Neuron-Level Sequential Editing for Large Language Models

Anonymous ACL submission

Abstract

This work explores sequential model editing in large language models (LLMs), a critical task that involves modifying internal knowledge within LLMs continuously through multi-round editing, each incorporating updates or corrections to adjust the model's outputs without the need for costly retraining. Existing model editing methods, especially those that alter model parameters, typically focus on single-round editing and often face significant challenges in sequential model editing-most notably issues of model forgetting and failure. To address these challenges, we introduce a new model editing method, namely **N**euron-level **S**equential **E**diting (NSE), tailored for supporting sequential model editing. Specifically, we optimize the target layer's hidden states using the model's original weights to prevent model failure. Furthermore, we iteratively select neurons in multiple layers for editing based on their activation values to mitigate model forgetting. Our empirical experiments demonstrate that NSE significantly outperforms current modifying parameters model editing methods, marking a substantial advancement in the field of sequential model editing. Our code is released on https://anonymous.4open.science/r/NSE-0A8D/.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in storing extensive factual knowledge during pre-training and recalling this information during inference (Brown et al., 2020; Petroni et al., 2019; Roberts et al., 2020). However, as real-world knowledge continuously evolves, the information within these models can become outdated or incorrect (Cao et al., 2021; Mitchell et al., 2022a; Fang et al., 2024). Retraining LLMs to incorporate new information is often prohibitively costly (Mitchell et al., 2022b; Meng et al., 2022). Consequently, recent years have witnessed a surge in *model editing* methods focusing on modifying specific knowledge without the complete retraining process. Specifically, they first identify the crucial layers for the target knowledge by calculating their casual effect on output. Then, by updating the weights of these layers, they manipulate the hidden states of these layers to modify the final output, allowing LLMs to seamlessly adapt to dynamic real-world information (Meng et al., 2023; Hartvigsen et al., 2023).

Although current direct model editing methods are effective for single-round modifications, real-world applications require a continuous learning process where models must retain previous edits during subsequent modifications (Yao et al., 2023). This has led to the concept of *sequential model editing*, which requires the performance of multiple consecutive edits on models. However, current direct model editing methods pose significant risks in this context (Meng et al., 2022, 2023). The primary risk is *model forgetting*, where cumulative changes in parameters from consecutive edits cause the model to forget previously modified knowledge, thus degrading overall performance (Gupta et al., 2024a). For example, as illustrated in Figure 1 (a), after editing the model with new knowledge about "The cat", the LLM forgets previously edited knowledge about "The latest Olympic".

Furthermore, the second risk is *model failure*, where excessive edits impair the model's ability to generate coherent text. Worse still, this impairment can potentially lead to model collapse, characterized by producing irrelevant, repetitive, or non-sensical text, as illustrated in Figure 1 (a). In view of this, recent research, such as memory-based methods (Mitchell et al., 2022b; Hartvigsen et al., 2023), has attempted to address these challenges by preserving the LLM parameters after each edit. However, the increasing storage requirements as the number of edits grows significantly limit the practicality of these methods.

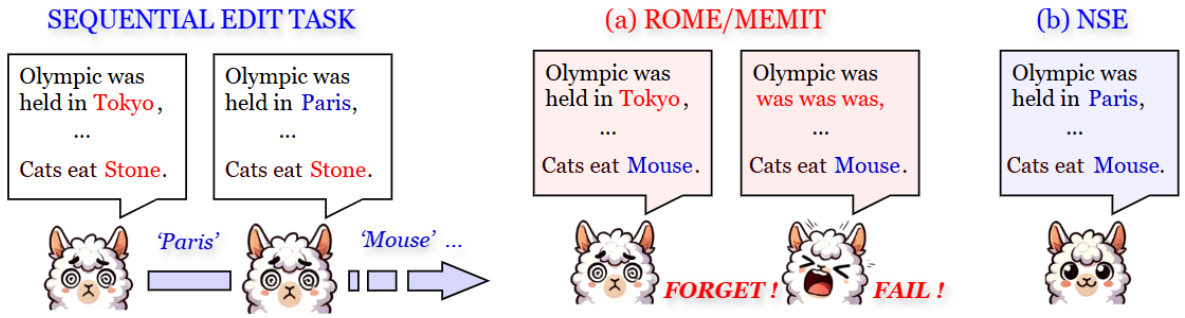

Figure 1: Example of sequential editing. (a) shows model forgetting and model failure issues in sequential editing using ROME/MEMIT, while (b) shows the accurate editing capabilities of our method without such issues.

