SeqMMR: Sequential Model Merging and LLM Routing for Enhanced Batched Sequential Knowledge Editing

Anonymous ACL submission

Abstract

Model knowledge editing enables the efficient correction of erroneous information and the continuous updating of outdated knowledge within language models. While existing research has demonstrated strong performance in single-instance or few-instance sequential editing and one-time massive editing scenarios, the batched sequential editing paradigm remains a significant challenge. The primary issue lies in the model's tendency to gradually forget previously edited knowledge and become increasingly unstable after multiple iterations of batched editing. To address these challenges, we propose **SeqMMR**, an enhanced framework for batched sequential knowledge editing that leverages **Seq**uential **M**odel **M**erging and a model **R**outer. Our approach iteratively merges parameters from current batch-edited models with those of their predecessors, ensuring that newly emerging knowledge is integrated while mitigating the forgetting of previously edited knowledge. Furthermore, the model router directs queries unrelated to the edited knowledge to an unedited model backup, preventing unintended alterations in model predictions. Extensive experiments across various datasets demonstrate that our approach effectively mitigates knowledge forgetting, improves performance across all previous batches, and better preserves the model's general capabilities.

1 Introduction

Large language models (LLMs) have exhibited remarkable performance across a wide range of natural language processing (NLP) tasks, serving as repositories of extensive factual knowledge within their parameters (Touvron et al., 2023; OpenAI, 2023; DeepSeek-AI et al., 2025). However, their knowledge remains inherently limited in coverage and accuracy, often relying on outdated information (Onoe et al., 2022; Dhingra et al., 2022; Liška et al., 2022) or generating erroneous, hallucinatory, and biased content (Zhao et al., 2023; Ji et al., 2023; Lazaridou et al., 2021; Agarwal and Nenkova, 2022; Gallegos et al., 2023). Given the continuous evolution of world knowledge and the necessity of correcting inaccuracies, the field of *knowledge editing* has garnered increasing research attention.

Existing knowledge editing methods can be broadly classified into parameter-updating and parameter-preserving approaches. Parameter-updating methods (Cao et al., 2021; Meng et al., 2022a,b; Li et al., 2023; Fang et al., 2024) follow the locate-then-edit paradigm to modify specific model parameters associated with knowledge storage. In contrast, parameter-preserving methods (Mitchell et al., 2022a; Tan et al., 2024; Zheng et al., 2023; Zhong et al., 2023; Hartvigsen et al., 2023; Yu et al., 2024a) either train hypernetworks to dynamically adjust model outputs or modify outputs by appending constructed prompts to input queries, leaving the model parameters unchanged.

While prior research has primarily focused on single-instance sequential editing or one-time massive editing, real-world model maintenance requires *batched and sequential* editing to continuously update knowledge as it evolves. ICL-based methods (Zheng et al., 2023; Zhong et al., 2023) face inefficiencies and temporary edits, while meta-learning-based methods (Mitchell et al., 2022a; Tan et al., 2024) are optimized for individual instances, limiting batch effectiveness. In contrast, parameter-updating approaches (Meng et al., 2022b; Li et al., 2023; Fang et al., 2024) enable large-scale editing in a single step, making them more practical for continuous model updates.

A critical challenge in batched sequential editing is ensuring the stability of sequential parameter updates. Recent studies (Gu et al., 2024; Ma et al., 2024; Fang et al., 2024) have shown that parameter-updating methods in sequential editing tasks suffer from model degradation due to the accumulation of parameter shifts, prompting efforts

to mitigate this issue. (Gupta et al., 2024b) also observed that models continuously forget previously edited knowledge and loses the ability to perform downstream tasks. Addressing these limitations is crucial for developing scalable and stable knowledge editing frameworks capable of supporting continuous, large-scale updates in LLMs. To this end, we propose an enhanced batched sequential knowledge editing framework based on sequential model merging and a model router.

Specifically, we iteratively merge the parameters of the current batch-edited model with those of the previous model, enabling the merged model to retain the latest edited knowledge while preventing the forgetting of previously stored knowledge. Similar to sequential parameter updates, sequential model merging needs to address interference between merging parameters. Unlike multi-task model merging, sequential merging does not have access to model parameters from future time steps, necessitating an interference handling solution based on self-awareness at the current time step. Furthermore, to better handle knowledge outside the editing scope, we design a model router that routes edit-unrelated queries to the unedited model backup, ensuring their predictions are unaffected by the editing process. The main contributions of our work can be summarized as follows:

1. We apply model merging methods to the batched sequential knowledge editing task, iteratively enabling the model to acquire new knowledge while preserving the original knowledge of the predecessor model. We treat a batch of edits as a "*task*" in model merging and perform interference handling based on self-awareness on the corresponding *task vectors* to alleviate potential parameter conflicts across time steps.

2. We introduce a model router component to route queries unrelated to knowledge editing to the unedited model backup, while ensuring editing-related queries are routed to the edited model. This improves the accuracy of processing the model's original knowledge without compromising the editing performance.

3. Extensive experiments on different datasets demonstrate that our proposed method mitigates the forgetting of previously edited knowledge, leading to comprehensive performance improvements across all past batches while better preserving the model's general capabilities.

2 Related Works

2.1 Knowledge Editing

The knowledge editing task aims to correct erroneous knowledge or update outdated knowledge within a language model while ensuring that other knowledge remains unaffected. Previous works (Mitchell et al., 2022a; Meng et al., 2022b; Li et al., 2023; Qiao et al., 2024; Tan et al., 2024; Mitchell et al., 2022b; Jiang et al., 2024) have made gradual progress and achieved excellent performance on standard knowledge editing datasets, such as CounterFact (Meng et al., 2022a) and ZsRE (Levy et al., 2017). Recently, the issue of model degradation in sequential knowledge editing scenarios has garnered widespread attention.

