On the Retention of Edited Knowledge in Fine-Tuned Language Models

Fufang Wen
Columbia University
fw2325@columbia.edu

Shichang Zhang
Harvard University
shzhang@hbs.edu

Abstract

Large language models (LLMs) store vast amounts of knowledge, which often requires updates to correct factual errors, incorporate newly acquired information, or adapt model behavior. Model editing methods have emerged as efficient solutions for such updates, offering localized and precise knowledge modification at significantly lower computational cost than continual training. In parallel, LLMs are frequently fine-tuned for a wide range of downstream tasks. However, the effect of fine-tuning on previously edited knowledge remains poorly understood. In this work, we systematically investigate how different fine-tuning objectives interact with various model editing techniques. Our findings show that edited knowledge is substantially more susceptible to forgetting during fine-tuning than intrinsic knowledge acquired through pre-training. This analysis highlights a key limitation of current editing approaches and suggests that evaluating edit robustness under downstream fine-tuning is critical for their practical deployment. We further find that freezing layers associated with edited content can significantly improve knowledge retention, offering insight into how future editing methods might be made more robust.

1 Introduction

LLMs have shown the ability to store vast amounts of knowledge Radford et al. (2019). The training of LLMs often involves with different stages, including pretraining and finetuning. Finetuning is important as it can align LLMs with human intent Ouyang et al. (2022). LLMs can acquire fundamental knowledge during the pretraining stage and subsequently adapt to specific tasks through downstream fine-tuning. However, when a gap exists between the pretraining and downstream finetuning stage, LLMs face the risk of catastrophic forgetting Wang et al. (2023).

As real-world information continually evolves, software must be constantly updated to keep pace to avoid becoming outdated Lazaridou et al. (2021). LLMs are simply another type of software that requires regular updates, particularly for knowledge refresh. Standard fine-tuning can bring large computation costs and perform poorly in locality and generalization Mitchell et al. (2022b); Gangadhar & Stratos (2024). To this end, the concept of knowledge editing (KE) has been proposed, enabling data-efficient modifications to the model behavior, while ensuring no adverse impact on other inputs. Recently, many methods have been proposed to correct the erroneous or obsolete knowledge Fang et al. (2024); Meng et al. (2023b;a); Mitchell et al. (2022b). The KE methods can be classified into 3 main categories: 1. Locate-then-edit 2. Meta-learning 3. Memory-based. Each category has some representing methods that proved to be efficient and effective.

Previous work has primarily evaluated KE methods based on reliability, portability, locality and efficiency in post-edit models Yao et al. (2023). However, the impact of downstream finetuning on edited knowledge remains poorly understood. In this paper, we address a critical question: **how does downstream finetuning affect knowledge edited by KE methods, and how can we preserve them?**

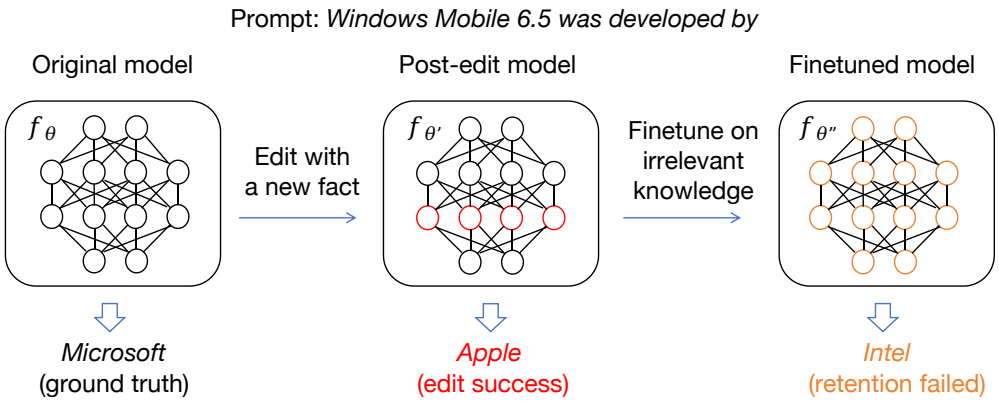

Figure 1: Demonstration of model editing and downstream model finetuning and their impact on the knowledge in LLMs. The original model is edited with a single instance of new fact: *Windows Mobile 6.5 was developed by Apple*, and the edited model is finetuned by an irrelevant dataset, which does not contain subject, relation and object from the edited knowledge. Although the edit can be successful, it is vulnerable to different downstream finetuning tasks. f_θ, $f_{\theta'}$, $f_{\theta''}$ denote the pretrained models, edited model and finetuned model respectively.

The downstream fine-tuning can also impair the model's intrinsic knowledge Lange et al. (2022); Wu et al. (2022); Wang et al. (2023). To differentiate the finetuning impact on model edited knowledge and intrinsic knowledge, we create a dataset for each of them, and compare their retention rate of post-edit model and finetuned model. We conduct experiments on different KE methods-including standard finetuning, Locate-then-edit, and meta-learning method and then finetune these post-edit models by downstream tasks including finetuning with unstructured and structured dataset, classification task as well as supervised finetuning(SFT) on multiple representative LLMs, including GPT-2 XL Radford et al. (2019) and Llama3-8B Dubey et al. (2024). We also compare the knowledge retention result for single-editing and batch-editing methods. For all the downstream tasks, we perform data filtering on the training set, exclusively preserving samples that are irrelevant to the edit knowledge. **We show different type of downstream finetuning tasks can affect differently to the intrinsic knowledge and edited knowledge.** In addition, how the downstream finetuning task affect edited knowledge can depend on the edit method. An example is shown in Figure 1, even edit can be successful for some edit methods, the edited knowledge can be vulnerable to downstream fine-tuning tasks.

