedAlpaca[3]), financial (FinGPT (Yang et al., 2023a)) and mathematical (MathInstruct[4]) domain, respectively.

As shown in Table 8, the public data consists of four domain-specific instruction datasets and a general instruction dataset, which are CodeAlpaca, MedAlpaca, FinGPT, MathInstruct, and Alpaca,

---

[2] `https://huggingface.co/datasets/sahil2801/CodeAlpaca-20k`

[3] `https://huggingface.co/datasets/medalpaca`

[4] `https://huggingface.co/datasets/TIGER-Lab/MathInstruct`

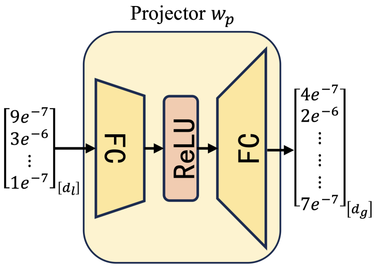

Projector w_p
FC
ReLU
FC
$[9e^{-7}, 3e^{-6}, \vdots, 1e^{-7}]_{[d_l]}$
$[4e^{-7}, 2e^{-6}, \vdots, \vdots, \vdots, 7e^{-7}]_{[d_g]}$

Figure 6: The projector used on the server side for feature alignment, which consists of two fully connected layers and a ReLU activation layer in between.

Table 8: Train set information of each domain.

Dataset name	Domain	N_{sample}
CodeAlpaca	Code	20,022
MedAlpaca	Medical	33,955
FinGPT	Financial	76,772
MathInstruct	Mathematical	224,567
Alpaca	General	52,002

respectively. For each domain's FedDIT, we randomly select 1000 samples from the in-domain instruction and split them into 10 shards as each client's local dataset. The rest of the instructions are used as the public dataset D^p.

Table 9: Test set information of each domain.

Dataset name	Domain	N_{sample}	Metric
HumanEval	Code	164	Pass@1
MMLU-Med	Medical	1,089	Acc
FPB	Financial	152	Acc
FiQA	Financial	35	Acc
TFNS	Financial	299	Acc
GSM8K	Mathematical	1,319	Exact Match

Table 9 shows the detail of the test set used for evaluation: HumanEval is used for the code domain; MMLU-Med is used for the medical domain; FPB, FiQA and TFNS are used for the financial domain, and GSM8K is used for the mathematical domain. Specifically, MMLU-Med uses subjects `anatomy`, `clinical_knowledge`, `college_biology`, `college_medicine`, `medical_genetics` and `professional_medicine` in MMLU.

A.6 IMPLEMENTATION DETAILS

We consider FedDIT in the cross-device scenario, N = 10 clients, $\mathcal{R}$ = 30 rounds, where we randomly sample 2 clients to be available for each round. Then, each available client performs FedDIT for 10 steps with AdamW optimizer, and the batch size is $B = 32$ in a round. The initial learning rate is $5e - 5$ with a cosine learning rate scheduler. Our experiment utilizes the widely used LLM, Llama3-8B[5] as the base model with 2048 max sequence length and adopts LoRA tuning method. The rank of LoRA is 16, and the scalar alpha is 16. For k-means (Wu, 2012), we set cluster num ξ = 10 and for FedDCA we set the similarity threshold α = 0.7. For FedDCA*, we set the temperature parameter τ = 0.5 for contrastive learning (Khosla et al., 2020). We utilize `bge-large-en-v1.5`[6] as both the client and server's encoder as default, which outputs embed-

---

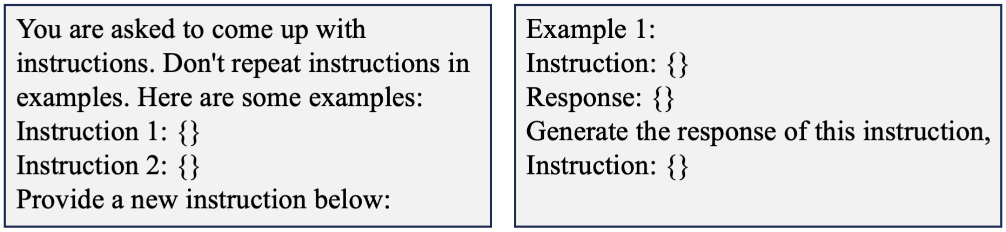
[5] `https://huggingface.co/meta-llama/Meta-Llama-3-8B`

[6] `https://huggingface.co/BAAI/bge-large-en-v1.5`

dings of 1024 dimensions. While we utilize `all-MiniLM-L6-v2`[7] as the client's small encoder which outputs embeddings of 384 dimensions and `bge-large-en-v1.5` as the server's encoder for FedDCA*.

