COLLABEDIT: TOWARDS NON-DESTRUCTIVE COLLABORATIVE KNOWLEDGE EDITING

Jiamu Zheng [1]* **Jinghuai Zhang** [4] **Futing Wang** [2] **Tianyu Du** [3] **Tao Lin** [2]
University of Electronic Science and Technology of China [1] Westlake University [2]
Zhejiang University [3] University of California, Los Angeles [4]
zhengjaamie@gmail.com jinghuai1998@g.ucla.edu
wangfuting@westlake.edu.cn
tydusky@gmail.com lintao@westlake.edu.cn

ABSTRACT

Recently, collaborative fine-tuning large language models (LLMs) has emerged as a new paradigm for utilizing private data from different parties in a manner that guarantees both efficiency and privacy. Meanwhile, the practical needs of the "right to be forgotten" and the frequent demands to update outdated information, have led to a burgeoning in the techniques of knowledge editing (KE) for LLMs. However, current KE methods are all designed for a single model, and directly adapting current KE methods to collaborative learning scenarios encounters severe performance decreases. In this study, we propose a non-destructive collaborative knowledge editing framework COLLABEDIT that utilizes novel model fusion strategy to preserve overall editing performance. Empirical studies on two canonical datasets demonstrate the effectiveness and superiority of our method compared with other destructive baselines.

1 INTRODUCTION

Large Language Models (LLMs) (Achiam et al., 2023; Qiao et al., 2023) recently have emerged as the promising solution toward general artificial intelligence. However, deploying LLMs in practice usually requires customizing LLMs with specific knowledge (Meng et al., 2022), where re-training LLMs may be expensive and unacceptable (Jang et al., 2023). Accordingly, Knowledge Editing (KE) (Meng et al., 2022; Mitchell et al., 2022; Tan et al., 2024; Zhang et al., 2023), which allows efficient modification of knowledge stored in existing models, has been proposed as a remedy.

Current KE methods only consider the centralized case where all edit requests from different parties need to be first globally collected, which usually violates privacy concerns: the edit request itself contains sensitive private information and thus becomes infeasible for sharing. All these motivate resorting to the collaborative learning paradigm (Wu et al., 2023; Kairouz et al., 2021)—by only communicating the locally-updated-models, rather than uploading a list of risky edit requests—namely collaborative knowledge editing for LLMs.

However, existing KE methods are all designed for the single model scenario (Meng et al., 2022; Mitchell et al., 2022; Tan et al., 2024; Meng et al., 2023) (see Figure 1). As our first (side)-contribution, we examine a naive combination of local knowledge editing and global model fusion methods, where these naive collaborating editing methods are all destructive.

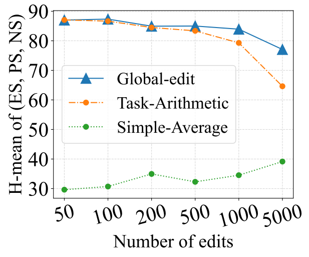
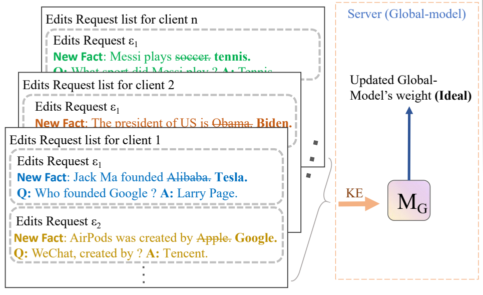
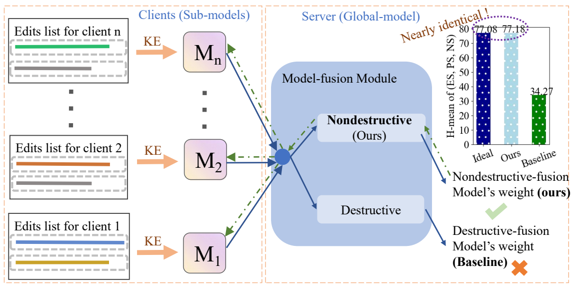


Figure 1: **Limits of existing KE methods** under the collaborative editing scenarios on the Multi-CounterFact dataset (Meng et al., 2022). We conduct independent knowledge editing on each client locally, and then use model fusion techniques like Simple-Average (Chronopoulou et al., 2023) and Task-Arithemetic (Ilharco et al., 2023) to merge local models into a global model. We compare the performance of naive collaborative editing methods with the optimal GLOBAL-EDIT: as the number of edits increases, the performance gap also widens.

To this end, we first analyze the performance gap between naive collaborative editing methods and the global editing method (GLOBAL-EDIT) from a theoretical perspective, upon which we further design


Edits Request list for client n
Edits Request ε_1
New Fact: Messi plays ~~soccer.~~ tennis.
Edits Request list for client 2
Edits Request ε_1
New Fact: The president of US is ~~Obama.~~ Biden.
Edits Request list for client 1
Edits Request ε_1
New Fact: Jack Ma founded ~~Alibaba.~~ Tesla.
Q: Who founded Google ? A: Larry Page.
Edits Request ε_2
New Fact: AirPods was created by ~~Apple.~~ Google.
Q: WeChat, created by ? A: Tencent.
Server (Global-model)
Updated Global-Model's weight (Ideal)
KE
M_G

(a) Global Editing



(b) Collaborative Editing

Figure 2: **Comparison of global knowledge editing (GLOBAL-EDIT) and collaborative knowledge editing.**

a novel framework COLLABEDIT that allows non-destructive collaborative editing. Our contributions can be summarized as follows:

- We identify the performance gap between the naive collaborative knowledge editing method and the upper bound performance (i.e., GLOBAL-EDIT) through theoretical and empirical analysis.
- To the best of our knowledge, we are the first to propose a non-destructive knowledge editing framework in the context of collaborative learning. Our framework is designed to be versatile, allowing seamless integration of existing knowledge-editing methods.
- Our empirical results demonstrate the effectiveness of our proposed framework compared with baselines. Our discussions may shed light on future research for collaborative knowledge editing.