2 Related work

KE has emerged as a promising paradigm for updating LLMs to adapt to dynamically evolving information. There are plenty of works about KE methods, which can be classified into 3 main categories: 1. Locate-then-edit 2. Meta-learning 3. Memory-based.

Fine-tuning (FT). Vanilla FT is a straightforward method to modify the knowledge in LLMs (Zhu et al., 2020). Model can be finetuned using the training corpus in the pretraining stage. Continual pre-training with a domain-specific corpus is an effective way to incorporate domain knowledge into a pre-trained model and enhance the its performance in that domain. However, the training corpus for fine-tuning often differs from the pre-training data. This can serve as a baseline for other model edit methods.

Locate-then-edit. The locate-and-edit method for LLMs knowledge editing involves first identifying the specific parts of the model where the target knowledge is stored (Dai et al., 2022; Meng et al., 2023b;a). Once localized, the method directly modifies the model's

weights or representations in those areas to update or correct the knowledge. This approach aims to precisely edit knowledge while minimizing broader impacts on the model's overall performance. ROME and MEMIT are examplars of this category. ROME edits factual knowledge in LLMs by identifying and updating specific rank-one subspaces in the model's weights, allowing precise, localized changes without retraining (Meng et al., 2023a). It leverages causal tracing to locate key layers and modifies them efficiently to correct or update facts while preserving the model's overall performance.

Meta-learning Method. Meta-learning for LLMs knowledge editing involves training a hyper-network to generate targeted parameter shifts that update the model's knowledge without full retraining Mitchell et al. (2022); Cao et al. (2022). This approach leverages the hyper-network to transform standard fine-tuning gradients into precise edits, ensuring generalization to semantically equivalent inputs while preserving unrelated knowledge. MALMEN is a representative of the meta-learning-based editing approach. MALMEN edits large language models by using a hyper-network to compute parameter shifts as a least squares problem, solved via the normal equation, enabling efficient and scalable updates while minimizing interference with unrelated knowledge.

Memory-based Methods. Memory-based methods store edits in a explicit memory without modifying the model's parameters Mitchell et al. (2022b); Zheng et al. (2023); Hartvigsen et al. (2023). For example, SERAC is a gradient-free memory-based model editing method that stores edits in an explicit memory and uses a scope classifier to determine if a test input is within the scope of any cached edits. If within scope, a counterfactual model predicts the label based on the most relevant edit example; otherwise, the base model's prediction is used Mitchell et al. (2022b).

Our work focuses specifically on these the first two parameter-modifying approaches, as they enable more fundamental alterations to the model's knowledge representations. Some work has show that KE methods fail to retain the model's accuracy on irrelevant knowledge and general ability Gupta et al. (2024); Gu et al. (2024), and the evaluation paradigm for model editing has been investigated (Cohen et al., 2024; Zhong et al., 2023). Besides, recent work reveals significant limitations in current methods, showing that their performance on real-world hallucinations often falls short of expectations, and highlights the need for further improvements in the field (Huang et al., 2025).

3 Retention analysis of edited knowledge after fine-tuning

3.1 Experiment setup

We evaluate whether a language model encodes specific knowledge using COUNTERFACT, a dataset contains counterfactual statements, is designed to distinguish superficial lexical changes from meaningful alterations in factual knowledge. In COUNTERFACT, each record is knowledge in a simple form that contains a subject, a relation and a object Meng et al. (2023a). Each COUNTERFACT instance is derived from PARAREL and consists of a true knowledge tuple $t^c = (s, r, o^c)$ and a false knowledge tuple $t^* = (s, r, o^*)$, where s stands for subject, r stands for relation, o^c stands for correct object and o^* stands for false object. The subject and relation can form a prompt. Edit methods in this paper are performed to edit the corresponding object. We create our edited knowledge dataset from COUNTERFACT train split and intrinsic knowledge dataset from the test split.

Edited knowledge dataset The edited knowledge dataset is constructed by sampling from the model's existing knowledge. To achieve this, we filter the dataset by retaining only instances that given the prompt $p = (s, r)$, the model assigns the true target token with the highest probability that significantly surpasses that of other tokens by a substantial margin. From this filtered set, we sample 50 instances to form the edited knowledge dataset.

Intrinsic knowledge dataset We create the intrinsic knowledge dataset for benchmark purpose. Similar to the edited knowledge dataset, the intrinsic knowledge dataset is crafted by sampling 100 data that the true target token is assigned with the significant highest probability. We make sure that there is no overlap between intrinsic knowledge dataset and

edited knowledge dataset for the subject, relation as well as object. More details about the dataset are shown in Appendix A.

