Resolving UnderEdit & OverEdit with Iterative & Neighbor-Assisted Model Editing

Anonymous ACL submission

Abstract

Large Language Models (LLMs) are used in various downstream language tasks, making it crucial to keep their knowledge up-to-date, but both retraining and fine-tuning the model can be costly. Model editing offers an efficient and effective alternative by a single update to only a key subset of model parameters. While being efficient, these methods are not perfect. Sometimes knowledge edits are unsuccessful, i.e., UnderEdit, or the edit contaminated neighboring knowledge that should remain unchanged, i.e., OverEdit. To address these limitations, we propose **iterative model editing**, based on our hypothesis that a single parameter update is often insufficient, to mitigate UnderEdit, and **neighbor-assisted model editing**, which incorporates neighboring knowledge during editing to minimize OverEdit. Extensive experiments demonstrate that our methods effectively reduce UnderEdit up to 38 percentage points and OverEdit up to 6 percentage points across multiple model editing algorithms, LLMs, and benchmark datasets.

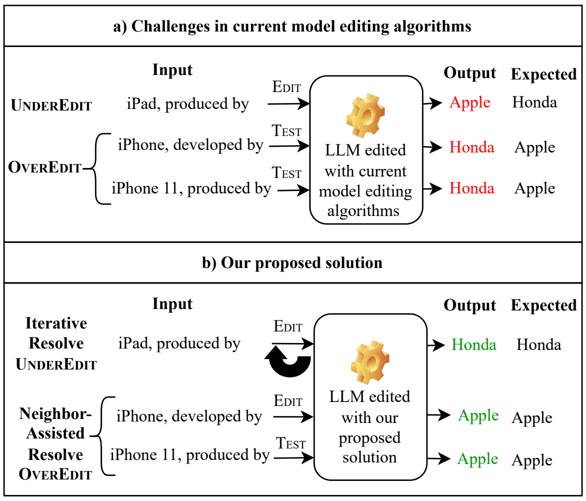

Figure 1: The example from CounterFact updates *iPad* producer from *Apple* to *Honda*. UnderEdit fails to make the desired update in the Edit sentence, while OverEdit introduces the undesired change in the Test sentences as shown in (a). The proposed iterative model editing mitigated UnderEdit and neighbor-assisted model editing reduced OverEdit by incorporating related knowledge in Edit stage as shown in (b).

1 Introduction

LLMs have been widely used as repositories of factual and specialized knowledge (Petroni et al., 2020; Jiang et al., 2021; Roberts et al., 2020; Youssef et al., 2023). However, the world is constantly changing, with knowledge and information evolving rapidly, such as significant government policy changes and their wide impacts across various domains. Thus, it is essential for many NLP applications, such as text generation, question answering, and knowledge retrieval, to have models that can adapt to knowledge changes both effectively and efficiently. Re-training an LLM is resource-intensive (Patterson et al., 2021). Standard supervised fine-tuning is data hungry and less effective (Meng et al., 2023b). Model-editing, which directly modifies important model parameters for making the prediction, has emerged as a more efficient alternative for updating outdated information (Meng et al., 2023a,b; Li et al., 2024). These methods adopt a "locate and edit" approach, wherein they first identify the parameter locations associated with outdated knowledge and then update the parameters to enable the model to incorporate and predict the new knowledge.

The effectiveness of the methods is evaluated from two perspectives. The first is whether the method successfully updates the knowledge, failure on this leaves certain facts unedited, causing UnderEdit. Secondly, whether the update introduces unintended modifications to neighboring knowledge — a phenomenon we call OverEdit. Existing methods suffer from both UnderEdit and OverEdit as shown in Figure 1.

To address this, we propose methods to mitigate both UnderEdit and OverEdit. For UnderEdit, we hypothesize that the parameter update is insufficient to achieve the desired knowledge change. The editing process performed a rank-one update on the layer parameters to achieve the desired update. We empirically showed that the approximation introduces errors, leading to UnderEdit. To this end, we proposed iterative model editing, wherein editing is performed multiple times. For OverEdit, we hypothesize that model editing can benefit from including neighboring knowledge during the editing stage. We thus introduce neighbor-assisted model editing, a procedure that integrates neighboring knowledge during the editing process to keep the test neighboring knowledge unchanged.

In summary, we propose solutions to two fundamental challenges in model editing: UnderEdit, where edits fail, and OverEdit, where neighboring knowledge is erroneously modified. We evaluate our approach using three "locate and edit" model editing algorithms, ROME (Meng et al., 2023a), MEMIT (Meng et al., 2023b), and PMET (Li et al., 2024), and applied to three LLMs: GPT-2 XL (1.5B) (Radford et al., 2019), GPT-J (6B) (Wang and Komatsuzaki, 2021), and Llama 2 (7B) (Touvron et al., 2023). Our experiments are conducted on two widely used factual knowledge editing benchmarks: COUNTERFACT (Meng et al., 2023a) and ZsRE (Levy et al., 2017). Our results show that iterative model editing improves edit success while also reducing the approximation error introduced by the rank-one update. Furthermore, we demonstrate that incorporating neighboring knowledge during model editing leads to fewer unintended modifications to neighboring knowledge at test time, resulting in stronger edit performance.

2 Background

In this section, we provide background on the locate and edit model editing framework along with the notation used throughout the paper.

An autoregressive LLM is a function $f_\theta\colon \mathcal{X}^T \rightarrow \Delta(\mathcal{X})$, that takes as input a sequence of tokens $x = (x_1, x_2, \cdots, x_T)$ of length T with x_i in the dictionary $\mathcal{X}$, and uses model parameters θ to return a probability distribution $\Delta(\mathcal{X})$ to model the next token x', i.e,. $f_\theta(x)[x'] \approx \Pr(X' = x'|X = x)$, where X and X' are random variables representing the sequence of input tokens and the next token, respectively. The internal computations of an LLM relies on a grid of hidden states h_t^l, where l corresponds to the layer and t corresponds to the token position in the sequence (using tokens $x_1, x_2, \cdots, x_t$). Each layer is a standard transformer block with the self-attention module, MLP module, etc (Vaswani et al., 2017).

Prior work focuses on editing the factual knowledge within the LLM. Factual knowledge is represented as a triplet (s, r, o), where subject $s \in \mathcal{X}^{T_s}$, relation $r \in \mathcal{X}^{T_r}$, and object $o \in \mathcal{X}$ are sequences of tokens, e.g., The *iPad* $[s]$ is *produced by* $[r]$ *Apple* $[o]$. We consider only single token objects in this representation, following previous work (Meng et al., 2023a,b; Li et al., 2024). The model editing task is to make the model place a higher likelihood on a new object o^* than an old object o when presented with $x = (s, r)$, i.e., find new parameters θ', such that $f_{\theta'}(x)[o^*] > f_{\theta'}(x)[o]$. Model editing is not limited to a single edit, but can encompass a batch of m desired edits $D = \{(s_i, r_i, o_i, o_i^*)\}_{i=1}^m$.

Locate and edit model editing algorithms hypothesize that factual knowledge locates within specific layers of the LLM, and updating parameters in these layers is sufficient to induce the desired change in object (Pearl, 2013; Vig et al., 2020; Meng et al., 2023a) . These methods employ causal tracing to identify these layers responsible for the factual knowledge, referred to as causal layers $\{l_1, \ldots, l_c\}$, where c denotes the number of layers in this range. The last MLP in these layers has been found to have a major impact on the object token distribution when presented with the subject tokens (Meng et al., 2023a,b; Geva et al., 2021). Due to this major impact, locate and edit methods focus on only updating these MLP weights.

The weight update is performed in two stages: OPTIMIZATION stage finds ideal values for the network's hidden state in certain transformer layer to make o^* likely and SPREAD stage updates the weights of the last MLP in casual layer(s) to approximate this ideal hidden state. We detail these stages below for MEMIT (Meng et al., 2023b) and discuss its differences to PMET (Li et al., 2024) and ROME (Meng et al., 2023a).

OPTIMIZATION Stage: Learning the Ideal State. The goal of the OPTIMIZATION stage is to find what outputs in the causal layers would lead to a high likelihood on o^*. The methods we investigate search for ideal outputs in different locations. MEMIT searches for an ideal output, $\bar{h}_t^{l_c}$, for the last casual layer at t the last token index of the subject s. The search is performed by finding a vector δ to add to current hidden state value $h_t^{l_c}$.

Algo	OPTIMIZATION	SPREAD
MEMIT	h^{l_c}	$W^{l_1}\dots W^{l_c}$
PMET	$Attn^{l_c}$ & z^{l_c}	$W^{l_1}\dots W^{l_c}$
ROME	z^{l_*}	W^{l_*}

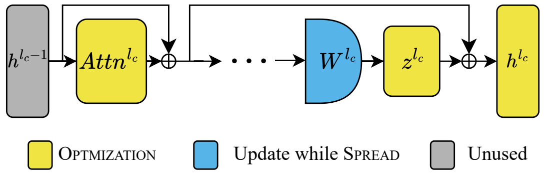

Figure 2: The table compares three memory editing algorithms, each of which optimizes different parameters within a single transformer layer and propagates changes across causal layer(s). Despite their differences, all algorithms follow the same OPTIMIZATION and SPREAD stages. The figure below shows a simplified transformer layer, composed of an attention module and MLP modules. Only the last MLP module is shown, as all algorithms modify parameters within it.

We represent the output probability distribution of the model using $h_t^{l_c}$ and δ as $f_\theta(x, h_t^{l_c}+\delta)$.

To make the hidden state δ change robust to diverse contexts, these methods add a random prefix to the prompt, i.e., the network takes as input $x_i = (\xi_i, s, r)$, where ξ_i is one of n random prefixes. The loss function for δ is to minimize the average negative log likelihood of o^*, i.e.,

$$g(\delta) \doteq -\frac{1}{n}\sum_{i=1}^{n} \ln f_\theta\left(x_i, h_t^{l_c}+\delta\right)[o^*] + D_{\mathrm{KL}}\left(f_\theta\left(s, h_t^{l_c}+\delta\right)||f_\theta\left(s\right)\right)$$

where D_{KL} is the Kullback–Leibler divergence, which is added to constrain the model's output to be close to the original.

The ideal hidden state for the prompt $x = (s, r)$ is $\bar{h}_t^{l_c} = h_t^{l_c} + \delta^*$, where δ^* is found by performing gradient descent on g. This ideal hidden state is then used in computing weight update in the next stage. PMET differs from MEMIT by searching for an ideal outputs for the attention module and MLP modules in layer l_c. ROME searches for an ideal output for the MLP module of a single layer in the set of causal layers.

SPREAD Stage: Propagating the Change. The goal of the SPREAD stage is to find new weights θ' such that the hidden state after the update $\hat{h}_t^{l_c}$is close to the ideal hidden state $\bar{h}_t^{l_c}$ for all desired edits in D. Not all weights in the network are updated, only the weights W^l corresponding to the weights of the last MLP layer in causal layers are updated. The weight update methods are derived from an rank-one approximation to make $\hat{h}_t^{l_c} \approx \bar{h}_t^{l_c}$, which can lead to failed . The different algorithms also update a different set of weights; MEMIT and PMET update W^l for all causal layers, while ROME only updates one W^l. The algorithm differences are summarized in Figure 2.

3 Method

The memory-editing algorithms mentioned above face challenges, such as failing to edit certain knowledge i.e., UnderEdit or changing neighbor knowledge that should remain unchanged i.e., OverEdit. In this section, we present our proposed method to address these issues. Specifically, we introduce iterative model editing (3.1) to mitigate UnderEdit and neighbor-assisted model editing (3.2) to reduce OverEdit.

