FedLEKE: Federated Locate-then-Edit Knowledge Editing for Multi-Client Collaboration

Zongkai Zhao[1*], **Guozeng Xu**[1*], **Xiuhua Li**[1†], **Kaiwen Wei**[2†], **Jiang Zhong**[2]
[1]School of Big Data & Software Engineering, Chongqing University, China
[2]College of Computer Science, Chongqing University, China
{zongkaizhao, archipes}@stu.cqu.edu.cn, {lixiuhua, weikaiwen, zhongjiang}@cqu.edu.cn

Abstract

Locate-then-Edit Knowledge Editing (LEKE) is a key technique for updating large language models (LLMs) without full retraining. However, existing methods assume a single-user setting and become inefficient in real-world multi-client scenarios, where decentralized organizations (e.g., hospitals, financial institutions) independently update overlapping knowledge, leading to redundant mediator knowledge vector (MKV) computations and privacy concerns. To address these challenges, we introduce **Fed**erated **L**ocate-then-**E**dit **K**nowledge **E**diting (FedLEKE), a novel task that enables multiple clients to collaboratively perform LEKE while preserving privacy and reducing computational overhead. To achieve this, we propose FedEdit, a two-stage framework that optimizes MKV selection and reuse. In the first stage, clients locally apply LEKE and upload the computed MKVs. In the second stage, rather than relying solely on server-based MKV sharing, FedLEKE allows clients retrieve relevant MKVs based on cosine similarity, enabling knowledge re-edit and minimizing redundant computations. Experimental results on two benchmark datasets demonstrate that FedEdit retains over 96% of the performance of non-federated LEKE while significantly outperforming a FedAvg-based baseline by approximately twofold. Besides, we find that MEMIT performs more consistently than PMET in the FedLEKE task with our FedEdit framework. Our code is available at https://github.com/zongkaiz/FedLEKE.

1 Introduction

Locate-then-Edit Knowledge Editing (LEKE) has emerged as a key paradigm for updating large language models (LLMs) by directly identifying and modifying model parameters associated with newly acquired knowledge, eliminating the need for costly full-model retraining (Meng et al., 2022a,b; Gupta et al., 2024). It has proven effective in mitigating hallucinations (Huang et al., 2024), detoxifying outputs (Wang et al., 2024), and improving factual recall (Wei et al., 2023a; Zhang et al., 2024).

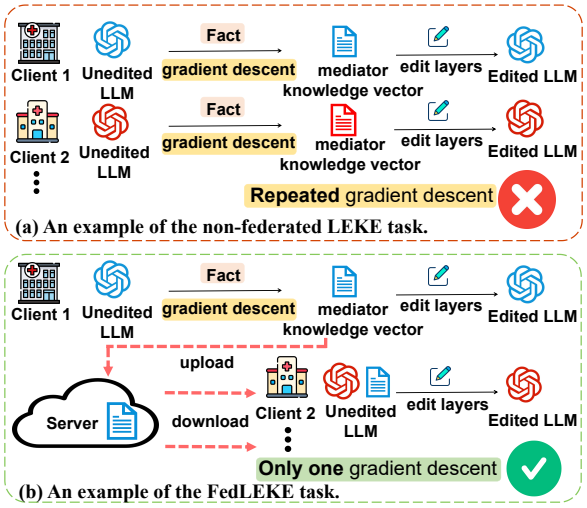

Figure 1: Comparison between (a) non-federated LEKE and (b) the proposed FedLEKE task, where the former requires the computation of the mediator knowledge vector multiple times for the same knowledge through gradient descent, while the latter computes it only once.

However, existing methods are all conducted in single-client scenarios. Considering real-life applications, as shown in Fig. 1(a), traditional LEKE methods suffer from redundant gradient descent computations of mediator knowledge vectors (MKVs) (Meng et al., 2022a,b; Li et al., 2024), leading to inefficiencies in knowledge updates, especially for organizations within the same domain (e.g., different hospitals) that often process overlapping information. This not only exacerbates these inefficiencies but also raises privacy concerns due to data sharing (El Ouadrhiri and Abdelhadi, 2022; Yazdinejad et al., 2024). To mitigate these issues, federated learning (FL) (Konečnỳ, 2016; McMahan

[*]Equal Contribution
[†]Corresponding Author

Findings of the Association for Computational Linguistics: ACL 2025, pages 14247–14258

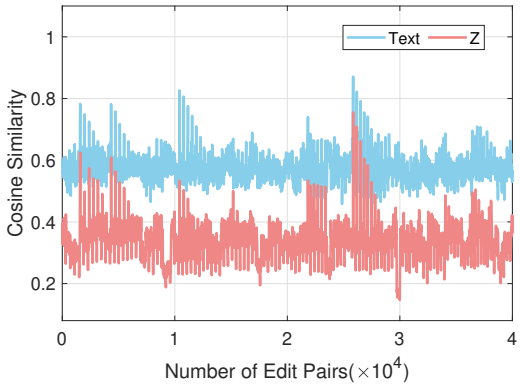

Figure 2: Cosine similarity between core text in zsRE dataset and the corresponding z_i vectors.

et al., 2017a; Yang et al., 2018) enables collaborative model training while preserving data privacy, making it particularly suitable for such sensitive domains like healthcare and finance.

To extend the LEKE task to federated settings, we propose a new task: **Fed**erated **L**ocate-then-**E**dit **K**nowledge **E**diting (FedLEKE), enabling multiple clients to collaboratively edit knowledge while reducing computational costs and preserving privacy. As shwon in Fig. 1(b), In FedLEKE, each client runs the LEKE algorithm locally to generate MKVs representing knowledge updates. These MKVs are uploaded to a central server, where they are stored and shared, preventing redundant computations. When a predefined time slot arrives, clients retrieve relevant MKVs from the server to refine their knowledge.

To accomplish FedLEKE, several critical challenges need to be addressed: (1) *How to define MKVs for client update*: Unlike traditional LEKE in federated settings, where homogeneous clients redundantly recompute MKVs multiple times for identical knowledge edits, FedLEKE computes them only once and shares them across clients via a central server, so selecting appropriate MKVs for upload is crucial. They must effectively encode essential knowledge while remaining computationally efficient. (2) *How to retrieve relevant MKVs for client download*: Efficient retrieval is crucial to minimize computational and storage costs while ensuring clients access only the most relevant MKVs. A key issue is dynamically selecting MKVs that best match each client's needs, balancing retrieval efficiency and knowledge quality.

To address the first challenge, we explored various representations for MKVs and found z_i vectors introduced in Meng et al. (2022b) to be particularly suitable. As shown in Fig. 2, our analysis on the zsRE dataset (Levy et al., 2017) revealed strong semantic alignment textual knowledge and corresponding z_i vectors. Statistical analysis on 2,000 selected edit pairs (cosine similarity > 0.65) confirmed a strong positive correlation, with a Pearson coefficient of 0.74 (Cohen et al., 2009). These findings show that z_i vectors effectively encode original knowledge while improving computational efficiency, making them well-suited as MKVs.