2 METHODOLOGY

2.1 INTRODUCTION TO KNOWLEDGE EDITING IN A SINGLE LLM

LLMs can answer natural-language queries about facts based on implicit knowledge encoded within the parameters. Following Meng et al. (2023), we define a fact f as "*(subject* s*, relation* r*, object* o*)*", e.g., "*(*s = *Danielle Darrieux,* r = *spoke the language,* o = *French)*". Given a sequence of facts $\mathcal{E} = \{f_i | f_i = (s_i, r_i, o_i)\}$ to edit, knowledge editing aims to maximize the likelihood that the updated LLM $\mathcal{M}_{\boldsymbol{\theta}}$ predicts the desired object o_i for any factual prompt $\mathbf{x} \oplus p(s_i, r_i)$, which involves a prefix $\mathbf{x}$ and a templated prompt $p(s_i, r_i)$:

$$\arg\min_{\mathcal{M}_{\boldsymbol{\theta}}} \frac{1}{|\mathcal{E}|} \sum_{i=1}^{|\mathcal{E}|} \mathbb{E}_{\mathbf{x}} - \log \Pr_{\mathcal{M}_{\boldsymbol{\theta}}} \left[o_i | \mathbf{x} \oplus p(s_i, r_i) \right] . \tag{1}$$

The state-of-the-art knowledge editing methods (Meng et al., 2022; 2023; Tan et al., 2024) found that modifying a small sequence of MLP layers in the critical path of LLM is sufficient to edit its factual associations. In particular, linear operation $\mathbf{W}^l$ in an MLP layer can operate as a key-value store for input keys $\mathbf{K}^l$ and the memory/knowledge values $\mathbf{M}^l$, where input keys correspond to the intermediate feature vector of the model from a set of edit requests. Knowledge editing modifies each MLP layer such that it associates $\mathbf{K}^l$ to the desired $\mathbf{M}^l$ by solving $\mathbf{W}^l\mathbf{K}^l \approx \mathbf{M}^l$. For brevity, we will describe knowledge editing for a specific layer and omit l throughout the paper.

Given a set of facts $\mathcal{E}$ to edit, we first obtain their input keys $\mathbf{K} = [\mathbf{k}_1, \ldots, \mathbf{k}_{|\mathcal{E}|}]$ to the layer l via a single feed-forward. We also obtain the desired memory values $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_{|\mathcal{E}|}]$ of layer l that maximize $\Pr\left[o_i | \mathbf{x} \oplus p(s_i, r_i)\right]$. The goal of editing the layer l can be formulated as optimizing the $\boldsymbol{\Delta}$ such that the updated weight $\mathbf{W} + \boldsymbol{\Delta}$ associates the input keys $\mathbf{K}$ to the desired memory values $\mathbf{M}$. Note that the MLP layer also contains previously stored memories of existing knowledge, which should be preserved during the knowledge editing. Therefore, we also maintain the associations between input keys of existing knowledge $\mathbf{K}_{\text{init}}$ and their memory values ($\mathbf{W}\mathbf{K}_{\text{init}}$). Following MEMIT (Meng et al., 2023), we derive the closed form of Δ for a specific layer l as:

$$\boldsymbol{\Delta} = \mathbf{R}\mathbf{K}^\top(\mathbf{C} + \mathbf{K}\mathbf{K}^\top)^{-1}, \tag{2}$$

where $\mathbf{C} = \mathbf{K}_{\text{init}}\mathbf{K}_{\text{init}}^\top$ indicates the covariance matrix of the input keys of existing knowledge, and $\mathbf{R} = \mathbf{M} - \mathbf{W}\mathbf{K}$ represents the residual error in the output space of layer l. See more details in Appendix A.

2.2 Destructive Fusion Encounters Collaborative Knowledge Editing

Knowledge editing in practice requires editing the factual associations of LLM (e.g., correcting the hallucinations) using some edit requests simultaneously, in which multiple clients access and jointly contribute to the same LLM service. Though global editing (Global-Edit) illustrated in Figure 2(a) represents the ideal editing cases, it also necessitates each client to directly share the edit requests with the server, which violates the privacy constraints. Collaborative editing (in Figure 2(b)) instead allows each client to edit on its local LLM and only rely on the server to aggregate the edit updates.

However, existing knowledge editing algorithms are all designed for a single client and cannot be trivially generalized to the collaborative editing case. As evidenced in Figure 1, naively extending existing editing methods or model fusion methods yields a dramatic performance drop compared to that of the Global-Edit upper bound, especially when the number of edits increases. Given the limits of these destructive collaborative editing methods, in the following section, we aim to develop a non-destructive collaborative editing method that can achieve a similar editing performance as Global-Edit, even with a large number of edits.

2.3 CollabEdit: Non-destructive Collaborative Editing

To better understand the performance drop, we first explicitly model the relationship between the weight updates $\mathbf{\Delta}_G$ of the global model using Global-Edit and that of each client model $\mathbf{\Delta}_i$ using local editing. For ease of presentation, we consider the collaborative editing scenario with N clients and each client model has M edit requests.

Lemma 2.1 (The relationship between the weight updates from Global-Edit and local editing)**.** *Take the knowledge editing method* MEMIT *as an example. Following the definitions in Section 2.1, let's denote* $\mathbf{C}$ *as an aggregated statistic over the previously stored keys of existing knowledge and use* $\mathbf{K}_i$ *to represent the new keys derived from client* i*'s edit. Then, the relationship between* $\mathbf{\Delta}_G$ *and* $\mathbf{\Delta}_i$ *is measured as:*

$$\mathbf{\Delta}_G = \textstyle\sum_{i=1}^{N} \mathbf{\Delta}_i \cdot \Big(\boldsymbol{\alpha}_i := (\mathbf{C} + \mathbf{K}_i\mathbf{K}_i^\top)(\mathbf{C} + \textstyle\sum_{j=1}^{N} \mathbf{K}_i\mathbf{K}_i^\top)^{-1}\Big) . \tag{3}$$

See detailed proof in Appendix B.1.