3.1 Iterative Model Editing

There are two possible reasons for UnderEdit to occur. The first is that $\bar{h}_t^{l_c}$ does not reflect a hidden state for a successful edit. The second is that the weight update results in $\hat{h}_t^{l_c} \not\approx \bar{h}_t^{l_c}$. We hypothesize that both of these potential problems can be addressed by running the memory edit process multiple times because: 1) it allows for potentially finding better $\bar{h}_t^{l_c}$ after updating the model parameters so that $\|\hat{h}_t^{l_c} - \bar{h}_t^{l_c}\| \leq \|h_t^{l_c} - \bar{h}_t^{l_c}\|$, and 2) on the next iteration, the approximation used in the SPREAD stage for the weight update will be better since $\hat{h}_{l_c}^{t}$ is closer to $\bar{h}_t^{l_c}$ than $h_t^{l_c}$. We detail this iterative process below for MEMIT, but it can also be adapted to ROME and PMET by replacing $\bar{h}_t^{l_c}$ with the targets of the optimization procedure for those algorithms.

Iterative model editing works as follows. At iteration k, OPTIMIZATION computes the ideal hidden state $\bar{h}_{t,k}^{l_c}$ based on the hidden state produced with the model parameters θ_k, i.e., $\bar{h}_{t,k}^{l_c} = h_{t,k}^{l_c} + \delta_k^*$, where δ_k^* is obtained by optimizing $g(\delta)$ using θ_k as the model parameters. SPREAD stage updates the model parameters to θ_{k+1} based on the computed $\bar{h}_{t,k}^{l_c}$, producing a new hidden state $\hat{h}_{t,k}^{l_c}$. Note that $\hat{h}_{t,k}^{l} = h_{t,k+1}^{l}$.

The iterations end when the model perplexity using $\hat{h}_{t,k}^{l_c}$ is within ϵ of the perplexity using $\bar{h}_{t,k}^{l_c}$,

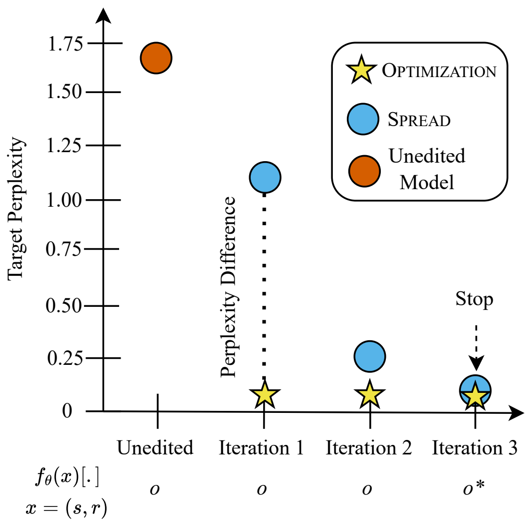

Figure 3: An editing example of using MEMIT to edit GPT-J. Iterative model editing resolving UnderEdit. As the iteration proceeds the perplexity differences eventually reduces to $\leq \epsilon$, leading to the model predicting new object. The perplexity values are Box-Cox transformed to better visualize extreme high and low values.

i.e.,

$$|p(\theta_{k+1}, \hat{h}^{l_c}_{t,k}) - p(\theta_k, \bar{h}^{l_c}_{t,k})| \leq \epsilon,$$

where p is the perplexity of the target token over the m edits in D

$$p(\theta, h) \doteq \frac{1}{m} \sum_{i=1}^{n} e^{-\ln f_\theta(x_i, h)[o_i^*]}.$$

For brevity, we use Δp_k to denote the above difference in perplexity in iteration k. Empirically, we found $\epsilon = 1$ to be a sufficient threshold for the data sets used in this paper.

Figure 3 illustrates how iterative model editing allows $\hat{h}^{l_c}_{t,k}$ to have a perplexity closer to $\bar{h}^{l_c}_{t,k}$ over time. The figure also makes it clear that most of the improvement of iterative model editing comes from applying SPREAD multiple times since the perplexity of of $\bar{h}^{l_c}_{t,k}$ does not change as much.

3.2 Neighbor-Assisted Model Editing

Model editing not only needs to modify the model's output from o to o^* given (s, r), but also preserve the model's output for neighboring knowledge, i.e $(\tilde{s}, r, o)$, where $\tilde{s}$ is a new subject sharing the same relation r. An example of this preservation is shown in Figure 1a: iPhone 11 $[\tilde{s}]$ is still produced by $[r]$ Apple $[o]$ despite iPad is edited to produced by Honda[1]. Existing model editing algorithms struggle to preserve neighboring knowledge because OPTIMIZATION is designed solely to maximize the likelihood of the new knowledge, (s, r, o^*). Moreover, iterative model editing can exacerbate this OverEdit issue, as each iteration continues to reinforce the new knowledge without explicitly preserving neighboring knowledge.

Gangadhar and Stratos (2024) argue that incorporating neighboring knowledge while learning new facts through fine-tuning is more effective at preserving such neighbors compared to conventional model editing. Inspired by this observation, we hypothesize that incorporating neighboring knowledge into the OPTIMIZATION stage can help to mitigate OverEdit.

We propose neighbor-assisted model editing, which optimizes $\bar{h}^{l_c}_t$ to maximize the likelihood of the new knowledge (s, r, o^*) and the neighboring knowledge $(\tilde{s}, r, o)$. To accomplish this we define the loss function for δ as:

$$\begin{aligned}\tilde{g}(\delta) \doteq\ & -\frac{1}{n}\sum_{i=1}^{n} \ln f_\theta\left(x_i, h^{l_c}_t + \delta\right)[o^*] \\ & + D_{\mathrm{KL}}\left(f_\theta\left(s, h^{l_c}_t + \delta\right) || f_\theta(s)\right) \\ & - \ln f_\theta\left(\tilde{x}, h^{l_c}_t + \delta\right)[o],\end{aligned}$$

where $\tilde{x} = (\tilde{s}, r)$ without any prefix. In our experiments we only included a single neighboring knowledge fact $(\tilde{s}, r, o)$, but it should be extensible to multiple neighboring knowledge facts. We omit the iteration notation k here for simplicity. The proposed loss function change could be easily applied in different iterations.

4 Experimental Details

In this section, we detail the experiments to demonstrate the effectiveness of iterative and neighboor assisted model editing with MEMIT, PMET, and ROME. We evaluate these algorithms with our modifications across three LLMs: GPT-2 XL (1.5B), GPT-J (6B), and Llama-2 (7B). We use the EasyEdit[2] framework (Wang et al., 2024b) with their default hyperparameters. Additional implementation details are provided in Appendix B.

4.1 Datasets

To evaluate model editing across different datasets, we use the COUNTERFACT (Meng et al., 2023a) and

[1] In the COUNTERFACT dataset

[2] https://github.com/zjunlp/EasyEdit

Model	Algo	k	Efficacy (↑)		Generalization (↑)		Specificity (↑)		Score (↑)		Perplexity (↓)	$\|\Delta p_k\|$(↓)
			Accuracy	Success	Accuracy	Success	Accuracy	Success	Accuracy	Success	ME-PPL-50	
GPT-2 XL (1.5B)	**Unedited**	0	0.00%	21.67%	0.00%	31.33%		56.67%		31.33%	54.66	
	ROME	1	1.00%	56.00%	1.00%	55.00%	0.00%	92.00%	1.00%	64.00%	7.07E+03	7.15E+05
	MEMIT	1	79.00%	92.67%	22.00%	65.67%	**75.00%**	**99.00%**	42.67%	83.00%	58.66	11359.60
		4	**99.67%**	**99.67%**	**35.67%**	**76.00%**	68.67%	**99.00%**	**56.67%**	**90.00%**	62.44	0.47
	PMET	1	21.67%	57.00%	3.00%	42.00%	**91.67%**	**100.00%**	8.33%	58.33%	55.60	103785.71
		10	**99.33%**	**99.67%**	**24.67%**	**70.33%**	75.33%	99.00%	**47.00%**	**87.33%**	63.92	0.42
GPT-J (6B)	**Unedited**	0	9.33%	38.00%	9.00%	38.33%		82.00%		37.33%	39.80	
	ROME	1	1.00%	57.00%	1.00%	55.00%	0.00%	76.00%	1.00%	61.00%	2.15E+05	1.21E+14
	MEMIT	1	99.00%	100.00%	75.00%	95.67%	**69.33%**	**89.33%**	77.67%	94.33%	42.20	1.22
		2	**99.33%**	**100.00%**	**80.67%**	**98.00%**	66.33%	88.33%	**79.00%**	**95.00%**	43.92	0.03
	PMET	1	98.00%	**99.67%**	76.00%	95.00%	**68.33%**	**88.67%**	77.67%	93.67%	41.27	1.15
		3	**99.00%**	**99.67%**	**76.67%**	**95.67%**	**68.33%**	**88.67%**	**78.33%**	**94.00%**	41.15	0.05
Llama-2 (7B)	**Unedited**	0	15.00%	13.67%	15.00%	15.00%		84.33%		19.67%	30.63	
	ROME	1	0.00%	48.00%	0.00%	49.00%	0.00%	76.00%	0.00%	55.00%	1.45E+04	8.10E+05
	MEMIT	1	91.67%	98.00%	70.33%	93.33%	29.33%	67.33%	50.67%	83.67%	42.10	198.79
		2	14.33%	79.00%	9.67%	73.67%	6.67%	70.67%	9.00%	74.67%	9.37E+03	4664.24
	PMET	1	94.33%	97.00%	68.33%	86.67%	**76.33%**	**89.00%**	77.33%	90.33%	30.73	3.32
		2	**95.33%**	**98.33%**	**70.00%**	**88.67%**	75.33%	88.67%	**78.00%**	**91.67%**	30.76	0.09

Table 1: Iterative model editing results on COUNTERFACT for at most 10 iterations (denoted by k). We compare the evaluation metrics of iteration that met stopping criterion $|\Delta p_k| \leq 1$ to that of their corresponding first iteration and **bold** the higher value. PMET on GPT-2 XL require more than 5 iterations to achieve our stopping criteria. Results for all iterations are provided in Table 6 (Appendix D). While ROME is known to collapse (results reported in Table 6), we observed a unique case of collapse with Llama-2 (7B) specifically when using MEMIT. We discuss this in Section 5.

ZsRE (Levy et al., 2017) datasets. Both datasets consist of approximately 20k factual knowledge instances. Due to hardware limitations, for each model-editing experiment, we ran it on a subset of m = 1,000 edits for each dataset. We repeat the editing task three times each using a different set of m edits sampled from the whole dataset. We ensured that the edits in these trials were mutually exclusive and report the averages across them.

It is not uncommon for a model to "collapse" (fail on a downstream task) after editing. To evaluate model collapse we use the ME-PPL-50 dataset (Yang et al., 2024). ME-PPL-50 comprises 50 utterances, each averaging 22 tokens, sampled from LLMs' pre-training corpora. Yang et al. (2024) demonstrated that high perplexity on this dataset correlates with failures in various downstream tasks, making it an efficient proxy for evaluating model collapse. They also observed that this behavior remains consistent regardless of dataset size. Thus, we use this smaller set. We analyze the impact of our proposed methods on model collapse in Section 5.

