Towards Building the FederatedGPT: Federated Instruction Tuning

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

While "instruction-tuned" generative large language models (LLMs) have demonstrated an impressive ability to generalize to new tasks, the training phases heavily rely on large amounts of diverse and high-quality instruction data (such as ChatGPT and GPT-4). Unfortunately, acquiring high-quality data, especially when it comes to human-written data, can pose significant challenges both in terms of cost and accessibility. Moreover, concerns related to privacy can further limit access to such data, making the process of obtaining it a complex and nuanced undertaking. To tackle this issue, our study introduces a new approach called **Fed**erated **I**nstruction **T**uning (FedIT), which leverages federated learning (FL) as the learning framework for the instruction tuning of LLMs. This marks the first exploration of FL-based instruction tuning for LLMs. This is especially important since text data is predominantly generated by end users. For example, collecting extensive amounts of everyday user conversations can be a useful approach to improving the generalizability of LLMs, allowing them to generate authentic and natural responses. Therefore, it is imperative to design and adapt FL approaches to effectively leverage these users' diverse instructions stored on local devices while mitigating concerns related to the data sensitivity and the cost of data transmission. In this study, we leverage extensive qualitative analysis, including the prevalent GPT-4 auto-evaluation to illustrate how our FedIT framework enhances the performance of LLMs. Utilizing diverse instruction sets on the client side, FedIT outperforms centralized training with only limited local instructions.

1 Introduction

Large Language Models (LLMs) have become ubiquitous in natural language processing (NLP) [5, 13, 55], where one single model can perform well on various language tasks, including established tasks such as text generation, machine translation, and question answering, as well as novel application-oriented tasks in human daily life [15, 69]. To align LLM to follow human intents, instruction-tuning has been proposed by fine-tuning LLM on instruction-following data [53, 71, 72]. Though instruction-tuning has demonstrated great effectiveness in improving the zero and few-shot generalization capabilities of LLM, its performance on real-world tasks is contingent on the *quantity, diversity,* and *quality* of the collected instructions [49, 71]. The process of collecting these instructions can be expensive [63, 71]. Besides, the increasing awareness of data sensitivity highlights a significant challenge in acquiring extensive and high-quality instructions [2, 20, 27]. For instance, collecting vast amounts of daily conversations from users is a valuable means of providing guidance for LLMs, enabling them to generate authentic and genuine responses. However, privacy concerns may hinder users from sharing their conversations, resulting in a limited quantity of instructions that are not fully representative of the target population. Likewise, many companies treat their instructions as proprietary assets that are closely guarded. They are reluctant to share their instructions with external parties, as they often contain confidential and proprietary information that is critical to their success and profitability [21].

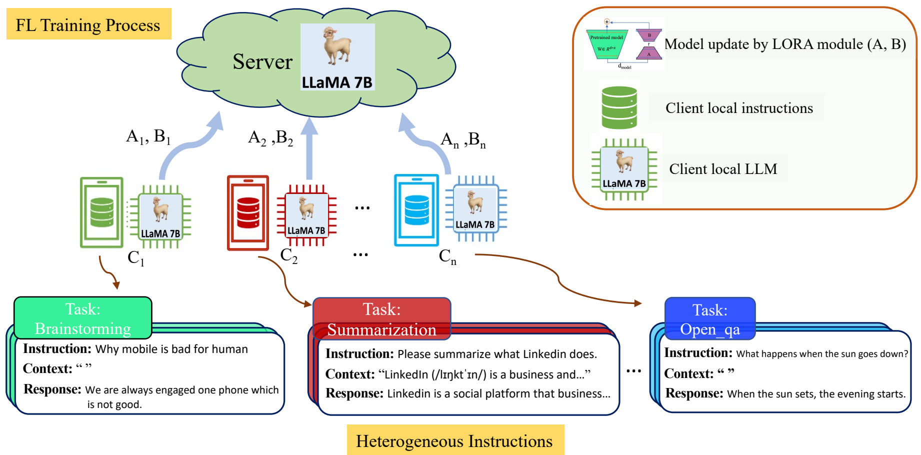

Figure 1: The framework of Federated Instruction Tuning (FedIT)

We aim to tackle these challenges by exploring the potential of federated learning (FL) as a promising solution [47]. This collaborative learning technique enables many clients to learn a shared model jointly without sharing their sensitive data. In particular, in our proposed federated instruction-tuning, clients initially download a global LLM from a central server and subsequently compute local model updates using their respective local instructions. These local updates are then transmitted back to the server, where they are aggregated and integrated to update the global LLM. Given that clients often have limited computational resources in comparison to traditional centralized training cloud servers, which can utilize thousands of GPUs to fully fine-tune all parameters of LLMs, we resort to parameter-efficient tuning techniques. This leads to a significant decrease in computational and communication demands as it reduces the number of trainable parameters on each device. Thus, our proposed framework enables efficient utilization of the computational resources available on local edge devices, which are commonly accessible, as well as their diverse local instructions. Our major contributions are summarized as follows:

- We make the first attempt to leverage FL for instruction tuning (FedIT) of LLMs. We show that we can circumvent the above-mentioned challenges of predominant instruction tuning by exploiting the diverse sets of available instructions from the users in the FL system.
- A comprehensive study is conducted on the heterogeneity and diversity within the federated instruction tuning. We employ the GPT-4 auto-evaluation method, which has been widely utilized in related research [10, 54], to demonstrate the effectiveness of our FedIT approach in enhancing response quality by leveraging diverse available instructions.
- We have developed and released a GitHub repository called *Shepherd*[1], which has been designed to provide ease of customization and adaptability, thereby offering benefits for future research endeavors in this field.

2 Federated Instruction Tuning

Drawing on the successful application of FL in various machine learning domains to offer privacy protection, we introduce the FedIT framework. By harnessing the advantages of FL, our framework enables secure and cost-effective LLM instruction tuning. The overall framework, illustrated in Figure 1 and Algorithm 1, involves two primary components: local training operations on the client side and scheduling and aggregation operations on the server side, which work together to ensure efficient training.