A.7 PROMPTS USED IN THE SELF-INSTRUCT DATA GENERATION

To generate the Self-Instruct data, we prompt GPT-3.5 to generate the instruction with the designed prompt in Figure 7. Specifically, we randomly sample two examples from the client's local data to guide GPT-3.5 generating the in-domain instruction and one example from the client's local data for one-shot in-context learning to guide GPT-3.5 generating responses into the example's format.



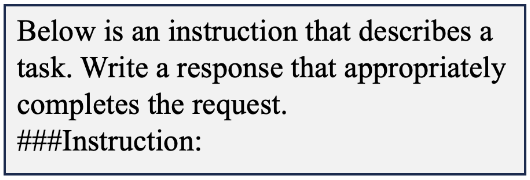
You are asked to come up with instructions. Don't repeat instructions in examples. Here are some examples:
Instruction 1: {}
Instruction 2: {}
Provide a new instruction below:

Example 1:
Instruction: {}
Response: {}
Generate the response of this instruction,
Instruction: {}

Figure 7: Prompts used in the Self-Instruct data generation. (a) Prompt for generating new instructions. Two examples are randomly sampled from the client's local data for in-context demonstration. (b) Prompt for generating responses. We prompt GPT-3.5 to generate responses with a randomly selected example for one-shot in-context learning.

A.8 PROMPT USED FOR MEMORY EXTRACTION ATTACK

As we use Llama3-8B as our base model and format the instructions and responses into the Alpaca's format, to utilize the auto-regression nature of LLM to extract the instruction, we prompt the model to generate the instruction using the prompt in Figure 8, which is exactly the prefix of the Alpaca's template.



Below is an instruction that describes a task. Write a response that appropriately completes the request.
###Instruction:

Figure 8: Prompt used for memory extraction attack.

A.9 FURTHER ANALYSIS

To further study the effect of different hyperparameters of FedDCA, we undertake a thorough analysis including various retrieval amounts and different k-means cluster numbers ξ. In addition, we perform the ablation study on whether using the similarity threshold α in the greedy client center selection.

Effect of Retrieval Number. We report the performance and the corresponding domain coverage of FedDCA with different retrieval amounts on the four domains in Figure 9, respectively. We can see that the domain coverage of FedDCA is increasing along with the retrieval amount in different trends, as well as the performance. Specifically, the domain coverage of each domain increases by 6.36%, 18.69%, 5.04%, and 8.06% in relative, respectively. Along with domain coverage increasing, the performance of FedDCA is increasing by 3.05%, 2.20%, 4.08%, and 6.95% for each domain. Noted that although the code domain is more about following a certain paradigm, which could

---

[7] `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`

perform well with a few data and more fine-tuning rounds, it still could benefit from the instruction augmentation.

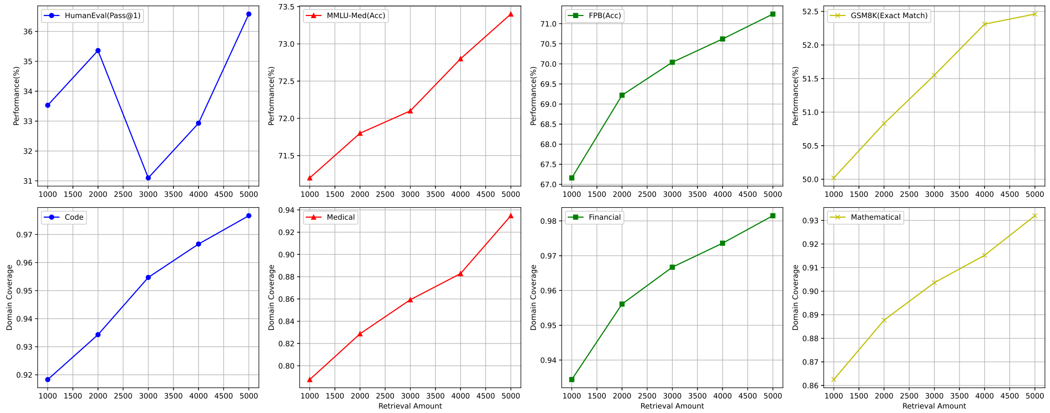


Figure 9: Effect of different retrieval amounts on the performance of FedDCA and its domain coverage. We show the results on four domains separately. Here, we use the FPB test set to evaluate the performance in the financial domain.