4.2 Evaluation Metrics

We are primarily concerned with evaluating how well iterative and neighbor assisted model editing reduce the frequency of UnderEdit and OverEdit. We measure how successful the editing algorithms were by examining *efficacy* and *generalization* scores. Efficacy measures the success of introducing new knowledge edits in the dataset. Generalization tests whether the edit is robust and not overfit by evaluating the model on paraphrases of the examples in the dataset. To undertand how well the algorithms were at avoiding OverEdits, we measure the *specificity* of the model, i.e., how much of the neighboor knowledge remained unchanged. To summarize the overall performance in a *score*, we use a harmonic mean of efficacy, generalization, and specificity.

For each of these metrics, we report two evaluation scores: *success* and *accuracy*. Success is the percentage of edits where $f_{\bar{\theta}}(x_i)[o_i^*] > f_{\bar{\theta}}(x_i)[o_i]$ (or $f_{\bar{\theta}}(\tilde{x}_i)[o_i] > f_{\bar{\theta}}(\tilde{x}_i)[o_i^*]$ for specificity) with $\bar{\theta}$ being the final weights after editing. Accuracy is the percentage of edits where o^* (or o in the case of specificity) is the most likely next token.

5 Results and Discussions

We show the experimental results for both iterative model editing and neighbor-assisted model editing in this section. The hardware used on running these experiments are detailed in Appendix C.

Model	Algo	k	Efficacy (↑)		Generalization (↑)		Specificity (↑)		Score (↑)		Perplexity (↓)	$\lvert\Delta p_k\rvert$(↓)
			Accuracy	Success	Accuracy	Success	Accuracy	Success	Accuracy	Success	ME-PPL-50	
GPT-2 XL (1.5B) #739	Unedited	0	1.00%	9.00%	1.00%	22.00%		100.00%		18.00%	54.66	
	MEMIT	4	**99.00%**	**99.00%**	**38.00%**	**74.00%**	52.00%	77.00%	54.00%	82.00%	62.27	0.05
	NA_MEMIT	4	**99.00%**	**99.00%**	36.00%	70.00%	**86.00%**	**95.00%**	**60.00%**	**86.00%**	64.89	0.19
	PMET	8	**99.00%**	**99.00%**	**31.00%**	**68.00%**	67.00%	86.00%	52.00%	82.00%	63.47	0.31
	NA_PMET	9	98.00%	98.00%	29.00%	64.00%	**85.00%**	**97.00%**	**53.00%**	**83.00%**	66.29	0.29
GPT-J (6B) #960	Unedited	0	0.00%	8.00%	1.00%	10.00%		100.00%		12.00%	39.80	
	MEMIT	2	**99.00%**	**100.00%**	**79.00%**	**97.00%**	63.00%	83.00%	78.00%	93.00%	44.84	0.64
	NA_MEMIT	2	**99.00%**	99.00%	75.00%	92.00%	**81.00%**	**95.00%**	**84.00%**	**95.00%**	45.24	0.33
	PMET	1	**99.00%**	**100.00%**	**72.00%**	**93.00%**	65.00%	84.00%	76.00%	92.00%	40.79	0.25
	NA_PMET	6	98.00%	99.00%	69.00%	89.00%	**80.00%**	**94.00%**	**81.00%**	**94.00%**	43.54	0.44
Llama-2 (7B) #1340	Unedited	0	38.33%	57.00%	37.00%	56.00%		59.67%		55.67%	33.69	
	PMET	3	96.00%	98.00%	72.00%	89.00%	70.00%	92.00%	78.00%	93.00%	31	0.44
	NA_PMET	10	62.00%	79.00%	47.00%	75.00%	26.00%	76.00%	40.00%	76.00%	1414.01	

Table 2: Neighbor-Assisted model editing results on CounterFact. We present iteration where our proposed stopping criteria is achieved for both neighbor-assisted (NA_) and without neighbor runs of the model editing algorithms. We compare their evaluation metrics and **bold** the higher value. Results among models and from Table 1 are not comparable due to difference in neighboring samples as explained in Appendix. B.2. Hence, we report the no. of examples (#) used to run experiment for each model. NA_PMET on Llama-2 (7B) stands as an exception that didn't achieved the stopping criteria within 10 iteration and showed a performance decrease. Results for all iterations are provided in Table 8.

5.1 Iterative Model Editing Results

We conducted iterative model editing experiments across all datasets, LLMs, and editing algorithms, running each configuration for at most 10 iterations. The evaluation results are presented in Table 1 for CounterFact and Table 7 for ZsRE (provided in Appendix 7 due to space constraints). From these experiments results, we drew several conclusions.

First, iterative model editing consistently improves performance, with the overall success scores increasing across iterations for most models and algorithms. The overall success improvement stems from enhanced efficacy and generalization capabilities, which means fewer cases of UnderEdit. Specifically, we observed an increase in success accuracy of up to 38 percentage points, with a greater improvement in efficacy accuracy of up to 77 percentage points (PMET on GPT-2 XL). We conducted more analysis in section 6.1 to showcase the efficacy improvement is mostly coming from UnderEdit examples. Secondly, as iteration goes, the perplexity difference constantly goes down in most cases. Finally, the proposed stopping criterion ($|\Delta p_k| \leq \epsilon$) consistently halts the process, validating its reliability. Using this criterion also yields better overall scores compared to executing the algorithm only once.

While efficacy and generalization improve significantly with iterative model editing, specificity decreased in some experiments, indicating an increase in OverEdit. We argue that this occurs because maximizing the likelihood of new knowledge through updates to causal layer weight parameters inadvertently affects neighboring knowledge due to shared weights. However, the overall performance increase outweighs the drop in specificity.

Although iterative editing is effective in most cases, we also observed model collapse as editing progresses, indicated by high model perplexity on ME-PPL-50. This collapse behavior aligns with the continuous editing failures observed in previous work (Gupta et al., 2024; Meng et al., 2023a). This suggests that when the combination of the model and editing algorithm succeeds in continuous editing, an essential experimental setting, iterative editing can further improve model performance. Specifically, ROME collapse under iterative [3] editing, which is consistent with prior work's finding on its inability to do continuous editing (Gupta et al., 2024). We also found that Llama-2 (7B) collapse only when edited with MEMIT, similar finding fromYang et al. (2024) confirmed this. We hypothesize two contributing factors: (1) model-specific characteristics, as no GPT models exhibited similar collapse behavior, suggesting that certain training-related strategy may be more vulnerable to instability, and (2) MEMIT's less precise weight updates compared to PMET, which

[3] ROME does not support batch editing, as it can only modify one fact at a time (Meng et al., 2023b). We discuss this further in Appendix A.

may have introduced excessive parameter changes (Appendix A).

5.2 Neighbor-Assisted Model Editing Results

To evaluate neighbor-assisted model editing method, we only conducted experiments on COUNTERFACT due to data limitations of ZsRE explained in Appendix B. We perform the iterative model editing with the modified neighbor loss function. We excluded the collapsed experimental settings. The evaluation results are shown in Table 2.

We observed consistently higher specificity across all iterations when using neighbor-assisted editing denoted by NA_ compared to setups without it. This shows the effectiveness of enforcing neighboring knowledge unchanged during the OPTIMIZATION stage. With increase in specificity, we observed an increase in overall score as well. Specifically, we observed an increase in success accuracy of up to 6 percentage points, with a greater improvement in specificity accuracy of up to 34 percentage points (NA_MEMIT on GPT-2 XL). Although there is a decrease in generalization, the improvement in efficacy is more significant, resulting in an overall increase in the score.

Moreover, the proposed stopping criteria $|\Delta p_k| \leq \epsilon$ defined for iterative model editing remains effective for neighbor-assisted model editing. We did observed an exception when Llama-2 (7B) neighbor-assisted edited with PMET showed an performance decrease instead of increase across all metrics. We attribute this degradation to increase in tendency to collapse indicated by high model perplexity measured by ME-PPL-50. Notably, Llama-2 (7B) was again the only model to exhibit this collapse behavior, reinforcing our earlier hypothesis that model-specific factors contribute to instability. However, pinpointing the training-related factors responsible for this behavior requires deeper analysis, which we leave for future work.

In conclusion, while both our proposed methods—iterative and neighbor-assisted model editing—do not inherently prevent collapse, they remain susceptible when other contributing factors are present.

6 Analysis

This section dive deeper to the proposed methods. Specifically, we aim to understand how iterative model editing address UnderEdit, the impact of the number of neighbors added in neighbor-assisted model editing for resolving OverEdit, and the effectiveness of the stopping criteria.

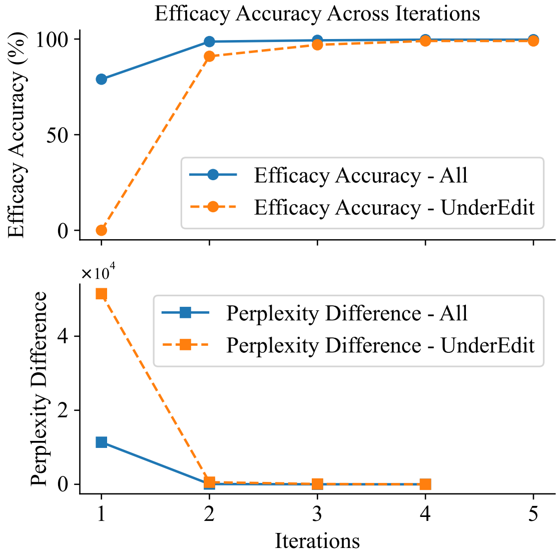

Figure 4: Improvement in efficacy accuracy and reduction in $|\Delta p_k|$ for UnderEdit examples over iterative model editing. The results show that iterative editing mitigates UnderEdit cases in GPT-2 XL edited with MEMIT on COUNTERFACT, contributing to overall performance gains.

6.1 How does iterative model editing address UnderEdit?

We hypothesize that the iterative model editing approach can reduce the number of UnderEdit cases. To test this hypothesis, we identified UnderEdit examples in GPT-2 XL edited with MEMIT on COUNTERFACT after the first iteration, i.e, edits (s_i, r_i, o_i, o_i^*) where $f_{\theta_2}(x_i)[o_i^*] < f_{\theta_2}(x_i)[o_i]$. We then tracked $|\Delta p_k|$ and efficacy accuracy across subsequent iterations. Figure 4 illustrates the results of the analysis. We observe that the rate of $|\Delta p_k|$ decrease and accuracy improving over iterations is much more pronounced for the UnderEdit examples. This observation tells us that multiple iterations of SPREAD is the larger contributor to getting higher performance.

6.2 How prefixes influence neighbor-assisted model editing behavior?

Using random prefixes aid generalization across contexts in the memory editing process when only the target knowledge edit is known. So we pose the question, does adding random prefixes to the neighbor knowledge prompts help prevent OverEdit?

Algo	k	Efficacy (↑)	Generalization (↑)	Specificity (↑)	Score (↑)
		Accuracy	Accuracy	Accuracy	Accuracy
Unedited	0	1.00%	1.00%		
MEMIT	4	**99.00%**	**38.00%**	52.00%	54.00%
NA_MEMIT	4	**99.00%**	36.00%	**86.00%**	60.00%
NAP_MEMIT	4	**99.00%**	37.00%	84.00%	**61.00%**
PMET	8	**99.00%**	**31.00%**	67.00%	52.00%
NA_PMET	9	98.00%	29.00%	**85.00%**	53.00%
NAP_PMET	8	98.00%	30.00%	84.00%	**54.00%**

Table 3: Results of prefix-free (NA_) and with prefix(NAP_) neighbor-assisted model editing on GPT-2 XL on 739 samples of CounterFact. We compare their evaluation metrics and **bold** the higher value. Full results with success and perplexity performance for all iterations are reported in Table 9.