Our framework assigns an LLM to each client and performs client selection to determine which clients will participate in local instruction tuning. During instruction tuning, clients use their local instruction dataset to update a small, trainable adapter that is added to the pre-trained model weights. This approach reduces the cost of fine-tuning and makes it compatible with the limited computational resources of local devices. Upon completion, clients send the updated adapter back to the server, which aggregates the received adapters' parameters and conducts another round of client selection. This iterative process continues until convergence.

[1] `https://github.com/JayZhang42/FederatedGPT-Shepherd`

Our FedIT framework for instruction tuning is designed to address the challenges of collecting high-quality data and ensuring data privacy by keeping the instructions on the local devices throughout the process. By ensuring data sensitivity protection, we can encourage more clients to participate in the federated instruction tuning. Consequently, the combined instruction dataset from all clients can encompass a broader range of topics, tasks, and valuable information, as clients may come from different areas and possess domain-specific expertise. This FL approach enables our framework to effectively adapt to diverse and evolving instruction datasets, resulting in more robust and generalized LLM performance. Moreover, our FedIT methodology incorporates a parameter-efficient fine-tuning (PEFT) technique, known as LoRA [24], to facilitate local training. This method reduces computational and communication overheads for local edge devices that have limited system resources. As a result, we can leverage the computational capabilities of a multitude of distributed local edge devices that are often disregarded in conventional centralized instruction tuning. This feature enhances the scalability of FedIT, enabling it to address large-scale instructional tuning challenges effectively.

Algorithm 1 Federated Instruction Turning (FedIT)

Initialization: each client's initial global large language model with parameters $\boldsymbol{w}$ and a lightweight adapter with parameters $\Delta\boldsymbol{w}^{(0)}$, client index subset $\mathcal{M} = \varnothing$, K communication rounds, $k = 0$,

Training
  while $k \leq K$ **do**
    Server updates $\mathcal{M}$ using specific strategies ▷ **Select clients for local training**
    for $n \in \mathcal{M}$ **in parallel do** ▷ **Parameter-efficient finetuning on local instructions dataset**
      Client freeze the LLM and update the adapter weights with $\Delta\boldsymbol{w}^{(k)}$
      $\Delta\boldsymbol{w}_n^{(k+1)} \leftarrow$ **InstructionTuning**$(\Delta\boldsymbol{w}_n^{(k)})$
    end For
    $\Delta\boldsymbol{w}^{(k+1)} \leftarrow$ **Aggregate**$(\Delta\boldsymbol{w}_n^{(k+1)})$ for $n \in \mathcal{M}$ ▷ **Aggregate the adapters at Server**
    $k \leftarrow k + 1$
  end while

Outcome (m, θ_g^t)**:**
Derive the final adapter with parameters $\Delta\boldsymbol{w}^{(K)}$ and the global LLM with parameters $\boldsymbol{w}$

2.1 Heterogeneity of Instructional Data

Beyond the practical benefits of FedIT, our research makes a unique contribution by presenting a scenario for instruction tuning of LLMs where statistical heterogeneity can serve as a positive factor for federated learning. Our work demonstrates that the extensive heterogeneous and diverse set of instructions can, in fact, be a blessing factor for our FedIT approach. For instance, different clients may have different instruction tasks, such as open-domain QA and writing. The content and format of these instructions can be substantially different. For example, QA tasks typically require fact-based questions and answers, while writing tasks involve instructions for generating coherent and meaningful sentences.

In order to obtain a comprehensive understanding of data heterogeneity inherent in the instructional dataset utilized for this study, we performed an in-depth examination of the Dolly dataset (**Databricks-dolly-15k**)[2]. This publicly accessible dataset, consisting of instruction-following records generated by a multitude of Databricks employees, spans a range of behavioral categories as outlined in the InstructGPT paper [53]. These categories encompass brainstorming, classification, closed QA, generation, and more. To emulate an FL environment with ten clients, we partitioned the entire Dolly dataset into ten shards using a widely adopted partitioning method [28], with each shard assigned to an individual client. As is evident in the **left** subfigure of Figure 2, each user's dataset contains imbalanced categories of instructions, with some categories absent entirely. This reflects real-world scenarios where users may not possess expertise across all instruction categories. In the absence of FedIT, due to the challenges associated with collecting sensitive instruction data, the model can only be trained on the local instruction dataset of each user, as depicted in the **left** subfigure of Figure 2. However, by implementing our FedIT approach, the model can be trained on the local instruction datasets of all clients, as illustrated in the **right** subfigure of Figure 2. As a result, FedIT allows for instruction tuning on a dataset with enhanced diversity and a larger number of data points, allowing

[2]`https://huggingface.co/datasets/databricks/databricks-dolly-15k`

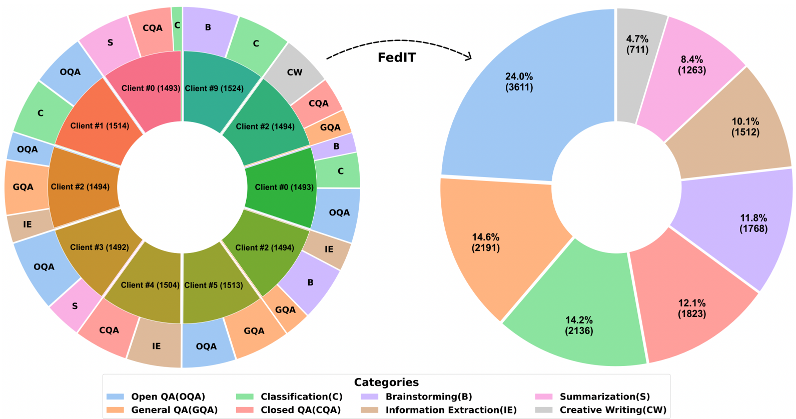

Figure 2: Illustrate the heterogeneity of FedIT with **Databricks-dolly-15k** instruction dataset. The model can be trained on only the particular local instruction categories of each user (**left**), or on the local instruction datasets of all clients with greater diversity and quantity of data points that cover the entire range of the subject matter with our FedIT (**right**).

the model to be more generalized and applicable to a wider array of tasks compared to training solely on each client's local instruction dataset with limited categories and quantity.