Knowledge editing This study investigates three distinct KE methods: FT, ROME and MALMEN, that representing three categories of approaches: 1) Brutal-force Fine-tuning, 2) Locate-then-Edit, and 3) Meta-learning respectively. To enhance locality, we fine-tune only a specific layer while freezing all others, as this approach demonstrates superior locality compared to full-model fine-tuning Gangadhar & Stratos (2024). A study has been conducted to determine the optimal layer for knowledge edits Hase et al. (2023); Meng et al. (2023a). For FT and ROME method, we choose layer 1 and layer 6 to be the edit layer respectively. In all of these experiments, including FT, ROME and MALMEN, we stop editing until the loss goes below a threshold. As a result, we achieve perfect edit success rate.

Downstream fine-tuning We conduct four different type of downstream fine-tuning tasks 1) Fine-tuning with unstructured text, 2) Fine-tuning with structured factual text, 3) Classification tasks, and 4) Question-answer supervised finetuning task on GPT-2 XL and Question-answer supervised finetuning on Llama3-8B. To fairly compare these edit methods, it is essential to develop a metric to quantify the extent to which fine-tuning influences the model. To ensure that the fine-tuning impact is consistent across the four methods (FT, ROME, MEMIT and MALMEN), for each of these downstream fine-tuning tasks, we sample and fix an evaluation dataset from the fine-tuning dataset and standardize the training stopping criteria, and employ the same finetuning hyperparameter.

- **Fine-tuning with the Unstructured Dataset** To systematically assess the impact on both the model's intrinsic knowledge and the edited knowledge, we employed an unstructured dataset distinct from all pre-training corpora. Specifically, for our experiments on the GPT-2 XL, which was pre-trained on webtext, we choose the Common Crawl dataset for fine-tuning. Since a small subset of the data suffices to demonstrate the influence, we sampled 60k data for our training set. We evaluate the finetuning influence on a validation set, which contains of 1k instances sampled from the training set. We employ loss as a metric to measure the volume of knowledge acquired from fine-tuning. The finetuning is stopped once the validation loss goes below the threshold of 3.1.

- **Finetuning with the Structured Factual Dataset** To investigate the influence of structured factual data on model behavior, we construct a fine-tuning dataset that mirrors the structure of our edit data while containing distinct factual content. Specifically, we sample data from COUNTERFACT train split, which has no overlap with the one used in upstream edit. We select only instances where the pretrained model fails to predict the true target, and sample 3,000 of them. For supervision, we use the true target from each example as the training label. To monitor progress, we evaluate model performance on a validation set, which contains 100 instances sampled from the training set. Training terminates once the model achieves above 70% accuracy on this validation set.

- **Classification Task** To assess the impact of downstream fine-tuning on pretrained knowledge, we employ a classification task as our benchmark evaluation. Specifically, we utilize the IMDB sentiment analysis dataset which consists of 25k movie reviews paired with binary sentiment labels. For the classification architecture, we append a fully connected layer to the final hidden state of the end-of-sequence (EOS) token. Classification accuracy on the validation set can be a fair metric, as it is the target index people are aiming to improve. The validation set contains 100 samples, which has no overlap with the training set. We finetune the model until the model achieves a classification accuracy exceeding 95% on the validation set.

- **Supervised Finetuning Task** Supervised fine-tuning has emerged as a prevalent downstream adaptation method for LLMs. In this paradigm, each training instance consists of an instruction (question) paired with its corresponding response (answer). For our experiments, we employ the Tulu-3-SFT-mixture dataset Lambert et al. (2024) for instruction fine-tuning, which consists of 10k movie reviews paired with binary sentiment labels. To assess the training influence, we evaluate model performance on a validation set, which contains 1k instances sampled from the training set. Training

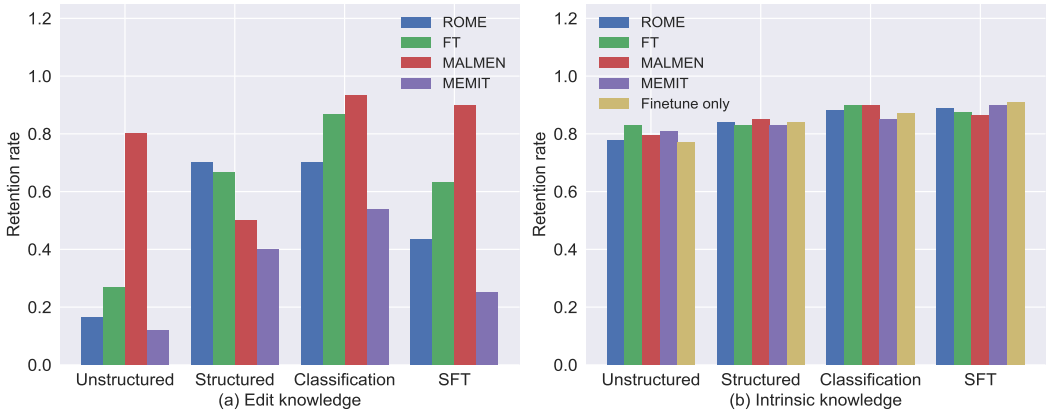

Figure 2: Edited and intrinsic knowledge retention rate after model edit and finetuning for different combination of upstream edit methods and downstream finetuning methods. For ROME method, we choose layer 6 as the edit layer. For FT method, we choose layer 1 as the edit layer.

terminates once the model achieves above 70% accuracy on this validation set. We evaluate the training progress by compute the model's loss on the validation set. We set the stopping criteria as validation loss goes below the threshold of 2.5.