To tackle these challenges, we introduce a new model editing method, termed **N**euron-level **S**equential **E**diting (**NSE**). Specifically, to address the model failure, NSE uses weights rewinding for value computation by preserving the model's original weights as a significant reference when manipulating the hidden states of the crucial layers. This process can effectively mitigate the impairment of previous knowledge accumulated over various edits. Furthermore, to address model forgetting, NSE selectively collects "influential neurons" to update weights by sorting neuron activation within the crucial layers, rather than updating all weights in the critical layers as in previous work (Meng et al., 2022, 2023). This selective modification maximizes the protection of model functionality from degradation. Additionally, for large-scale LLMs containing numerous neurons, an iterative multi-layer editing is introduced to streamline the neuron selection process, enabling NSE to achieve massive knowledge updates effectively in a single editing.

Through theoretical analysis and extensive experiments conducted on GPT2-XL (1.5B) (Radford et al., 2019), GPT-J (6B) (Wang and Komatsuzaki, 2021), and Llama3 (8B), we validate the effectiveness and efficiency of our NSE. Compared to current model editing methods (*e.g.*, MEND (Mitchell et al., 2022a), ROME (Meng et al., 2022), MEMIT (Meng et al., 2023) and GRACE (Hartvigsen et al., 2023)), NSE shows substantial improvements *with respect to* five commonly used metrics such as specificity and consistency.

2 Preliminary

2.1 Autoregressive Language Model

An autoregressive language model predicts the next token in a sequence based on the tokens that have come before it. Given a L layer transformer model and an input sequence $x = (x_0, x_1, \ldots, x_T)$, the model aims to predict the next token in the sequence. The probability of the next token x_{t+1} is given by:

$$\mathbb{P}(x_{t+1} \mid x_0, x_1, \ldots, x_t) = \text{Softmax}(\mathbf{W}_e \mathbf{h}_t^N), \quad (1)$$

where $\mathbf{W}_e$ represents embedding matrix, and $\mathbf{h}_t^N$ represents the final hidden state at the topmost layer N. The hidden state $\mathbf{h}_t^l$ at layer l is calculated as:

$$\begin{aligned} \boldsymbol{h}_t^l(x) &= \boldsymbol{h}_t^{l-1}(x) + \boldsymbol{a}_t^l(x) + \boldsymbol{v}_t^l(x), \\ \boldsymbol{a}_t^l &= \text{attn}^l(\boldsymbol{h}_0^{l-1}, \boldsymbol{h}_1^{l-1}, \ldots, \boldsymbol{h}_t^{l-1}), \\ \boldsymbol{v}_t^l &= \boldsymbol{W}_{\text{out}}^l \sigma(\boldsymbol{W}_{\text{in}}^l \gamma(\boldsymbol{h}_t^{l-1} + \boldsymbol{a}_t^l)), \end{aligned} \quad (2)$$

where $\mathbf{a}_t^l$ represents the output of the attention block and $\boldsymbol{v}_t^l$ represents the output of the FFN layer. $\boldsymbol{W}_{\text{in}}^l$ and $\boldsymbol{W}_{\text{out}}^l$ are weight matrices, σ is a non-linear activation function, and γ represents the layer norm.

2.2 Previous Model Editing Method

Sequential model editing refines a pre-trained model $f_{\boldsymbol{\theta}_0}$ through multiple updates, each incorporating new facts (s, r, o), such as s = "The latest Olympic", r = "was held in", o = "Paris". After each edit, the updated model $f_{\boldsymbol{\theta}_t}$ is optimized to produce the correct outputs for the current edit set $\mathbb{D}_{edit_t}$ while maintaining the precision of previous tasks, ensuring both adaptation to new information and preservation of previous performance.