Some studies have explored the instability of models in sequential knowledge editing, highlighting issues such as forgetting previously edited knowledge (Gupta et al., 2024b; Huang et al.) and degradation of general capabilities (Li et al.). Consequently, recent works have focused on achieving lifelong knowledge editing (Hartvigsen et al., 2023; Chen et al., 2024; Hu et al., 2024a; Gupta et al., 2024a), though most are designed for sequential editing in single-instance scenarios. Meanwhile, other studies (Gu et al., 2024; Ma et al., 2024) also aim to stabilize model parameters to preserve general capabilities. The recently proposed AlphaEdit (Fang et al., 2024) achieves strong editing performance and stable general capabilities in batched sequential editing scenarios.

Moreover, some studies have explored knowledge editing in different paradigms, such as multi-hop editing (Zhong et al., 2023; Bi et al., 2024; Lu et al., 2024), ripple effects of edits (Cohen et al., 2024), commonsense knowledge editing (Huang et al., 2024), event-level editing (Liu et al., 2024), and long-form evaluation (Rosati et al., 2024). These efforts introduce meaningful directions and challenges, further advancing research in the field of knowledge editing.

2.2 Model Merging and LLM Router

Model merging was initially introduced as a training-free approach to integrate multiple models fine-tuned on downstream tasks. The process involves computing the parameter differences between fine-tuned models and the base model, re-

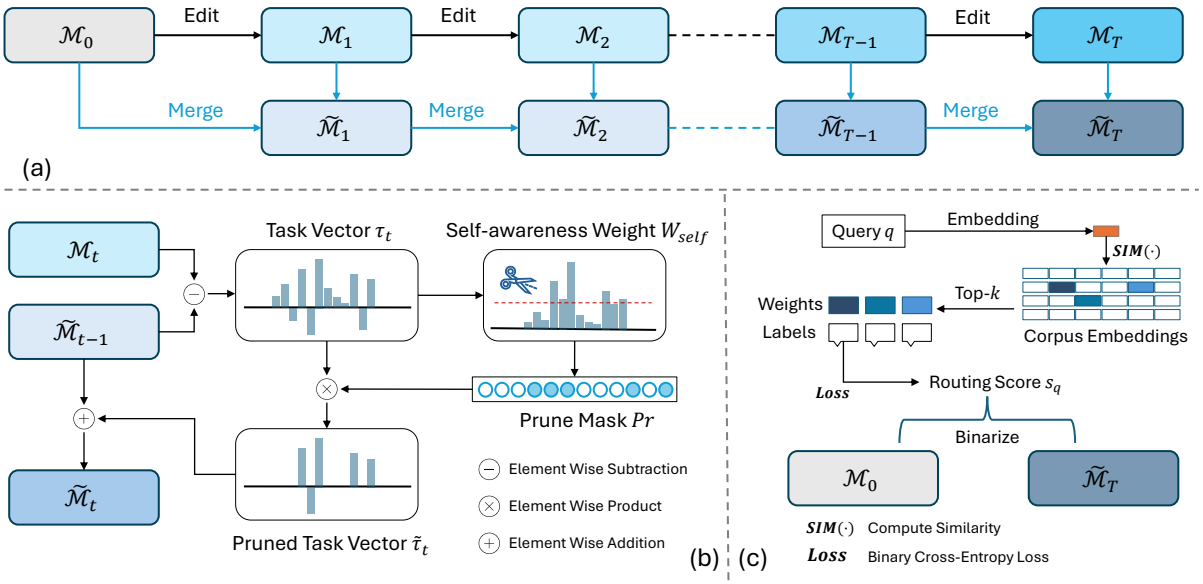

Figure 1: Overview of our methods. Figure (a) illustrates our batched sequential model merging workflow, which iteratively merges the current sequentially edited model $\mathcal{M}_t$ with the last timestep merged model $\tilde{\mathcal{M}}_{t-1}$, resulting in the merged model $\tilde{\mathcal{M}}_t$ for the current set of editing batches. Figure (b) details the model merging process within a single time step. The task vector τ_t is computed as the parameter difference between $\mathcal{M}_t$ and $\tilde{\mathcal{M}}_{t-1}$. Self-awareness weights W_{self} are then computed to guide the pruning of redundant parameters. The pruned task vector $\tilde{\tau}_t$ is subsequently integrated with the predecessor merged model $\tilde{\mathcal{M}}_{t-1}$ to update the model parameters. Figure (c) illustrates the model router process. Given a user query q, the router calculates the similarity between its embedding and the corpus embeddings. The corpus samples with top-k similarity weights are selected to compute a routing score s_q, which is then binarized and used as the routing label l_q for the query.

ferred to as *task vector* (Ilharco et al.). The core challenge in model merging is managing interference among multiple task vectors. Recent approaches, such as weighted merging (Matena and Raffel, 2022), sign election (Yadav et al., 2023), and parameter sparsification (Yu et al., 2024b; Du et al., 2024), have contributed to advancements in model merging tasks.

Model router aims to address the challenge of balancing time, computational costs, and task performance across numerous large models with varying capabilities and costs (Stripelis et al., 2024; Hu et al., 2024b). It typically involves analyzing or training preference datasets corresponding to different models (Ong et al., 2024; Shnitzer et al., 2023). Various model routers can be designed, including training-free or model-based approaches.

3 Method

3.1 Preliminaries

Batched Sequential Knowledge Editing Existing mainstream methods capable of batch editing typically involve adding a trained perturbation $\mathbf{\Delta}$ to the model's parameters $\mathbf{\Theta}$, thereby altering the model's predictions for specified queries. When a piece of knowledge is formalized as (s, r, o), the subject s, relation r, and object o constitute a factual statement (e.g., s ="Cybertruck", r = "is manufactured by", o = "Tesla"). It can be represented as $k - v$ pairs in the model parameter, where k encodes the query prompt (s, r) and v encodes the answer o.

Given a batch of new knowledge $\mathcal{E} = \{e_i\}_{i=1}^{B}$, where $e_i = (s_i, r_i, o_i)$, the perturbation $\mathbf{\Delta}$ is optimized under the constraint: $(\mathbf{\Theta} + \mathbf{\Delta})\mathbf{K} = \mathbf{V}$, where $\mathbf{K}$ and $\mathbf{V}$ are the collections of k_i and v_i, respectively. In the context of batched sequential editing, for a total of T editing batches, each batch generates a $\mathbf{\Delta}_t$ based on the predecessor model, responsible for updating the parameters. Therefore, for a given knowledge editing method $\mathcal{G}$, at the t-th editing batch, the process of batched sequential knowledge editing can be formally expressed as:

$$\mathcal{M}_t = \mathcal{G}(\mathcal{M}_{t-1}, \mathbf{\Delta}_t), t \in \{1, 2, ..., T\} \quad (1)$$

where M_t is the model after t batches of editing.