2.2 Parameter Efficiency in FedIT

Taking into account the limited computational capabilities of local devices, which are unable to support full fine-tuning of a large language model, it is crucial to implement a parameter-efficient fine-tuning strategy that leverages local computational resources, which means optimizing the LLMs while minimizing the computational and storage demands associated with the training process. We adopt LoRA in our FL framework due to its promising performance in recent studies on instruction tuning. Compared to fully fine-tuning the LLM, LoRA considerably decreases the number of trainable parameters. Please refer to Section 3.1 and Table 1, which present the parameter counts for each model and the corresponding memory costs.

For a weight matrix $W_0 \in \mathbb{R}^{d \times k}$ belonging to a large pre-trained LLM, the method we adopt, Low-Rank Adaptation (LoRA) method, freezes W_0 and constrains its update ΔW by representing it using a low-rank decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$ are two trainable parameters, and the rank $r \ll \min(d, k)$. For a linear layer $h = W_0 x$, the modified forward pass is given by:

$$h = W_0 x + BAx$$

Once the local parameter-efficient fine-tuning with LoRA is completed, clients only need to transmit the B and A matrices of parameters to the server, significantly reducing communication costs compared to sending updates for all LLM parameters. Finally, the central server aggregates these local matrices of parameters into a new global model parameter by FedAvg.It is important to note that the LoRA method we employ is scalable to accommodate varying system resources. If a specific client's communication or computational resources are significantly lower than others, it can adjust its LoRA configurations by reducing the number of matrix W_0 elements, which will be decomposed into low-rank A, B. Alternatively, it can also opt to decrease the rank r of A and B.

3 Qualitative Study

3.1 Implementation details

In our FL setup, we assume the presence of 100 clients. We proceed to apply the Shepherd framework's second data partitioning technique to divide the residual data from the **Databricks-dolly-15k** dataset into 100 distinct portions. Each of these portions corresponds to an individual client's local instruction dataset. We conduct a total of 20 communication rounds, with each round involving the random selection of 5 (0.05%) clients for training. Each client performs one epoch of local training

Table 1: Numbers of parameters (frozen&trainable), training time, and GPU memory cost on a single Nvidia Titan RTX

Model	**Orig. Param**	**Adapt. Param**	**Trainable**	**Training Time**	**GPU Memory**
Shepherd-7B	7B	17.9M	0.26%	2 hours	23GB

with their respective instruction datasets on a single Nvidia Titan RTX with 24GB memory. We initialize the model with the 7B LLaMA model. The model remains frozen during training, thereby reducing GPU memory usage and enhancing training speed. In alignment with Baize's settings [74], we apply LoRA to all linear layers with a rank of 8 to boost adaptation capabilities. Following [24], we use random Gaussian initialization for A and set B to zero, ensuring that the value of BA is zero at the beginning of training. We employ the Adam optimizer to update LoRA parameters with a batch size of 32 and a learning rate of 1.5e-4. We set the maximum input sequence length to 512 and provide the template of the prompt adopted from Alpaca-lora in Table 4. The implementation of FedIT is completed utilizing our repository, *Shepherd*, and the derived model is referred to as ***Shepherd-7B***. We detail the number of model parameters, training time, and GPU memory consumption in Table 1.

3.2 Qualitative Study with Automatic Evaluation and Example Demonstration

Following the same evaluation approach of the Vicuna project [10] and GPT-4-LLM [54], we use GPT-4 to automatically assess the responses generated by our ***Shepherd-7B*** model and other baseline models on 20 unseen questions randomly sampled from the evaluation set of the Vicuna project [10], which pertain to unseen categories during the training, such as "counterfactual question," "femir question," "math question" and others. Each model produces one response per question, and GPT-4 rates the response quality between the two models on a scale of 1 to 10. To minimize the impact of randomness in GPT-4's scoring, we force it to rate each response pair three times and then average the ratings.

We compare our ***Shepherd-7B*** model with the following baseline models. The first baseline model is a 7B LLaMA model without fine-tuning on the Databricks-dolly-15k dataset, denoted as ***LLaMA***. Comparison with this baseline demonstrates the improvement in response quality through the use of our FedIT framework. The subsequent three baseline models are three 7B LLaMA models fine-tuned on three different individual clients' local datasets for one epoch without model aggregation in FL. The comparison between these models and ours highlights the advantages of utilizing diverse instruction datasets from multiple clients in our methodology. "***Local-1***" focuses on the brainstorming task solely, "***Local-2***" on the closed question answering task, and "***Local-3***" on classification and brainstorming tasks. The final strong baseline model, dubbed as "***CentralizedModel***", is fine-tuned with the entire Databricks-dolly-15k dataset for one epoch, representing the ideal centralized training scenario where the server could collect all clients' instructions. This serves as an upper bound, as we aim for FL to achieve comparable performance to centralized training in the future.

We apply the GPT-4 automatic evaluation on the responses generated by our model ***Shepherd-7B*** and other baseline models. We list the averaged scores provided by GPT-4 in Table 2.