Following current work (Meng et al., 2022, 2023), we treat the weights of the Transformer's (Vaswani et al., 2017) FFN layer as a linear associative memory. That is, linear operations within the FFN layer can be viewed as key-value storage for information retrieval (Kohonen, 1972; Anderson, 1972).

Given the weights $\boldsymbol{W}_{\text{in}}^l$ of the l-th FFN layer when prompted by (s_i, r_i), we identify the activation output of the last subject token S as the key $\boldsymbol{k}_i^l$. In the following, this key $\boldsymbol{k}_i^l$ is processed through the output weights $\boldsymbol{W}_{\text{out}}^l$, producing the value $\boldsymbol{v}_i^l$. In the context of sequential model editing, we start with an initial set of key-value associations for knowledge facts stored in the l-th FFN layer, denoted respectively by $\boldsymbol{K}_0 = \{\boldsymbol{k}_i\}_{i=1}^n$ and $\boldsymbol{V}_0 = \{\boldsymbol{v}_i\}_{i=1}^n$. In the following, our objective is to introduce new key value associations m, denoted as $\boldsymbol{K}_1 = \{\boldsymbol{k}_i\}_{i=n+1}^{n+m}$ and $\boldsymbol{V}_1 = \{\boldsymbol{v}_i\}_{i=n+1}^{n+m}$, while retaining all existing associations unchanged. Drawing on prior work (Meng et al., 2023), the optimization objective is as follows:

$$\boldsymbol{\Delta}^* = \arg\min_{\boldsymbol{\Delta}} (\|(\boldsymbol{W} + \boldsymbol{\Delta})\boldsymbol{K}_1 - \boldsymbol{V}_1\|^2 + \|(\boldsymbol{W} + \boldsymbol{\Delta})\boldsymbol{K}_0 - \boldsymbol{V}_0\|^2), \quad (3)$$

where $\boldsymbol{W}$ represents the weights of $\boldsymbol{W}_{\text{out}}$ in the target FFN layer, Δ denotes the update to W, and V_1 can be directly trained using the fine-tuning loss predicted by the model through backpropagation. Since $\boldsymbol{K}_0$ and $\boldsymbol{V}_0$ represent the knowledge retained in LLM, we can express this relationship as $\boldsymbol{W}\boldsymbol{K}_0 = \boldsymbol{V}_0$. Therefore, we can obtain the closed-form solution of Eqn. 3 using the least squares method (Lang, 2012):

$$\boldsymbol{\Delta}^* = \boldsymbol{R}\boldsymbol{K}_1^T \left(\boldsymbol{K}_0\boldsymbol{K}_0^T + \boldsymbol{K}_1\boldsymbol{K}_1^T\right)^{-1}, \quad (4)$$

where $\boldsymbol{R} = \boldsymbol{V}_1 - \boldsymbol{W}\boldsymbol{K}_1$. Furthermore, MEMIT focuses on editing a specific set of layers denoted as $\mathcal{R} = \{l^* - |\mathcal{R}| + 1, \ldots, l^*\}$. The required the weight update $\boldsymbol{\Delta}^l$ for layer $l \in \mathcal{R}$ is expressed as:

$$\boldsymbol{\Delta}^l = \boldsymbol{R}^l {\boldsymbol{K}_1^l}^T \left(\boldsymbol{K}_0^l {\boldsymbol{K}_0^l}^T + \boldsymbol{K}_1^l {\boldsymbol{K}_1^l}^T\right)^{-1}, \quad (5)$$

where $\boldsymbol{R}^l = \frac{\boldsymbol{R}^{l^*}}{l^* - l + 1}$. These modifications are implemented sequentially, starting from the lower layers and progressing to the upper layers.