Model Merging Model merging can integrate the task-specific capabilities of two or more models by computing and processing task vectors. Techniques such as pruning and sparsification serve as

important strategies for mitigating interference between task vectors. Given N task vectors $\{\tau_i\}_{i=1}^{N}$ derived from N fine-tuned models' parameters $\{\theta_{\mathcal{M}_i}\}_{i=1}^{N}$, they are individually or collectively processed using interference mitigation methods $\mathcal{F}$, after which the processed task vectors $\{\hat{\tau}_i\}_{i=1}^{N} = \mathcal{F}(\{\tau_i\}_{i=1}^{N})$ are merged into the base model $\mathcal{M}$:

$$\tilde{\mathcal{M}} \leftarrow \theta_{\mathcal{M}} + \sum_{i=1}^{N} \lambda_i \tilde{\tau}_i \tag{2}$$

where λ_i are the merge coefficients for different task vectors.

To align with the batched sequential editing task, the perturbation $\mathbf{\Delta}$ computed during batch editing can be regarded as a task vector, hereafter denoted as τ. When applying model merging sequentially during knowledge editing, each merge at the t-th batch involves two models: the currently edited model $\mathcal{M}_t$ and the predecessor merged model $\tilde{\mathcal{M}}_{t-1}$. Since only a single task vector τ_t is generated at each step, no interference occurs between task vectors within the same timestep. However, due to the accumulation of task vectors over timesteps, interference mitigation remains necessary to reduce the impact on model parameter distribution and prevent potential conflicts arising across different timesteps.

3.2 Sequential Model Merging for Batched Knowledge Editing

Our approach aims to integrate the recently edited knowledge of the current model with the memory of previous batches of edits from the predecessor model through model merging, the workflow as illustrated in Figure 1(a).

We define the model after t batches of sequential editing as $\mathcal{M}_t$ and the model after our sequential merging process as $\tilde{\mathcal{M}}_t$, where $\mathcal{M}_0 = \tilde{\mathcal{M}}_0$ represents the unedited base model. In each batch of editing, we merge the model after the current batch of edits with the predecessor merged model, the corresponding task vector is computed as:

$$\tau_t = \theta_{\mathcal{M}_t} - \theta_{\tilde{\mathcal{M}}_{t-1}} \tag{3}$$

Similar to the instability and forgetting issues caused by the iterative accumulation of parameters in sequential editing, sequential model merging also involves the iterative accumulation of task vectors. Therefore, handling interference between task vectors across time steps is crucial for achieving optimal merging performance and stability. Since the task vectors to be generated in future batches are not visible during the current batch, and maintaining task vectors from past batches incurs additional memory overhead as the number of iterations increases, we focus on optimizing the current task vector to reduce any potential interference and conflicts. Inspired by (Du et al., 2024), we employ a self-awareness weight-based pruning approach to sparsify the task vectors.

Given a current task vector τ_t, we first compute its normalized Hadamard-product H with itself to quantify the importance of its parameters:

$$H = Normalize(\tau_t \odot \tau_t) \tag{4}$$

Subsequently, we apply the $Softmax$ function as a nonlinear activation to emphasize parameters with significant contributions while suppressing redundant parameters with minor contributions, thereby obtaining self-awareness weights W_{self} for parameter pruning:

$$W_{self} = [h_1, h_2, ..., h_D], h_i = \frac{e^{H_i}}{\sum_{j=1}^{D} e^{H_j}} \tag{5}$$

where H_i denote the i-th row of H, D is the dimension of task vector. We then specify a pruning ratio r to sparsify the task vector based on the self-awareness weights. Specifically, we retain only the top $(1-r)\%$ of the parameters while discarding the rest, the pruning mask Pr can be defined as:

$$Pr_i = \begin{cases} 1, & \text{if } h_i \text{ in top-}(1-r)(W_{self}) \\ 0, & \text{else} \end{cases} \tag{6}$$

In our experiments, we observed that the majority of the parameters are redundant, with over 80% of them being prunable.

After obtaining the pruned task vector, we apply it to the predecessor merged model to generate the merged model for the current iteration:

$$\begin{aligned} \tilde{\tau}_t &= Pr \odot \tau_t \\ \tilde{\mathcal{M}}_t &\leftarrow \theta_{\tilde{\mathcal{M}}_{t-1}} + \tilde{\tau}_t \end{aligned} \tag{7}$$

Figure 1(b) illustrates the merging process within a single time step, as described above. Due to the highly sparse nature of effective parameters in the task vector, extensive pruning based on self-awareness weights effectively retains the current knowledge while reducing potential interference and conflicts with future task vectors. This merging process consistently preserves knowledge from

Method	Score ↑	Efficacy ↑	Generalization ↑	Locality ↑	Fluency	Consistency
Unedited	12.87	7.85	10.58	89.48	635.44	24.15
FT	60.08	87.95	80.90	38.17	391.14	5.54
PMET	71.92	79.50	77.48	61.62	529.97	20.66
PRUNE	77.20	86.15	83.35	65.56	610.79	29.26
RECT	81.80	92.40	85.55	70.61	622.20	30.44
MEMIT	78.37	89.35	87.20	64.02	614.21	32.11
$\text{SeqMMR}_{\text{MEMIT}}$	85.15	93.35	89.55	<u>74.89</u>	618.44	32.10
AlphaEdit	<u>88.01</u>	<u>99.85</u>	**96.02**	73.22	<u>622.88</u>	**34.82**
$\text{SeqMMR}_{\text{Alpha}}$	**90.95**	**99.90**	<u>95.00</u>	**80.34**	**623.68**	<u>33.86</u>

Table 1: Performance comparison of LLaMA3-8B model on the **CounterFact** dataset. The batch size is set to 200, with a total of 10 sequential editing batches, resulting in 2,000 knowledge edits. $\textbf{SeqMMR}_{\textbf{MEMIT}}$ and $\textbf{SeqMMR}_{\textbf{Alpha}}$ represent the results of applying model merging and the model router under the MEMIT and AlphaEdit knowledge editing methods, respectively. The best results are in bold, second-best are underlined.

the previous model during sequential execution, mitigating the issue of forgetting previously edited knowledge and leading to comprehensive improvements in editing performance across all batches.