Intuition: If we can estimate $\mathbf{\Delta}_G$ using $\mathbf{\Delta}_i$, then we can merge $\{\mathbf{\Delta}_i\}_{i=1}^N$ to obtain the same global model as Global-Edit and, therefore, obtain non-destructive collaborative editing.

Details of CollabEdit: Indeed $\mathbf{\Delta}_G$ can be represented as the weighted sum of different local weight updates $\mathbf{\Delta}_i$ with coefficient α_i. However, the coefficient α_i relies on the value of $\mathbf{K}_i$ of all the clients: it breaks the privacy, given the fact that $\mathbf{K}_i$ is an intermediate feature vector of the model from a set of edit requests and any external party can easily reconstruct the edit requests if $\mathbf{K}_i$ is leaked. As a remedy, our CollabEdit instead proposes to directly communicate $\mathbf{K}_i\mathbf{K}_i^\top$, in which we prove in Appendix B.2 that $\mathbf{K}_i\mathbf{K}_i^\top$ is non-trivial to attack. See our pseudo-code in Appendix D.

Remark 2.2. *Currently, we consider two mainsteam knowledge editing methods (Akyürek et al., 2023), namely (1) locate and edit activations (e.g., MEMIT (Meng et al., 2023) and ROME (Meng et al., 2022)); and (2) train an auxiliary model to directly predict parameters (e.g., MEND (Mitchell et al., 2022) and MALMEN (Tan et al., 2024)). Note that our framework* CollabEdit *is general enough to integrate many other knowledge editing methods, and we leave them for future work.*

Justifying the performance drop for destructive editing approaches. We further analyze the performance degradation for destructive editing approaches when the number of edits increases, as illustrated in Figure 1. For the sake of simplicity, we take the Task-Arithmetic (Ilharco et al., 2023) with MEMIT as an example. The drop can be explained by:

$$\mathbf{\Delta}_G - \mathbf{\Delta}'_G = \textstyle\sum_{i=1}^{N} \mathbf{\Delta}_i \left[(\mathbf{C} + \mathbf{K}_i\mathbf{K}_i^\top)(\mathbf{C} + \textstyle\sum_{j=1}^{N} \mathbf{K}_i\mathbf{K}_i^\top)^{-1} - \lambda\mathbf{I}\right] , \tag{4}$$

where $\mathbf{\Delta}_G$ and $\mathbf{\Delta}'_G$ represent the weight updates derived from CollabEdit (our non-destructive collaborative editing) and a destructive collaborative editing using Task-Arithmetic, respectively. We can see that the impact of new knowledge $\mathbf{K}_i\mathbf{K}_i^\top$ is negligible compared to existing knowledge $\mathbf{C}$ when the number of edits is small, resulting in $(\mathbf{C} + \sum_{j=1}^{N} \mathbf{K}_i\mathbf{K}_i^\top)^{-1} \approx \mathbf{C}$ and thus $\mathbf{\Delta}_G \approx \mathbf{\Delta}'_G$ when $\lambda = 1$. The gap becomes wider when the number of edits increases, contributing to the continuous decline in Task-Arithmetic's performance in Figure 1 compared to Global-Edit.

Table 1: **Overall editing performance on MCF and zsRE.** GLOBAL-EDIT is 5000×1, which means we edit 5000 records in one model (global model) at one time. GLOBAL-EDIT is an ideal situation. Others are merging methods (500×10) where we edit 10 models and each model will be edited by 500 records. We merge 10 models into the global model by different methods and evaluate the final global model's performance.

Method	MCF						zsRE			
	NS ↑	**PS ↑**	**ES ↑**	**Score ↑**	**NE ↑**	**RS ↑**	**NA ↑**	**PA ↑**	**EA ↑**	**Score ↑**
GLOBAL-EDIT	65.08	80.66	89.66	77.08	622.05	36.97	25.25	64.71	68.96	43.12
TIES-MERGING	**78.46**	26.35	27.16	34.27	**627.15**	31.56	24.94	25.99	27.59	26.12
TASK-ARITHMETIC	66.84	55.19	61.66	60.85	613.72	31.83	24.97	33.66	34.80	30.45
SIMPLE-AVERAGE	76.90	29.97	33.06	39.15	626.47	31.49	**25.78**	29.26	30.62	28.4
COLLABEDIT	65.26	**80.67**	**89.70**	**77.18**	622.38	**37.10**	25.21	**64.27**	**68.40**	**42.95**

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

Datasets and models. For datasets, we use Multi-CounterFact (MCF) (Meng et al., 2022) and zsRE (Levy et al., 2017). For models, we use GPT2-XL (Radford et al., 2019).

Baselines. We compare our method with three naive collaborative editing methods (i.e., using current knowledge editing methods to update the local model and then use model fusion techniques to merge local updates to the global model), including SIMPLE-AVERAGE (Chronopoulou et al., 2023), TASK-ARITHMETIC (Ortiz-Jimenez et al., 2023), and TIES-MERGING (Yadav et al., 2023).

Evaluation metrics. Following Meng et al. (2022; 2023), for MCF, we use Efficacy Score (ES), Paraphrase Score (PS), Neighborhood Score (NS), N-gram Entropy (NE), Reference Score (RS), and Score (harmonic mean of ES, PS, NS); for zsRE, we use Neighborhood Accuracy (NA), Paraphrase Accuracy (PA), Efficacy ccuracy (EA), and Score (harmonic mean of NA, PA, EA). The exact definitions refer to Appendix C.