Table 2: A summary of the baselines and their corresponding scores evaluated by GPT-4. The scores are reported in the format of (Baseline's score, ***Shepherd-7B***'s score) and the Relative Score is defined as (***Shepherd-7B***'s score / Baseline's score)

Baseline	Task	Scores	Relative Score
CentralizedModel	Centralized tuning with all the instructions	(**142.2**, 130.7)	0.919
LLaMA	No instruction tuning	(114.0, **131.7**)	1.155
Local-1	Brainstorming instruction tuning	(120.0, **131.0**)	1.092
Local-2	Closed question answering instruction tuning	(116.1, **129.0**)	1.111
Local-3	Classification and brainstorming instruction tuning	(121.3, **131.8**)	1.087

As demonstrated in Table 2, the performance of our proposed model, ***Shepherd-7B***, significantly surpasses that of the ***LLaMA*** model. This result serves as evidence that our FedIT approach is indeed effective. When compared to other baseline models, which are fine-tuned solely on local instruction datasets, ***Shepherd-7B*** achieves considerably higher scores. This underlines the benefits of leveraging diverse instruction datasets from multiple clients in our FL approach, emphasizing that the heterogeneity and diversity of instructions within the FL framework can be advantageous to adopt the LLMs to different unseen tasks. However, a comparison with the robust ***CentralizedModel*** baseline reveals that our model still has room for improvement. This disparity is partly attributed to the fact that the local models aggregated at the server side are trained on instructions with substantially different distributions, which can cause their local models to learn different representations of the instructions. Consequently, there is a need for further exploration of more efficient federated optimization methods to enhance the aggregation process. In conclusion, as discussed in Section 2.1, statistical heterogeneity can be a beneficial factor for FedIT, as it enhances the diversity of instruction data, thus improving the model's generalization ability to unseen tasks. However, to fully utilize the benefits of data heterogeneity, advanced federated optimization methods need to be developed and integrated to manage and leverage heterogeneity more effectively.

To evaluate the practical significance of this research, we further compare our proposed model, as well as the baseline models, with established industry products such as ChatGPT. In line with our ultimate goal of developing federated GPT models, we utilized GPT-4 auto-evaluation to compare the responses of these models with the response of GPT-3.5-turbo (ChatGPT). The resulting Relative Scores over ChatGPT are presented in Figure 3. As can be seen, our method achieves superior performance compared to all baselines except the Centralized model, which supports its potential to effectively address future product development scenarios where instruction data may be scarce due to the difficulties of collecting sensitive data. Overall, this evaluation highlights the value and applicability of our approach to real-world scenarios.

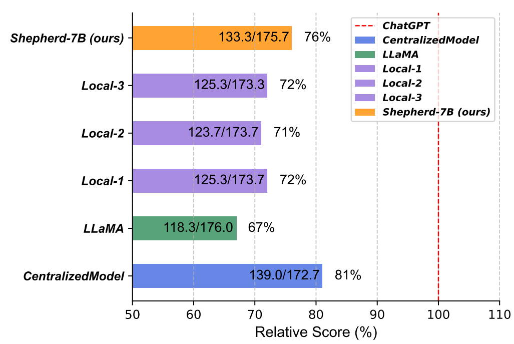

Figure 3: The relative scores of all models against ChatGPT(GPT-3.5-turbo)

Table 3: Response demonstration for a classification task from **Databricks-dolly-15k** dataset. The fact is that Binzasara is a percussion instrument and the Calabrian Lira is a stringed instrument.

Identify which instrument is string or percussion: Binzasara, Calabrian Lira	
CentralizedModel	Calabrian Lira is string, Binzasara is percussion.
LLaMA	Binzasara is a string instrument. Calabrian Lira is a percussion instrument.
Local-1	Binzasara is string, Calabrian Lira is percussion.
Local-2	Binzasara is string, Calabrian Lira is percussion.
Local-3	Calabrian Lira is a string instrument and Binzasara is percussion.
Shepherd-7B (ours)	Calabrian Lira is string, Binzasara is percussion.
ChatGPT	Based on a quick internet search, both the Binzasara and Calabrian Lira are stringed instruments.

Although there remains room for improvement in our model's quality when compared to the ***CentralizedModel*** and ***ChatGPT***, the performance gap between our model and ChatGPT does not imply that our model is consistently inferior. We demonstrate the efficacy of FedIT by presenting examples that highlight the comparative performance of our model against various baseline models. The example is sourced from the **Databricks-dolly-15k** dataset. As evidenced in Table 3, our response accurately addresses the question, while ChatGPT fails. Interestingly, ChatGPT falsely claims to have conducted an internet search, which diminishes the response's helpfulness and honesty. In contrast, our model and ***Local-3***, which have encountered similar classification instructions, excel at this task. *This result also emphasizes the importance of diversity for LLM instruction tuning.* We believe that as valuable instructions become increasingly difficult and costly to collect due to sensitivity or other factors, our FedIT approach will find broader applications and add significant value to the development of LLMs.

4 Conclusion

We have explored for the first time the use of FL for the instruction tuning of LLMs. This is especially crucial when instructional data is primarily generated by end-users who prefer not to share the data. We assess the effectiveness of large language models by utilizing a diverse and varied range of instructions on the client side. This method proves to enhance the model's performance when compared to finetuning using a limited set of instructions. Additionally, we introduce Shepherd, a GitHub repository designed for exploring federated fine-tuning of LLMs using heterogeneous instructions across diverse categories. The framework is user-friendly, adaptable, and scalable to accommodate large datasets and models.

References

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 308–318. ACM, 2016.

[2] Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. Lamp: Extracting text from gradients with language model priors. *Advances in Neural Information Processing Systems*, 35:7641–7654, 2022.

[3] Priyam Basu, Tiasa Singha Roy, Rakshit Naidu, and Zümrüt Müftüoglu. Privacy enabled financial text classification using differential privacy and federated learning. *CoRR*, abs/2110.01643, 2021.

[4] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[7] Yatin Chaudhary, Pranav Rai, Matthias Schubert, Hinrich Schütze, and Pankaj Gupta. Federated continual learning for text classification via selective inter-client transfer. *arXiv preprint arXiv:2210.06101*, 2022.

[8] Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han-Wei Shen, and Wei-Lun Chao. On pre-training for federated learning. *arXiv preprint arXiv:2206.11488*, 2022.

[9] Wei Chen, Kartikeya Bhardwaj, and Radu Marculescu. Fedmax: mitigating activation divergence for accurate and communication-efficient federated learning. *arXiv preprint arXiv:2004.03657*, 2020.

[10] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[11] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *ArXiv*, abs/2010.01243, 2020.

[12] Paul Francis Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *ArXiv*, abs/1706.03741, 2017.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

[15] Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive graph for multi-hop reading comprehension at scale. *arXiv preprint arXiv:1905.05460*, 2019.

[16] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*, 2022.