To assess whether the model possesses the target knowledge, we input the prompt into the model and compute the probability distribution over the output token. The model can sometimes assigns the highest probability to a stop word but not the target word. However, this does not necessarily imply a lack of knowledge. In such cases, we employ greedy decoding to generate subsequent tokens until a non-stop token is produced. We consider the model to retain the knowledge if the first non-stop token matches the target token. We define the edited knowledge retention rate as the proportion of prompts for which the target token is the highest-ranked non-stop word:

$$\mathbb{E}_{x,y\sim\{(x,y)\}} 1\left\{\arg\max_{y} f_\theta(y \mid x) = y_t, y \notin S\right\} \tag{1}$$

Where y_t is the target token, $\{(x,y)\}$ is our edited/intrinsic dataset, S is the set of stopping word first token. We can obtain a intrinsic knowledge retention rate similar to this. We evaluate both the edited knowledge retention rate and intrinsic knowledge retention rate on both post-edit and finetuned model. Details about these model edit and finetuning hyper-parameters are shown in Appendix B.

For a combination of edit method and downstream finetuning task, we:

- conduct knowledge edition on the original model using this edit method; For single-edit method(ROME, FT and MALMEN), we update one knowledge from edited knowledge dataset at a time. For the batch-edit method(MEMIT), We partition the edited knowledge dataset into groups that containing multiple knowledge tuples, and update multiple knowledge for the model to derive the post-edit model at a time.
- Fine-tuning the post-edit model on the downstream tasks
- Assess the edited knowledge and intrinsic knowledge retention rate for both post-edit model and finetuned knowledge on the single edited knowledge and intrinsic knowledge dataset.

3.2 Finetuning impact on edited and intrinsic knowledge

Figure 2(a) and (b) show edited and intrinsic knowledge retention rates after knowledge edit and different finetuning tasks, respectively. The basic model is GPT-2 XL.

Our analysis reveals four key findings:

- **The edited knowledge exhibit lower retention rate than the intrinsic knowledge, which suggests that even when a fact is successfully inserted into a model, it is still inherently different from the intrinsic knowledge.** MALMEN achieves the best performance: it presents similar edit and intrinsic knowledge retention rate for unstructured data finetuning, classification, and SFT task, but lower retention rate for structured finetuning task. Since MEMIT edits multiple facts simultaneously, it introduces more extensive changes to the model compared to ROME. Consequently, it has lower edit knowledge retention rate. Among these edit methods, we find that ROME/MEMIT demonstrates lowest edited knowledge retention rates for unstructured, classification and SFT task. However, it is interesting that for the structured task, ROME presents the highest one. We hypothesize this occurs because ROME conditions the model to process information in the COUNTERFACT prompt format. When downstream tasks employ data formats similar to the pretraining data (including edited content), they tend to preserve pretraining knowledge more effectively. These results indicate that knowledge introduced through ROME editing is particularly vulnerable to format mismatches during downstream fine-tuning.
- **In contrast to the edited knowledge retention rate, the intrinsic knowledge retention rate remains relatively stable before and after all these downstream fine-tuning tasks.** The model edition decrease the intrinsic knowledge retention rate from 100% to around 80%. However, after finetuning, it remains around 80% for all of these tasks. This suggests that the model's inherent knowledge demonstrates greater robustness compared to newly edited knowledge across all four downstream tasks.
- Across different editing methods, intrinsic knowledge retention rates show negligible variation. Unlike edited knowledge retention rates—which is highly sensitive to the editing method—the preservation of intrinsic model knowledge remains relatively consistent regardless of the method for each downstream task.
- When performing downstream fine-tuning without upstream knowledge editing, we observe comparable intrinsic knowledge retention rates to cases with upstream knowledge editing. This demonstrates that upstream knowledge editing has negligible impact on the model's ability to retain its intrinsic knowledge.
- MEMIT achieves a lower knowledge retention rate than ROME, as it processes multiple fact edits (10 in our experiment) concurrently, while ROME applies edits sequentially, one fact at a time.

We also conduct experiments on the task of Question-answer supervised finetuning on a model with larger size (Llama3-8B), and this shows similar result. The detailed experiment results are shown in Appendix C.

3.3 Strategies to improve knowledge retention rate

For Locate-then-Edit method, the layer that the edited knowledge being inserted into is where the knowledge located Meng et al. (2023a); Geva et al. (2021). We hypothesize that preserving edited knowledge can be improved by avoiding fine-tuning of layers containing the target knowledge. To test this hypothesis, we evaluate two layer-specific fine-tuning strategies on GPT-2 XL:

- Freeze the early layers and fine-tune the layers beyond a specified threshold layer. As GPT-2 XL contains 48 transformer layers, we set the finetuning layer threshold to be 10, 20, 30, 40 and compare their result.
- Freeze all layers except a window of layers. We focus on two methods: ROME and FT, because both of them incorporate knowledge by editing a single layer. We set the window size to be 5. To reserve a larger room of layers for the experiment, for both FT and ROME, we choose layer 10 instead of 6.