3 Methodology

In this section, we introduce NSE, a method adapted for sequential model editing, as illustrated in Figure 2. Initially, we identify the values computation method in Section 3.1. Subsequently, in Section 3.2, we detail the editing method that selectively filters certain neurons for the corresponding parameter updates. Finally, in Section 3.3, we introduce the approach of iterative multi-layer editing.

3.1 Weights Rewinding for Value Computation

First, our primary goal is to find a hidden vector that encodes the new association (s_i, r_i, o_i) and replaces the value $\boldsymbol{v}_i^l$ in the l-th layer as described in Section 2.2. In practical implementation, as shown in Figure 2 (a), we optimize $\boldsymbol{\delta}_i$ through gradient descent by maximizing the probability that the model outputs o_i to compute $\boldsymbol{z}_i = \boldsymbol{h}_i^l + \boldsymbol{\delta}_i$, where $\boldsymbol{h}_i^l$ denotes the hidden state of the LLM in layer l. And the value ${\boldsymbol{v}_i}^l$ can be calculated as ${\boldsymbol{v}_i}^l \mathrel{+}= \boldsymbol{\delta}_i$. In the process of sequential editing, we observed that using the updated model parameters f_{θ_t} to calculate $\boldsymbol{z}_i$ after each edit leads to significant model degradation over multiple edit rounds. This indicates that the cumulative parameter updates from each editing round can lead to a change in value computation. In contrast, using the original model parameters $f_{\boldsymbol{\theta}_0}$ to compute $\boldsymbol{z}_i$ effectively prevents this issue. Hence, we propose a weight rewinding method for the value computation, which is based on the initial model weights $f_{\boldsymbol{\theta}_0}$ to ensure that $\boldsymbol{z}_i$ is computed using $f_{\boldsymbol{\theta}_0}$ for each edit. The optimization objective is as follows:

$$\boldsymbol{\delta}_i^* = \arg\min_{\boldsymbol{\delta}_i} -\log \mathbb{P}_{f_{\boldsymbol{\theta}_0}(\boldsymbol{h}_i^l + \boldsymbol{\delta}_i)} \left[o_i \mid (s_i, r_i)\right], \quad \boldsymbol{z}_i = \boldsymbol{h}_i^l + \boldsymbol{\delta}_i^*, \quad (6)$$

where $f_{\boldsymbol{\theta}_0}(\boldsymbol{h}_i^l \mathrel{+}= \boldsymbol{\delta}_i)$ represents the original model with $\boldsymbol{h}_i^l$ updated to $\boldsymbol{h}_i^l + \boldsymbol{\delta}_i$. Subsequently, the value ${\boldsymbol{v}_i}^l$ can be updated to ${\boldsymbol{v}_i}^l \mathrel{+}= \boldsymbol{z}_i - \boldsymbol{h}_i^l$, which will be used to support subsequent model edits. Note that when calculating $\boldsymbol{z}_i$ using the original model parameters $f_{\boldsymbol{\theta}_0}$, it is sufficient to save only the weight matrix $\boldsymbol{W}_{\text{out}}$ that needs to be updated, rather than storing the entire model parameters and we use the original weights only during the computation of $\boldsymbol{z}_i$.

3.2 Neuron-level Weights Updating

In Section 3.1, we primarily discussed the method to calculate z_i to replace the value in the target layer. In this section, we will specifically elaborate on how to utilize the computed value for weight updates at the neuron level, as shown in Figure 2 (b).

Based on previous work, it is established that neurons in FFN contain abundant information (Dai et al., 2022; Wang et al., 2022; Schwettmann et al., 2023a; Pan et al., 2023; Wu et al., 2023). Hence, we selectively optimize a subset of neurons rather