Dataset		CounterFact				ZsRE			
Model	Algo	k	Score (↑) Acc	$\lvert\Delta p_k\rvert$(↓)	Δ_{p2} (↓)	k	Score (↑) Acc	$\lvert\Delta p_k\rvert$(↓)	Δ_{p2}(↓)
GPT-2 XL (1.5B)	**MEMIT**	4	**56.67%**	0.47	8.65	3	**46.67%**	0.03	39.36
						4	45.00%	0.01	0.02
	PMET	10	**47.00%**	0.42	1.29	6	**54.00%**	0.19	1.29
						7	53.33%	0.08	0.11
GPTJ (6B)	**MEMIT**	2	**79.00%**	0.03	1.20	2	74.67%	0.01	1.65
						3	**75.00%**	0.00	**0.02**
	PMET	1	77.67%	1.15		2	74.00%	0.09	4.06
		3	**78.33%**	0.05	4.18	3	**74.33%**	0.02	0.07
		4	78.33%	0.02	0.03				
Llama-2 (7B)	**PMET**	2	78.00%	0.09	3.18	2	78.00%	0.07	6.28
		3	**78.33%**	0.16	0.07	3	**78.67%**	0.02	0.05

Table 4: Comparing stopping criteria. We compare our proposed stopping criteria (green) to the two alternate stopping criteria, monotonic decrease (orange), and small change (purple). We **bold** the higher scores among them. We report results for all iterations in Table 10.

To answer this question we run an experiment by adding random prefixes to the neighboring knowledge used during edit. Table 3 shows an increase in specificity accuracy[4]. However, this increase is less compared to the improvement of using neighbor-assisted editing versus no neighbor-assist. Regardless, the prefix-free neighbor-assisted edits (NA_) achieved better overall performance denoted by Accuracy in Score due to their significant boost in specificity.

6.3 How effective is the stopping criterion?

We tested two alternate stopping criteria to the proposed stopping criteria $|\Delta p_k| \leq 1$. The first is that $|\Delta p_k|$ should monotonically decrease i.e., $|\Delta p_{k+1}| < |\Delta p_k|$, otherwise stop and use θ_k. The second is to stop when the difference in perplexity between consecutive iterations, i.e after Spread stage, is small, i.e., $\Delta_{p2} = |p(\theta_{k+1}, h^{l_c}_{t,k+1}) - p(\theta_k, h^{l_c}_{t,k})| \leq 1$. We found our proposed criteria to be most the most effective in these experiments as shown in Table 4. Moreover, the second criteria suffered with two major drawbacks—a) it always needed at least two iterations to terminate and b) it always took one extra iteration longer than the proposed stopping criteria.

7 Related Work

In this work, we extensively discussed locate and edit model editing methods—ROME (Meng et al., 2023a), MEMIT (Meng et al., 2023b), and PMET (Li et al., 2024). In addition, there is a body of research that employs meta-learners to guide the parameter updates required for specific edits. For example, KE (Cao et al., 2021) uses a hyper-network to update model parameters, while MEND (Mitchell et al., 2022a) trains gradient-based, lightweight model editor networks. MALMEN (Tan et al., 2024) builds upon MEND to address scalability challenges.

Another line of research adds new knowledge without altering the model's parameters. SERAC (Mitchell et al., 2022b), GRACE (Hartvigsen et al., 2023), and WISE (Wang et al., 2024a) achieve this by employing additional memory to store new knowledge. A router network is then trained to decide whether to retrieve knowledge from the original model or the additional memory, ensuring the intended knowledge is accessed without modifying the model's core parameters. We specifically focus on locate-and-edit model editing methods due to their effectiveness and efficiency in updating only the important parameters. Our proposed method introduces simple changes to existing techniques while still demonstrating effectiveness.

8 Conclusion

In this work, we addressed key challenges in model editing—UnderEdit and OverEdit —by proposing iterative and neighbor-assisted model editing techniques. Our iterative approach effectively resolves UnderEdit by reducing the approximation error to ensure sufficient weight updates, while neighbor-assisted editing mitigates OverEdit by preserving neighboring knowledge. Extensive experiments across diverse editing algorithms, LLMs, and datasets validate the efficacy of our methods. These contributions pave the way for more reliable model editing, with broad applicability to dynamic knowledge updates in LLMs.

[4] The full results with success on each measurement and perplexity performance are in Table 9

9 Limitations

The results shows that our proposed iterative and neighbor-assisted model editing approaches are highly effective resolving UnderEdit and OverEdit, respectively. However, we did notice some trade-offs where the former negatively impacted specificity and later generalization. We believe, these trade-offs stem from the fundamental challenge faced by direct model editing methods where LLM parameters are shared across different types of stored knowledge and currently no method exits to isolate parameters related to a knowledge. Our experiments and results highlight these challenges and encourages the research community to explore further resolving these challenges. So, we would recommend the adopters of our methods to prioritize between specificity or generalization depending on the application-specific requirements.

Limited computational resources restricted us from experimenting with larger batch sizes and additional LLMs, such as GPT-NeoX (20B) and larger Llama-2 models. We hypothesize that the experimental result trend will remain the same, and we leave the verification of this hypothesis for future work.

References

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. *Preprint*, arXiv:2104.08164.

Govind Krishnan Gangadhar and Karl Stratos. 2024. Model editing by standard fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5907–5913, Bangkok, Thailand. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. 2024. 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*.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *CoRR*, abs/1706.04115.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18564–18572.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2023a. Locating and editing factual associations in gpt. *Preprint*, arXiv:2202.05262.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023b. Mass-editing memory in a transformer. *Preprint*, arXiv:2210.07229.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. *Preprint*, arXiv:2110.11309.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. *Preprint*, arXiv:2206.06520.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *Preprint*, arXiv:2104.10350.

Judea Pearl. 2013. Direct and indirect effects. *Preprint*, arXiv:1301.2300.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. *Preprint*, arXiv:2005.04611.

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

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Preprint*, arXiv:1706.03762.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024a. WISE: Rethinking the knowledge memory for lifelong model editing of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. 2024b. EasyEdit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93, Bangkok, Thailand. Association for Computational Linguistics.

Wanli Yang, Fei Sun, Xinyu Ma, Xun Liu, Dawei Yin, and Xueqi Cheng. 2024. The butterfly effect of model editing: Few edits can trigger large language models collapse. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5419–5437, Bangkok, Thailand. Association for Computational Linguistics.

Paul Youssef, Osman Koraş, Meijie Li, Jörg Schlötterer, and Christin Seifert. 2023. Give me the facts! a survey on factual knowledge probing in pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15588–15605, Singapore. Association for Computational Linguistics.

A MEMIT vs PMET vs ROME

As discussed in Section 2, model editing algorithms—MEMIT (Meng et al., 2023b), PMET (Li et al., 2024), and ROME (Meng et al., 2023a)—operate on the hypothesis that updating the final MLP parameters is sufficient to increase the likelihood of a new object o^* over the original object o when presented with a (subject, relation) pair $x = (s, r)$ as input to the LLM. Specifically, the final MLP weight matrix W functions as a linear associative memory that stores a key-value mapping $[k, v]$[5](Meng et al., 2023a). Here, the key encodes the last subject token, while the value represents the relation-object pair (r, o) as a property of the subject s. Within the transformer architecture, the key corresponds to the output of the first fully connected MLP layer, whereas the value corresponds to the output of the second fully connected MLP layer z, as shown in Figure2. This key-value mapping $[k, v]$ is derived by computing the inner product between the key k and the final MLP weight matrix W as $Wk \approx v$.

Model editing involves modifying a batch of M desired edits, represented as D= $\{(s_m, r_m, o_m, o^*m)\}m = 1^M$, which translates to inserting M new key-value pairs $[K_M, V_M]$ by updating the final MLP weights W at the causal layers $\{l_1, \ldots, l_c\}$.

MEMIT's objective in the SPREAD stage is to update final MLP weight matrix W_l at each causal layer l with a small change Δ, producing new weights W_l^* such that the M new key-value mappings $[K_M, V_M]$ are incorporated. Simultaneously, MEMIT seeks to preserve E existing key-value mappings $[K_E, V_E]$, where $K_E = [k_e]_{e=1}^E$ and $V_E = [v_e]_{e=1}^E$ are pre-existing vector keys and their corresponding values.

MEMIT obtains W_l^* by solving an optimization problem that seek a transformation $\hat{W}_l$ minimizing the sum of squared euclidean distances between the

[5] Distinct from key-value pairs in attention mechanisms

transformed key vectors $\hat{W}_l k_i$ and their respective target vectors v_i:

$$W_1 \triangleq \arg\min_{\hat{W}} \left(\sum_{i=1}^{E} \left\| \hat{W} k_i - v_i \right\|^2 + \sum_{i=E+1}^{E+M} \left\| \hat{W} k_i - v_i \right\|^2 \right).$$

The term $\sum_{i=E+1}^{E+M} \left\| \hat{W} k_i - v_i \right\|^2$ enforces the modification of $M \gg 1$ knowledge entries, while $\sum_{i=1}^{E} \left\| \hat{W} k_i - v_i \right\|^2$ iensures the preservation of existing knowledge. For a complete solution, we refer the reader to Meng et al. (2023b).

PMET follows the same optimization objective as MEMIT in the SPREAD stage but differs slightly in the OPTIMIZATION stage. As illustrated in Figure 2, PMET searches for an ideal self-attention output $\bar{attn}_t^{l_c}$ and an ideal MLP output $\bar{z}_t^{l_c}$. It hypothesizes that the self-attention module encodes general knowledge patterns, and thus, its contribution to the hidden state $h_t^{l_c}$ is not required when editing specific knowledge in the MLP. Consequently, PMET's SPREAD stage updates the final MLP weights W^l in the causal layers using only the ideal MLP output $\bar{z}_t^{l_c}$. This modification allows PMET to achieve more precise model edits compared to MEMIT, leading to improved efficacy and generalization performance.

ROME is the predecessor of MEMIT. Unlike MEMIT and PMET, ROME assumes that any single causal layer can sufficiently store the new knowledge. Thus, in the OPTIMIZATION stage, it identifies the ideal MLP output $\bar{z}^{lt}$ at a single arbitrary causal layer l_* and subsequently updates only the final MLP weights W^{l_*} at that layer in the SPREAD stage, as shown in Figure 2.

ROME partially follows MEMIT's optimization objective but with a stricter constraint: it aims to compute the updated MLP weight $\hat{W}^l$ such that it preserves all existing E key-value mappings $[K_E, V_E]$ while inserting only a single new key-value mapping (k_*, v_*). Meng et al. (2023a) achieve this via the following closed-form solution:

$$\text{minimize } ||\hat{W}^{l_*} K_E - V_E||$$
$$\text{such that } \hat{W}^{l_*} k_* = v_*$$

For further details and the complete derivation, we refer the reader to Meng et al. (2023a).

ROME, due to its strict equality constraint, does not support batch editing[6]. Notably, ROME produces the most precise edits compared to MEMIT and PMET when modifying a single fact, as its constraint ensures maximal accuracy. However, this same constraint becomes a disadvantage in batch editing scenarios, as it results in model collapse (Gupta et al., 2024; Yang et al., 2024). The collapse occurs due to a sharp increase in model perplexity, likely caused by the compounding effects of sequential edits.