Finetune after a layer threshold: From the Figure 3(a), we find that larger layer threshold can improve the edited knowledge retention rate while having similar intrinsic knowledge

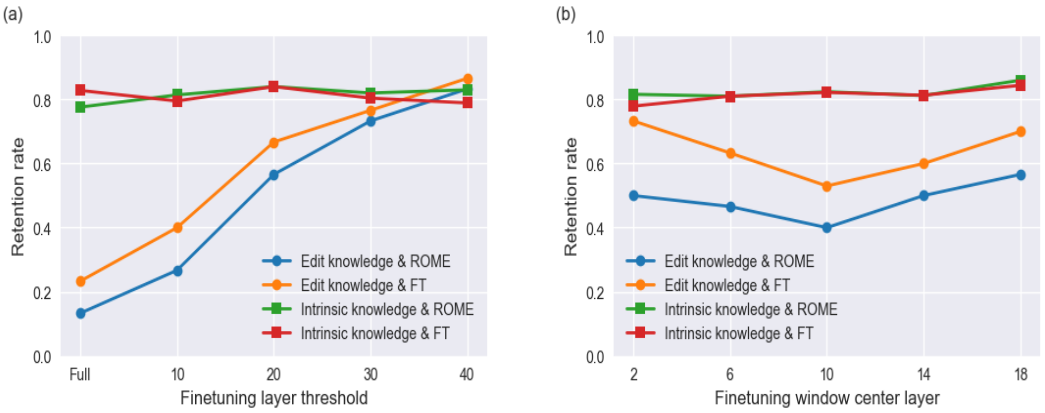

Figure 3: (a) Edit and intrinsic knowledge retention rate for different finetuning layer threshold for Rome and FT edit. We choose layer 1 as the edit layer for both ROME and FT method. (b) Edit and intrinsic knowledge retention rate for different finetuning window center layer for ROME and FT edit. We choose layer 10 as the edit layer for both ROME and FT method.

retention rate. If only layers after 40 are finetuned, the edited knowledge reaches the same level of knowledge retention with intrinsic knowledge. In addition, finetuning after a layer threshold has similar intrinsic knowledge retention rate with full finetuning retention rate, showing that this method does not impair the intrinsic knowledge.

Finetune a window of layers: From the Figure 3(b), we observe that fine-tuning achieves lowest edited knowledge retention rate when the window centered at layer 10-where the edited knowledge is located-while exhibiting higher edited knowledge retention rate for window does not include the edit layer, or even centered farther from this layer. In contrast, intrinsic knowledge maintains a similar knowledge retention rate across different window center positions.

3.4 Token distribution

We further analyze the distribution of output token during downstream finetuning after model editing. We classify output tokens into four categories:

Edited target token: The target editing token in t^*

True token: The original correct token in t^c

Related-term token: Tokens belong to the same category (e.g., piano, violin, and guitar as musical instrument; or England, France, and Brazil as country). The sentence's meaning would be changed if substituted

Other tokens: Completely unrelated tokens

Figure 4 illustrates the evolution of output token distributions throughout the fine-tuning process for both edit knowledge and intrinsic knowledge, when given prompt. We conduct experiments on ROME as the editing method and using unstructured datasets for fine-tuning. We average the probability of the first output token of all the 50 experiments.

There is a huge difference between edited knowledge and intrinsic knowledge probability: edited knowledge target token probability is above 99%, while intrinsic knowledge target token probability is around 60%. Despite this, after finetuning, the intrinsic knowledge can be better retained.

We see that the probability of generating edit token first decrease quickly, and the probability of generating related-term token increase quickly, and both of them finally converge. The probability of generating true target token is low. Our results show that for all of the three

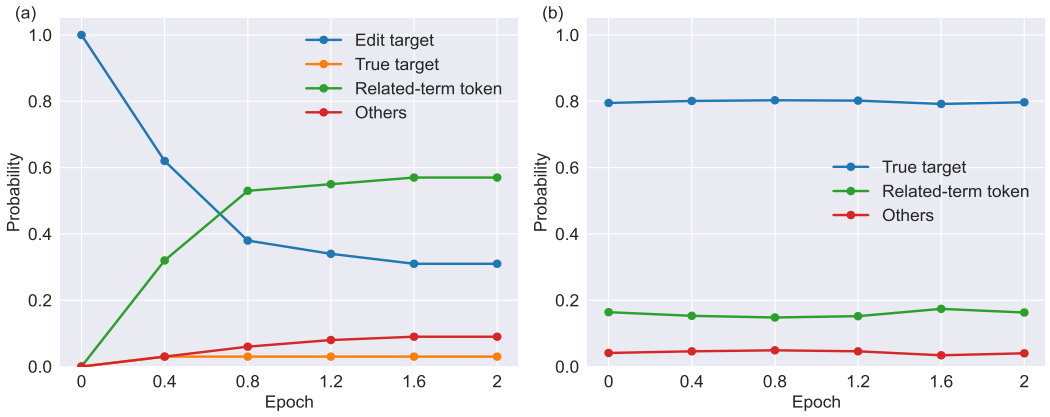

Figure 4: (a) First generated token distribution vs training epoch for edited knowledge. (b) First generated token distribution vs training epoch for intrinsic knowledge.

edit method, the model would unlikely generate the original true target. It would usually generate the related-term token, where the wrongly generated sentence is still fluent. This indicate that ROME works well on erasing the previous knowledge. Contrast to the edit knowledge, the token distribution fo intrinsic knowledge is more stable, and thus more robust to finetuning. After the model edition, the intrinsic knowledge drops from 100% to around 80%. However, the downstream finetuning has little impact to it compared to the edited knowledge.