3.2 RESULTS AND DISCUSSION

Superior collaborative knowledge editing performance. As shown in Table 1, *our privacy-preserving solution* COLLABEDIT *achieves on-par editing performance with that of* GLOBAL-EDIT*, and significantly outperforms other naive model fusion methods in terms of the "Score"*. Though other baselines have a relatively higher NS value compared to GLOBAL-EDIT and our COLLABEDIT, we conjecture that it might be caused by the under-fitting phenomenon: these model fusion methods are not specifically designed for merging the weight updates from knowledge editing, which is reflected by their low values of PS, ES, and Score.

Discussion on the "overlapped" knowledge editing. In a collaborative knowledge editing system, multiple parties may apply the same or conflict editing action locally and simultaneously, and then be aggregated in the global model.

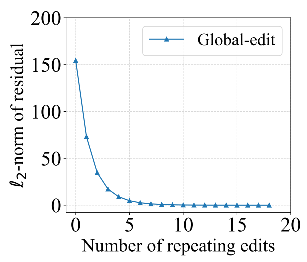


Figure 3: The ℓ_2-norm of residual $\mathbf{R}$ when data replication happens.

Such a pattern leads to overlapped or conflicting knowledge editing records, which may jeopardize the overall model performance (Li et al., 2024). Motivated by (2), we leverage the following equation[1], i.e., residual $\mathbf{R}_{\text{new}} := \mathbf{R}_{\text{old}} - \mathbf{\Delta K} = \mathbf{R}_{\text{old}} - \mathbf{R}_{\text{old}}\mathbf{K}^\top(\mathbf{C} + \mathbf{K}\mathbf{K}^\top)^{-1}\mathbf{K}$, to track the dynamics of knowledge editing. Preliminary experiments in Figure 3 show that as the number of repeating edits increases, the ℓ_2-norm of residual $\mathbf{R}$ reduces rapidly and becomes smaller than 0.01 when repeating edits for 12 times. This implies that the ℓ_2-norm of $\mathbf{R}$ can be used to check whether

[1]When editing some knowledge, we can obtain weights update $\mathbf{\Delta}$ and residual $\mathbf{R}_{\text{old}}$ by the input key $\mathbf{K}$. If we edit the same knowledge (i.e., same $\mathbf{K}$) after update the LLM with $\mathbf{\Delta}$, we can get the new residual $\mathbf{R}_{\text{new}} = \mathbf{R}_{\text{old}} - \mathbf{\Delta K}$.

"overlap editing" happens, which may be helpful for practitioners to avoid the decrease in model performance.

Limitations and future work. Our empirical observation and discussion on the overlapped knowledge editing shed light on exploring many interesting yet unsolved perspectives of collaborative knowledge editing, including but not limited to:

1. Conflict editing. For example, one party edits the fact that "The mother tongue of Danielle Darrieux is *Spain*", while another party edits "The mother tongue of Danielle Darrieux is *English*".
2. Multi-round editing. While our method could achieve perfect editing in a single round, it may not extend to multiple rounds, which is essential for ensuring long-term continual collaborative knowledge editing. Generalizing our method to multi-round scenarios is an important future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.

Afra Akyürek, Eric Pan, Garry Kuwanto, and Derry Wijaya. Dune: Dataset for unified editing. In Proceedings of Empirical Methods in Natural Language Processing (EMNLP), 2023.

Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. Adaptersoup: Weight averaging to improve generalization of pretrained language models. In Findings of the Association for Computational Linguistics (ACL), pp. 2009–2018, 2023.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In The Eleventh International Conference on Learning Representations (ICLR), 2023.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge unlearning for mitigating privacy risks in language models. In Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL), 2023.

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.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL), pp. 333–342, 2017.

Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. Unveiling the pitfalls of knowledge editing for large language models. In International Conference on Learning Representations (ICLR), 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In Neural Information Processing Systems (NeurIPS), 2022.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In International Conference on Learning Representations (ICLR), 2023.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In International Conference on Learning Representations (ICLR), 2022.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In 37th Conference on Neural Information Processing Systems (NeurIPS), 2023.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. Reasoning with language model prompting: A survey. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), 2023.

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

Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning. In International Conference on Learning Representations (ICLR), 2024.

Leijie Wu, Song Guo, Junxiao Wang, Zicong Hong, Jie Zhang, and Jingren Zhou. On knowledge editing in federated learning: Perspectives, challenges, and future directions. arXiv preprint arXiv:2306.01431, 2023.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS), 2023.

Ningyu Zhang, Yunzhi Yao, and Shumin Deng. Editing large language models. In Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: Tutorial Abstract (IJCNLP-AACL), pp. 23–28, 2023.

A Details of Knowledge Editing in a Single LLM

Details of identifying the critical path of MLP layers. Following MEMIT (Meng et al., 2023), we apply causal tracing to LLMs (e.g., GPT-2 XL) and identify the critical path of MLP layers to edit. For consistency, we edit the same set of layers $\mathcal{R}$ as MEMIT such as the 13-17th layers of GPT-2 XL.

Details of the closed form optimization of Δ for a single layer. We optimize the following objective to obtain the optimal weights $\mathbf{W}^*$ of layer l:

$$\mathbf{W}^* \triangleq \underset{\hat{\mathbf{W}}}{\arg\min} \left(\sum_{i=1}^{n} \left\| \hat{\mathbf{W}}\mathbf{k}_i - \mathbf{m}_i \right\|^2 + \sum_{i=n+1}^{n+|\mathcal{E}|} \left\| \hat{\mathbf{W}}\mathbf{k}_i - \mathbf{m}_i \right\|^2 \right), \tag{5}$$

where k_i $(1 \le i \le n)$ indicates the old keys derived from existing knowledge and k_i $(n+1 \le i \le n+|\mathcal{E}|)$ indicates the new keys derived from the edit requests $\mathcal{E}$.