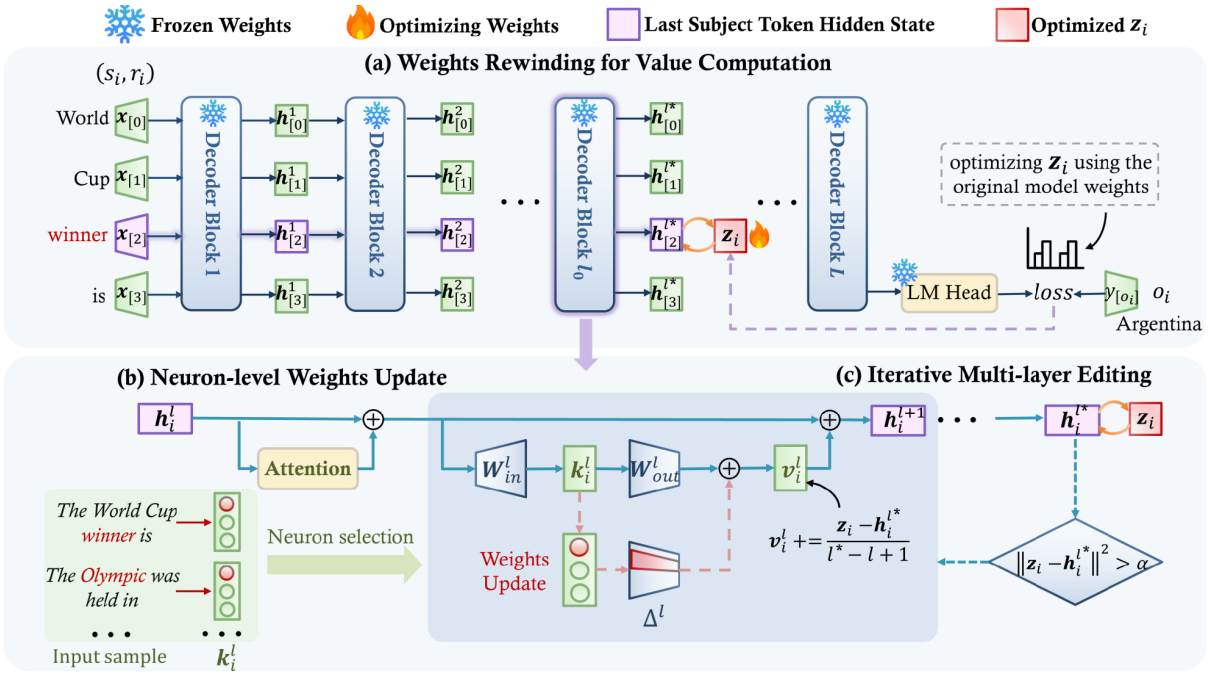

Figure 2: Overview of sequential model editing with NSE. (a) describes the process of weights rewinding for value computation. (b) illustrates the neuron selection and neuron-level weights update. (c) shows the process of iterative multi-layer editing.

than alter the entire weight matrix for each edit. Specifically, for a given knowledge fact (s_i, r_i, o_i), we use the activation values $\boldsymbol{k}_i$ of the neurons and compute scores $\mathbf{Q}_i = |\boldsymbol{k}_i|$. Neurons are ranked based on these scores, and a subset is chosen such that the cumulative score of selected neurons surpasses a predetermined percentage p of the total score:

$$\mathcal{I} = \underset{\mathcal{I}\subseteq\{1,\ldots,N\}}{\arg\min} |\mathcal{I}| \quad \text{s.t.} \quad \sum_{j\in\mathcal{I}} \boldsymbol{Q}_{ij} \geq p \times \sum_{j=1}^{N} \boldsymbol{Q}_{ij}, \tag{7}$$

where $\boldsymbol{Q}_{ij}$ represents the score of the j-th neuron, $\mathcal{I}$ is the set of indices for selected neurons. Hereafter, we introduce how to update $\boldsymbol{W}$ (*i.e.*, $\boldsymbol{W}_{\text{out}}$) using the selected neuron set $\mathcal{I}$. For clarity, we decompose $\boldsymbol{W}$ into two submatrices along the row dimension: $\hat{\boldsymbol{W}}$ (rows indexed by $\mathcal{I}$) and $\tilde{\boldsymbol{W}}$ (remaining rows), with corresponding decomposition of the knowledge matrix $\boldsymbol{K} = [\hat{\boldsymbol{K}}, \tilde{\boldsymbol{K}}]$ along the column dimension. Let $\hat{\boldsymbol{\Delta}}$ denote the update submatrix for $\hat{\boldsymbol{W}}$. Our objective is to modify only $\hat{\boldsymbol{W}}$ while keeping $\tilde{\boldsymbol{W}}$ unchanged, which aligns with the optimization goal in Eqn. 3. The original output can be expressed as:

$$\boldsymbol{V} = \hat{\boldsymbol{K}}\hat{\boldsymbol{W}} + \tilde{\boldsymbol{K}}\tilde{\boldsymbol{W}}. \tag{8}$$