Table 1: Model's Top-3 tokens and their probability in different stages for the prompt of *"Windows Mobile 6.5 was developed by"*, true target of *"Microsoft"* and the edit target of *"Apple"*.

Model	Top3-tokens
Original model	Microsoft (0.258) — Nokia (0.201) — the (0.107)
Post-edit model	Apple (0.992) — Nokia (0.004) — Google (0.001)
0.4 finetuned epoch	Apple (0.297) — Intel (0.179) — Nokia (0.004)
0.8 finetuned epoch	Intel (0.183) — Apple (0.126) — IBM (0.056)
1.2 finetuned epoch	Intel (0.170) — Nokia (0.051) — Apple (0.049)
1.6 finetuned epoch	Intel (0.254) — Nokia (0.210) — Google (0.106)
2 finetuned epoch	Intel (0.247) — Nokia (0.147) — Google (0.08)

We present a case study using a knowledge tuple chosen from COUNTERFACT in Table 1. The knowledge is edited using ROME, followed by fine-tuning the post-edit model on unstructured datasets. Initially, the model predict the target token of *"Microsoft"* correctly. After model edition, the post-edit model predict an extremely high probability of 0.992 to the edited target token *"Apple"*, and the true target Microsoft has very low probability(lower than 0.001). However, after just one and a half epoch of fine-tuning, this token disappears from the top-3 predicted tokens. Interestingly, while the true target *"Microsoft"* does not achieve high probability, *"Intel"*—a related-term token—shows increasing probability. The top-3 token rankings stabilize after one epoch of fine-tuning. Notably, even after removing all the text that contains non-stop keyword tokens from the prompt (*"window"*, *"mobile"*, and *"develop"*) in the fine-tuning dataset, this related-term tokens can still achieve the highest ranking after fine-tuning.

4 Discussion

4.1 Experiment fairness

To prevent potential conflicts between the fine-tuning data and edited knowledge—where the former may contain contradictory information (either correct or incorrect) that could degrade the edited knowledge—we systematically remove all text containing non-stop words from both the prompt and the true target. This ensures that the fine-tuning process does not inadvertently expose the model to conflicting knowledge. The fine-tuning data may also enhance the true intrinsic knowledge, so we also remove all text containing non-stop words from the intrinsic dataset.

For a finetuning task, ensuring consistent fine-tuning impacts across different post-edit models is challenging but critical for fairness. We set the same stopping criteria for finetuning different post-edit models. Some experiments take more training epoch to reach the stopping criteria. For example, we note that finetuning with larger layer threshold needs larger training epoch. Notably, experiments with larger layer thresholds require more training epochs to meet these criteria, as fewer parameters are updated, demanding greater training effort to incorporate the same volume of knowledge into the model. For FT and ROME, we also try different edited layer. We find that for both of the methods, edit early layer (prior to layer 10) can achieve similar result. More details about this experiment are shown in Appendix D.

4.2 Possible reason why edited knowledge has low retention rate

There is a key difference between edited knowledge and intrinsic knowledge: The intrinsic knowledge in models is gained from diverse expressions in the unstructured pre-training corpus, which makes the knowledge more robust to subsequent finetuning. However, edited knowledge gained from existing KE methods typically enforce modifications only on certain layers of the model or rely on a single expression pattern, making them less natural than intrinsic knowledge. As a result, it is a trade-offs between editing convenience and long-term stability: unstructured pre-training has long-term stability but low efficiency, and Editing method has higher efficiency but lower knowledge stability.

Section 3.3.2 shows that avoiding downstream fine-tuning the edited layer, and finetune layers farther from edited layer would helps preserve edited knowledge. However, intrinsic knowledge appears to be less sensitive to the choice of fine-tuning layers compared to edited knowledge. We hypothesize that this is because the edited knowledge (acquired from ROME or FT with some layers) is more localized than intrinsic knowledge (acquired during pretraining). Specifically, edited knowledge is concentrated within thee modified layer, whereas intrinsic knowledge is likely distributed across multiple layers, though some layer is more prominent than others (Meng et al., 2023a).

5 Limitations

While our layer-freezing approach provides a solution for preserving single-edited knowledge, it introduces key limitations. Firstly, it restricts the model's ability to acquire new knowledge during finetuning. Secondly, this method would decrease the training efficiency. Lastly, it cannot handle the extreme case that multiple edits that all layers become occupied by prior edits. As a result, development of a method for preserving multiple edits knowledge without compromising model plasticity or training efficiency would be an important future direction.

6 Conclusion and Future Work

We examine how different downstream fine-tuning tasks affect previous edited knowledge. First, we demonstrate that knowledge incorporated via KE methods are particularly sensitive to downstream fine-tuning data, and none of ROME, FT and MALMEN can edit the

knowledge as robust as intrinsic knowledge that suit for all the downstream finetuning task. Locate-and-edit(ROME) fails at finetuning with unstructured data, while FT and MALMEN fails at structured data. We hypothesize that this occurs because locate-and-edit methods condition the model to process structured knowledge tuples. As there is a format difference between structured knowledge and unstructured knowledge, downstream fine-tuning on knowledge with different structure will consequently impair the edited knowledge. Second, our analysis reveals that when the model fails to retain edited knowledge, it typically outputs related tokens that belongs to the same category rather than the true target token or completely unrelated ones. Finally, we propose potential mitigation strategies for this issue, primarily by avoiding modifying layers that relating to the single-edited knowledge. An interesting future direction is to see how this method works in the situation of synchronous editing for multiple cases. Another future direction is to develop more robust editing method that can resist fine-tuning drift. For future KE methods, post-edit retention should be assessed not only immediately after editing, but also after fine-tuning.