To address the second challenge, we propose **FedEdit**, it operates in two stages: first, at predefined intervals, clients apply existing LEKE algorithms to update multiple layers of their models, uploading the computed MKVs to the server. Then, in the re-editing stage, clients periodically evaluate the similarity between their local data and the vectors stored on the server. And a re-editing condition is established, if the similarity meets a predefined threshold, the server's vectors can be reused for further editing, allowing clients to refine their models without redundant computations.

We reorganize two large-scale counterfactual datasets zsRE and COUNTERFACT (Meng et al., 2022a) to simulate the FedLEKE task,. Extensive experiments on GPT-J (6B) (Wang and Komatsuzaki, 2021) and GPT-NeoX (20B) (Black et al., 2022) show that even in the FedLEKE setting, the proposed FedEdit method retains at least 96% of the performance of state-of-the-art methods in non-federated environments. The key contributions of this work are summarized as follows:

1) We introduce FedLEKE, a task enabling multi-client collaborative knowledge editing in dynamic scenarios. To the best of our knowledge, this is the first work to apply LEKE in the federated setting.

2) We introduce FedEdit, a two-stage editing framework designed to improve multi-client editing efficiency for related knowledge, where a re-editing condition is established to efficiently select mediator knowledge vectors from the server.

3) We reorganize the zsRE and COUNTERFACT datasets to simulate FedLEKE. Experimental results show that, under FedLEKE conditions, FedEdit achieves performance at or above 96% of that of state-of-the-art methods in non-federated settings.

2 Related Work

Locate-then-Edit Knowledge Editing. The locate-then-edit approach in knowledge editing

identifies and modifies specific weights in pre-trained models to achieve desired outputs (Mitchell et al., 2022; Yao et al., 2023). Various methods have been proposed within this framework(Wei et al., 2021). ROME (Meng et al., 2022a) updates the feedforward network to encode new knowledge, while MEMIT (Meng et al., 2022b) extends this for large-scale editing. PMET (Li et al., 2024) enhances MEMIT's performance with a residual attribution strategy. Additionally, ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b) use input prompts to locate and edit knowledge neurons. However, existing works do not address multi-client scenarios and multi-editing tasks (Song et al., 2022; Wei et al., 2023b). In this paper, we propose a federated locate-then-edit knowledge editing framework to improve editing efficiency in such settings.

Federated Learning in LLMs. Research on combining large language models (LLMs) and federated learning (FL) primarily focuses on pre-training and prompt engineering (Chen et al., 2024). Pre-trained models, trained on large datasets, serve as a foundation for FL, significantly reducing training time (Tan et al., 2022; Liu et al., 2024) and helping address data and system heterogeneity (Nguyen et al., 2022). Some studies incorporate pre-trained models into FL frameworks for various tasks (Agarwal et al., 2023; Zhang et al., 2023). Prompt-based techniques have shown strong performance in LLMs (Guo et al., 2023). The pFedPT framework personalizes models efficiently using personalized prompts (Li et al., 2023), while DiPrompT (Bai et al., 2024) applies adaptive prompts to tackle domain generalization challenges in FL. To the best of our knowledge, this is the first work to apply FL for optimizing LEKE in LLMs.

3 Method

In this section, we provide a detailed introduction to the FedLEKE task and the FedEdit framework. First, we discuss the relationship among the hidden states of each Transformer layer in the LLM and the relationship between the hidden states and the input in section 3.1, which is essential for calculating the MKVs. Next, we introduce the FedLEKE task and explain its connection to the LEKE task in section 3.2, and we also analyze how to optimize and solve it. Then, we focus on solving the LEKE task and extracting the relevant knowledge vector in section 3.3. Finally, we propose the FedEdit framework to address the FedLEKE task in section 3.4.

3.1 Preliminaries

This section introduces the foundational concepts of autoregressive and decoder-only LLM models, focusing on the relationship between the hidden states of each Transformer layer and the input. These foundations are essential for calculating the MKVs.

Autoregressive and decoder-only LLMs denoted as $\mathcal{F}_\theta$ encode input sequences x into z token sequences $x_1, ..., x_z$, which are processed through L Transformer decoder layers. The probability of the next token x_{z+1} is computed as:

$$\begin{aligned}\mathcal{F}_\theta(x_1, ..., x_z) &= \text{softmax}\left(W_{\text{E}}\gamma\left(h_z^{L-1} + a_z^L + m_z^L\right)\right) \\ &= \mathbb{P}\left(x_{z+1}|x_1, ..., x_z\right),\end{aligned} \tag{1}$$

where W_{E} and γ are the embedding matrix and layer norm, respectively, and a_z^L, m_z^L are the hidden states of the MHSA and FFN at the L-th layer. a_j^l, m_j^l for the j-th token at layer l are:

$$\begin{aligned}a_j^l &= W_{O^{\text{MHSA}}}^l \text{MHSA}^l\left(\gamma\left(h_1^{l-1}, h_2^{l-1}, ..., h_j^{l-1}\right)\right), \\ m_j^l &= W_{O^{\text{FFN}}}^l \sigma\left(W_I^l \gamma\left(h_j^{l-1}\right)\right),\end{aligned} \tag{2}$$

where $W_{O^{\text{MHSA}}}$ and $W_{O^{\text{FFN}}}$ are weights for MHSA and FFN, and σ is the activation function.

3.2 FedLEKE Task Formulation

In this section, we present the FedLEKE task and explain its connection to the traditional LEKE task. The FedLEKE refers to the collaborative execution of the LEKE task by multiple clients in a federated scenario. Assuming that each client c has a fact data set $\mathcal{E}_c^t$ to be edited in time slot t, the goal of FedLEKE is to insert the fact data $\mathcal{E}$ of all clients by editing the internal parameters of LLM. Overall, for each client c between predefined time slots, FedLEKE optimizes an objective function to obtain target weights (Meng et al., 2022b):

$$\begin{aligned}W_c^t \triangleq \underset{\tilde{W}_c^t}{\text{argmin}} \Bigg(&\sum_{i=1}^{n} \left\| \tilde{W}_c^t k_{ci}^t - v_{ci}^t \right\|^2 \\ &+ \sum_{i=n+1}^{n+u} \left\| \tilde{W}_c^t k_{ci}^t - v_{ci}^t \right\|^2 \Bigg),\end{aligned} \tag{3}$$

here, $k_{ci}^t \triangleq k_{ci}^{tl}$ and $v_{ci}^t \triangleq v_{ci}^{tl}$ represent the sets of keys and values, respectively, encoding the subject-related knowledge in the l-th layer at time t on