Next, we denote $\mathbf{W}$ as the model weights before knowledge editing, $\mathbf{K}_{\text{init}} = [\mathbf{k}_1, \ldots, \mathbf{k}_n]$ as the set of old keys derived from existing knowledge and $\mathbf{K} = [\mathbf{k}_{n+1}, \ldots, \mathbf{k}_{n+|\mathcal{E}|}]$ as the set of new keys derived from the edit requests $\mathcal{E}$. Moreover, $\mathbf{M}_{\text{init}} = [\mathbf{m}_1, \ldots, \mathbf{m}_n] = \mathbf{W}\mathbf{K}_{\text{init}}$ represents the memory values of $\mathbf{K}_{\text{init}}$ that are previously stored and $\mathbf{M} = [\mathbf{m}_{n+1}, \ldots, \mathbf{m}_{n+|\mathcal{E}|}]$ represents the desired memory values of $\mathbf{K}$ that we aim to store. We can solve the Equation (5) by applying *the normal equation*:

$$\begin{aligned} (\mathbf{W} + \mathbf{\Delta})(\mathbf{K}_{\text{init}}\mathbf{K}_{\text{init}}^\top + \mathbf{K}\mathbf{K}^\top) &= \mathbf{M}_{\text{init}}\mathbf{K}_{\text{init}}^\top + \mathbf{M}\mathbf{K}^\top, \\ \mathbf{W}\mathbf{K}_{\text{init}}\mathbf{K}_{\text{init}}^\top + \mathbf{W}\mathbf{K}\mathbf{K}^\top + \mathbf{\Delta}\mathbf{K}_{\text{init}}\mathbf{K}_{\text{init}}^\top + \mathbf{\Delta}\mathbf{K}\mathbf{K}^\top &= \mathbf{M}_{\text{init}}\mathbf{K}_{\text{init}}^\top + \mathbf{M}\mathbf{K}^\top. \end{aligned} \tag{6}$$

In addition, we define two variables: (1) $\mathbf{C} \triangleq \mathbf{K}_{\text{init}}\mathbf{K}_{\text{init}}^\top$, which represents the covariance matrix of the input keys of existing knowledge. (2) $\mathbf{R} \triangleq \mathbf{M} - \mathbf{W}\mathbf{K}$, which represents the residual error of the new associations when evaluated on the old weights $\mathbf{W}$. Then, we can obtain the closed-form solution of the weight updates $\mathbf{\Delta}$ as:

$$\mathbf{\Delta} = \mathbf{R}\mathbf{K}^\top(\mathbf{C} + \mathbf{K}\mathbf{K}^\top)^{-1}. \tag{7}$$

We compute $\mathbf{C} = \mu \cdot \mathbb{E}_k\left[\mathbf{k}\mathbf{k}^\top\right]$, where $\mathbb{E}_k\left[\mathbf{k}\mathbf{k}^\top\right]$ is estimated as an uncentered covariance statistic collected using an empirical sample of vector inputs to the layer (e.g., 100,000 Wikipedia records). μ is a hyperparameter that balances the weighting of new v.s. old associations (a typical value of μ is 1.5×10^4 according to MEMIT).

Details of the implementation on simultaneously editing multiple layers. Previously we focus on illustrating how existing knowledge editing algorithms edit a single layer in the LLM. To simultaneously edit multiple layers of $l \in \mathcal{R}$, existing editing algorithms (e.g., MEMIT (Meng et al., 2023)) firstly obtain the desired output vector $\mathbf{z}_i$ of final layer in $\mathcal{R}$ that can maximize $\Pr[o_i|\mathbf{x} \oplus p(s_i, r_i)]$. Then, they spread the whole residual over all the layers in $\mathcal{R}$ by computing partial residual $\mathbf{r}_i^l = \frac{\mathbf{z}_i - \mathbf{W}_i^l \mathbf{k}_i^l}{L-l+1}$ of each layer, i.e., $l \in \mathcal{R}$. Then, the desired memory value of layer l can be computed as $m_i^l = \mathbf{W}_i^l \mathbf{k}_i^l + r_i^l$ and we can use Equation (7) to edit each layer. For details of the implementation, please also refer to Meng et al. (2023). In this work, we strictly follow their implementation to simultaneously edit multiple layers.

B Theoretical Analysis of the Methods

For ease of understanding, we will describe knowledge editing for a specific layer l and omit l for brevity. We denote $\mathbf{\Delta}_G$ and $\mathbf{\Delta}_i$ as the weight updates derived from GLOBAL-EDIT and client i's edit. $\mathbf{K}_G$ and $\mathbf{K}_i$ represent the new keys derived from all the edit requests and client i's edits requests. According to Section 2.1, $\mathbf{R}_G$ and $\mathbf{R}_i$ represent the residual errors in the output space of layer l derived from all the edit requests and client i's edits requests, respectively. $\mathbf{C}$ represents the aggregated statistic over the previously stored keys of existing knowledge. We consider the collaborative editing scenario with N clients and each client model has M edit requests.