B Implementation details

B.1 Iterative model editing

Currently, our implementation requires running an additional iteration to compute $p(\theta_{k+1}, \hat{h}_{t,k}^{l_c})$ for iteration k. As a result, $p(\theta_{k+1}, \hat{h}_{t,k}^{l_c})$ for the 5th iteration is not reported in Tables 6, 7, 8, and 9. We are updating our code to compute $p(\theta_{k+1}, \hat{h}_{t,k}^{l_c})$ at the end of the kth iteration without requiring the next iteration. This update involves running only the OPTIMIZATION stage with a single gradient step, using the initialization vector, before proceeding to the next iteration.

B.2 Neighbor-assisted model editing

We observed that different models often produce varying objects for the same neighboring knowledge. To calculate specificity, we used the model's actual output as the object (o), which should remain unchanged during editing. However, this introduced a challenge when employing neighbor-assisted model editing to guide the finding the ideal hidden state $\bar{h}_t^{l_c}$ during OPTIMIZATION, as models produced inconsistent outputs for neighboring knowledge used for evaluation. This inconsistency caused conflicts among neighboring knowledge when selecting a single instance for neighbor-assisted editing.

To address this, we filtered out neighboring knowledge samples that did not yield the same model output as the original ground truth reported in the dataset. This strategy ensured that any remaining neighboring knowledge sample could be randomly selected for neighbor-assisted editing, resolving the conflict.

This approach also introduced an additional constraint on the edit data points: each data point

[6] Since ROME can only edit one fact at a time, batch edits are performed sequentially.

must have at least two neighboring knowledge samples—one for editing and the others for evaluation. Unfortunately, the ZsRE dataset contains only one neighboring knowledge sample per data point, restricting us to the COUNTERFACT dataset. Even within COUNTERFACT, only a limited number of samples met the required constraints. The number of qualifying samples varied depending on the model, as shown in Table 2.

C Hardware Details

Table 5 outlines the GPU resources utilized to conduct edits of bach size 1000 across various models, algorithms, and datasets. It highlights the specific hardware configurations, such as GPU type (e.g., NVIDIA L40S with 48GB memory or NVIDIA A100 with 80GB memory), used for each experiment.

Model	Algo	Dataset	GPU
GPT-2 XL (1.5B)	**ROME**	COUNTERFACT ZsRE	L40S (48GB) L40S (48GB)
	MEMIT	COUNTERFACT ZsRE	L40S (48GB) L40S (48GB)
	PMET	COUNTERFACT ZsRE (L40S)	L40S (48GB) L40S (48GB)
GPT-J (6B)	**ROME**	COUNTERFACT ZsRE	A100 (80GB) A100 (80GB)
	MEMIT	COUNTERFACT ZsRE	L40S (48GB) A100 (80GB)
	PMET	COUNTERFACT ZsRE	A100 (80GB) A100 (80GB)
Llama 2 (7B)	**ROME**	COUNTERFACT ZsRE	A100 (80GB) A100 (80GB)
	MEMIT	COUNTERFACT ZsRE	L40S (48GB) A100 (80GB)
	PMET	COUNTERFACT ZsRE	L40S (48GB) A100 (80GB)

Table 5: GPU requirements to conduct 1000 edits

D Iterative model editing results

We present the complete results of all iterations of iterative model editing in Table 6 for the COUNTERFACT dataset and Table 7 for the ZsRE dataset. For experiments that did not meet our proposed stopping criteria within 5 iterations, we extended the runs by an additional 5 iterations and included those results as well.

E Neighbor-assisted model editing results

We present the complete results of all iterations of neighbor-assisted model editing in Table 8. As the ZsRE dataset was unsuitable for this experiment (see Appendix B.2 for details), results are reported only for the COUNTERFACT dataset. Furthermore, since only a subset of samples from COUNTERFACT qualified for this experiment, we also include the performance of iterative model editing on these samples for comparison with neighbor-assisted model editing.

Our current implementation of neighbor-assisted model editing does not use prefixes. To analyze its behavior with prefixes, we conducted an additional set of experiments. The results, presented in Table 9, compare prefix-based neighbor-assisted model editing with its prefix-free counterpart and iterative model editing. A detailed analysis is provided in Section 6.2.

Model	Algo	k	Efficacy (↑)		Generalization (↑)		Specificity (↑)		Score (↑)		Perplexity (↓)			$\|\Delta p_k\|$ (↓)
			Accuracy	Success	Accuracy	Success	Accuracy	Success	Accuracy	Success	ME-PPL-50	$p(\theta_k, \bar{h}^{l_c}_{t,k})$	$p(\theta_{k+1}, \hat{h}^{l_c}_{t,k})$	
GPT2XL (1.5B)	**Unedited**	0	0.00%	21.67%	0.00%	31.33%		56.67%		31.33%	54.66		3.26E+05	
	ROME	1	1.00%	56.00%	1.00%	55.00%	0.00%	92.00%	1.00%	64.00%	7.07E+03	5.29E+03	7.20E+05	7.15E+05
		2	0.00%	55.00%	0.00%	51.00%	0.00%	93.00%	0.00%	62.00%	1.36E+04	9.10E+04	1.78E+06	1.69E+06
		3	0.00%	55.00%	0.00%	52.00%	0.00%	95.00%	0.00%	63.00%	1.26E+04	7.91E+05	3.68E+06	2.89E+06
		4	0.00%	55.00%	0.00%	50.00%	0.00%	88.00%	0.00%	60.00%	3.13E+04	2.10E+06	1.93E+05	-1.91E+06
		5	3.00%	64.00%	1.00%	58.00%	0.00%	91.00%	1.00%	68.00%	1.14E+04	7.86E+04		
	MEMIT	1	79.00%	92.67%	22.00%	65.67%	**75.00%**	**99.00%**	42.67%	83.00%	58.66	1.04	11360.64	11359.60
		2	98.67%	99.67%	33.67%	75.67%	70.33%	99.00%	55.67%	89.67%	60.47	1.03	78.15	77.12
		3	99.33%	99.67%	35.33%	76.00%	69.33%	99.00%	56.67%	90.00%	61.58	1.02	10.14	9.11
		4	**99.67%**	**99.67%**	**35.67%**	**76.00%**	68.67%	**99.00%**	**56.67%**	**90.00%**	62.44	1.02	1.49	0.47
		5	99.67%	99.67%	35.67%	76.00%	68.67%	99.00%	56.67%	90.00%	63.28	1.02		
	PMET	1	21.67%	57.00%	3.00%	42.00%	**91.67%**	**100.00%**	8.33%	58.33%	55.60	12598.80	116384.51	103785.71
		2	65.67%	85.33%	14.67%	58.67%	84.00%	99.33%	31.67%	77.33%	57.01	1333.32	26852.75	25519.43
		3	86.00%	93.67%	20.00%	65.33%	80.67%	99.00%	40.67%	83.33%	58.00	97.51	5123.07	5025.56
		4	93.00%	97.00%	21.67%	68.00%	78.67%	99.00%	43.33%	85.33%	58.82	15.96	2045.77	2029.80
		5	96.00%	98.33%	22.67%	69.00%	77.67%	99.00%	44.00%	86.67%	59.47	3.90	228.44	224.53
		6	97.33%	99.67%	22.67%	69.67%	77.00%	99.00%	44.67%	86.67%	60.46	1.81	50.77	48.96
		7	98.67%	99.67%	23.67%	70.00%	77.00%	99.00%	45.67%	87.00%	61.52	1.42	22.44	21.02
		8	98.67%	99.67%	24.00%	70.00%	76.00%	99.00%	46.00%	87.00%	62.58	1.35	12.10	10.75
		9	99.33%	99.67%	24.00%	70.33%	76.00%	99.00%	46.00%	87.00%	63.20	1.32	3.03	1.71
		10	**99.33%**	**99.67%**	**24.67%**	**70.33%**	75.33%	99.00%	**47.00%**	**87.33%**	63.92	1.31	1.73	0.42
GPTJ (6B)	**Unedited**	0	9.33%	38.00%	9.00%	38.33%		82.00%		37.33%	39.80		5.98E+05	
	ROME	1	1.00%	57.00%	1.00%	55.00%	0.00%	76.00%	1.00%	61.00%	2.15E+05	1.19E+08	1.21E+14	1.21E+14
		2	1.00%	67.00%	2.00%	62.00%	0.00%	71.00%	1.00%	66.00%	8.51E+05	1.75E+12	4.17E+06	-1.75E+12
		3	1.00%	71.00%	1.00%	63.00%	0.00%	71.00%	1.00%	68.00%	9.00E+05	8.09E+05	2.44E+06	1.64E+06
		4	1.00%	72.00%	1.00%	64.00%	0.00%	73.00%	1.00%	69.00%	8.29E+05	7.36E+05	8.36E+05	9.97E+04
		5	1.00%	72.00%	1.00%	65.00%	0.00%	71.00%	1.00%	69.00%	7.10E+05	3.36E+05		
	MEMIT	1	99.00%	100.00%	75.00%	95.67%	**69.33%**	**89.33%**	77.67%	94.33%	42.20	1.03	2.25	1.22
		2	**99.33%**	**100.00%**	**80.67%**	**98.00%**	66.33%	88.33%	**79.00%**	**95.00%**	43.92	1.02	1.05	0.03
		3	99.67%	100.00%	82.00%	98.33%	65.33%	88.33%	79.00%	95.00%	46.33	1.02	2.88	1.86
		4	99.67%	100.00%	83.67%	98.33%	64.67%	88.00%	79.33%	95.00%	46.95	1.01	1.04	0.03
		5	99.67%	100.00%	83.67%	98.33%	64.33%	88.00%	79.33%	95.00%	47.20	1.01		
	PMET	1	98.00%	**99.67%**	76.00%	95.00%	**68.33%**	**88.67%**	77.67%	93.67%	41.27	1.08	2.24	1.15
		2	98.67%	99.67%	75.67%	95.00%	68.33%	89.00%	78.00%	94.33%	41.16	1.06	5.29	4.23
		3	**99.00%**	**99.67%**	**76.67%**	**95.67%**	**68.33%**	**88.67%**	**78.33%**	**94.00%**	41.15	1.06	1.11	0.05
		4	99.33%	99.67%	77.00%	95.67%	68.33%	88.67%	78.33%	94.00%	41.19	1.06	1.08	0.02
		5	99.33%	99.67%	77.00%	95.67%	68.00%	89.00%	78.33%	94.33%	41.28	1.06		
Llama2 (7B)	**Unedited**	0	15.00%	13.67%	15.00%	15.00%		84.33%		19.67%	30.63		2789.16	
	ROME	1	0.00%	48.00%	0.00%	49.00%	0.00%	76.00%	0.00%	55.00%	1.45E+04	3.80E+03	8.14E+05	8.10E+05
		2	0.00%	56.00%	0.00%	54.00%	0.00%	64.00%	0.00%	58.00%	7.27E+04	6.81E+05	2.34E+08	2.33E+08
		3	0.00%	55.00%	0.00%	53.00%	0.00%	57.00%	0.00%	55.00%	3.86E+05	1.55E+08	7.55E+09	7.39E+09
		4	0.00%	57.00%	0.00%	55.00%	0.00%	54.00%	0.00%	55.00%	1.26E+06	6.04E+09	5.81E+10	5.20E+10
		5	0.00%	56.00%	0.00%	52.00%	0.00%	57.00%	0.00%	55.00%	1.38E+06	4.36E+10		
	MEMIT	1	91.67%	98.00%	70.33%	93.33%	29.33%	67.33%	50.67%	83.67%	42.10	1.01	199.80	198.79
		2	14.33%	79.00%	9.67%	73.67%	6.67%	70.67%	9.00%	74.67%	9.37E+03	16.77	4681.01	4664.24
		3	26.67%	89.67%	12.33%	82.67%	5.67%	66.67%	9.67%	78.33%	4.35E+04	8.40	884.21	875.80
		4	22.00%	94.67%	5.67%	82.00%	5.67%	68.67%	7.00%	80.33%	8.63E+04	1.89	1381.59	1379.71
		5	38.33%	95.67%	10.67%	77.67%	4.67%	66.67%	8.67%	78.33%	7.60E+04	2.37		
	PMET	1	94.33%	97.00%	68.33%	86.67%	**76.33%**	**89.00%**	77.33%	90.33%	30.73	1.09	4.41	3.32
		2	**95.33%**	**98.33%**	**70.00%**	**88.67%**	75.33%	88.67%	**78.00%**	**91.67%**	30.76	1.14	1.23	0.09
		3	95.33%	98.33%	69.67%	88.67%	75.33%	88.67%	78.33%	91.67%	30.78	1.14	1.30	0.16
		4	95.33%	98.33%	70.00%	89.00%	75.33%	88.67%	78.33%	91.67%	30.79	1.14	1.14	0.00
		5	95.33%	98.33%	70.00%	89.00%	75.33%	88.67%	78.33%	91.67%	30.79	1.14		