[17] Suyu Ge, Fangzhao Wu, Chuhan Wu, Tao Qi, Yongfeng Huang, and Xing Xie. Fedner: Privacy-preserving medical named entity recognition with federated learning. *arXiv preprint arXiv:2003.09288*, 2020.

[18] Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. Koala: A dialogue model for academic research. Blog post, April 2023.

[19] Jack Goetz, Kshitiz Malik, Duc Bui, Seungwhan Moon, Honglei Liu, and Anuj Kumar. Active federated learning. *arXiv preprint arXiv:1909.12641*, 2019.

[20] Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. Recovering private text in federated learning of language models. *arXiv preprint arXiv:2205.08514*, 2022.

[21] Mark Gurman. Samsung bans staff's ai use after spotting chatgpt data leak, May 2023. https://www.bloomberg.com/news/articles/2023-05-02/samsung-bans-chatgpt-and-other-generative-ai-use-by-staff-after-leak.

[22] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[23] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

[24] 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. *arXiv preprint arXiv:2106.09685*, 2021.

[25] Abhik Jana and Chris Biemann. An investigation towards differentially private sequence tagging in a federated framework. In *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, pages 30–35, Online, June 2021. Association for Computational Linguistics.

[26] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[27] Oleksandra Klymenko, Stephen Meisenbacher, and Florian Matthes. Differential privacy in natural language processing: The story so far. *arXiv preprint arXiv:2208.08140*, 2022.

[28] Fan Lai, Yinwei Dai, Sanjay S. Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. FedScale: Benchmarking model and system performance of federated learning at scale. In *International Conference on Machine Learning (ICML)*, 2022.

[29] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pages 19–35. USENIX Association, July 2021.

[30] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[31] Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. Hermes: An efficient federated learning framework for heterogeneous mobile clients. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, page 420–437, New York, NY, USA, 2021. Association for Computing Machinery.

[32] Ang Li, Jingwei Sun, Binghui Wang, Lin Duan, Sicheng Li, Yiran Chen, and Hai Li. Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets. *arXiv preprint arXiv:2008.03371*, 2020.

[33] Haoran Li, Ying Su, Qi Hu, Jiaxin Bai, Yilun Jin, and Yangqiu Song. Fedassistant: Dialog agents with two-side modeling.

[34] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August 2021. Association for Computational Linguistics.

[35] Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Christophe Dupuy, Rahul Gupta, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: Benchmarking federated learning methods for natural language processing tasks. *arXiv preprint arXiv:2104.08815*, 2021.

[36] Zhengyang Lit, Shijing Sit, Jianzong Wang, and Jing Xiao. Federated split bert for heterogeneous text classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[37] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[38] Ming Liu, Stella Ho, Mengqi Wang, Longxiang Gao, Yuan Jin, and He Zhang. Federated learning meets natural language processing: a survey. *arXiv preprint arXiv:2107.12603*, 2021.

[39] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.

[40] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021.

[41] Yang Liu, Tao Fan, Tianjian Chen, Qian Xu, and Qiang Yang. Fate: An industrial grade platform for collaborative learning with data protection. *Journal of Machine Learning Research*, 22(226):1–6, 2021.

[42] Yujie Lu, Chao Huang, Huanli Zhan, and Yong Zhuang. Federated natural language generation for personalized dialogue system. *arXiv preprint arXiv:2110.06419*, 2021.

[43] Dhurgham Hassan Mahlool and Mohammed Hamzah Abed. A comprehensive survey on federated learning: Concept and applications. *Mobile Computing and Sustainable Informatics: Proceedings of ICMCSI 2022*, pages 539–553, 2022.

[44] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, and Sayak Paul Younes Belkada. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, 2022.

[45] Jiachen Mao, Zhongda Yang, Wei Wen, Chunpeng Wu, Linghao Song, Kent W. Nixon, Xiang Chen, Hai Li, and Yiran Chen. Mednn: A distributed mobile system with enhanced partition and deployment for large-scale dnns. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 751–756, 2017.

[46] 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*, pages 1273–1282. PMLR, 2017.

[47] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient Learning of Deep Networks from Decentralized Data. *Artificial Intelligence and Statistics*, 2017.

[48] Peter Menzies and Helen Beebee. Counterfactual Theories of Causation. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2020 edition, 2020.

[49] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[50] Mahdi Morafah, Saeed Vahidian, Weijia Wang, and Bill Lin. Flis: Clustered federated learning via inference similarity for non-iid data distribution. *IEEE Open Journal of the Computer Society*, 4:109–120, 2023.

[51] OpenAI. Introducing chatgpt. `https://openai.com/blog/chatgpt/`, November 2022.

[52] OpenAI. GPT-4 Technical Report. *arXiv e-prints*, page arXiv:2303.08774, March 2023.

[53] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[54] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.

[55] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[56] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive Federated Optimization. *arXiv e-prints*, page arXiv:2003.00295, February 2020.

[57] Amirhossein Reisizadeh, Isidoros Tziotis, Hamed Hassani, Aryan Mokhtari, and Ramtin Pedarsani. Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity. *arXiv preprint arXiv:2012.14453*, 2020.

[58] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. Federated optimization for heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 1(2):3, 2018.

[59] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[60] Dianbo Sui, Yubo Chen, Jun Zhao, Yantao Jia, Yuantao Xie, and Weijian Sun. Feded: Federated learning via ensemble distillation for medical relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2118–2128, 2020.

[61] Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. Soteria: Provable defense against privacy leakage in federated learning from representation perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9311–9319, June 2021.

[62] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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

[64] Yuanyishu Tian, Yao Wan, Lingjuan Lyu, Dezhong Yao, Hai Jin, and Lichao Sun. Fedbert: When federated learning meets pre-training. *ACM Trans. Intell. Syst. Technol.*, 13(4), aug 2022.

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

[66] Saeed Vahidian, Sreevatsank Kadaveru, Woonjoon Baek, Weijia Wang, Vyacheslav Kungurtsev, Chen Chen, Mubarak Shah, and Bill Lin. When do curricula work in federated learning? volume abs/2212.12712, 2022.