Since $\tilde{\boldsymbol{K}}\tilde{\boldsymbol{W}}$ remains fixed during editing, we reformulate Eqn. 3 by focusing on the editable component:

$$\hat{\boldsymbol{\Delta}}^* = \underset{\hat{\boldsymbol{\Delta}}}{\arg\min}\Big(\Big\|(\hat{\boldsymbol{W}} + \hat{\boldsymbol{\Delta}})\hat{\boldsymbol{K}}_1 - \boldsymbol{V}_1\Big\|^2 + \Big\|(\hat{\boldsymbol{W}} + \hat{\boldsymbol{\Delta}})\hat{\boldsymbol{K}}_0 - \boldsymbol{V}_0\Big\|^2\Big), \tag{9}$$

where $\hat{\boldsymbol{K}}_0$ and $\hat{\boldsymbol{K}}_1$ are submatrices of $\boldsymbol{K}_0$ and $\boldsymbol{K}_1$ indexed by $\mathcal{I}$. This reformulation explicitly decouples the editable ($\hat{\boldsymbol{W}}$) and fixed ($\tilde{\boldsymbol{W}}$) components. The optimal solution is:

$$\hat{\boldsymbol{\Delta}}^* = \hat{\boldsymbol{R}}\hat{\boldsymbol{K}}_1^\top \hat{\boldsymbol{C}}^{-1}, \tag{10}$$

where $\hat{\boldsymbol{R}} = \boldsymbol{V}_1 - \hat{\boldsymbol{W}}\hat{\boldsymbol{K}}_1$ and $\hat{\boldsymbol{C}} = \hat{\boldsymbol{K}}_0\hat{\boldsymbol{K}}_0^\top + \hat{\boldsymbol{K}}_1\hat{\boldsymbol{K}}_1^\top$. Following (Meng et al., 2023), we approximate $\boldsymbol{K}_0\boldsymbol{K}_0^\top$ as $\lambda\mathbb{E}[\boldsymbol{k}\boldsymbol{k}^\top]$ with hyperparameter λ. The submatrix $\hat{\boldsymbol{K}}_0\hat{\boldsymbol{K}}_0^\top$ is obtained by selecting rows/columns through $\mathcal{I}$ from this approximation. During continual editing, newly acquired knowledge is incrementally added to $\boldsymbol{K}_0\boldsymbol{K}_0^\top$.

3.3 Iterative multi-layer Editing

As described in Section 2.2, current work (Meng et al., 2023) propagates edits through layers by computing the value $\boldsymbol{v}_i^l$ as $\boldsymbol{v}_i^l += \frac{\boldsymbol{\delta}_i}{l^*-l+1}$ $(l \in \mathcal{R})$ (Meng et al., 2023; Gupta et al., 2024b).Here, $\boldsymbol{\delta}_i = \boldsymbol{z}_i - \boldsymbol{h}_i^{l^*}$ represents the residual difference.

The fundamental purpose is that, as each layer is updated, the hidden state $\boldsymbol{h}_i^{l^*}$ progressively approaches the target $\boldsymbol{z}_i$, thus diminishing the residual $\boldsymbol{\delta}_i$. Detailed analyses can be found in the Appendix F. However, due to errors in the fitting process, some knowledge proves difficult to edit, resulting in $\boldsymbol{v}_i^{l^*}$ not adequately approximating $\boldsymbol{z}_i$, and consequently leading to editing failures.