3.3 Top-k Similarity Weighted Router

Although our model merging method mitigates forgetting of previously edited knowledge, frequent parameter updates inevitably affect the model's prior knowledge outside the editing scope, which should ideally be handled by the unedited model. To address this, we designed a training-free model router that routes queries related to edited knowledge to the sequentially merged model, while directing unrelated queries to the unedited backup model.

To formally describe the details of the routing process, let $\mathcal{E} = \{e_i\}_{i=1}^{n}$ denote the set of n newly edited knowledge examples, labeled as 1, and $\mathcal{U} = \{u_i\}_{i=1}^{m}$ denote m examples unrelated to editing, sampled from data outside the test set (detailed in Appendix B), labeled as -1. Here, the label indicates whether the example should be routed to the edited model or the original model. The computation corpus $\mathcal{C}$ consists of $\mathcal{E}$ and $\mathcal{U}$, i.e., $\mathcal{C} = \{\mathcal{E}, \mathcal{U}\}$. We then use an embedding model to compute embeddings ϵ_i for each instance in the corpus. For a given query q, we compute its similarity weight β_i with each corpus instance as follows:

$$\beta_i = \exp\left(1 + \frac{\epsilon_q \cdot \epsilon_i}{||\epsilon_q|| ||\epsilon_i||}\right) \tag{8}$$

This similarity weight is then used to compute the routing score. To ensure the significance of high-weighted samples, we retain the top-k largest weights and set the remaining weights to 0, thereby eliminating the influence of weakly similar samples on the score computation, i.e., $\hat{\beta}_i = \text{top-}k(\beta_i)$. Then, the routing score s_q can be computed as follows:

$$s_q = \arg\min_{\tilde{s}_q} \sum_{i=1}^{n+m} [\hat{\beta}_i \cdot L(\tilde{s_q}, l_i)] \tag{9}$$

where $l_i \in \{-1, 1\}$ denote the model label of examples, L represent the Binary Cross-Entropy loss. The routing label l_q used for final model selection is obtained by simply binarizing s_q:

$$l_q = \begin{cases} 1, & \text{if } s_q \geq 0 \\ -1, & \text{if } s_q < 0 \end{cases} \tag{10}$$

where a label of 1 routes the query to the edited model, while a label of -1 routes it to the unedited model. The overall process of the router is shown in Figure 1(c).

4 Experiments

4.1 Datasets and Baselines

We evaluate our method on widely used knowledge editing datasets, **CounterFact** (Meng et al., 2022a) and **KnowEdit** (Zhang et al., 2024), with detailed descriptions and examples provided in Appendix C.

To evaluate the general capabilities of the model, we conducted tests on the General Language Understanding Evaluation **(GLUE)** benchmark (Wang et al., 2018), which includes six downstream tasks: Stanford Sentiment Treebank (**SST**) (Socher et al., 2013), Massive Multi-task Language Understanding (**MMLU**) (Hendrycks et al., 2021), Microsoft Research Paraphrase Corpus (**MRPC**) (Dolan and Brockett, 2005), Recognizing Textual Entailment

Method	Edit Succ. ↑		Portability ↑		Locality ↑
	Rewrite Acc.	Res. Acc.	Subj. Ali.	Logic. Gen.	Rel. Sepc.
Unedited	31.48	50.92	22.17	52.83	-
MEMIT	70.17	54.66	29.15	46.84	58.59
$\text{SeqMMR}_{\text{MEMIT}}$	75.05	56.16	30.72	**49.44**	**65.53**
AlphaEdit	<u>93.68</u>	<u>58.63</u>	**33.85**	47.91	59.98
$\text{SeqMMR}_{\text{Alpha}}$	**93.78**	**58.83**	<u>33.31</u>	<u>48.63</u>	<u>65.35</u>

Table 2: Performance comparison of LLaMA3-8B model on the ZsRE subset of **KnowEdit** dataset. The batch size is set to 100, with a total of 10 sequential editing batches, resulting in 1,000 knowledge edits. The best results are in bold, second-best are underlined.

(**RTE**) (Bentivogli et al., 2009), Corpus of Linguistic Acceptability (**CoLA**) (Warstadt et al., 2019), and Natural Language Inference (**NLI**) (Williams et al., 2018).

We select the fine-tuned model and representative knowledge editing methods as baselines for comparison, including **PMET** (Li et al., 2023), **RECT** (Gu et al., 2024), **PRUNE** (Ma et al., 2024), **MEMIT** (Meng et al., 2022b), and **AlphaEdit** (Fang et al., 2024). Since our method can be broadly applied to parameter-updating knowledge editing methods, we choose MEMIT, which performs well in batched knowledge editing, and the recently proposed state-of-the-art AlphaEdit as base models to test the effectiveness of our method across different knowledge editing approaches.

4.2 Metrics and Settings

We use a variety of metrics across different datasets to evaluate our method. In the CounterFact dataset, three main knowledge editing metrics are included: **Efficacy**, **Generalization** and **Locality**, with detailed definitions provided in Appendix A. The overall edit **Score** is represented by the harmonic mean of the above three metrics. Additionally, **Fluency** measures excessive repetition in the model outputs, while **Consistency** evaluates the cosine similarity between the TF-IDF vectors of the model outputs and a reference Wikipedia text. In the KnowEdit dataset, the evaluation metrics are also categorized into three types: **Edit Success**, **Portability**, **Locality**, as detailed in Appendix A. For evaluating the general capabilities of the model, we use the **F1** scores on six downstream tasks from the GLUE dataset as the evaluation metric.