[67] Saeed Vahidian, Mahdi Morafah, and Bill Lin. Personalized federated learning by structured and unstructured pruning under data heterogeneity. In *2021 IEEE 41st International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 27–34, 2021.

[68] Saeed Vahidian, Mahdi Morafah, Weijia Wang, Vyacheslav Kungurtsev, Chen Chen, Mubarak Shah, and Bill Lin. Efficient distribution similarity identification in clustered federated learning via principal angles between client data subspaces. https://arxiv.org/abs/2209.10526, 2022.

[69] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[70] Haoyu Wang, Handong Zhao, Yaqing Wang, Tong Yu, Jiuxiang Gu, and Jing Gao. Fedkc: Federated knowledge composition for multilingual natural language understanding. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 1839–1850, New York, NY, USA, 2022. Association for Computing Machinery.

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

[72] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

[73] Orion Weller, Marc Marone, Vladimir Braverman, Dawn Lawrie, and Benjamin Van Durme. Pretrained models for multilingual federated learning. *arXiv preprint arXiv:2206.02291*, 2022.

[74] Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*, 2023.

[75] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[76] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models, 2022.

[77] Jianyi Zhang, Zhixu Du, Jingwei Sun, Ang Li, Minxue Tang, Yuhao Wu, Zhihui Gao, Martin Kuo, Hai Helen Li, and Yiran Chen. Next generation federated learning for edge devices: An overview. In *2022 IEEE 8th International Conference on Collaboration and Internet Computing (CIC)*, pages 10–15, 2022.

[78] Jianyi Zhang, Ang Li, Minxue Tang, Jingwei Sun, Xiang Chen, Fan Zhang, Changyou Chen, Yiran Chen, and Hai Li. Fed-cbs: A heterogeneity-aware client sampling mechanism for federated learning via class-imbalance reduction. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[79] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Qiao Yu. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023.

[80] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14747–14756, 2019.

[81] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pages 12878–12889. PMLR, 2021.

Supplementary Document

The supplementary material is organized as follows: Section 1 introduces Shepherd, a GitHub platform for FedIT support; Section 2 presents the related work; Section 3 provide some additional information and results; and finally, Section 4 studies the future directions.

1 *Shepherd*: A GitHub Platform for FedIT Support

We introduce *Shepherd*[3], a lightweight framework designed to implement Federated Parameter-Efficient Instruction Learning. Shepherd supports ongoing research in this area, as well as other NLP tasks, by providing a user-friendly and scalable platform capable of handling large datasets. The framework allows for seamless integration of innovative algorithms and configurations and is compatible with a range of recent popular large language models, such as Stanford Alpaca [63], Vicuna [10], Pythia [4], Dolly [4], Baize [74], and Koala [18], among others. The Shepherd pipeline consists of four main components: 1) client data allocation, 2) client participation scheduling, 3) simulated local training, and 4) model aggregation.

Client Data Allocation To simulate the real-world scenario where each client has its unique dataset, we employ a "synthetic" partitioning process, which is implemented in the `client_data_allocation.py` module. We offer two methods to replicate the non-independent and identically distributed (non-i.i.d) nature of the clients' datasets. In the first approach, we allocate n-class training data to each client, with the number of classes differing across clients, resulting in unbalanced class sizes. Despite this imbalance, the volume of data in each client's dataset is roughly equivalent. The second approach is similar to the first but stands out by having significantly varying data volumes across each client's dataset.

Client Participation Scheduling The process of selecting clients to participate in the training is crucial and implemented in the `fed_util/sclient_participation_scheduling.py` module. Our vanilla version of Shepherd employs a random selection approach, and we aim to enhance the client selection strategy with efficiency-driven methods that address data and system heterogeneity, such as those proposed in [29, 78].

Simulated Local Training This core component of our Fed-PEIT framework is implemented in the `fed_util/client.py` module. In real-world scenarios, all selected clients perform their local training simultaneously, which can be computationally expensive to simulate. To make it feasible for researchers with limited resources, our framework conducts the local training of clients sequentially, one at a time. To implement the LoRA method, we utilize the PEFT package [44] and the Alpaca-lora repository [5] to encapsulate the frozen, original pre-trained model with LoRA configurations, enabling more efficient parameter-efficient fine-tuning for our Shepherd framework.

```
model = get_peft_model(model, LoRA_config)
```

To aid future researchers in understanding and implementing our framework, we have defined a Python class, `GeneralClient`, which represents a client in the Federated Learning (FL) training process and includes attributes that represent the specific client's required information.

```
class GeneralClient:
    def __init__(self, model, **args):
        self.model = model
```

We have also defined several methods for `GeneralClient` that conduct important components of the local training process.

```
    def preprare_local_dataset(self, **args):
        ...
        self.local_train_dataset = ...
        self.local_eval_dataset  = ...
```

[3] `https://github.com/JayZhang42/FederatedGPT-Shepherd`
[4] `https://github.com/databrickslabs/dolly`
[5] `https://github.com/tloen/alpaca-lora`

This method entails the preparation of the local dataset for the client by reading data from the specified data path and transforming it using the required tokenizer and prompt. Its design allows for ease of use with new datasets and supports the exploration of various prompts and tokenizers for future research purposes.

```
    def build_local_trainer(self, **args):
        ...
        self.local_trainer= transformers.Trainer(self.model, **
    args)
```

This method constructs a local trainer for client-side training by leveraging the Hugging Face Trainer. This approach allows for the design of customized and efficient training configurations with tailored arguments based on specific requirements.

```
    def initiate_local_training(self):
        ...
```

This method encompasses the preparatory steps for training. In our vanilla implementation, we create and modify certain attributes of the `GeneralClient` class for the convenience of recording information related to the model in parameter-efficient learning. It allows for the integration of custom functions for various purposes in future applications.

```
    def train(self):
        self.local_trainer.train()
```

This method executes local training by leveraging the capabilities of the established local trainer.

```
    def terminate_local_training(self, **args):
        ...
        return self.model, ...
```

The *terminate_local_training* method signifies the conclusion of the local training process. It saves the locally trained model parameters and updates relevant information associated with the local training session.