Table 6: Iterative model editing results on CounterFact for at most 10 iterations (denoted by k). We compare the evaluation metrics of iteration that met stopping criterion $|\Delta p_k| \leq 1$ (green rows) to that of their corresponding first iteration and **bold** the higher value. PMET on GPT-2 XL require more than 5 iterations to achieve our stopping criteria. While ROME is known to collapse (red rows), we observed a unique case of collapse with Llama-2 (7B) specifically when using MEMIT. We discuss this in Section 5.

Model	Algo	k	Efficacy (↑)		Generalization (↑)		Specificity (↑)		Score (↑)		Perplexity (↓)			$\lvert\Delta p_k\rvert$ (↓)
			Accuracy	Success	Accuracy	Success	Accuracy	Success	Accuracy	Success	ME-PPL-50	$p(\theta_k, \bar{h}^{l_c}_{t,k})$	$p(\theta_{k+1}, \hat{h}^{l_c}_{t,k})$	
GPT-2 XL (1.5B)	Unedited	0	22.00%	87.33%	21.00%	86.33%		56.33%		73.67%	54.66		195351.10	
	ROME	1	3.00%	40.00%	3.00%	37.00%	1.00%	77.00%	2.00%	46.00%	8.59E+03	1.21E+04	2.64E+05	2.52E+05
		2	0.00%	97.00%	0.00%	97.00%	29.00%	71.00%	0.00%	86.00%	5.20E+03	5.75E+04	2.67E+05	2.10E+05
		3	0.00%	21.00%	0.00%	18.00%	0.00%	76.00%	0.00%	25.00%	1.21E+04	9.44E+04	3.27E+05	2.32E+05
		4	0.00%	100.00%	0.00%	100.00%	0.00%	68.00%	0.00%	86.00%	6.01E+03	8.27E+04	1.02E+06	9.36E+05
		5	0.00%	0.00%	0.00%	0.00%	0.00%	67.00%	0.00%	67.00%	1.85E+04	4.38E+05		
	MEMIT	1	69.00%	**100.00%**	58.67%	99.67%	**32.33%**	**85.00%**	**48.33%**	**94.00%**	61.74	1.02	2035.34	2034.31
		2	98.33%	100.00%	87.00%	100.00%	24.33%	84.33%	47.67%	94.00%	67.68	1.02	40.41	39.39
		3	**100.00%**	**100.00%**	**88.00%**	**100.00%**	23.33%	84.00%	46.67%	**94.00%**	69.40	1.02	1.05	0.03
		4	100.00%	100.00%	89.00%	100.00%	22.00%	84.00%	45.00%	94.00%	70.02	1.02	1.03	0.01
		5	100.00%	100.00%	89.67%	100.00%	21.67%	84.00%	44.33%	94.00%	70.16	1.02		
	PMET	1	34.67%	97.67%	30.33%	95.67%	45.00%	85.67%	35.33%	93.00%	56.77	5375.18	116427.63	111052.45
		2	67.00%	100.00%	52.00%	99.33%	38.33%	85.33%	49.67%	94.33%	60.44	15.17	3109.90	3094.74
		3	89.67%	100.00%	66.33%	99.67%	35.00%	85.00%	55.00%	94.33%	62.37	2.16	491.43	489.28
		4	94.33%	100.00%	69.67%	100.00%	33.33%	84.67%	54.67%	94.00%	64.42	1.34	50.04	48.69
		5	98.00%	100.00%	73.00%	100.00%	32.00%	84.67%	54.00%	94.00%	65.22	1.25	2.70	1.45
		6	**99.33%**	**100.00%**	**73.67%**	**100.00%**	31.00%	84.00%	**54.00%**	**94.00%**	65.93	1.23	1.41	0.19
		7	99.00%	100.00%	75.00%	100.00%	30.33%	84.00%	53.33%	94.00%	66.41	1.22	1.30	0.08
		8	99.67%	100.00%	74.67%	100.00%	30.00%	84.00%	53.00%	94.00%	66.71	1.21	1.25	0.04
		9	99.67%	100.00%	75.00%	100.00%	29.33%	84.00%	52.33%	94.00%	66.77	1.22	1.22	0.01
		10	100.00%	100.00%	75.33%	100.00%	29.33%	84.00%	52.33%	94.00%	66.89	1.20	1.21	0.01
GPT-J (6B)	Unedited	0	27.33%	91.00%	26.33%	90.00%		60.00%		77.33%	39.80		5.02E+04	5.02E+04
	ROME	1	5.00%	90.00%	5.00%	89.00%	0.00%	65.00%	5.00%	80.00%	3.93E+05	1.63E+04	1.33E+05	1.17E+05
		2	13.00%	95.00%	10.00%	94.00%	0.00%	67.00%	11.00%	83.00%	6.19E+05	8.72E+03	1.58E+04	7.04E+03
		3	17.00%	99.00%	12.00%	97.00%	0.00%	66.00%	14.00%	84.00%	6.88E+05	3.75E+03	9.70E+03	5.94E+03
		4	14.00%	100.00%	11.00%	99.00%	0.00%	64.00%	12.00%	84.00%	1.39E+06	4.61E+03	1.28E+04	8.24E+03
		5	10.00%	99.00%	8.00%	99.00%	0.00%	63.00%	9.00%	83.00%	3.32E+06	7.89E+03		
	MEMIT	1	98.67%	100.00%	89.33%	**100.00%**	**52.67%**	80.00%	**74.67%**	92.00%	41.56	1.01	2.68	1.67
		2	**99.33%**	**100.00%**	**92.67%**	**100.00%**	51.67%	**80.33%**	**74.67%**	**92.33%**	41.71	1.02	1.03	0.01
		3	100.00%	100.00%	94.00%	100.00%	51.33%	80.00%	75.00%	92.33%	41.88	1.02	1.02	0.00
		4	100.00%	100.00%	93.67%	100.00%	51.33%	80.00%	75.00%	92.33%	41.88	1.01	1.01	0.00
		5	100.00%	100.00%	93.67%	100.00%	51.33%	80.00%	75.00%	92.33%	41.87	1.01		
	PMET	1	95.67%	**100.00%**	87.33%	**100.00%**	52.00%	**80.00%**	72.67%	92.00%	41.94	1.10	5.20	4.10
		2	**98.67%**	**100.00%**	**88.00%**	99.67%	**52.00%**	**80.00%**	**74.00%**	**92.00%**	41.61	1.06	1.14	0.09
		3	99.67%	100.00%	89.67%	100.00%	52.00%	80.00%	74.33%	92.00%	41.62	1.06	1.08	0.02
		4	100.00%	100.00%	89.67%	100.00%	52.00%	80.00%	74.33%	92.00%	41.59	1.06	1.06	0.00
		5	100.00%	100.00%	89.67%	100.00%	52.00%	80.00%	74.33%	92.00%	41.58	1.06		
Llama2 (7B)	Unedited	0	38.33%	57.00%	37.00%	56.00%		59.67%		55.67%	33.69		2.01E+04	2.01E+04
	ROME	1	9.00%	93.00%	8.00%	92.00%	0.00%	72.00%	9.00%	84.00%	2.49E+04	7.73E+01	7.05E+04	7.04E+04
		2	13.00%	99.00%	11.00%	99.00%	1.00%	73.00%	3.00%	88.00%	4.49E+04	4.85E+02	9.17E+03	8.69E+03
		3	22.00%	100.00%	16.00%	99.00%	0.00%	72.00%	18.00%	88.00%	3.73E+04	1.07E+03	6.18E+03	5.11E+03
		4	24.00%	100.00%	17.00%	99.00%	0.00%	74.00%	20.00%	89.00%	3.53E+04	1.78E+03	1.04E+04	8.61E+03
		5	24.00%	100.00%	17.00%	98.00%	0.00%	74.00%	20.00%	89.00%	3.50E+04	2.47E+03		
	MEMIT	1	79.50%	99.50%	76.50%	99.00%	31.00%	74.50%	51.50%	89.00%	41.04	1.05	23.10	22.06
		2	6.50%	88.00%	6.00%	86.00%	5.50%	80.00%	6.00%	84.50%	4435.72	1.07	2.38E+04	2.38E+04
		3	13.50%	96.00%	11.00%	95.00%	3.50%	70.00%	6.50%	85.00%	120046.73	1.68	9.45E+03	9.45E+03
		4	6.00%	94.00%	4.50%	91.50%	4.50%	72.00%	5.00%	84.50%	34779.75	1.65	1.74E+04	1.74E+04
		5	6.50%	86.00%	6.00%	83.00%	1.00%	68.00%	2.00%	78.00%	40680.54	1.70		
	PMET	1	90.00%	98.67%	**83.00%**	96.33%	**66.00%**	**74.67%**	77.33%	**88.33%**	34.63	1.07	7.40	6.33
		2	**92.00%**	**99.00%**	**83.33%**	**96.67%**	**66.00%**	**74.67%**	**78.00%**	**88.33%**	34.47	1.05	1.12	0.07
		3	92.33%	99.00%	84.33%	96.67%	66.00%	74.67%	78.67%	88.33%	34.44	1.05	1.07	0.02
		4	93.00%	99.00%	84.67%	96.67%	65.67%	74.67%	78.67%	88.33%	34.42	1.05	1.06	0.00
		5	93.00%	99.00%	84.67%	97.00%	66.00%	74.67%	79.00%	88.33%	34.44	1.05		

Table 7: Iterative model editing results on ZsRE for at most 10 iterations (denoted by k). We compare the evaluation metrics of iteration that met stopping criterion $|\Delta p_k| \leq 1$ (green rows) to that of their corresponding first iteration and **bold** the higher value. PMET on GPT-2 XL require more than 5 iterations to achieve our stopping criteria. While ROME is known to collapse (red rows), we observed a unique case of collapse with Llama-2 (7B) specifically when using MEMIT. We discuss this in Section 5.