B.1 Analysis of the non-destructive collaborative knowledge editing

Note that $\mathbf{\Delta}_i$ and $\mathbf{\Delta}_G$ can be computed via (2) as:

$$\begin{aligned} \mathbf{\Delta}_G &= \mathbf{R}_G \mathbf{K}_G^\top (\mathbf{C} + \mathbf{K}_G \mathbf{K}_G^\top)^{-1}, \\ \mathbf{\Delta}_i &= \mathbf{R}_i \mathbf{K}_i^\top (\mathbf{C} + \mathbf{K}_i \mathbf{K}_i^\top)^{-1}. \end{aligned} \tag{8}$$

Following the definitions of $\mathbf{K}$ and $\mathbf{R}$ in Section 2.1, we have:

$$\begin{aligned} \mathbf{K}_i &= [\mathbf{k}_{i\times(M-1)+1}, \mathbf{k}_{i\times(M-1)+2}, \cdots, \mathbf{k}_{i\times M}], \\ \mathbf{R}_i &= [\mathbf{r}_{i\times(M-1)+1}, \mathbf{r}_{i\times(M-1)+2}, \cdots, \mathbf{r}_{i\times M}], \\ \mathbf{K}_G &= [\mathbf{k}_1, \mathbf{k}_2, \cdots, \mathbf{k}_{N\times M}] = [\mathbf{K}_1, \mathbf{K}_2, \cdots, \mathbf{K}_N], \\ \mathbf{R}_G &= [\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_{N\times M}] = [\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_N]. \end{aligned} \tag{9}$$

Then we have:

$$\mathbf{R}_G \mathbf{K}_G^\top = \mathbf{R}_1 \mathbf{K}_1^\top + \mathbf{R}_2 \mathbf{K}_2^\top + \cdots + \mathbf{R}_N \mathbf{K}_N^\top. \tag{10}$$

According to Equations (8) and (10), we can obtain:

$$\begin{aligned} \mathbf{\Delta}_G(\mathbf{C} + \textstyle\sum_{j=1}^N \mathbf{K}_j \mathbf{K}_j^\top) &= \mathbf{\Delta}_G(\mathbf{C} + \mathbf{K}_1 \mathbf{K}_1^\top \cdots + \mathbf{K}_N \mathbf{K}_N^\top) \\ &= \mathbf{\Delta}_G(\mathbf{C} + \mathbf{K}_G \mathbf{K}_G^\top) \\ &= \mathbf{R}_G \mathbf{K}_G^\top \\ &= \mathbf{R}_1 \mathbf{K}_1^\top + \mathbf{R}_2 \mathbf{K}_2^\top + \cdots + \mathbf{R}_N \mathbf{K}_N^\top \\ &= \mathbf{\Delta}_1(\mathbf{C} + \mathbf{K}_1 \mathbf{K}_1^\top) + \cdots + \mathbf{\Delta}_N(\mathbf{C} + \mathbf{K}_N \mathbf{K}_N^\top) \\ &= \textstyle\sum_{i=1}^N \mathbf{\Delta}_i(\mathbf{C} + \mathbf{K}_i \mathbf{K}_i^\top). \end{aligned} \tag{11}$$

According to the Equation (11), we can finally reach the following conclusion:

$$\mathbf{\Delta}_G = \textstyle\sum_{i=1}^N \mathbf{\Delta}_i(\mathbf{C} + \mathbf{K}_i \mathbf{K}_i^\top)(\mathbf{C} + \sum_{j=1}^N \mathbf{K}_j \mathbf{K}_j^\top)^{-1}. \tag{12}$$

B.2 CollabEdit is privacy-preserving via directly sharing $\mathbf{K}\mathbf{K}^\top$

Next, we show that the design of sharing $\mathbf{K}\mathbf{K}^\top$ is privacy-preserving. Let's define input keys $\mathbf{K}$ as:

$$\mathbf{K} = [\mathbf{k}_1, \mathbf{k}_2, \cdots, \mathbf{k}_M] \in \mathbb{R}^{d\times M}, \tag{13}$$

where d indicates the dimension of the feature vector and M indicates the number of edit requests. In particular, we aim to prove that given $\mathbf{K}\mathbf{K}^\top$, it is nontrivial to reconstruct the $\mathbf{K}$. The problem is equal to proving that given any specific $\mathbf{K}\mathbf{K}^\top$, there exists an infinite number of $\mathbf{K}$ (different $\mathbf{K}$ may involve different M) that will lead to the same $\mathbf{K}\mathbf{K}^\top$.

To start with, let's assume there exists a matrix operation $\mathbf{W}' \in \mathbb{R}^{M \times M'}$, which can transform $\mathbf{K}$ into $\mathbf{K}'$ through $\mathbf{K}' = \mathbf{K} \cdot \mathbf{W}'$ and ensure that $\mathbf{K}'\mathbf{K}'^\top = \mathbf{K}\mathbf{K}^\top$. Then we have:

$$\mathbf{K}'\mathbf{K}'^\top = \mathbf{K}\mathbf{W}'^\top(\mathbf{K}\mathbf{W}')^\top = \mathbf{K}(\mathbf{W}'\mathbf{W}'^\top)\mathbf{K}^\top = \mathbf{K}\mathbf{K}^\top. \tag{14}$$

According to the Equation (14), we observe that any **orthogonal matrix** $\mathbf{W}'$ such that $\mathbf{W}'\mathbf{W}'^\top = \mathbf{I}$ will lead to the $\mathbf{K}'$ which has the same covariance matrix as $\mathbf{K}$. Since there exists an infinite number of orthogonal matrices when $M > 1$, we prove that it is nontrivial to reconstruct the $\mathbf{K}$ given $\mathbf{K}\mathbf{K}^\top$.