Impact of Different Cluster Number. The hyperparameter ξ is the number of clusters in the k-means algorithm. The experiment is conducted on $\xi = [N, 2N, 4N, 8N]$, where N is the number of clients. Following the setting in Appendix A.6, we set $N = 10$. We report the domain coverage of the augmented dataset via FedDCA with different cluster numbers ξ on the four domains in Figure 10, respectively. Results show that there is no best ξ for all domains. Specifically, the best ξ of code, medical, financial, and mathematical domains are 80, 80, 40, and 10, respectively.

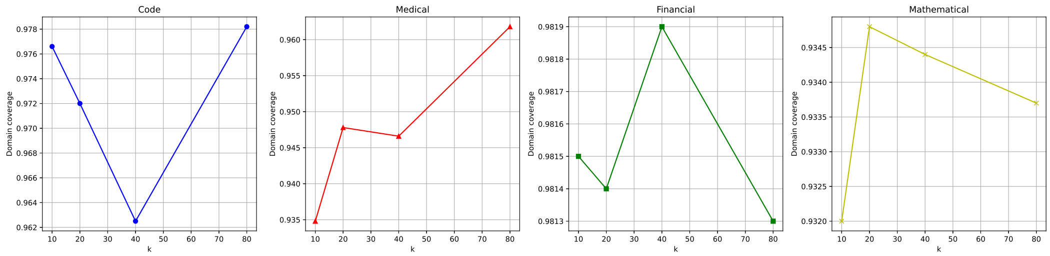
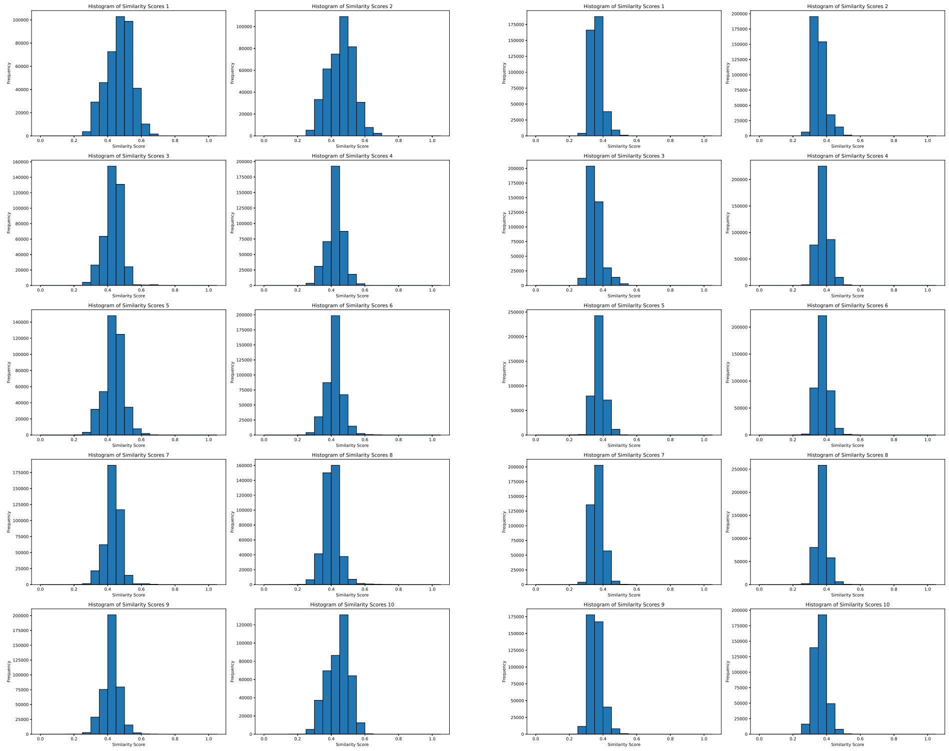


Figure 10: Impact of different cluster number ξ on the cross-client domain coverage. We report the domain coverage of the augmented dataset via FedDCA with different cluster numbers ξ on the four domains.