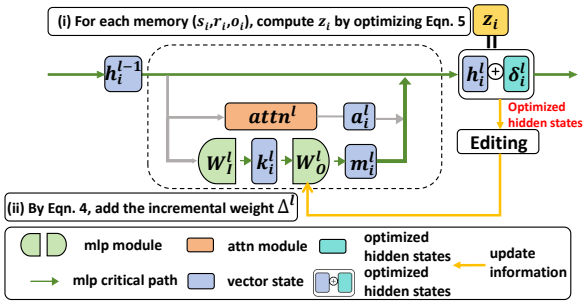

Figure 3: The overview of the classic LEKE method named MEMIT (Meng et al., 2022b).

client c. The term $\sum_{i=1}^{n} \left\| \tilde{W}_c^t k_{ci}^t - v_{ci}^t \right\|^2$ indicates that we aim to retain n pieces of knowledge, while $\sum_{i=n+1}^{n+u} \left\| \tilde{W}_c^t k_{ci}^t - v_{ci}^t \right\|^2$ suggests that we intend to modify a much larger number of knowledge pieces, denoted as $u \gg 1$. Here, the keys and values are represented as matrices stacked horizontally: $\left[k_{c1}^t \mid k_{c2}^t \mid \cdots \mid k_{cn}^t\right] \triangleq K_c^t$ and $\left[v_{c1}^t \mid v_{c2}^t \mid \cdots \mid v_{cn}^t\right] \triangleq V_c^t$. The target weight W_c^t is the sum of the original weight $\tilde{W}_c^t$ and the incremental weight Δ_c^t, i.e., $W_c^t = \tilde{W}_c^t + \Delta_c^t$. Based on the derivation from MEMIT (Meng et al., 2022b), the formal expression for the incremental weight is given as:

$$\Delta_c^t = R_c^t K_c^{t^T} (C_0 + K_c^t K_c^{t^T})^{-1}, \tag{4}$$

where $R_c^t \triangleq V_c^t - \tilde{W}_c^t K_c^t$ represents the residual between the values V_c^t (namely the target knowledge representations) corresponding to the keys K_c^t of the target knowledge and the client c model's original knowledge $\tilde{W}_c^t K_c^t$. $C_0 \triangleq \lambda \mathbb{E}_k \left[kk^T\right]$ is an estimate of the set of previously memorized keys obtained through sampling, and λ is a hyperparameter that balances the degree of model modification and preservation.

3.3 LEKE

This section delves into the LEKE method, emphasizing how knowledge updates are performed across multiple layers of the Transformer. For instance, as shown in Fig. 3, MEMIT (Meng et al., 2022b) employs optimized transformer layer hidden states to perform subtle updates on the FFN weights. In contrast, PMET (Li et al., 2024) simultaneously optimizes the transformer component hidden states of both MHSA and FFN, but only applies the optimized TC hidden states to the FFN. In this paper, we take MEMIT as an example of a LEKE method and further elaborate on its approach to updating multiple layers

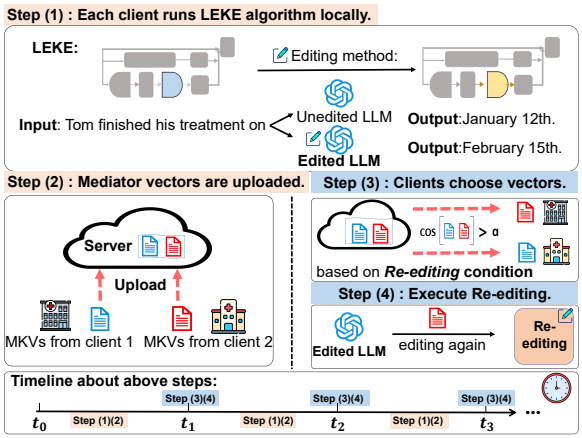

Figure 4: The workflow of the proposed FedEdit.

in the FedLEKE task. Specifically, we calculate the target knowledge set of the first and last critical layer $L_0 = \min(\mathcal{R}), L = \max(\mathcal{R})$. For each edit $(s_{ci}, r_{ci}, o_{ci}) \in \mathcal{E}_c$ (sucbject s, relation r, object o) on client c, we (i) compute z_{ci} to replace h_{ci}^L such that adding $\delta_{ci} \triangleq z_{ci} - h_{ci}^L$ to the hidden state at layer L. Then, for each layer, we (ii) modify the MLP at layer l by spreading Δ_c^{tl} over layer l.

(i) **Computing z_{ci}.** For the i-th edit on client c, z_{ci} is derived by optimizing the residual vector δ_{ci} via gradient descent:

$$\begin{aligned} z_{ci} = & h_{ci}^L + \underset{\delta_{ci}}{\operatorname{argmin}} \frac{1}{P} \sum_{j=1}^{P} \\ & - \log \mathbb{P}_{\mathcal{F}_c(h_{ci}^L += \delta_{ci})} \left[o_{ci} \mid x_{cj} \oplus p(s_{ci}, r_{ci})\right]. \end{aligned} \tag{5}$$

In words, we optimize δ_{ci} to maximize the client c model's prediction accuracy for the desired object o_{ci}, given a set of factual prompts $\{x_{cj} \oplus p(s_{ci}, r_{ci})\}$ that concatenate random prefixes x_{cj} to a templated prompt to aid generalization across contexts. $\mathcal{F}_c(h_{ci}^L += \delta_{ci})$ indicates that we modify the transformer execution by substituting the modified hidden state z_{ci} for h_{ci}^L.

(ii) **Spreading Δ_c^{tl} over layer l.** We follow the same algorithm steps as MEMIT that are presented in Algorithm 2 in Appendix C. Next, we'll mainly describe how to implement our FedEdit framework with the update step.

3.4 FedEdit Framework

In this section, we propose the FedEdit framework to address the FedLEKE task in a federated setting. The framework is designed to adapt LEKE tasks to a federated scenario, where each client interacts with the server and collaboratively edits the knowledge. As shown in Fig. 4, the workflow of the FedEdit framework is as the following steps:

Algorithm 1: FedEdit

Input: similarity threshold α, the number of time slots m, records $\mathcal{E}$, unedited model $\mathcal{M}$

1 Initialize $client \leftarrow [client_1, client_2, \ldots, client_n]$, $server \leftarrow []$, $t \leftarrow 0$, $T \leftarrow [t_1, t_2, \ldots, t_m]$, $selected\_z \leftarrow []$;
2 t begins to increment ;
3 **for** $c \in client$ ***in parallel*** **do**
4 $\quad edited\_model_c, Z_c^{t_i} \leftarrow$ **Edit**$(model_c, \mathcal{E}_c)$;
5 $\quad server$.append$(Z_c^{t_i})$;
6 $\quad$ **if** $t \in T$ ***and Select***$\_Z(server, Z_c^{t_i}) \neq \emptyset$ **then**
7 $\quad\quad$ **Edit**$(edited\_model_c,$ **Select**$\_Z(server, Z_c^{t_i}))$;
8 **function Select**$\_Z(server, Z_c^{t_i})$:
9 $\quad$ **for** $Z_{sq}^{t_i} \in server$ **do**
10 $\quad\quad similarities \leftarrow$ **cosine_similarity**$(Z_{sq}^{t_i}, Z_c^{t_i})$;
11 $\quad\quad$ **if** $\sum(similarities > \alpha) \geq \frac{len(similarities)}{2}$ **then**
12 $\quad\quad\quad selected\_z$.append$(Z_{sq}^{t_i})$;
13 $\quad$ **return** $selected\_z$;

Step (1): Starting at $t = 0$, each client runs the Edit algorithm locally, which can be any LEKE method. In this paper, we select MEMIT (Meng et al., 2022b) and PMET (Li et al., 2024). This process generates the MKVs.