B.3 Analysis of the gap between two editing methods

According to the Equation (12), we obtain the relationship between $\mathbf{\Delta}_G$ with $\mathbf{\Delta}_i$ as:

$$\mathbf{\Delta}_G = \mathbf{\Delta}_1(\mathbf{C} + \mathbf{K}_1\mathbf{K}_1^\top)\mathbf{A}^{-1} + \cdots + \mathbf{\Delta}_N(\mathbf{C} + \mathbf{K}_N\mathbf{K}_N^\top)\mathbf{A}^{-1}. \tag{15}$$

Furthermore, we denote the weight updates derived from the destructive collaborative knowledge editing method using "Task-Arithmetic (TA)" as $\mathbf{\Delta}'_G$. We have:

$$\mathbf{\Delta}'_G = \lambda \times (\mathbf{\Delta}_1 + \mathbf{\Delta}_2 + \cdots + \mathbf{\Delta}_N). \tag{16}$$

Then, the gap between $\mathbf{\Delta}_G$ and $\mathbf{\Delta}'_G$ can be calculated as:

$$\begin{aligned}\mathbf{\Delta}_G - \mathbf{\Delta}'_G &= \sum_{i=1}^{N}(\mathbf{\Delta}_i(\mathbf{C} + \mathbf{K}_i\mathbf{K}_i^\top)\mathbf{A}^{-1} - \sum_{i=1}^{N}\lambda \times \mathbf{\Delta}_i \\ &= \sum_{i=1}^{N}\mathbf{\Delta}_i\left[(\mathbf{C} + \mathbf{K}_i\mathbf{K}_i^\top)(\mathbf{C} + \sum_{j=1}^{N}\mathbf{K}_i\mathbf{K}_i^\top)^{-1} - \lambda\mathbf{I}\right].\end{aligned} \tag{17}$$

C Evaluation Metrics

C.1 Metrics for Multi-CounterFact

Multi-CounterFact (MCF) contains an assortment of prompts and texts for evaluating model rewrites. For (s_i, r_i), knowledge editing aims to rewrite the old object o_i^c with the new desired object o_i. We use the same metrics as previous works (Meng et al., 2023) for evaluation:

- **Efficacy Success** (ES) is the proportion of cases where the new object o_i exceeds the old object o_i^c in probability:
$$\mathbb{E}_i\left[\Pr_{\mathcal{M}_{\boldsymbol{\theta}}}[o_i|p(s_i,r_i)] \geq \Pr_{\mathcal{M}_{\boldsymbol{\theta}}}[o_i^c|p(s_i,r_i)]\right]. \tag{18}$$
- **Paraphrase Success** (PS) is the proportion of cases where the new object o_i exceeds the old object o_i^c in probability on rephrasings of the original statement:
$$\mathbb{E}_i\left[\mathbb{E}_{p\in \text{paraphrases}(s_i,r_i)}\left[\Pr_{\mathcal{M}_{\boldsymbol{\theta}}}[o_i|p] > \Pr_{\mathcal{M}_{\boldsymbol{\theta}}}[o_i^c|p]\right]\right]. \tag{19}$$
- **Neighborhood Success** (NS) is the proportion of neighborhood prompts (all such prompts have the same old object o_i^c) where the model still assigns higher probability to the old object:
$$\mathbb{E}_i\left[\mathbb{E}_{p\in \text{neighborhood prompts}(s_i,r_i)}\left[\Pr_{\mathcal{M}_{\boldsymbol{\theta}}}[o_i|p] < \Pr_{\mathcal{M}_{\boldsymbol{\theta}}}[o_i^c|p]\right]\right]. \tag{20}$$

Additionally, the generation tests contain the following metrics:

- **Reference Score** (RS) measures the consistency of the model $\mathcal{M}_{\boldsymbol{\theta}}$'s free-form generations. To compute it, we first prompt $\mathcal{M}_{\boldsymbol{\theta}}$ with the subject s, then compute TF-IDF vectors for both $\mathcal{M}_{\boldsymbol{\theta}}(s)$and a reference Wikipedia text about o; RS is defined as their cosine similarity. Intuitively, $\mathcal{M}_{\boldsymbol{\theta}}(s)$ would match better with o's reference text if it has more consistent phrasing and vocabulary.

- We also check for excessive repetition (a common failure case with model editing) using **Generation Entropy** (N-gram Entropy, NE), which relies on the entropy of n-gram distributions:

$$-\left(\frac{2}{3}\sum_{k} f_2(k)\log_2 f_2(k) + \frac{4}{3}\sum_{k} f_3(k)\log_2 f_3(k)\right). \tag{21}$$

  Here, $f_n(\cdot)$ is the n-gram frequency distribution.

C.2 Metrics for zsRE

For the sake of consistency, we report the same three accuracy-based metrics as the previous work (Meng et al., 2023) to evaluate the editing performance on zsRE:

- **Efficacy Accuracy** (EA) is the proportion of edits that the model $\mathcal{M}_{\boldsymbol{\theta}}$ recalls with top-1 accuracy. Specifically, an edited model $\mathcal{M}_{\boldsymbol{\theta}}$ should correctly recall the target object o_i with the largest probability given a templated prompt $p(s_i, r_i)$ containing s_i and r_i:

$$\mathbb{E}_i\left[o_i = \arg\max_{o'_i} \mathrm{Pr}_{\mathcal{M}_{\boldsymbol{\theta}}}\left[o'_i | p(s_i, r_i)\right]\right]. \tag{22}$$

- **Paraphrase Accuracy** (PA) is the accuracy of rephrasings of the original statement:

$$\mathbb{E}_i\left[\mathbb{E}_{p\in \text{paraphrases}(s_i,r_i)}\left[o_i = \arg\max_{o'_i} \mathrm{Pr}_{\mathcal{M}_{\boldsymbol{\theta}}}\left[o'_i | p\right]\right]\right]. \tag{23}$$

- **Neighborhood Accuracy** (NA) is the proportion of neighborhood prompts that the model gets correct for the old object o_i^c:

$$\mathbb{E}_i\left[\mathbb{E}_{p\in \text{neighborhood prompts}(s_i,r_i)}\left[o_i^c = \arg\max_{o'_i} \mathrm{Pr}_{\mathcal{M}_{\boldsymbol{\theta}}}\left[o'_i | p\right]\right]\right]. \tag{24}$$

D Algorithm of Our CollabEdit

Algorithm 1 CollabEdit: Non-destructive Collaborative Knowledge Editing

Require: The number of clients N, edit requests $\mathcal{E}_i$ of each client $(1 \le i \le N)$ where $\mathcal{E}_i = \{(s_{ij}, r_{ij}, o_{ij}|j)\}$, language model $\mathcal{M}_{\boldsymbol{\theta}}$ with weights $\mathbf{W}^l$ of layer l, a set of MLP layers to edit $\mathcal{R}$, covariance matrix $\mathbf{C}$ of existing knowledge (optional for direct editing methods, e.g., MEMIT), Hyper-network $\mathcal{H}$ with learnable parameter κ_l for layer l (optional for hypernetwork-based editing methods, e.g., MALMEN), a set of prompt templates $\mathcal{P}$.