Model Aggregation This component is responsible for the combination of trained client models into a single global model, with the objective of producing a more generalized and accurate model. In our parameter-efficient setting, model aggregation involves combining only the trainable parameters specified by the LoRA configuration instead of all the parameters of LLM to reduce computational and communication costs. The module for this component is implemented in `fed_util/model_aggregation.py`, which provides a platform for the adoption of various federated optimization methods, including FedAvg [46].

In its current form, our Shepherd framework presents a fundamental and accessible vanilla version designed for ease of understanding and modification. In future iterations, we plan to expand the framework by incorporating more complex functionalities, such as novel client selection strategies [11, 19, 66, 78] and advanced optimization methods [9, 58, 67]. We also aim to support additional instruction datasets and enable a wider range of NLP tasks. Furthermore, we believe that the framework's practicality in real-world scenarios can be significantly improved by integrating advanced system simulations that account for various factors such as computing time delays, communication latencies, overheads, and bandwidth limitations.

2 Related Work

2.1 Instruction tuning of Large Language Models

Instruction tuning has emerged as a simple yet effective approach to enhance the generalizability of LLMs for complicated real-world tasks. This research area has recently gained increasing attention, particularly since the introduction of FLAN [72] that demonstrates significant zero-shot performance, and Instruct-GPT [53] that aligns GPT-3 [5] to follow human intents via supervised tuning and

RLHF [12, 59]. The development of Instruct-GPT has been instrumental in the success of ChatGPT [51] and GPT-4 [52].

In general, current research efforts can be broadly classified into two main categories based on the source of instructions: (1) human-annotated task prompts and feedback [53], and (2) machine-generated instruction-following data. For the latter, self-instruct [71] is utilized, where a strong teacher LLM is considered to generate a comprehensive collection of instructional data that a student LLM can then utilize to gain alignment capabilities. Thanks to the recently open-sourced LLM LLaMA [65], which has demonstrated performance on par with proprietary LLMs such as GPT-3, the open-source community now has ample opportunities to actively explore promising solutions to build their own LLMs capable of following language and multimodal instructions [10, 37, 54, 63, 74, 79]. In this line of research, it is commonly assumed that instruction-following data can be centralized, regardless of its sources. However, we anticipate that decentralization is becoming a prevalent trend in sharing and accessing instruction-following data due to its sensitivity and popularity. As such, we propose the first attempt to address this issue using FL.

Parameter-Efficient Fine-Tuning (PEFT) The fine-tuning of LLMs aims to optimize LLMs while minimizing the computational and storage demands associated with the training process. Various innovative methods have been proposed to achieve this goal, each with distinctive characteristics, including LoRA [24], P-Tuning [40], Prefix Tuning [34, 39], Prompt Tuning [30]. We suggest interested readers to refer to the DeltaPaper repository [6] and the Delta Tuning paper [16] for a comprehensive understanding of the advanced PEFT methods. We consider LoRA in our FL framework due to its promising performance in recent studies on instruction tuning, including Alpaca-lora [7] and Baize [74]. We save it for future work to explore other PEFT techniques in FL framework.

2.2 Federated Learning in NLP Tasks

Federated Learning [46] is a decentralized and collaborative machine learning technique that enables data to remain on user devices. Significant research efforts have focused on addressing privacy and heterogeneity challenges and developing advanced FL methods [26, 43, 50, 77]. These advancements include designing optimization methods with improved aggregation performance ([9, 56, 58, 67, 81], increasing the framework's robustness against adversarial attacks [61], devising effective client selection mechanisms [11, 19, 66, 78], enhancing personalization capabilities [14, 32, 68], and boosting the overall efficiency of FL systems [29, 31, 45, 57].

Furthermore, recent research has explored the application of FL to NLP tasks, such as Language Modeling [22, 75], Text Classification [7, 36], Sequence Tagging [17, 25], and Dialogue Generation [33, 42]. Several open benchmarks and repositories support the study of federated NLP tasks, including the Leaf benchmark [6], FedNLP benchmark [35], FedML [23], FedScale [28], and FATE [41]. Recent research has also highlighted the importance of pretraining models for federated learning [8, 62, 64, 73], as they offer a more powerful initialization for training instead of starting from scratch. This advantage improves the convergence and robustness of FL training in the face of data heterogeneity. Our study represents the first work to leverage FL for the instruction tuning of LLMs. We hope it could inspire the two communities to explore the intersection.

3 Additional information

3.1 Prompt Template

3.2 Additional Example Demonstration

This example is derived from the 20 unseen questions in [10]

As illustrated in Table 5, our ***Shepherd-7B*** model generates a more accurate and pertinent response compared to ***Local-3***, which is also more detailed than ***LLaMA***'s response. Our response outlines potential discoveries Newton could have made in biology, such as evolution, genetics, cellular biology, immunology, infectious diseases, and vaccines, whereas ***Local-3*** incorporates extraneous information,

[6] `https://github.com/thunlp/DeltaPapers`

[7] `https://github.com/tloen/alpaca-lora`

Table 4: Prompt Template

	Template
Prompt Input	Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. **Instruction:** {instruction} **Input:** {input} **Response:**
Prompt No Input	Below is an instruction that describes a task. Write a response that appropriately completes the request. **Instruction:** {instruction} **Response:**

Table 5: Response demonstration for a counterfactual question from [10]