Model	Algo	k	Efficacy (↑)		Generalization (↑)		Specificity (↑)		Score (↑)		Perplexity (↓)			$\lvert\Delta p_k\rvert$ (↓)
			Accuracy	Success	Accuracy	Success	Accuracy	Success	Accuracy	Success	ME-PPL-50	$p(\theta_k, \bar{h}^{l_c}_{t,k})$	$p(\theta_{k+1}, \hat{h}^{l_c}_{t,k})$	
GPT-2 XL (1.5B) #739	**Unedited**	0	1.00%	9.00%	1.00%	22.00%		100.00%		18.00%	54.66			
	MEMIT	1	87.00%	94.00%	30.00%	67.00%	56.00%	81.00%	48.00%	79.00%	58.79	1.04	12808.17	12807.13
		2	98.00%	99.00%	37.00%	73.00%	51.00%	77.00%	53.00%	82.00%	59.75	1.03	59.15	58.12
		3	99.00%	99.00%	38.00%	74.00%	50.00%	77.00%	53.00%	82.00%	60.74	1.02	4.17	3.15
		4	**99.00%**	**99.00%**	**38.00%**	**74.00%**	52.00%	77.00%	54.00%	82.00%	62.27	1.02	1.07	0.05
		5	99.00%	99.00%	38.00%	74.00%	53.00%	78.00%	55.00%	83.00%	63.88	1.02		
	NA_MEMIT	1	78.00%	83.00%	22.00%	53.00%	87.00%	97.00%	43.00%	73.00%	57.56	1.07	1248.69	1247.62
		2	99.00%	99.00%	36.00%	71.00%	82.00%	93.00%	60.00%	86.00%	59.94	1.04	52.51	51.47
		3	99.00%	99.00%	36.00%	70.00%	84.00%	95.00%	60.00%	86.00%	61.95	1.03	3.67	2.64
		4	**99.00%**	**99.00%**	36.00%	70.00%	86.00%	95.00%	**60.00%**	**86.00%**	64.89	1.03	1.22	0.19
		5	99.00%	99.00%	35.00%	68.00%	87.00%	96.00%	60.00%	85.00%	66.73	1.03		
	PMET	1	0.29	0.49	0.06	0.36	0.92	0.98	0.15	0.52	55.61	3166.97	14020.32	10853.35
		2	0.72	0.85	0.2	0.57	0.78	0.93	0.39	0.75	56.73	146.75	11323.25	11176.5
		3	0.88	0.93	0.25	0.63	0.73	0.91	0.46	0.8	57.73	14.57	4457.16	4442.59
		4	0.95	0.97	0.27	0.66	0.71	0.89	0.48	0.82	58.68	2.69	2399.39	2396.7
		5	0.97	0.98	0.28	0.67	0.71	0.88	0.5	0.82	59.77	1.4	1073.72	1072.32
		6	0.98	0.99	0.29	0.67	0.67	0.87	0.5	0.82	60.99	1.33	171.9	170.57
		7	0.99	0.99	0.3	0.68	0.67	0.86	0.51	0.82	62.4	1.32	12.43	11.11
		8	**0.99**	**0.99**	**0.31**	**0.68**	0.67	0.86	0.52	0.82	63.47	1.31	1.62	0.31
		9	0.99	0.99	0.3	0.68	0.66	0.86	0.52	0.82	64.34	1.29	1.4	0.11
		10	0.99	0.99	0.31	0.68	0.66	0.86	0.52	0.82	65.16	1.28	1.33	0.05
	NA_PMET	1	0.29	0.48	0.06	0.35	0.93	0.99	0.15	0.5	55.59	740.72	2872.44	2131.72
		2	0.7	0.83	0.19	0.54	0.84	0.97	0.39	0.73	56.54	46.86	2336.83	2289.97
		3	0.88	0.92	0.25	0.61	0.84	0.96	0.48	0.8	57.36	7.04	1193.64	1186.6
		4	0.94	0.96	0.26	0.63	0.87	0.96	0.5	0.82	58.54	2.18	1651.91	1649.73
		5	0.96	0.97	0.28	0.64	0.86	0.96	0.52	0.82	60.37	1.36	851.28	849.92
		6	0.97	0.98	0.29	0.64	0.87	0.96	0.53	0.83	62.03	1.31	193.5	192.19
		7	0.97	0.98	0.29	0.64	0.85	0.97	0.53	0.83	63.48	1.3	24.63	23.33
		8	0.98	0.98	0.29	0.64	0.85	0.97	0.53	0.83	64.92	1.27	3.37	2.1
		9	0.98	0.98	0.29	0.64	**0.85**	**0.97**	**0.53**	**0.83**	66.29	1.26	1.55	0.29
		10	0.98	0.98	0.3	0.65	0.83	0.96	0.54	0.84	67.86	1.24	1.51	0.27
GPT-J (6B) #960	**Unedited**	0	0.00%	8.00%	1.00%	10.00%		100.00%		12.00%	39.80			
	MEMIT	1	99.00%	100.00%	71.00%	93.00%	67.00%	86.00%	77.00%	93.00%	42.79	1.03	4.24	3.21
		2	**99.00%**	**100.00%**	**79.00%**	**97.00%**	63.00%	83.00%	78.00%	93.00%	44.84	1.02	1.66	0.64
		3	99.00%	100.00%	79.00%	98.00%	62.00%	83.00%	77.00%	93.00%	48.58	1.01	17.10	16.09
		4	100.00%	100.00%	79.00%	98.00%	59.00%	82.00%	76.00%	93.00%	49.43	1.01	1.74	0.73
		5	100.00%	100.00%	81.00%	98.00%	58.00%	81.00%	76.00%	92.00%	50.50	1.01		
	NA_MEMIT	1	98.00%	99.00%	63.00%	82.00%	84.00%	95.00%	79.00%	91.00%	42.71	1.04	2.56	1.52
		2	**99.00%**	99.00%	75.00%	92.00%	**81.00%**	**95.00%**	**84.00%**	**95.00%**	45.24	1.02	1.35	0.33
		3	99.00%	100.00%	74.00%	91.00%	83.00%	95.00%	84.00%	95.00%	47.19	1.02	3.29	2.27
		4	99.00%	100.00%	74.00%	90.00%	80.00%	95.00%	83.00%	95.00%	49.77	1.02	1.43	0.41
		5	100.00%	100.00%	73.00%	91.00%	79.00%	95.00%	82.00%	95.00%	50.94	1.02		
	PMET	1	**99.00%**	**100.00%**	**72.00%**	**93.00%**	65.00%	84.00%	76.00%	92.00%	40.79	1.06	1.31	0.25
		2	99.00%	100.00%	73.00%	93.00%	65.00%	84.00%	77.00%	92.00%	40.90	1.06	219.70	218.64
		3	99.00%	100.00%	73.00%	93.00%	65.00%	84.00%	77.00%	92.00%	40.92	1.06	2.25	1.19
		4	99.00%	100.00%	73.00%	94.00%	65.00%	84.00%	77.00%	92.00%	41.08	1.05	1.06	0.01
		5	99.00%	100.00%	74.00%	94.00%	65.00%	84.00%	77.00%	92.00%	41.28	1.05		
	NA_PMET	1	99.00%	100.00%	72.00%	91.00%	75.00%	90.00%	80.00%	93.00%	41.01	1.06	2.38	1.32
		2	98.00%	99.00%	71.00%	90.00%	82.00%	94.00%	82.00%	94.00%	41.23	1.12	4.91	3.79
		3	99.00%	99.00%	70.00%	89.00%	83.00%	95.00%	82.00%	94.00%	41.66	1.13	24.94	23.81
		4	97.00%	98.00%	68.00%	88.00%	84.00%	95.00%	82.00%	93.00%	41.99	1.13	4.13	3.00
		5	98.00%	98.00%	69.00%	89.00%	81.00%	94.00%	81.00%	94.00%	42.71	1.12	273.05	271.93
		6	98.00%	99.00%	69.00%	89.00%	**80.00%**	**94.00%**	**81.00%**	**94.00%**	43.54	1.12	1.56	0.44
		7	99.00%	99.00%	68.00%	89.00%	76.00%	93.00%	79.00%	94.00%	44.62	1.13	1.57	0.44
		8	98.00%	99.00%	67.00%	88.00%	75.00%	92.00%	78.00%	93.00%	45.70	1.13	1.64	0.51
		9	98.00%	99.00%	67.00%	88.00%	72.00%	92.00%	77.00%	93.00%	47.48	1.14	2.11	0.97
		10	99.00%	99.00%	68.00%	88.00%	70.00%	91.00%	76.00%	93.00%	50.68	1.14		-1.14
Llama 2 (7B) #1340	**Unedited**	0	38.33%	57.00%	37.00%	56.00%		59.67%		55.67%	33.69			
	PMET	1	94.00%	97.00%	70.00%	87.00%	71.00%	92.00%	77.00%	92.00%	30.95	1.09	2.29	1.2
		2	96.00%	98.00%	72.00%	89.00%	71.00%	92.00%	78.00%	93.00%	31.03	1.14	25.72	24.58
		3	96.00%	98.00%	72.00%	89.00%	70.00%	92.00%	78.00%	93.00%	31	1.14	1.58	0.44
		4	96.00%	98.00%	72.00%	89.00%	71.00%	92.00%	78.00%	93.00%	31.01	1.14	1.14	0
		5	96.00%	98.00%	72.00%	89.00%	71.00%	92.00%	78.00%	93.00%	31	1.14		
	NA_PMET	1	92.00%	95.00%	67.00%	80.00%	76.00%	94.00%	77.00%	89.00%	30.43	1.14	15.27	14.13
		2	96.00%	97.00%	75.00%	87.00%	73.00%	93.00%	80.00%	92.00%	30.52	1.02	2.86	1.84
		3	96.00%	96.00%	76.00%	87.00%	71.00%	94.00%	80.00%	92.00%	31.19	1.04	2.61	1.57
		4	95.00%	95.00%	76.00%	86.00%	66.00%	92.00%	77.00%	91.00%	32.19	1.04	2.27	1.23
		5	94.00%	94.00%	75.00%	85.00%	60.00%	91.00%	74.00%	90.00%	35.28	1.05	186	184.95
		6	90.00%	91.00%	74.00%	84.00%	55.00%	89.00%	70.00%	88.00%	40.1	1.06	13.09	12.03
		7	87.00%	88.00%	70.00%	81.00%	47.00%	86.00%	64.00%	85.00%	53.95	1.06	30.27	29.21
		8	80.00%	83.00%	66.00%	80.00%	42.00%	83.00%	58.00%	82.00%	104.98	1.06	84.27	83.21
		9	73.00%	83.00%	58.00%	78.00%	30.00%	79.00%	47.00%	80.00%	373.87	1.07	358.05	356.98
		10	62.00%	79.00%	47.00%	75.00%	26.00%	76.00%	40.00%	76.00%	1414.01	1.07		

Table 8: Neighbor-Assisted model editing results on CounterFact. We compare evaluation metrics for both neighbor-assisted (NA_) and without neighbor runs of the model editing algorithms where $|\Delta p_k| \leq 1$ (green rows) and **bold** the higher value. Results among models and from Table 6 are not comparable due to difference in neighboring samples (Appendix. B.2). Hence, we report the no. of examples (#) used to run experiment for each model. NA_PMET on Llama-2 (7B) stands as an exception that didn't achieved the stopping criteria within 10 iteration and showed a performance decrease.