Step (2): The MKVs are then uploaded to the server.

Step (3): At a predetermined time slot, each client selects some MKVs from the server according to the re-editing conditions defined later.

Step (4): If at least one vector is chosen by the client, it continues editing on the model.

On the timeline, steps (1) and (2) occur within the intervals between given time slots, while steps (3) and (4) are executed when the predetermined time slot is reached.

Furthermore, to define the MKVs and the re-editing conditions, we summarize our framework FedEdit in Algorithm 1, which consists of two main steps:

(i) Editing. Between the time slots in T, each client executes the Edit algorithm parallelly and independently (Step (1)). Here we still take MEMIT as an example i.e., Algorithm 2 in Appendix C. In this algorithm process, z_{ci}, k_{ci}^l are related to the data records $\mathcal{E}_c$, and we define the mediator knowledge vectors (MKVs) of client c at time t as Z_c^t:

$$Z_c^t = \left\{ (z_{ci}, k_{ci}^l) \right\}, \tag{6}$$

where (z_{ci}, k_{ci}^l) are all generated by client c during the time interval from $t-1$ to t, $(s_{ci}, r_{ci}, o_{ci}) \in \mathcal{E}_c$, and the keys k_{ci}^l at the l-th layer are defined as follows (Meng et al., 2022b):

$$k_{ci}^l = \frac{1}{P} \sum_{j=1}^{P} k(x_{cj} + s_{ci}), \tag{7}$$

where $k(x) = \sigma(W_I^l \gamma(h_{ci}^{l-1}(x)))$. Once a client has finished editing, it uploads the obtained Z_c to the server (Step (2)).

(ii) Re-editing. Once the time reaches any time $t_i \in T (i = 1, ..., m, m$ is the total number of time slots$)$, where server s distributes the previously stored $Z_s^{t_i}$ between t_{i-1} to t_i to each client. Each client selects $Z_c^{t_i}$ from the $Z_s^{t_i}$ that are beneficial to it, i.e., positively correlated with its own data $\mathcal{E}_c$ indirectly, through the **"re-edit" condition**:

$$\sum(similarities > \alpha) \geq \frac{len(similarities)}{2}, \tag{8}$$

where *similarities* is the cosine similarity between the q-th traversed $Z_{sq}^{t_i}$ in the server and $Z_c^{t_i}$, i.e., line 10 of Algorithm 1. α means similarity threshold, which is a hyperparameter. $\sum(similarities > \alpha)$ is the number of MKVs in $Z_c^{t_i}$ that satisfy the similarity threshold requirement with $Z_{sq}^{t_i}$. $len(similarities)$ is the number of all MKVs in the client c as of the current time slot t. In summary, iterates through each $Z_{sq}^{t_i}$ in the server, calculates the cosine similarity between the $Z_{sq}^{t_i}$ and the $Z_c^{t_i}$ of client c, and if more than half of the MKVs in client c are greater than the similarity threshold α, then the $Z_{sq}^{t_i}$ is said to satisfy the current client's "re-edit" condition. Then the $Z_{sq}^{t_i}$ will be selected by client c (Step (3)).

When the screening process is finished, each client performs Algorithm 2 again on the basis of the model $edited\_model_c$ that has been edited earlier (Step (4)). The process is repeated until $t_i = t_m$.

4 Experiments

4.1 Experimental Setup

Datasets. We conducted counterfactual update experiments on two datasets: Zero-Shot Relation Extraction (zsRE) (Levy et al., 2017) and COUNTERFACT (Meng et al., 2022a). The zsRE dataset contains 10,000 real-world facts (Meng et al., 2022b), while COUNTERFACT includes 21,919 factual statements (Meng et al., 2022a). To simulate FedLEKE, we reorganized the datasets using different clustering methods. For zsRE, we

Editor	Score	Efficacy	Generalization	Specificity	Fluency	Consistency
GPT-J (6B)	22.4	15.2 (0.7)	17.7 (0.6)	83.5 (0.5)	622.4 (0.3)	29.4 (0.2)
FT-W	67.6	99.4 (0.1)	77.0 (0.7)	46.9 (0.6)	293.9 (2.4)	15.9 (0.3)
MEND	23.1	15.7 (0.7)	18.5 (0.7)	83.0 (0.5)	618.4 (0.3)	31.1 (0.2)
ROME	50.3	50.2 (1.0)	50.4 (0.8)	50.2 (0.6)	589.6 (0.5)	3.3 (0.0)
MEMIT	85.8	98.9 (0.2)	92.8 (0.4)	73.7 (0.5)	619.9 (0.3)	40.1 (0.2)
MEMITAvg	62.9 [73.3%]	60.4 [61.1%]	55.5 [59.8%]	**76.5 [103.8%]**	**619.8 [99.9%]**	35.2 [87.8%]
FedMEMIT	**85.1 [99.2%]**	**97.0 [98.1%]**	86.5 [93.2%]	**75.4 [102.3%]**	**614.9 [99.2%]**	37.8 [94.3%]
PMET	86.2	99.5 (0.1)	88.6 (0.4)	71.4 (0.5)	620.0 (0.3)	40.6 (0.2)
PMETAvg	35.9 [41.6%]	28.4 [28.5%]	27.8 [31.4%]	**80.8 [113.2%]**	**623.2 [100.5%]**	31.5 [77.6%]
FedPMET	**83.6 [97.0%]**	**98.9 [99.4%]**	**93.2 [105.2%]**	67.2 [94.1%]	**619.5 [99.9%]**	**39.6 [97.5%]**
GPT-NeoX (20B)	23.7	16.8 (1.9)	18.3 (1.7)	81.6 (1.3)	620.4 (0.6)	29.3 (0.5)
MEMIT	82.0	97.2 (0.8)	82.2 (1.6)	70.8 (1.4)	606.4 (1.0)	36.9 (0.6)
MEMITAvg	37.7 [46.0%]	30.7 [31.6%]	29.4 [35.8%]	**77.3 [109.2%]**	**618.0 [101.9%]**	30.9 [83.7%]
FedMEMIT	**80.8 [98.5%]**	**96.9 [99.7%]**	**89.6 [109.0%]**	64.1 [90.5%]	**598.5 [98.7%]**	**40.8 [110.6%]**
PMET	84.3	98.4 (0.2)	89.4 (0.5)	70.3 (0.5)	598.1 (0.6)	38.9 (0.2)
PMETAvg	36.2 [43.0%]	29.0 [29.5%]	28.0 [31.1%]	**79.7 [113.4%]**	**618.4 [103.4%]**	30.8 [79.2%]
FedPMET	**84.3 [100%]**	**95.6 [97.2%]**	**91.3 [102.1%]**	**70.7 [100.6%]**	**579.5 [96.9%]**	34.1 [87.7%]