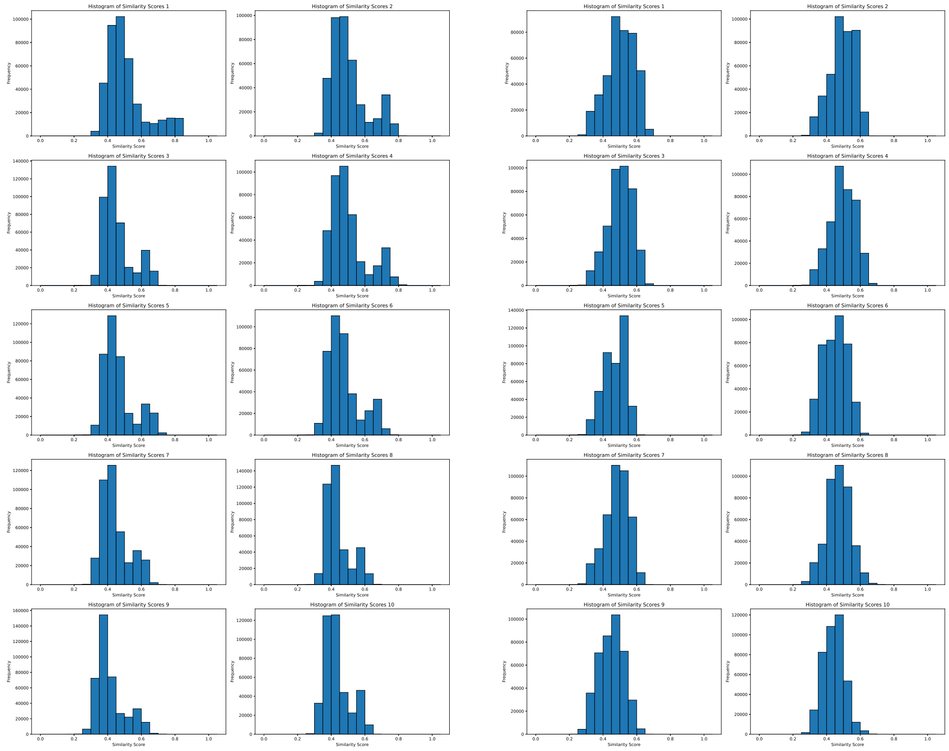
Ablation Study. We conduct the experiment with FedDCA w/o similarity threshold α on the four domains based on the FedAvg FL strategy. The performance and the corresponding domain coverage are shown in Table 10, where FedDCA without the similarity threshold α is marked as FedDCA†. The result shows that the performance of FedDCA with similarity threshold α is slightly better than FedDCA without using α in code, financial, and mathematical domains, as the similarity scores in the medical domain are relatively lower. We show the similarity score distribution of the four domains in Figure 11. For each domain, we plot each similarity score's distribution of 10 clients. The similarity score is computed between the selected client center and the public data. Then, we show the similarity score distribution using the histogram plot.

A.10 AUGMENTATION STRATEGY VISUALIZATION

To more intuitively compare the domain coverage of different instruction augmentation methods, we randomly sample 5,000 instructions obtained through these methods and 10,000 in-domain instructions as the background, representing the distribution of specific domains in the public dataset. We



(a) Code (b) Medical



(c) Financial (d) Mathematical

Figure 11: The similarity score distribution of the four domains. For each domain, we plot each similarity score's distribution of 10 clients.

Table 10: Ablation study on the performance and domain coverage of FedDCA w/o similarity threshold α.