Model	Algo	k	Efficacy (↑)		Generalization (↑)		Specificity (↑)		Score (↑)		Perplexity (↓)			$\lvert\Delta p_k\rvert$ (↓)
			Accuracy	Success	Accuracy	Success	Accuracy	Success	Accuracy	Success	ME-PPL-50	$p(\theta_k, \bar{h}^{l_c}_{t,k})$	$p(\theta_{k+1}, \hat{h}^{l_c}_{t,k})$	
GPT-2 XL (1.5B) #739	Unedited	0	1.00%	9.00%	1.00%	22.00%		100.00%		18.00%	54.66			
	MEMIT	1	87.00%	94.00%	30.00%	67.00%	56.00%	81.00%	48.00%	79.00%	58.79	1.04	12808.17	12807.13
		2	98.00%	99.00%	37.00%	73.00%	51.00%	77.00%	53.00%	82.00%	59.75	1.03	59.15	58.12
		3	99.00%	99.00%	38.00%	74.00%	50.00%	77.00%	53.00%	82.00%	60.74	1.02	4.17	3.15
		4	**99.00%**	**99.00%**	**38.00%**	**74.00%**	52.00%	77.00%	54.00%	82.00%	62.27	1.02	1.07	0.05
		5	99.00%	99.00%	38.00%	74.00%	53.00%	78.00%	55.00%	83.00%	63.88	1.02		
	NA_MEMIT	1	78.00%	83.00%	22.00%	53.00%	87.00%	97.00%	43.00%	73.00%	57.56	1.07	1248.69	1247.62
		2	99.00%	99.00%	36.00%	71.00%	82.00%	93.00%	60.00%	86.00%	59.94	1.04	52.51	51.47
		3	99.00%	99.00%	36.00%	70.00%	84.00%	95.00%	60.00%	86.00%	61.95	1.03	3.67	2.64
		4	**99.00%**	**99.00%**	36.00%	70.00%	**86.00%**	**95.00%**	60.00%	**86.00%**	64.89	1.03	1.22	0.19
		5	99.00%	99.00%	35.00%	68.00%	87.00%	96.00%	60.00%	85.00%	66.73	1.03		
	NAP_MEMIT	1	79.00%	84.00%	23.00%	54.00%	84.00%	96.00%	45.00%	74.00%	56.9	1.07	1020.79	1019.72
		2	99.00%	99.00%	38.00%	72.00%	77.00%	93.00%	61.00%	86.00%	58.97	1.04	81.73	80.69
		3	99.00%	99.00%	37.00%	70.00%	84.00%	94.00%	62.00%	86.00%	60.59	1.03	3.18	2.15
		4	**99.00%**	**99.00%**	37.00%	69.00%	84.00%	94.00%	**61.00%**	85.00%	63.4	1.03	1.24	0.21
		5	99.00%	99.00%	37.00%	69.00%	84.00%	94.00%	61.00%	85.00%	65.98	1.03		
	PMET	1	29.00%	49.00%	6.00%	36.00%	92.00%	98.00%	15.00%	52.00%	55.61	3166.97	14020.32	10853.35
		2	72.00%	85.00%	20.00%	57.00%	78.00%	93.00%	39.00%	75.00%	56.73	146.75	11323.25	11176.5
		3	88.00%	93.00%	25.00%	63.00%	73.00%	91.00%	46.00%	80.00%	57.73	14.57	4457.16	4442.59
		4	95.00%	97.00%	27.00%	66.00%	71.00%	89.00%	48.00%	82.00%	58.68	2.69	2399.39	2396.7
		5	97.00%	98.00%	28.00%	67.00%	71.00%	88.00%	50.00%	82.00%	59.77	1.4	1073.72	1072.32
		6	98.00%	99.00%	29.00%	67.00%	67.00%	87.00%	50.00%	82.00%	60.99	1.33	171.9	170.57
		7	99.00%	99.00%	30.00%	68.00%	67.00%	86.00%	51.00%	82.00%	62.4	1.32	12.43	11.11
		8	**99.00%**	**99.00%**	**31.00%**	**68.00%**	67.00%	86.00%	52.00%	82.00%	63.47	1.31	1.62	0.31
		9	99.00%	99.00%	30.00%	68.00%	66.00%	86.00%	52.00%	82.00%	64.34	1.29	1.4	0.11
		10	99.00%	99.00%	31.00%	68.00%	66.00%	86.00%	52.00%	82.00%	65.16	1.28	1.33	0.05
	NA_PMET	1	29.00%	48.00%	6.00%	35.00%	93.00%	99.00%	15.00%	50.00%	55.59	740.72	2872.44	2131.72
		2	70.00%	83.00%	19.00%	54.00%	84.00%	97.00%	39.00%	73.00%	56.54	46.86	2336.83	2289.97
		3	88.00%	92.00%	25.00%	61.00%	84.00%	96.00%	48.00%	80.00%	57.36	7.04	1193.64	1186.6
		4	94.00%	96.00%	26.00%	63.00%	87.00%	96.00%	50.00%	82.00%	58.54	2.18	1651.91	1649.73
		5	96.00%	97.00%	28.00%	64.00%	86.00%	96.00%	52.00%	82.00%	60.37	1.36	851.28	849.92
		6	97.00%	98.00%	29.00%	64.00%	87.00%	96.00%	53.00%	83.00%	62.03	1.31	193.5	192.19
		7	97.00%	98.00%	29.00%	64.00%	85.00%	97.00%	53.00%	83.00%	63.48	1.3	24.63	23.33
		8	98.00%	98.00%	29.00%	64.00%	85.00%	97.00%	53.00%	83.00%	64.92	1.27	3.37	2.1
		9	98.00%	98.00%	29.00%	64.00%	**85.00%**	**97.00%**	**53.00%**	**83.00%**	66.29	1.26	1.55	0.29
		10	98.00%	98.00%	30.00%	65.00%	83.00%	96.00%	54.00%	84.00%	67.86	1.24	1.51	0.27
	NAP_PMET	1	1.00%	9.00%	1.00%	22.00%		100.00%		18.00%	54.66	768.86	2983.29	2214.43
		2	28.00%	48.00%	6.00%	35.00%	92.00%	98.00%	15.00%	50.00%	55.54	47.43	2914.13	2866.7
		3	70.00%	83.00%	19.00%	54.00%	84.00%	97.00%	39.00%	73.00%	56.41	6.87	2892.36	2885.49
		4	88.00%	92.00%	25.00%	60.00%	82.00%	96.00%	47.00%	79.00%	57.29	2.09	1652.55	1650.46
		5	94.00%	96.00%	27.00%	62.00%	83.00%	95.00%	50.00%	81.00%	58.34	1.31	925.36	924.05
		6	96.00%	97.00%	28.00%	64.00%	83.00%	95.00%	52.00%	82.00%	59.69	1.28	222.03	220.75
		7	98.00%	98.00%	29.00%	64.00%	84.00%	95.00%	53.00%	83.00%	60.9	1.26	21.49	20.23
		8	98.00%	**99.00%**	30.00%	64.00%	84.00%	94.00%	**54.00%**	**83.00%**	62.26	1.24	2.19	0.95
		9	98.00%	99.00%	29.00%	64.00%	84.00%	95.00%	54.00%	83.00%	63.16	1.22	1.44	0.22
		10	98.00%	99.00%	30.00%	65.00%	82.00%	94.00%	53.00%	83.00%	64.04	1.21	1.47	0.26

Table 9: Results of prefix-free (NA_) and with prefix(NAP_) neighbor-assisted model editing on COUNTERFACT. We compare their evaluation metrics when our stopping criteria $|\Delta p_k| \leq 1$ (green rows) is met and **bold** the higher value. Results among models and from Table 6 are not comparable due to difference in neighboring samples as explained in Appendix B.2. Hence, we report the no. of examples (#) used to run experiment for each model.

Dataset			COUNTERFACT				ZsRE			
Model	Algo	k	Score (↑)		$\lvert\Delta p_k\rvert$ (↓)	Δ_{p2} (↓)	Score (↑)		$\lvert\Delta p_k\rvert$ (↓)	Δ_{p2} (↓)
			Accuracy	Success			Accuracy	Success		
GPT-2 XL (1.5B)	**Unedited**	0		31.33%				73.67%		
	MEMIT	1	42.67%	83.00%	11359.60		48.33%	94.00%	2034.31	
		2	55.67%	89.67%	77.12	1.13E+04	47.67%	94.00%	39.39	1994.93
		3	56.67%	90.00%	9.11	68.01	**46.67%**	**94.00%**	0.03	39.36
		4	**56.67%**	**90.00%**	0.47	8.65	45.00%	94.00%	0.01	0.02
		5	56.67%	90.00%			44.33%	94.00%		
	PMET	1	8.33%	58.33%	103785.71		35.33%	93.00%	111052.45	
		2	31.67%	77.33%	25519.43	8.95E+04	49.67%	94.33%	3094.74	113317.73
		3	40.67%	83.33%	5025.56	2.17E+04	55.00%	94.33%	489.28	2618.47
		4	43.33%	85.33%	2029.80	3077.3	54.67%	94.00%	48.69	441.40
		5	44.00%	86.67%	224.53	1805.25	54.00%	94.00%	1.45	47.33
		6	44.67%	86.67%	48.96	175.57	**54.00%**	**94.00%**	0.19	1.29
		7	45.67%	87.00%	21.02	27.94	53.33%	**94.00%**	0.08	0.11
		8	46.00%	87.00%	10.75	10.27	53.00%	94.00%	0.04	0.05
		9	46.00%	87.00%	1.71	9.04	52.33%	94.00%	0.01	0.03
		10	**47.00%**	**87.33%**	0.42	1.29	52.33%	94.00%	0.01	0.01
GPT-J (6B)	**Unedited**	0		37.33%				77.33%	5.02E+04	
	MEMIT	1	77.67%	94.33%	1.22		74.67%	92.00%	1.67	
		2	**79.00%**	**95.00%**	0.03	1.20	74.67%	**92.33%**	0.01	1.65
		3	79.00%	95.00%	1.86	1.83	**75.00%**	**92.33%**	0.00	0.02
		4	79.33%	95.00%	0.03	1.84	75.00%	92.33%	0.00	0.00
		5	79.33%	95.00%			75.00%	92.33%		
	PMET	1	77.67%	93.67%	1.15		72.67%	92.00%	4.10	
		2	78.00%	94.33%	4.23	3.05	74.00%	**92.00%**	0.09	4.06
		3	**78.33%**	**94.00%**	0.05	4.18	**74.33%**	**92.00%**	0.02	0.07
		4	78.33%	94.00%	0.02	0.03	74.33%	92.00%	0.00	0.01
		5	78.33%	94.33%			74.33%	92.00%		
Llama-2 (7B)	**Unedited**	0		19.67%				55.67%	2.01E+04	
	PMET	1	77.33%	90.33%	3.32		77.33%	88.33%	6.33	
		2	78.00%	**91.67%**	0.09	3.18	78.00%	**88.33%**	0.07	6.28
		3	**78.33%**	**91.67%**	0.16	0.07	**78.67%**	**88.33%**	0.02	0.05
		4	78.33%	91.67%	0.00	0.16	78.67%	88.33%	0.00	0.01
		5	78.33%	91.67%			79.00%	88.33%		

Table 10: Comparing stopping criteria. We compare our proposed stopping criteria $|\Delta p_k| \leq 1$ (green) to the two alternate stopping criteria, monotonic decrease i.e. $|\Delta p_{k+1}| < |\Delta p_k|$, otherwise stop and use θ_k (orange), and small change, i.e., $\Delta_{p2} = |p(\theta_{k+1}, h^{l_c}_{t,k+1}) - p(\theta_k, h^{l_c}_{t,k})| \leq 1$ (purple). We **bold** the higher scores among them.