Table 1: 10,000 counterfact edits on GPT-J (6B) and GPT-NeoX (20B) in federated and centralized scenarios. Parentheses indicate the 95% confidence interval, while brackets show federated scenario metrics as a percentage of the centralized scenario, with values exceeding 95% **bolded**.

clustered data based on the "src" value, which represents the subject (e.g., "What university did Watts Humphrey attend?" with the subject "Watts Humphrey"). We applied spectral clustering after transforming the text into word vectors to assign data to different clients. For COUNTERFACT, we grouped data with the same "relation id" into one client, and randomly assigned about 1/10 of the data from other clients to each client.

Baselines. We select six knowledge editing methods as baselines: (1) **FT-W** is a simple fine-tuning approach that applies weight decay to prevent forgetfulness. (2) **MEND** (Mitchell et al.) transforms the fine-tuning gradient of an updated fact by decomposing the weight matrix into rank-1 form using a pre-trained hyper-network. (3) **ROME** (Meng et al., 2022a) locates factual retrievals within a specific set of MLP modules and updates knowledge by directly writing new key-value pairs into the MLP module. (4) **MEMIT** (Meng et al., 2022b) extends ROME to insert multiple memories by modifying the MLP weights of several critical layers. (5) **PMET** (Li et al., 2024) uupdates FFN weights by optimizing the hidden states of both MHSA and FFN, using only the FFN hidden states for weight updates. (6) **EditAvg** is a variant of FedAvg for solving the FedLEKE task, where any LEKE method can replace "Edit." Please refer to Appendix B for the detail settings.

Metrics. Following Meng et al. (2022a), we use GPT-J (6B) (Wang and Komatsuzaki, 2021) and GPT-NeoX (20B) (Black et al., 2022) as the backbone for FedLEKE. Following prior work (Meng et al., 2022b), we evaluate models using the following metrics: (1) Efficacy, measuring editing success; (2) Paraphrase, assessing success on rephrasings of the original statement; (3) Specificity, ensuring unrelated facts remain unchanged; and (4) Score, the harmonic mean of these three metrics, balancing reliability (efficacy and paraphrase) and specificity. Additionally, in COUNTERFACT experiments, we include (5) Fluency, evaluating degradation due to repetition, and (6) Consistency, measuring semantic coherence in generated text. All results are weighted averages across clients.

Hyper-parameters. We set the number of clients to 8, with a total of approximately 10,000 edits, and define T to consist of 10 time slots. Covariance statistics are collected on GPT-J using 100,000 samples from Wikitext, and on GPT-NeoX using 50,000 samples from Wikitext. Please refer to Appendix D for more details.

4.2 Results of COUNTERFACT

Table 1 presents the results of all methods on 10K counterfactual edits. FedMEMIT and FedPMET achieve 99.2% and 97% of the performance of centralized methods, respectively. In contrast, apply-

Editor	Score	Efficacy	Generalization	Specificity
GPT-J	26.4	26.4 (±0.6)	25.8 (±0.5)	27.0 (±0.5)
FT-W	42.1	69.6 (±0.6)	64.8 (±0.6)	24.1 (±0.5)
MEND	20.0	19.4 (±0.5)	18.6 (±0.5)	22.4 (±0.5)
ROME	2.6	21.0 (±0.7)	19.6 (±0.7)	0.9 (±0.1)
MEMIT	50.7	96.7 (±0.3)	89.7 (±0.5)	26.6 (±0.5)
MEMITAvg	41.6 [82.1%]	55.7 [57.6%]	53.7 [59.9%]	28.1 [105.6%]
FedMEMIT	**50.5 [99.6%]**	**92.9 [96.1%]**	**87.3 [97.3%]**	**26.9 [101.1%]**
PMET	51.0	96.9 (±0.3)	90.6 (±0.2)	26.7 (±0.2)
PMETAvg	41.5 [81.4%]	55.5 [57.3%]	53.3 [58.8%]	**28.2 [105.6%]**
FedPMET	42.5 [82.4%]	66.5 [68.6%]	61.8 [68.2%]	**25.4 [95.1%]**

Table 2: 10,000 zsRE Edits on GPT-J (6B).

ing the FedAvg algorithm to MEMIT and PMET results in only 73.3% and 41.6%, respectively. This demonstrates that our method performs well in FedLEKE. It also highlights that simply combining federated learning algorithms like the classical FedAvg with knowledge editing methods does not yield effective results. In the trade-off between editing reliability and specificity, FedMEMIT and FedPMET, like MEMIT and PMET, prioritize reliability. On the other hand, MEND, MEMITAvg, and PMETAvg focus more on specificity. Moreover, FedMEMIT and FedPMET outperform non-federated methods in terms of specificity and generalization, respectively. However, in terms of specificity, they fall behind the meta-learning-based method MEND.

Next, we applied the FedEdit framework to perform 10K edits on GPTNeoX (20B) using the COUNTERFACT dataset. The results are shown in the lower part of Table 1. We find: FedMEMIT and FedPMET significantly outperform MEMITAvg and PMETAvg, consistently favoring reliability and consistency. Additionally, both FedMEMIT and FedPMET surpass their respective non-federated methods in generalization. This may be due to our proposed "re-edit" condition, which selects data with similar types for re-editing, thereby enhancing reliability. We further explore this in the following ablation experiments.

4.3 Results of ZsRE

The zsRE dataset tests the ability to add correct information. The results of editing 10K knowledge on the zsRE dataset are shown in Table 2. These results demonstrate that our method performs very close to the original method in the federated scenario, both in efficacy and generalization metrics, and even slightly outperforms it in terms of specificity. Specificity refers to the model's argmax accuracy on a randomly sampled, unrelated fact that should not have changed (Meng et al., 2022b).