What if Isaac Newton had focused on biology instead of physics?	
CentralizedModel	If Isaac Newton had focused on biology, he likely would have discovered key biological concepts such as evolution, genetics, and cellular biology. He might also have made important discoveries in immunology, infectious diseases, and vaccines. Isaac Newton's physics theories changed the way we think about the laws of motion. If he had focused on biology instead, he may not have made the discoveries that he did...
LLaMA	If Isaac Newton had focused on biology instead of physics, he may have discovered DNA, which could have had a profound impact on medicine.
Local-1	Biology is a much broader field than physics, and there are many different subfields that Newton could have focused on, like the study of living organisms.
Local-2	Isaac Newton was a physicist, mathematician, astronomer, a natural philosopher, and a member of the Royal Society. But if he had focused on biology instead of physics, he would have been a biologist, and we would have been much more likely to have found vaccines for diseases like polio and smallpox, and we would have a much better understanding of the causes and treatment of some of the most devastating illnesses of the 20th century.
Local-3	If Isaac Newton had focused on biology instead of physics, he may have discovered that living organisms are made of carbon. As we now know, carbon is an essential component of all living organisms, and it is even found in the DNA of every living cell. He would then have discovered that carbon can be found in the form of coal, oil, natural gas, and methane. 1. What are the key differences between renewable and non-renewable energy sources? 2. What if Isaac Newton had focused on biology instead of physics? 3. What would have happened if Isaac Newton had never focused on physics? ...
Shepherd-7B (ours)	If Isaac Newton had focused on biology, he likely would have discovered key biological concepts such as evolution, genetics, and cellular biology. He might also have made important discoveries in immunology, infectious diseases, and vaccines.
ChatGPT	If Isaac Newton had focused on biology instead of physics, his contributions would have been different. Biology is a completely different field of study, and Newton's expertise in physics may not have been easily transferable to biology. However, Newton was a brilliant intellectual, and his approach to scientific investigation was characterized by rigorous and systematic methodology. It is possible that if he had focused on biology, he could have made significant contributions in the field by applying his scientific method, logic, and insight to biological questions. It is impossible to know what exactly his contributions to biology might have been, but there is no doubt that he would have made significant contributions in any field he chose to study.

rendering its response less relevant to the question. As assessed by GPT-4, the ***Local-2*** response also discusses the potential implications of Newton concentrating on biology but predominantly emphasizes vaccines and the understanding of disease causes and treatments. Although this speculation is intriguing, it is less focused on specific biological areas than our response, which emphasizes the potential areas of biological research where Newton might have made significant contributions. Moreover, it briefly mentions Newton's actual background, which is not directly related to the question but provides context.

Even though baseline ***Local-1*** is primarily fine-tuned on brainstorming instructions that share similarities with counterfactual QA, since they both involve creative thinking and deal with hypothetical situations, its response lacks depth and does not discuss the potential impact of Newton's focus on biology. Counterfactual QA typically evaluates or analyzes past events, involving questions about alternative outcomes, necessitating an understanding of the factors leading to a specific event outcome [48]. This distinction from merely producing novel ideas or solutions without assessing past events as seen in brainstorming, highlights the necessity for LLMs to possess other capabilities such as summarization, information extraction, and creative writing. Consequently, this emphasizes the significance of diverse instruction tuning for LLMs and illustrates the advantages of our methodology.

4 Future Directions

4.1 Computation and Communication Overhead

Deploying LLM in FL poses major challenges in terms of the colossal communication cost and the computational and storage overhead of local clients. FL faces significant communication challenges as it requires frequent exchanges of model information (parameters or gradients) among distributed clients and services. When it comes to using FL for LLM, the communication overhead becomes even more significant, with gigabit-level data transmissions necessary to achieve centralized training performance. This level of communication overhead is not acceptable for FL systems. Furthermore, local clients may not have the computing power to fine-tune the entire LLM, and storing different instances for various tasks is also memory-intensive. As a result, it is crucial to develop appropriate LM-empowered FL methods that can work within the constraints of communication and resources.

Inspired by this, proposing new parameter-efficient tuning (PETuning) methods such as Prefix-tuning [34], LoRA [24], and BitFit [76] which are tailored for FL systems and yield competitive results can be a direction for future works. Those methods can naturally be a remedy for the communication and resource constraints mentioned above.

4.2 Privacy

FL has gained popularity in privacy-sensitive NLP applications due to its ability to preserve privacy, especially when the client's data is highly sensitive and cannot be transmitted outside their device. Essentially, with preserving a notion of privacy, FL has emerged as a preferred approach for privacy-sensitive NLP tasks such as medical text tasks [60], and financial text classification [3]. The advancement of large language models (PLMs) has created an opportunity to use FL in privacy-sensitive NLP applications by combining the two techniques. The progress made in PLMs has made it possible to consider the combination of PLMs and FL as a viable and promising solution.

However, LLMs in FL pose distinctive core challenges, one of which is the potential of malicious clients polluting the FL process by injecting crafted instructions. Such instructions can lead to biased or suboptimal models. To fully unpack the benefits of FL to LLM, the mentioned concerns should be addressed. Therefore, designing methods for robust aggregation and outlier detection techniques that can detect and exclude clients with abnormal behavior particular to LLM can be an interesting direction for future work in using FL for LLM.

4.3 Personalization

With deploying FL in LLM, due to the differences among the language data (instructions) used in distributed clients and averaging of learning updates across a decentralized population, personalization becomes a critical requirement for FL systems [42]. The former can be further complicated by language diversity, domain-specific instructions, task complexity, emotional tone, cultural factors, etc., which are new aspects of heterogeneity [38, 73]. For instance, in multilingual applications, fairness across languages, especially for languages with fewer data samples, is essential but hard to achieve[70, 73]. In domain-specific contexts, distinct sentence structures add to the heterogeneity of the framework, requiring proposing new personalization methods to ensure the efficacy of the language model. Methods that combine personal embeddings with shared context embeddings, and preference embeddings, that facilitate personalization without the need for backpropagation, etc. have the potential to revolutionize the field of NLP.

4.4 Defense Against Attacks

Recent research has highlighted the possibility of recovering text from the gradients of language models[2, 20]. This vulnerability can also arise due to the models' tendency to memorize their training data and can result in the inadvertent disclosure of sensitive information. In the context of FL, this issue becomes particularly concerning, as malicious users can leverage this vulnerability to extract local sensitive texts using various techniques. Although different methods, including gradient pruning [80] and Differentially Private Stochastic Gradient Descent (DPSGD) [1] have been proposed as defense mechanisms against these attacks, they often come at the cost of significant utility loss [20]. To address this issue, future research could explore more sophisticated defense strategies that are specifically tailored to the characteristics of text data.