References

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. *In The Tenth International Conference on Learning Representations, ICLR*, pp. 5338–5348, 2022.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *Trans. Assoc. Comput. Linguistics*, 12: 283—–298, 2024.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 8493—–8502, 2022.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, and Angela Fan. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Xiang Wang, Xiangnan He, and Tatseng Chua. Alphaedit: Null-space constrained knowledge editing for language models. *In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16801–16819, 2024.

Govind Gangadhar and Karl Stratos. Model editing by standard fine-tuning. *ACL*, pp. 5338–5348, 2024.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key value memories. *In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue. *In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16801–16819, 2024.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. *In Findings of the Association for Computational Linguistics: ACL*, pp. 15202–15232, 2024.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 2023.

Peter Hase, Mohit Bansal, Kim Been, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Conference on Neural Information Processing Systems*, pp. 5338–5348, 2023.

Baixiang Huang, Canyu Chen, Xiongxiao Xu, Ali Payani, and Kai Shu. Can knowledge editing really correct hallucinations? *ICLR*, pp. 5338–5348, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. *arXiv:2411.15124*, 2024.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Anal. Mach. Intell*, pp. 3366–3385, 2022.

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, and Sebastian Ruder. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, pp. 29348–29363, 2021.

Kevin Meng, David Bau, Alex Andonian, Emma Pierson, and Yonatan Belinkov. Locating and editing factual associations in GPT. *International conference on machine learning*, pp. 5338–5348, 2023a.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *ICLR*, pp. 00363, 2023b.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. *In The Tenth International Conference on Learning Representations, ICLR*, pp. 5338–5348, 2022.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory based model editing at scale. *ICML*, pp. 15817–15831, 2022b.

Long Ouyang, Jeff Wu, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, pp. 5338–5348, 2019.

Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language model via meta learning. *Conference on Neural Information Processing Systems*, pp. 5338–5348, 2023.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *CoRR*, 2023.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, and Guilin Qi. Pretrained language model in continual learning: A comparative study. *ICLR*, 2022.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. *In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10222–10240, 2023.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? *ACL*, pp. 5338–5348, 2023.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. Mquake: Assessing knowledge editing in language models via multi-hop questions. *In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 6-10:15686–15702, 2023.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix X. Yu, and Sanjiv Kumary. Modifying memories in transformer model. *CoRR*, pp. 00363, 2020.

A Details about dataset

Task name	Data source	#Train data points	#Validation data points
Unstructured	Common Crawl	60,000	1,000
Structured	CounterFact	3,000	100
Classification	IMDB	25,000	100
SFT (GPT-2 XL)	Tulu-3-SFT-mixture dataset	10,000	1,000
SFT (Llama3-8B)	Tulu-3-SFT-mixture dataset	100,000	1,000

Table 2: Information for finetuning dataset.

In CounterFact, given prompt, we observe that models can generate top-ranked tokens with nearly identical probabilities(difference less than 0.03), indicating the model is uncertain about the knowledge even it may predict the true target correctly. To mitigate this ambiguity and ensure the model confidently possesses the target knowledge, we filter both the edit and intrinsic knowledge datasets, retaining only samples where the true target token's probability exceeds the second-ranked token's probability by at least 0.1.

Applying this criterion, we filter out 1,434 and 4,939 samples from the CounterFact training split for GPT-2 XL and Llama3-8B, respectively. From the remaining data, we sample 50 prompts for the edit dataset and 100 prompts for the intrinsic dataset. Detailed statistics of the filtered datasets are provided in Table 3.

Task name	# Data points	Avg true target prob	Avg second ranked target prob
GPT-2 XL Edit	50	0.407	0.088
GPT-2 XL Intrinsic	100	0.409	0.092
Llama3-8B Edit	50	0.467	0.112
Llama3-8B Intrinsic	100	0.473	0.119

Table 3: Information for edit and intrinsic data for GPT-2 XL and Llama3-8B.

As the GPT-2 XL and Llama3-8B possess different knowledge on the CounterFact dataset, necessitating the construction of different edit datasets and intrinsic knowledge datasets for each model. Compared to GPT-2 XL, Llama3-8B not only demonstrates more comprehensive knowledge coverage but also exhibits significantly higher prediction confidence for the target facts in CounterFact.

B implementation details

B.1 Hyperparameter for model editing

Fine-tuning(FT) without constraint: We choose Adam optimizer with learning rate of 5e-5, maximum training step of 25, weight decay of 0 and early stopping loss of 0.01. We finetune one specific layer's mlp_{proj} of the model. We choose layer 1 as the edit layer for both GPT-2 XL and Llama3-8B.