Edits	Editor	Score	Efficacy	Generalization	Specificity
	GPT-J	26.4	25.8	27.0	26.4
1K	FedMEMIT	57.0	99.7	97.1	31
	w/o Z_c^t	54.1 (↓2.9)	99.6 (↓0.1)	97.2 (↑0.1)	28.5 (↓2.5)
	FedPMET	54.9	98.0	94.0	29.6
	w/o Z_c^t	54.3 (↓0.6)	97.4 (↓0.6)	93.6 (↓0.4)	29.2 (↓0.4)
5K	FedMEMIT	55.1	98.2	94.0	29.8
	w/o Z_c^t	53.2 (↓1.9)	96.1 (↓2.1)	91.6 (↓2.4)	28.5 (↓1.3)
	FedPMET	52.8	93.6	88.4	28.7
	w/o Z_c^t	51.9 (↓0.8)	91.6 (↓2.0)	85.6 (↓2.8)	28.4 (↓0.3)
10K	FedMEMIT	50.5	92.9	87.3	26.9
	w/o Z_c^t	49.6 (↓0.9)	87.7 (↓5.2)	83.0 (↓4.3)	27.0 (↑0.1)
	FedPMET	42.5	66.5	61.8	25.4
	w/o Z_c^t	40.8 (↓1.7)	65.5 (↓1.0)	60.5 (↓1.3)	24.0 (↓1.4)

Table 3: The ablation experiments, where w/o Z_c^t means that the re-editing condition control is removed.

EditAvg (MEMITAvg and PMETAvg), averages the Δ_c^t of each client before inserting it into the server's model, making it naturally stable in the case of random sampling. Additionally, compared to the original method, our approach includes an extra re-editing step. This step allows for editing additional vectors that are more suitable for the current client, improving performance.

4.4 Ablation Study

The ablation study in Table 3 examines the impact of removing the re-editing condition control from the FedMEMIT and FedPMET methods on their performance across different editing scales (1K, 5K, and 10K edits). The results show that: (1) The re-editing condition is crucial for the FedLEKE task. Removing it causes a decline in the score for all FedEdit-related experiments, indicating that the model updates facts unrelated to its own data. This negatively impacts the model's ability to edit its own knowledge accurately. (2) The more similar the knowledge edited on a single client, the better the model's reliability. A clear trend emerges: reliability (efficacy and paraphrasing) decreases more, while specificity decreases less. This suggests that as the number of client edits increases, the re-editing condition improves reliability.

4.5 Robustness Study

We conducted a robustness study on the proposed framework using the zsRE dataset. Specifically, we assigned 10,000 facts to 2, 3, 4, 5, 6, 7, 8 clients for editing. As shown in Fig. 5, the experimental charts show that as the number of clients increases, FedMEMIT consistently performs well and remains stable across all metrics. In contrast, while FedPMET's performance improves initially, it declines as the number of clients grows, likely due to the

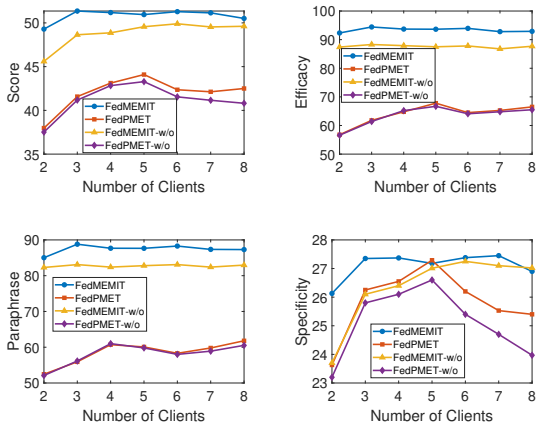

Figure 5: The editing performance of FedEdit and baselines with the number of clients. The suffix "w/o" indicates the Ablation experimental group.

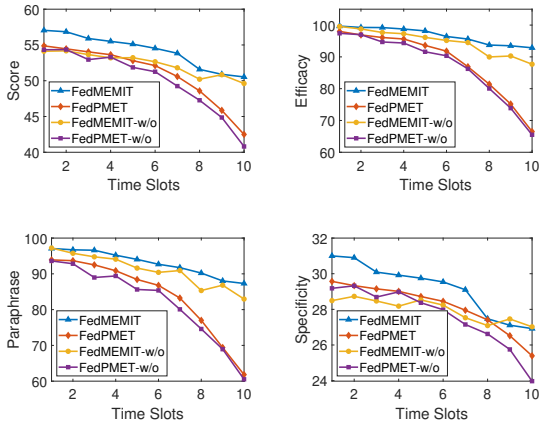

Figure 6: The editing performance of FedEdit and baselines with the number of time slots.

effects of multiple edits. Notably, the Specificity indicator fluctuates with the number of clients, which may be influenced by the number of single edits.

4.6 Impact of the Number of Time Slots

Fig. 6 illustrates the superior performance of FedMEMIT compared to FedPMET and the ablation variants, FedMEMIT-w/o and FedPMET-w/o. While both FedMEMIT and FedPMET show a decline in performance as the number of time slots increases, FedMEMIT consistently outperforms FedPMET across all metrics. FedMEMIT achieves higher scores, maintains better efficiency, and demonstrates more stable paraphrasing quality and specificity, especially in long-term editing tasks. The gradual decline in FedMEMIT indicates its better ability to preserve edit quality over time, compared to the more significant performance drops seen in FedPMET. This highlights FedMEMIT's robustness, showing advantages in efficiency, editing reliability, and specificity preservation. Although FedMEMIT-w/o shows improvements in stability, FedMEMIT remains the most effective for achieving high-quality, sustainable editing performance in federated scenarios.

Client 1:(case id: 15874)
Original Knowledge: Josef Albers, who has a citizenship from Germany
Edited Knowledge: Josef Albers, who has a citizenship from Canada
MEMITAvg: *Josef Albers, who has a citizenship from* ?0.0.
FedMEMIT: *Josef Albers, who has a citizenship from* Canada.
Client 2:(case id: 15911)
Original Knowledge: Ulrica Arfvidsson, who holds a citizenship from Sweden
Edited Knowledge: Ulrica Arfvidsson, who holds a citizenship from Kenya
MEMITAvg: *Ulrica Arfvidsson, who holds a citizenship from* Sweden.
FedMEMIT (Before 15874):*Ulrica Arfvidsson, who holds a citizenship from* Italy.
FedMEMIT (After 15874): *Ulrica Arfvidsson, who holds a citizenship from* Kenya.

Table 4: Results for two cases in COUNTERFACT from two clients based on GPT-J (6B). Prompt + subject are underlined and italicized. Words highlighted in green signify keywords that reflect correct behavior. Those in red denote keywords associated with incorrect behavior.