We use the widely used open-source model LLaMA3-8B as the backbone for experimental testing. For the hyperparameters, in model merging, we set the pruning ratio Pr to 85%, and in the model router, we set top-k value k to 2. For the embedding model used in the model router, we use the *text-embedding-3-small* model from OpenAI. All of our experiments were conducted on NVIDIA RTX A6000 48G GPUs.

5 Results

5.1 Performance on CounterFact

Table 1 presents the results of our method on the CounterFact dataset. We apply model merging and model routing to MEMIT and AlphaEdit methods, denoted as $\text{SeqMMR}_{\text{MEMIT}}$ and $\text{SeqMMR}_{\text{Alpha}}$, respectively. Both approaches demonstrate improvements in overall **Score**. Compared to MEMIT, $\text{SeqMMR}_{\text{MEMIT}}$ shows a significant enhancement in editing performance. Even when applied to AlphaEdit, a strong baseline known for both stability and effectiveness, $\text{SeqMMR}_{\text{Alpha}}$ further improves overall editing performance.

Among all evaluation metrics, locality shows the most notable improvement, primarily due to the combined effects of model merging, which enhances overall performance, and the model router, which directs editing-unrelated queries to the unedited model with optimal locality. These results demonstrate that our method better preserves the prior knowledge and mitigates the forgetting issue in batched sequential knowledge editing.

5.2 Performance on KnowEdit

We further evaluated our method on the ZsRE subset of the KnowEdit dataset, as it provides a more diverse set of evaluation metrics. Table 2 shows that $\text{SeqMMR}_{\text{MEMIT}}$ improves performance across all metrics compared to the MEMIT baseline. When applied to the stronger AlphaEdit baseline, $\text{SeqMMR}_{\text{Alpha}}$ shows a slight decrease in the Subject Aliasing Accuracy but achieves improvements on the remaining four metrics, ultimately leading to better overall performance.

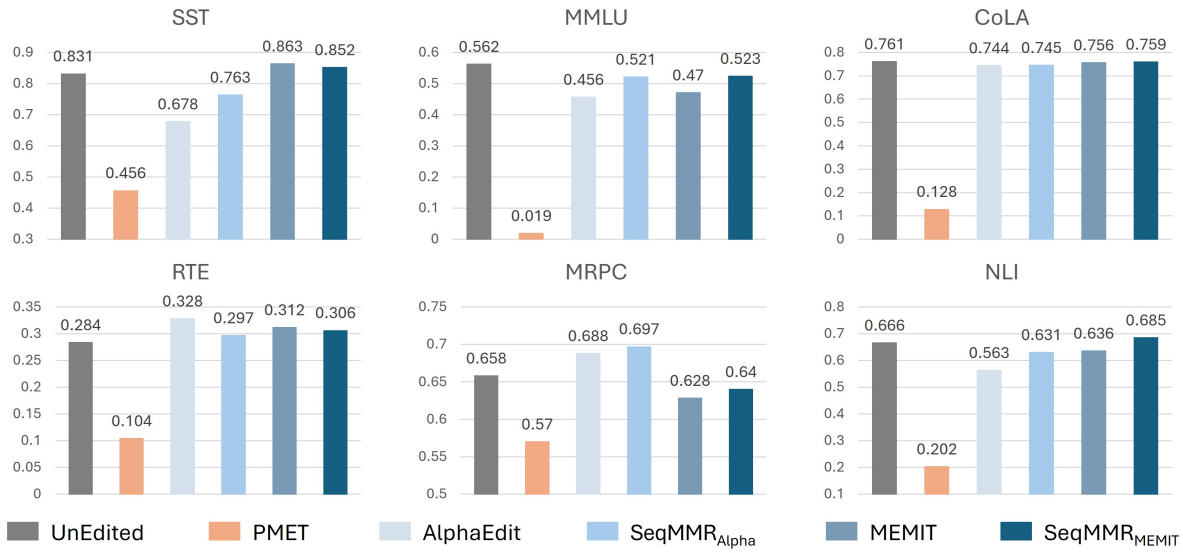

Figure 2: Comparison of general capabilities. The edited model is expected to retain the general capabilities of the unedited model.

5.3 General Capability

In addition to evaluating the knowledge editing capability, an important criterion for assessing the stability of knowledge editing methods is whether the ability to handle other downstream tasks is preserved. To this end, we tested the model's general capabilities across six downstream task datasets from the GLUE benchmark.

Figure 2 compares of **F1** scores of SeqMMR with other baseline models across six downstream tasks. SeqMMR outperforms its corresponding baselines in five tasks—SST, MMLU, CoLA, MRPC, and NLI—achieving scores comparable to the unedited model. Interestingly, in the RTE task, the model after editing performed better than the unedited model, while SeqMMR showed a slight decrease, aligning its performance more closely with the unedited model. Overall, SeqMMR improves the stability of the edited model's performance across various downstream tasks, effectively preserving the model's general capabilities.

5.4 Ablation Studies

We conducted comprehensive ablation experiments to evaluate the effectiveness of our proposed sequential model merging method and model router component, as well as to test the model's performance under different hyperparameter settings.

5.4.1 Ablation Study on Modules

We evaluate the contributions of the merging operation and the model router to each batch of edits on the CounterFact dataset. The results of MEMIT baseline as shown in Table 3 (more details in Table 6), and the AlphaEdit baseline, as shown in Table 5, demonstrate that applying our sequential model merging method to the baseline leads to an overall improvements across previously edited batches. Notably, earlier batches, which suffer from more severe forgetting, benefit more significantly from our approach. Only under the strong baseline AlphaEdit does the editing performance of the final batch experience some degradation, primarily due to task vector sparsification during the model merging process. However, with the further incorporation of the model router, editing performance improves further across all batches. Ultimately, the combined approach significantly mitigates the forgetting issue across all previously edited batches, leading to a notable enhancement in overall editing performance.