Ensure: Edited language model $\mathcal{M}_{\boldsymbol{\theta}}$ with updated weights $\mathbf{W}^* = \mathbf{W} + \boldsymbol{\Delta}$ of layer l.

```
1: $\boldsymbol{\Delta}_{list}$ = [ ] , $\mathbf{KKT}_{list}$ = [ ]
2: for $i \in \mathcal{N}$ do
3:     $\boldsymbol{\Delta}^i_{list}$ , $\mathbf{KKT}^i_{list} \leftarrow$ GetDeltaAndKKT ($\mathcal{E}_i$, $\mathcal{M}_{\boldsymbol{\theta}}$, $\mathbf{C}$, $\mathcal{H}$, $\mathcal{P}$)
4:     $\boldsymbol{\Delta}_{list}.append(\boldsymbol{\Delta}^i_{list})$ , $\mathbf{KKT}_{list}.append(\mathbf{KKT}^i_{list})$
5: for $l \in \mathcal{R}$ do
6:     $\mathbf{A} \leftarrow \mathbf{C}$
7:     $\mathbf{A} \leftarrow \kappa_l \mathbf{I}$
8:     for $i \in \mathcal{N}$ do
9:         $\mathbf{K}^l_i {\mathbf{K}^l_i}^\top = \mathbf{KKT}_{list}[i][l]$ , $\boldsymbol{\Delta}^l_i = \boldsymbol{\Delta}_{list}[i][l]$
10:        $\mathbf{A} \leftarrow \mathbf{A} + \mathbf{K}^l_i {\mathbf{K}^l_i}^\top$
11:        $\boldsymbol{\Delta}^l_i \leftarrow \boldsymbol{\Delta}^l_i \times (\mathbf{C} + \mathbf{K}^l_i {\mathbf{K}^l_i}^\top)$
12:        $\boldsymbol{\Delta}^l_i \leftarrow \boldsymbol{\Delta}^l_i \times (\kappa_l \mathbf{I} + \mathbf{K}^l_i {\mathbf{K}^l_i}^\top)$
13:        $\mathbf{W}^{*l} \leftarrow \mathbf{W}^l + \sum_{i=1}^{\mathcal{N}} \boldsymbol{\Delta}^l_i \times \mathbf{A}^{-1}$
```

Algorithm 2 GetDeltaAndKKT

```
1:  procedure GetDeltaAndKKT($\mathcal{E}_i$, $\mathcal{M}_{\boldsymbol{\theta}}$, $\mathcal{H}$, $\mathbf{C}$, $\mathcal{P}$)
2:      for $s_j, r_j, o_j \in \mathcal{E}_i$ do
3:          $L_j \leftarrow \frac{1}{|\mathcal{P}|}\sum_{k=1}^{|\mathcal{P}|} -\log \Pr_{\mathcal{M}_{\boldsymbol{\theta}}}[o_j|\mathcal{P}_k(s_j, r_j)]$
4:          optimize $\mathbf{z}_j \leftarrow \arg\min_{\mathbf{z}_j} L_j$
5:          Cache $L_j$
6:      $\boldsymbol{\Delta}_{list}$ = [], $\mathbf{KKT}_{list}$ = []
7:      for $l \in \mathcal{R}$ do
8:          $\mathbf{h}_i^l \leftarrow \mathbf{h}_i^{l-1} + \mathbf{a}_i^l + \mathbf{m}_i^l$
9:          for $s_j, r_j, o_j \in \mathcal{E}_{i,j}$ do
10:             $\mathbf{k}_i^l \leftarrow \mathbf{k}_i^l = \frac{1}{\mathcal{P}}\sum_{k=1}^{|\mathcal{P}|} \mathcal{P}_k(s_j, r_j)$
11:             $\mathbf{r}_i^l \leftarrow \frac{\mathbf{z}_j - \mathbf{W}^l\mathbf{k}^l}{\mathcal{R}[-1]-l+1}$
12:             $\mathbf{r}_i^l \leftarrow \mathcal{H}(\mathbf{k}_i^l, \nabla_{\mathbf{k}_i^l} L_j)\mathbf{k}_i^l$
13:          $\mathbf{K}^l \leftarrow [\mathbf{k}_1^l, ..., \mathbf{k}_i^l]$
14:          $\mathbf{R}^l \leftarrow [\mathbf{r}_1^l, ..., \mathbf{r}_i^l]$
15:          $\boldsymbol{\Delta}^l \leftarrow \mathbf{R}^l{\mathbf{K}^l}^\top(\mathbf{C}^l + \mathbf{K}^l{\mathbf{K}^l}^\top)^{-1}$
16:          $\boldsymbol{\Delta}^l \leftarrow \mathbf{R}^l{\mathbf{K}^l}^\top(\lambda_l\mathbf{I} + \mathbf{K}^l{\mathbf{K}^l}^\top)^{-1}$
17:          $\boldsymbol{\Delta}_{list}$.append($\boldsymbol{\Delta}^l$) , $\mathbf{KKT}_{list}$.append($\mathbf{K}^l{\mathbf{K}^l}^\top$)
18:      return $\boldsymbol{\Delta}_{list}$, $\mathbf{KKT}_{list}$
```