4.7 Case Study

Table 4 demonstrates that FedMEMIT correctly generates text in both cases, in contrast to MEMITAvg. This highlights the limitations of selecting Δ_c as the mediator vector and validates the appropriateness of choosing Z_c as the mediator vector. Moreover, the table illustrates a scenario where two highly similar cases are edited on two different clients. Specifically, case 15874 on client 1 is first edited, and the resulting Z_c vector is uploaded to the server. Client 2 then retrieves this vector from the server and edits it. As a result, when a similar case (case id: 15911) is edited, the text is generated correctly. However, if the vector has not been edited, client 2 generates incorrect text. This further demonstrates the effectiveness of FedEdit.

5 Conclusion

We introduce FedLEKE, a novel task that enables collaborative knowledge editing across multiple clients while ensuring privacy and reducing computational costs. To achieve this, we propose FedEdit, a two-stage framework comprising editing and re-editing. In the editing stage, clients locally perform knowledge editing and upload MKVs to a central server. In the re-editing stage, clients retrieve relevant MKVs via cosine similarity for further refinement. Experimental results demonstrate that FedEdit outperforms strong baselines in FedLEKE, paving the way for more effective knowledge editing in federated settings and inspiring future research in this direction.

Limitation

We acknowledge the following limitations in our work: (1) The FedLEKE task may face challenges due to non-IID data across clients. The heterogeneous data distributions can cause instability in the model, particularly when personalization is required for different tasks. While we have addressed this issue through the FedEdit framework, which uses clustering for selecting MKVs and re-editing conditions to improve the knowledge editing process, it remains a challenge in environments with diverse data. (2) Our work focuses on a simulated federated learning scenario, and thus does not account for certain external factors, such as environmental changes or system anomalies, that may impact the performance of the deployment in real-world settings. We plan to conduct additional experiments to further explore these challenges.

Ethics Consideration

In the development and application of federated learning systems, we prioritize ethical sourcing and privacy protection. Our proposed FedLEKE task ensures that the research complies with data privacy regulations, and all datasets used in this study (zsRE and COUNTERFACT) are open-source and publicly available. These datasets do not contain any personally identifiable information or sensitive data. To mitigate privacy risks, our proposed FedEdit framework ensures that only mediator knowledge vectors (MKVs) are uploaded to the server, rather than raw data. This design ensures that sensitive data is never directly shared, and knowledge editing is performed in a manner that prevents leakage of private information.

Additionally, while federated learning frameworks enable collaboration among different organizations, we acknowledge the importance of safeguarding intellectual property and ensuring fairness in model training. Our work is designed to facilitate efficient knowledge editing while preventing misuse or unintended consequences. As such, we have implemented careful oversight measures to ensure that the server-based aggregation of MKVs does not inadvertently expose confidential information.

Furthermore, all experiments were conducted with transparency and respect for the principles of fairness and data protection. We do not authorize the use of the datasets for any commercial purposes, and our results are strictly intended for academic and research purposes. Our study demonstrates the potential of federated learning to enhance the efficiency and privacy of knowledge editing tasks, while adhering to ethical standards of data use and model deployment.

Acknowledgements

This work is supported in part by Science and Technology Innovation Key R&D Program of Chongqing (Grants No. CSTB2024TIAD-STX0024 and CSTB2023TIAD-STX0035), National NSFC (Grants No. 62372072, 62102053, and 52272388), Chongqing Talent Program Contract System Project (Grant No. cstc2024ycjh-bgzxm0042), Haihe Lab of ITAI (Grant No. 22HHXCJC00002), the Natural Science Foundation of Chongqing, China (Grant No. CSTB2022NSCQ-MSX1104), Key Laboratory of Big Data Intelligent Computing, Chongqing University of Posts and Telecommunications (Grant No. BDIC-2023-B-003), Sichuan Science and Technology Program (Grant No. 2024YFHZ0097), Regional Science and Technology Innovation Cooperation Project of Chengdu City (Grant No. 2023-YF11-00023-HZ), and China Postdoctoral Science Foundation Funded Project (2024M763867).

References

Ankur Agarwal, Mehdi Rezagholizadeh, and Prasanna Parthasarathi. 2023. Practical takes on federated learning with pretrained language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 454–471.

Sikai Bai, Jie Zhang, Song Guo, Shuaicheng Li, Jingcai Guo, Jun Hou, Tao Han, and Xiaocheng Lu. 2024. Diprompt: Disentangled prompt tuning for multiple latent domain generalization in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27284–27293.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Chaochao Chen, Xiaohua Feng, Yuyuan Li, Lingjuan Lyu, Jun Zhou, Xiaolin Zheng, and Jianwei Yin. 2024. Integration of large language models and federated learning. *Patterns*, 5(12).

Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang,

and Israel Cohen. 2009. Pearson correlation coefficient. *Noise reduction in speech processing*, pages 1–4.

Ahmed El Ouadrhiri and Ahmed Abdelhadi. 2022. Differential privacy for deep and federated learning: A survey. *IEEE access*, 10:22359–22380.

Tao Guo, Song Guo, Junxiao Wang, Xueyang Tang, and Wenchao Xu. 2023. Promptfl: Let federated participants cooperatively learn prompts instead of models-federated learning in age of foundation model. *IEEE Transactions on Mobile Computing*.

Akshat Gupta, Sidharth Baskaran, and Gopala Anumanchipalli. 2024. Rebuilding rome: Resolving model collapse during sequential model editing. *arXiv preprint arXiv:2403.07175*.

Baixiang Huang, Canyu Chen, Xiongxiao Xu, Ali Payani, and Kai Shu. 2024. Can knowledge editing really correct hallucinations? *arXiv preprint arXiv:2410.16251*.

Jakub Konečnỳ. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Guanghao Li, Wansen Wu, Yan Sun, Li Shen, Baoyuan Wu, and Dacheng Tao. 2023. Visual prompt based personalized federated learning. *arXiv preprint arXiv:2303.08678*.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572.

I-Jieh Liu, Ci-Siang Lin, Fu-En Yang, and Yu-Chiang Frank Wang. 2024. Language-guided transformer for federated multi-label classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13882–13890.

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

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

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

John Nguyen, Jianyu Wang, Kshitiz Malik, Maziar Sanjabi, and Michael Rabbat. 2022. Where to begin? on the impact of pre-training and initialization in federated learning. *arXiv preprint arXiv:2206.15387*.

Jian Song, Di Liang, Rumei Li, Yuntao Li, Sirui Wang, Minlong Peng, Wei Wu, and Yongxin Yu. 2022. Improving semantic matching through dependency-enhanced pre-trained model with adaptive fusion. *arXiv preprint arXiv:2210.08471*.

Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. 2022. Federated learning from pre-trained models: A contrastive learning approach. *Advances in neural information processing systems*, 35:19332–19344.

Ben Wang and Aran Komatsuzaki. 2021. Gpt-j-6b: A 6 billion parameter autoregressive language model.

Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. 2024. Detoxifying large language models via knowledge editing. *arXiv preprint arXiv:2403.14472*.