5.4.2 Ablation Study on Pruning Ratio

We further investigated the impact of different pruning ratios in model merging on editing performance. Specifically, we evaluated the performance of each editing batch under four different pruning ratios using SeqMMR$_{\text{Alpha}}$. As shown in Figure 3, higher pruning ratios better preserve the editing performance of earlier batches, as the increased sparsity of the task vector reduces its impact on the parameter distribution of the merged model. However, excessive pruning can lead to substantial loss of recently edited knowledge, particularly for the latest

Epoch	Ablation of Modules		
	MEMIT	+MM	+MM+R
@1	67.96	73.06 (+5.10)	76.05 (+8.09)
@2	72.92	78.44 (+5.52)	81.15 (+8.23)
@3	75.31	80.00 (+4.69)	83.72 (+8.41)
@4	76.40	80.91 (+4.51)	84.49 (+8.09)
@5	80.13	84.25 (+4.12)	87.65 (+7.52)
@6	80.77	83.41 (+2.64)	87.99 (+7.22)
@7	80.72	83.01 (+2.29)	87.14 (+6.42)
@8	81.98	83.85 (+1.87)	88.68 (+6.70)
@9	83.09	84.26 (+1.17)	88.24 (+5.15)
@10	81.09	81.39 (+0.30)	84.96 (+3.87)
Overall	78.37	81.42 (+3.05)	85.15 (+6.78)

Table 3: Editing **Scores** under the ablation study. **+MM** and **+R** indicate the use of model merging and model routing methods, respectively. **@k** represents the results at the k-th sequential editing batch. The values in parentheses indicate the difference compared to the baseline model.

top-k	Edit. Req.	Para. Pro.	Neigh. Pro.
2	100	99.98	57.52
3	99.85	98.88	60.29
5	98.60	98.00	62.46
10	96.35	95.48	65.56

Table 4: Classification accuracy under different top-k values in the model router for the CounterFact dataset. *Edit. Req.*, *Para. Pro.*, and *Neigh. Pro.* represent Editing Request, Paraphrase Prompt, and Neighborhood Prompt, respectively.

batch. Since it has not yet been compensated by subsequent task vectors, aggressive pruning results in a sharp decline in editing performance. Therefore, we select 85% as an appropriate pruning ratio, ensuring a highly sparse task vector while maintaining the integrity of the knowledge from the most recently edited batches.

5.4.3 Ablation Study on Router Top-k

We also tested the impact of different values of k for selecting the top-k corpus embeddings in the model router. In the CounterFact dataset, in-scope queries (*Editing Request* and *Paraphrase Prompt*) should be routed to the edited model. Incorrectly routing them to the unedited model would severely degrade the knowledge editing performance, as the unedited model is not capable of handling the new knowledge. On the other hand, out-of-scope queries (*Neighborhood Prompt*) are expected to be routed to the unedited model, as it provides

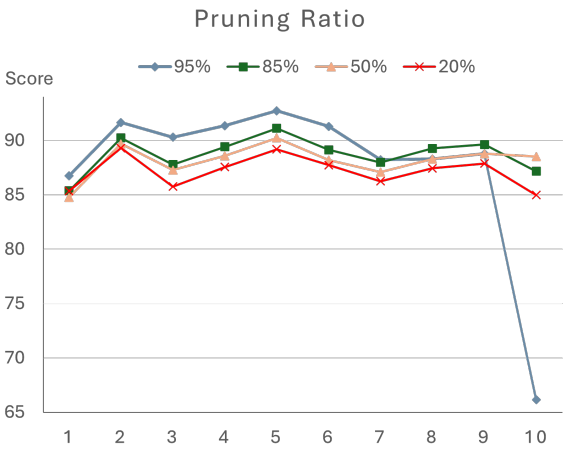

Figure 3: Performance of $\text{SeqMMR}_{\text{Alpha}}$ under different pruning ratios. The horizontal axis represents the batched sequential editing rounds, while the vertical axis represents the editing **Score**.

untouched original knowledge, leading to better locality. If routed to the edited model, the locality performance would follow that of the edited model.

Based on this, we prioritize ensuring accurate routing for in-scope queries while routing a portion of out-of-scope queries to the unedited model to improve locality without compromising editing performance. Table 4 shows the effect of different k-values on routing accuracy. As k increases, the proportion of out-of-scope queries routed to the unedited model also increases. However, this also impacts the routing accuracy for in-scope queries, which introduces greater risk. Therefore, we selected $k = 2$ to achieve the desired effectiveness.

6 Conclusion

In this work, we introduces SeqMMR, a novel approach to addressing the challenges in batched sequential knowledge editing for large language models. By iteratively merging the current batch-edited model with the previous merged one, our method preserves newly integrated knowledge while mitigating the forgetting of prior edits. Additionally, the incorporation of a model router enables editing-unrelated queries to be processed by an unedited model backup, leading to optimal locality on these queries. Extensive experiments show that SeqMMR effectively alleviates knowledge forgetting and enhances the model's performance across all previous edit batches, while also ensuring stable general capabilities. This framework provides a scalable and stable solution for continuous knowledge updates in large language models.

Limitations

Current work has shown success on standard knowledge editing datasets and general capability benchmarks. However, further exploration is needed for more diverse model ability tests, such as handling the ripple effects of knowledge editing and long-form questions. We plan to conduct experiments in future work to evaluate the effectiveness of our method across additional aspects of knowledge editing. Furthermore, for multi-hop question answering and event-level knowledge editing tasks, existing methods primarily rely on in-context-learning or chain-of-thought approaches, which are difficult to integrate with the parameter-updating-based approach we employ. Exploring solutions for these tasks within the parameter-updating paradigm will be a valuable direction for future research.

Additionally, our current model router is a training-free approach, which requires maintaining an embedding set of edited knowledge for computing routing scores. While other model-based routers do not require the maintenance of additional data, their accuracy often falls short of expectations. Therefore, exploring more effective model routing methods, or even leveraging the model itself for routing, represents a promising avenue for improving our work.

References

Oshin Agarwal and Ani Nenkova. 2022. Temporal effects on pre-trained models for language processing tasks. *Transactions of the Association for Computational Linguistics*, 10:904–921.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. *TAC*, 7(8):1.

Baolong Bi, Shenghua Liu, Yiwei Wang, Lingrui Mei, Hongcheng Gao, Junfeng Fang, and Xueqi Cheng. 2024. Struedit: Structured outputs enable the fast and accurate knowledge editing for large language models. *arXiv preprint arXiv:2409.10132*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models.