ROME: We emply the same hyper-parameters for ROME as the setting in original paper Meng et al. (2023a): We choose learning rate of 0.5, maximum training step of 50, weight

Edit method	ROME	FT	MALMEN
Edited knowledge	0.374	0.634	0.815
Intrinsic knowledge	0.824	0.831	0.817

Table 4: Edit and intrinsic knowledge retention rate after different edit methods and SFT for Llama3-8B.

decay of 0.5 and KL factor of 0.0625. We perform model edition on one layer of the model. We choose layer 6 as the edit layer for both GPT-2 XL and Llama3-8B.

MEMIT: We choose learning rate of 0.2, maximum training step of 50, weight decay of 0.003, editing layers ranging from 3 to 8 and KL factor of 0.0625. We edit 10 facts in each model edition, so we perform 5 model edition experiments for each finetuning task.

MALMEN: We adopt the same hyper-parameters for MALMEN as those used in the original paper Tan et al. (2023). We observe that choosing later layers can achieve better edition success rate. We select the model editing hyperparameter such that the edited model can predict true target for all the prompts in our edit dataset. Specifically, for GPT-2 XL, we edit layers ranging from 43 to 48(out of 48), while for Llama3-8B, we edit layers ranging from 27 to 32(out of 32).

B.2 Hyperparameter for finetuning

Unstructured finetuning: For GPT-2 XL, in align with its pretraining stage, we employ batch size of 256, AdamW optimizer with learning rate of 5e-5, beta of (0.9,0.999) and weight decay of 0.01. The training data set contains 60k data, and the validation data set contains 1k data. The model is evaluated on the validation set every 20 steps. The training stops once the validation loss goes below 3.1.

Structured finetuning: We employ batch size of 256, AdamW optimizer with learning rate of 2e-5, beta of (0.9,0.999) and weight decay of 0.01. The training data set contains 3,000 data, and the validation data set contains 100 data. The model is evaluated on the validation set every 20 steps. The training stops once the accuracy on the validation set goes above 70%.

Sentiment classification: We employ batch size of 64, AdamW optimizer with learning rate of 2e-5, beta of (0.9,0.999) and weight decay of 0.01. The training data set contains 25,000 data, and the validation data set contains 100 data. The model is evaluated on the validation set every 5 steps. The training stops once the accuracy on the validation set goes above 95%.

Supervised finetuning(SFT): For GPT-2 XL, we use an AdamW optimizer with learning rate of 2e-5, with beta of (0.9,0.999) and weight decay of 0.01. The training data set contains 10,000 data, and the validation data set contains 1,000 data. Training is performed with a batch size of 64, and the model is evaluated on the validation dataset every 20 steps. We halt training once the validation loss falls below 1.7.

For Llama3-8B, we employ AdamW optimizer with learning rate of 2e-5, beta of (0.9,0.999), batch size of 256, and weight decay of 0.01. The training data set contains 100,000 data, and the validation data set contains 1,000 data. The model is evaluated on the validation dataset every 20 steps. Due to its larger capacity and greater training requirements, it needs more training effort, so we set the validation loss of 1.0 as the training stopping criteria and evaluate the model on the validation set every 20 steps.

C Experiment results of Llama3-8B

To evaluate knowledge retention in a larger-scale model and a different model architecture, we conduct SFT experiments on Llama3-8B. We employ the same methodology as GPT-2 XL, but with a larger training dataset and stricter stopping criteria. Our results are shown in Table 4. We observe similar result as GPT-2 XL: Edited and intrinsic knowledge has similar

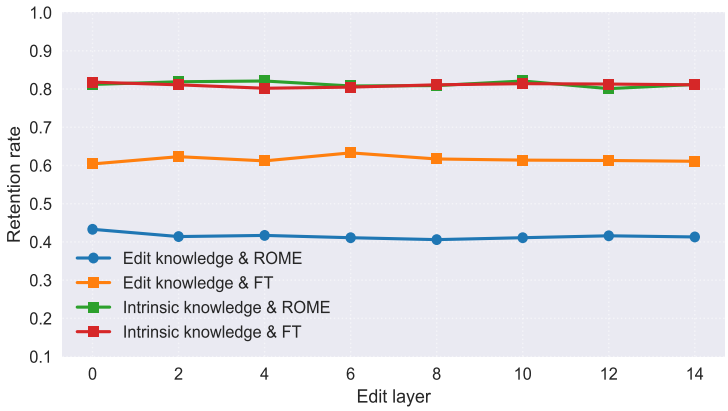

Figure 5: Edit and intrinsic knowledge retention rate for different edit layer for ROME and FT methods.

retention rate for MALMEN. However, for FT and ROME, edited knowledge retention rate is much lower than intrinsic knowledge.

Notably, while MALMEN maintains comparable retention rates for edited and intrinsic knowledge, this does not imply that edited knowledge is the inherently same as intrinsic knowledge. The intrinsic knowledge retention rate has a drop during model edition, from 1.0 to 0.8, whereas the edited knowledge retention rate starts at 1.0 post-editing.

D Experiment results for different edit layer

We analyze how the choice of editing layer influences model performance during downstream SFT on GPT-2 XL. Figure 5 compares knowledge retention rates for both edited and intrinsic knowledge across different layers when applying ROME and fine-tuning (FT) methods. Our experiments demonstrate that for early layers, ROME and FT achieve comparable edited knowledge and intrinsic knowledge retention rates across all edited layers.