Kaiwen Wei, Xian Sun, Zequn Zhang, Li Jin, Jingyuan Zhang, Jianwei Lv, and Zhi Guo. 2023a. Implicit event argument extraction with argument-argument relational knowledge. *IEEE Trans. Knowl. Data Eng.*, 35(9):8865–8879.

Kaiwen Wei, Xian Sun, Zequn Zhang, Jingyuan Zhang, Zhi Guo, and Li Jin. 2021. Trigger is not sufficient: Exploiting frame-aware knowledge for implicit event argument extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4672–4682. Association for Computational Linguistics.

Kaiwen Wei, Yiran Yang, Li Jin, Xian Sun, Zequn Zhang, Jingyuan Zhang, Xiao Li, Linhao Zhang, Jintao Liu, and Zhi Guo. 2023b. Guide the many-to-one assignment: Open information extraction via iou-aware optimal transport. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4971–4984. Association for Computational Linguistics.

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

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*.

Abbas Yazdinejad, Ali Dehghantanha, Hadis Karimipour, Gautam Srivastava, and Reza M Parizi. 2024. A robust privacy-preserving federated learning model against model poisoning attacks. *IEEE Transactions on Information Forensics and Security*.

Ao Zhang, Liming Zhao, Chen-Wei Xie, Yun Zheng, Wei Ji, and Tat-Seng Chua. 2023. Next-chat: An lmm for chat, detection and segmentation. *arXiv preprint arXiv:2311.04498*.

Zhuoran Zhang, Yongxiang Li, Zijian Kan, Keyuan Cheng, Lijie Hu, and Di Wang. 2024. Locate-then-edit for multi-hop factual recall under knowledge editing. *arXiv preprint arXiv:2410.06331*.

A Federated Learning

Training Objective. Federated learning aims to optimize the following objective function:

$$\min_{w} \mathcal{F}(w) \triangleq \sum_{i=1}^{N} p_i \mathcal{L}_i(w) \quad \text{where} \quad \mathcal{L}_i(w) = \mathbb{E}_{a \sim \mathcal{D}_i}[f_i(w, a)]. \tag{9}$$

In Eqn.(9), $\mathcal{L}_i(w)$ denotes the local training objective function of the client i and N denotes the number of clients. $w \in \mathbb{R}^d$ represents the parameters of the global model. a denotes each batch of data. The local training loss function $f_i(w, a)$ is often the same across all the clients, while $\mathcal{D}_i$ denotes the distribution of the local client data, which is often different across the clients, capturing the heterogeneity. p_i is defined as the training size proportion in Eqn. (4), where $|\mathcal{D}_i|$ is the training size of client i.

$$p_i = \frac{|\mathcal{D}_i|}{\sum_{i=1}^{N} |\mathcal{D}_i|} \tag{10}$$

Training Procedure. Federated learning is an iterative process shown in Figure 2. The server initializes the global model, followed by multiple communication rounds between the server and clients. In each *communication round*, there are four steps between the server and clients. 1) In round t, the server sends the global model w^t to all the clients. 2) After clients receive the global model w^t as the initialization of the local model, they start to train it using their own data for multiple epochs and obtain the local model changes Δw_i^t during the local training stage. 3) The clients send their local model changes to the server. 4) The server aggregates the local model changes Δw_i^t collected from different clients as Eqn. (3) shows, and then uses the $t\text{-}th$ round's global model w^t and the aggregated local model changes Δw_i^t to update the global model. As Eqn. (4) shows, w^{t+1} is the global model after the update. Here, n denotes the server learning rate. The server will send the updated model w^{t+1} to the clients, then the $(t+1)\text{-}th$ round starts.

The above procedure will repeat until the algorithm converges.

$$\Delta w^t = \sum_{i=1}^{N} p_i \Delta w_i^t \tag{11}$$

$$w^{t+1} = w^t - \eta \Delta w^t \tag{12}$$

FedAvg. Federated Averaging (FedAvg) (McMahan et al., 2017b) uses stochastic gradient descent (SGD) as the local training optimizer to optimize the training procedure and uses the same learning rate and the same number of local training epochs for all the clients.

B Details of EditAvg

In this approach, Δ_c^t is selected as the MKV for client c at time $t \in T$, and it is transferred to the server after execution of Algorithm 2 (equation (4)). The server then aggregates all Δ_c^t using the formula $\Delta_c^t = \sum_{c=1}^{N} p_c^t \Delta_c^t$, where p_c^t represents the proportion of edits made by client c from $t-1$ to t.

C MEMIT

In Algorithm 2, with the exception of the symbol r_i^l in line 8, the symbol definitions in the rest of the formulas are consistent with those defined in section 3. $r_i^l \leftarrow \frac{z_i - h_i^L}{L-l+1}$ is the residual, which is spread over layers $\mathcal{R}$.

D Detailed Hyper-parameters

We set the number of clients to 8, with a total of approximately 10,000 edits, and define T to consist of 10 time slots. Covariance statistics are collected on GPT-J using 100,000 samples from Wikitext, and on GPT-NeoX using 50,000 samples from Wikitext. $\mathcal{R} = \{3, 4, 5, 6, 7, 8\}$ for GPT-J

Algorithm 2: MEMIT

Input: Requested edits $\mathcal{E} = \{(s_i, r_i, o_i)\}$, generator G, layers to edit S, covariances C^l

Output: Modified generator containing edits from $\mathcal{E}$

for $s_i, r_i, o_i \in \mathcal{E}$ **do**
  $\delta_i \leftarrow argmin_{\delta_i} \frac{1}{P} \sum_{j=1}^{P} -\log \mathbb{P}_{G(h_i^L += \delta_i)} [o_i \mid x_j \oplus p(s_i, r_i)]$;
  $z_i \leftarrow h_i^L + \delta_i$;
for $l \in \mathcal{R}$ **do**
  $h_i^l \leftarrow h_i^{l-1} + a_i^l + m_i^l$;
  for $s_i, r_i, o_i \in \mathcal{E}$ **do**
    $k_i^l \leftarrow \frac{1}{P} \sum_{j=1}^{P} k(x_j + s_i)$;
    $r_i^l \leftarrow \frac{z_i - h_i^L}{L-l+1}$;
  $K^l \leftarrow \{k_i^l, \ldots, k_i^l\}$;
  $R^l \leftarrow \{r_i^l, \ldots, r_i^l\}$;
  $\Delta^l \leftarrow R^l K^{l^T} (C^l + K^l K^{l^T})^{-1}$;
  $W^l \leftarrow W^l + \Delta^l$

and $\mathcal{R} = \{6, 7, 8, 9, 10\}$ for GPT-NeoX. Further implementation details about LKE are the same as Meng et al. (2022b) and Li et al. (2024). For the computing resources, we utilize 8 NVIDIA A800 80GB GPUs.