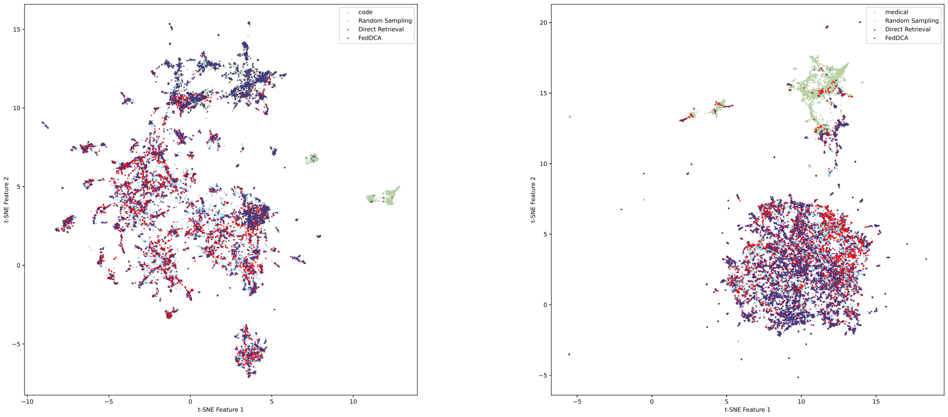
Metric	Performance(%)		Domain Coverage	
	FedDCA†	FedDCA	FedDCA†	FedDCA
H-Eval	35.97	**36.58**	0.8972	**0.9348**
M-Med	73.40	**73.40**	0.9348	**0.9348**
FPB	66.25	**67.24**		
FiQA	23.27	**35.27**	0.9353	**0.9815**
TFNS	69.34	**73.32**		
GSM8K	51.78	**52.46**	0.9128	**0.9320**

then visualized the results using t-SNE (van der Maaten & Hinton, 2008), as shown in Figure 12. The plot shows that FedDCA encompasses most of the in-domain data, which is consistent with FedDCA's domain coverage of each domain shown in Table 3. Also, we can observe that the random sampling strategy selects a lot of out-of-domain data while does not have good coverage in specific domains.

A.11 FREQUENTLY USED NOTATION

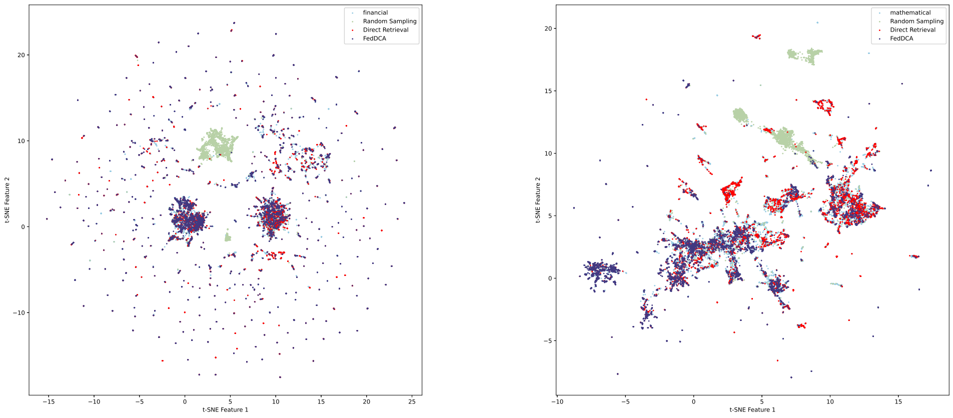
Table 11: Frequently used notation.

NOTATIONS	**REMARK**
N, C	Clients number, client set $C = \{c_1, c_2, \dots, c_N\}$.
D^p, D^d, D^c, D^l	The public datasets, the in-domain data, the cross-client dataset, client's local private data.
N_k^l, N_k^p	Number of local private data on the k-th client, number of the retrieved public data on the k-th client.
D_k^l, D_k^g, D_k	The local private data on the k-th client, the retrieved public data on the k-th client, the augmented dataset on the k-th client.
$\Lambda, \mathcal{P}, \mathcal{C}$	A specific sampling strategy that performs instruction augmentation on the server side, selected client center set, the cluster centers obtained locally and sent to the server for the greedy client center selection.
$\phi, \Delta\phi$	LLM's pre-trained parameters, additional LoRA parameters.
w, w_{enc}, w_p	Merged model parameters from the frozen LLM's parameters ϕ and the additional LoRA parameters $\Delta\phi$, encoder model, projector model.
$F_k(w; \mathcal{D}), l(w; x, y)$	Loss of model w over a specific dataset $\mathcal{D}$, the instructon tuning loss of model w over a data sample (x, y).



(a) Code

(b) Medical



(c) Financial

(d) Mathematical

Figure 12: Visualization of cross-client data distribution in different domains, performing t-SNE dimensionality reduction on retrieved instructions through various augmentation strategies. We randomly sample 10,000 in-domain samples as background while randomly sampling 5,000 samples from the cross-client augmented dataset for different instruction augmentation methods for comparison.