Qizhou Chen, Taolin Zhang, Xiaofeng He, Dongyang Li, Chengyu Wang, Longtao Huang, and Hui Xue'. 2024. Lifelong knowledge editing for LLMs with retrieval-augmented continuous prompt learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13565–13580, Miami, Florida, USA. Association for Computational Linguistics.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the*

Association for Computational Linguistics, 10:257–273.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim Kuan Goh, Ho-Kin Tang, Daojing He, and Min Zhang. 2024. Parameter competition balancing for model merging. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Xiang Wang, Xiangnan He, and Tat-seng Chua. 2024. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*.

Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. 2023. Bias and fairness in large language models: A survey. *Preprint*, arXiv:2309.00770.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. 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*. Association for Computational Linguistics.

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

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

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024a. WilKE: Wise-layer knowledge editor for lifelong knowledge editing. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3476–3503, Bangkok, Thailand. Association for Computational Linguistics.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024b. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*.

Xiusheng Huang, Jiaxiang Liu, Yequan Wang, and Kang Liu. Reasons and solutions for the decline in model performance after editing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xiusheng Huang, Yequan Wang, Jun Zhao, and Kang Liu. 2024. Commonsense knowledge editing based on free-text in LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14870–14880, Miami, Florida, USA. Association for Computational Linguistics.

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*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, Qun Liu, and Wei Wang. 2024. Learning to edit: Aligning LLMs with knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4689–4705, Bangkok, Thailand. Association for Computational Linguistics.

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. 2021. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363.

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

Qi Li, Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Xinglin Pan, and Xiaowen Chu. Should we really edit language models? on the evaluation of edited language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2023. Pmet: Precise model editing in a transformer. *arXiv preprint arXiv:2308.08742*.

Adam Liška, Tomáš Kočiský, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, Cyprien de Masson d'Autume, Tim Scholtes, Manzil Zaheer, Susannah Young, Ellen Gilsenan-McMahon Sophia Austin, Phil Blunsom, and Angeliki Lazaridou. 2022. Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models. *arXiv preprint arXiv:2205.11388*.

Jiateng Liu, Pengfei Yu, Yuji Zhang, Sha Li, Zixuan Zhang, Ruhi Sarikaya, Kevin Small, and Heng Ji. 2024. EVEDIT: Event-based knowledge editing for deterministic knowledge propagation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4907–4926, Miami, Florida, USA. Association for Computational Linguistics.

Yifan Lu, Yigeng Zhou, Jing Li, Yequan Wang, Xuebo Liu, Daojing He, Fangming Liu, and Min Zhang. 2024. Knowledge editing with dynamic knowledge graphs for multi-hop question answering. *arXiv preprint arXiv:2412.13782*.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. 2024. Perturbation-restrained sequential model editing. *CoRR*, abs/2405.16821.

Michael S Matena and Colin A Raffel. 2022. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716.

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

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. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *Preprint*, arXiv:2406.18665.

Yasumasa Onoe, Michael Zhang, Eunsol Choi, and Greg Durrett. 2022. Entity cloze by date: What LMs know about unseen entities. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 693–702, Seattle, United States. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Shanbao Qiao, Xuebing Liu, and Seung-Hoon Na. 2024. DistillMIKE: Editing distillation of massive in-context knowledge editing in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7639–7654, Bangkok, Thailand. Association for Computational Linguistics.

Domenic Rosati, Robie Gonzales, Jinkun Chen, Xuemin Yu, Yahya Kayani, Frank Rudzicz, and Hassan Sajjad. 2024. Long-form evaluation of model editing. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3749–3780, Mexico City, Mexico. Association for Computational Linguistics.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2023. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Dimitris Stripelis, Zhaozhuo Xu, Zijian Hu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Jipeng Zhang, Tong Zhang, Salman Avestimehr, and Chaoyang He. 2024. TensorOpera router: A multi-model router for efficient LLM inference. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 452–462, Miami, Florida, US. Association for Computational Linguistics.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language model via meta learning. In *The Twelfth International Conference on Learning Representations*.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages

353–355, Brussels, Belgium. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

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

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024a. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19449–19457.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024b. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *International Conference on Machine Learning*. PMLR.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. 2024. A comprehensive study of knowledge editing for large language models. *Preprint*, arXiv:2401.01286.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

A Details of Metrics

The definitions of the three main metrics in **CounterFact** are as follows:

- **Efficacy** measures the accuracy of the editing process, specifically reflecting the successful modification of the factual knowledge statement in the dataset.
- **Generalization** evaluates whether the edit can be effectively applied to paraphrased or contextually related sentences within the dataset.
- **Locality** refers to the preservation of original knowledge that unrelated to editing requests, ensuring it remains intact. This is evaluated using irrelevant natural questions or neighborhood questions in the dataset.

Formally, the metrics for **Efficacy** and **Generalization** are defined as follows:

$$\mathbb{E}[\mathcal{P}_{\tilde{\mathcal{M}}}(\tilde{o}|(s,r)) > \mathcal{P}_{\tilde{\mathcal{M}}}(o|(s,r))] \quad (11)$$

and the metric on **Locality** is defined as follows:

$$\mathbb{E}[\mathcal{P}_{\tilde{\mathcal{M}}}(\tilde{o}|(s,r)) < \mathcal{P}_{\tilde{\mathcal{M}}}(o|(s,r))] \quad (12)$$

where $\tilde{o}$ denote the subject corresponding to the new knowledge, and $\tilde{M}$ denote the post-edited model.

The detailed description of the metrics in **KnowEdit** is as follows:

- **Edit Success** is similar to the combination of Efficacy and Generalization in the CounterFact dataset. The edited model should not only provide correct answers to the original questions but also accurately respond to inputs with similar expressions.
- **Portability** evaluates whether the edited model can infer downstream knowledge related to the edited facts, comprising three aspects: **Alias**: Tests whether the same knowledge remains valid when presented with different subject aliases. **Reasoning**: Requires the edited model to perform reasoning based on the edited facts to infer related knowledge. **Logical Generalization**: Knowledge semantically related to the edited facts should also be modified, such as in cases of inverse relationship reasoning.
- **Locality**, primarily referring to Relation Specificity in the ZsRE subset, asserts that any other attributes of the previously updated subject should remain unchanged after the editing process.

B Data Sampling

Here, we provide a detailed description of how the data is sampled for constructing the computational corpus used in the model router.

For the CounterFact dataset, which contains 20,877 samples, we randomly selected 2,000 samples for knowledge editing and testing. The query prompts corresponding to these samples form the set $\mathcal{E} = \{e_i\}_{i=1}^{n}$, $(n = 2000)$. For the negative samples, we randomly selected 2,000 samples from the remaining 18,877, which are unrelated to the editing samples, using their query prompts to form $\mathcal{U} = \{u_i\}_{i=1}^{m}$, where $m = 2000$, thus constructing the computational corpus.

For the KnowEdit (ZsRE) dataset, which contains 1,301 samples, we randomly selected 1,000 samples for knowledge editing and testing. The corresponding query prompts for these samples form the positive set $\mathcal{E} = \{e_i\}_{i=1}^{n}$, $(n = 1000)$. For the negative set, we used the locality prompts from the remaining 301 samples, where each sample has two different locality prompts unrelated to the editing samples, thus forming $\mathcal{U} = \{u_i\}_{i=1}^{m}$, where $m = 602$, to construct the computational corpus.

C Details of Datasets

In the CounterFact dataset, each data example consists of a factual knowledge statement, 2 paraphrased sentences, and 10 neighborhood questions, an example as follows:

CounterFact Example

{
  "case_id": 2099,
  "requested_rewrite": {
    "prompt": "{}, produced by",
    "target_new": “str": "Toyota",,
    "target_true": "str": "Cadillac",,
    "subject": "Cadillac Fleetwood"
  },
  "paraphrase_prompts": [
    "Cadillac Fleetwood is produced by",
    "Cadillac Fleetwood is a product of"
  ],
  "neighborhood_prompts": [
    "Cadillac STS Wheels, created by",
    "Cadillac ATS is produced by",
    "Cadillac Type 51, developed by",
    "Cadillac Series 62 is produced by",
    "Cadillac Brougham, produced by",
    "M41 Walker Bulldog is created by",
    "Cadillac XLR is a product of",
    "Cadillac Series 62, developed by",
    "Cadillac STS Wheels is created by",
    "Cadillac ATS, created by"
  ]
}

In the ZsRE subset of KnowEdit (Zhang et al., 2024) dataset, each data example includes a new knowledge statement along with various test questions designed to assess different model capabilities, with each sample containing 5 test point, as example as follows:

KnowEdit Example

"subject": "GNOME Chess",
"target_new": "Python",
"prompt": "What programming language was used to write GNOME Chess?",
"ground_truth": "Vala",
"rephrase_prompt": "How is the programming language for GNOME Chess?",
"locality": {
  "Relation_Specificity": [
  { "prompt":
    "The platform of GNOME Chess is",
    "ground_truth":
    "Unix-like operating system", }
  { "prompt":
    "GNOME Chess platform",
    "ground_truth":
    "Unix-like operating system", }
  ],
},
"portability": {
  "Reasoning": [
    { "prompt":
    "Who created the programming language used to write GNOME Chess?",
    "ground_truth":
    "Guido van Rossum", },
  ],
}

D Ablation Study on Modules: More Details

Epoch	Ablation of Modules			
	AlphaEdit	+MM	+R	+MM+R
@1	84.58	85.37 (+0.87)	86.58 (+2.00)	87.11 (+2.53)
@2	89.62	90.23 (+0.61)	91.60 (+1.98)	92.02 (+2.40)
@3	86.97	87.78 (+0.81)	90.34 (+3.37)	90.75 (+3.78)
@4	88.45	89.42 (+0.97)	91.24 (+2.79)	91.81 (+3.36)
@5	90.01	91.12 (+1.11)	92.63 (+2.62)	93.28 (+3.27)
@6	87.99	89.15 (+1.16)	90.71 (+2.72)	91.44 (+3.45)
@7	86.85	87.98 (+1.13)	89.66 (+2.81)	90.22 (+3.37)
@8	88.18	89.28 (+1.10)	91.67 (+3.49)	91.97 (+3.79)
@9	88.54	89.64 (+1.10)	91.07 (+2.53)	91.57 (+3.03)
@10	88.39	87.17 (−1.22)	91.08 (+2.69)	88.91 (+0.52)
Overall	88.01	88.77 (+0.76)	90.70 (+2.69)	90.95 (+2.94)

Table 5: Editing **Scores** under the ablation study. **+MM** and **+R** indicate the use of model merging and model routing methods, respectively. **@k** represents the results at the k-th sequential editing batch. The values in parentheses indicate the difference compared to the baseline model.

Epoch	Ablation of Modules			
	MEMIT	+MM	+R	+MM+R
@1	67.96	73.06 (+5.10)	70.69 (+2.37)	76.05 (+8.09)
@2	72.92	78.44 (+5.52)	76.10 (+3.18)	81.15 (+8.23)
@3	75.31	80.00 (+4.69)	79.30 (+3.99)	83.72 (+8.41)
@4	76.40	80.91 (+4.51)	80.46 (+4.06)	84.49 (+8.09)
@5	80.13	84.25 (+4.12)	84.33 (+4.20)	87.65 (+7.52)
@6	80.77	83.41 (+2.64)	85.26 (+4.49)	87.99 (+7.22)
@7	80.72	83.01 (+2.29)	85.18 (+4.46)	87.14 (+6.42)
@8	81.98	83.85 (+1.87)	88.10 (+6.12)	88.68 (+6.70)
@9	83.09	84.26 (+1.17)	87.96 (+4.87)	88.24 (+5.15)
@10	81.09	81.39 (+0.30)	86.64 (+5.55)	84.96 (+3.87)
Overall	78.37	81.42 (+3.05)	82.69 (+4.32)	85.15 (+6.78)

Table 6: Editing **Scores** under the ablation study. **+MM** and **+R** indicate the use of model merging and model routing methods, respectively. **@k** represents the results at the k-th sequential editing batch. The values in parentheses indicate the difference compared to the baseline model.