Therefore, we propose iterative multi-layer editing to refine the multi-layer editing approach in MEMIT by iteratively selecting neurons to edit multiple layers, as depicted in Figure 2 (c). Specifically, considering that some knowledge is difficult to edit such that the corresponding value $\boldsymbol{v}_i^{l^*}$ cannot sufficiently approximate the optimized target value $\boldsymbol{z}_i$, we employ iterative multi-layer editing. After each round of multi-layer editing, we filter the knowledge samples in the current batch based on $\|\boldsymbol{z}_i - \boldsymbol{h}_i^{l^*}\|^2$. If $\|\boldsymbol{z}_i - \boldsymbol{h}_i^{l^*}\|^2 < \alpha$, the knowledge sample is considered edited successfully. In contrast, if $\|\boldsymbol{z}_i - \boldsymbol{h}_i^{l^*}\|^2 > \alpha$, the sample is deemed not yet successfully edited. The hyperparameter α, which is set differently for various LLMs, determines the threshold for editing success. These unedited samples are then filtered to form a new batch for further multi-layer editing, repeating this process until all knowledge samples in the batch meet $\|\boldsymbol{z}_i - \boldsymbol{h}_i^{l^*}\|^2 < \alpha$ or the iteration limit is reached. More details of the NSE algorithm are provided in Appendix A.

4 Experiments

We conducted experiments to demonstrate the effectiveness of our model editing method. The experiments aim to address the following research questions:

- **RQ1:** How does NSE perform on sequential editing tasks compared to existing methods?
- **RQ2:** What is the impact of adjusting the batch size of edits on the performance of NSE?
- **RQ3:** Can the LLM, after undergoing NSE editing, retain its original general capabilities, and how does it perform on general capability tests?

4.1 Experimental Settings

Datasets & Evaluation Metrics. To evaluate the effectiveness of our method, we use two datasets: Counterfact (Meng et al., 2022) and ZsRE (Levy et al., 2017). For the Counterfact dataset, we employ five evaluation metrics as defined in previous work (Meng et al., 2022, 2023): **Efficacy** (efficiency success), **Generalization** (paraphrase success), **Specificity** (neighborhood success), **Fluency** (generation entropy), and **Consistency** (reference score). For the ZsRE dataset, we use three evaluation metrics also defined in previous work (Mitchell et al., 2022a; Meng et al., 2022, 2023): **Efficacy**, **Generalization**, and **Specificity**. For more details, see Appendix C.

Models & Baselines. Our comparative analysis evaluates the performance of various editing methods in three LLMs, GPT2-XL (1.5B) (Radford et al., 2019), GPT-J (6B) (Wang and Komatsuzaki, 2021) and Llama3 (8B). For baseline comparisons, we primarily select model editing methods that modify the model's parameters, including fine-tuning the specific layer (FT-L) (Zhu et al., 2020), MEND (Mitchell et al., 2022a), ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), MALMEN (Tan et al., 2024) and DAFNeT (Zhang et al., 2024b). Furthermore, we incorporated memory-based editing methods, GRACE (Hartvigsen et al., 2023), as a baseline. More details are provided in Appendix B.

4.2 Performance Comparison (RQ1)

In this subsection, we provide a comprehensive comparison of NSE with existing methods on the sequential model editing task. The experiments are conducted with a total of 2000 edited samples and an editing batch size of 100 (batch size refers to the number of samples edited simultaneously in each editing round during the sequential editing process)s. The results of all the evaluation methods, on all datasets, are presented in Table 1. Furthermore, we test the methods on the edited samples after each edit round on the GPT2-XL model using Counterfact. We presented the result in Figure 3. We also provide additional experimental results in the Appendix G. According to these, we can find that:

- **Observation 1: NSE outperforms other baseline methods in almost all critical metrics in both datasets and models in the sequential editing task.** Specifically, compared to baseline methods for parameter modification, NSE shows significant improvements in all metrics. In particular, on Llama3 (8B) editing, NSE achieves an average enhancement of around 30.33% on multiple metrics. Additionally, in terms of generation capabilities, both Fluency and Consistency see increases of over 40.75%. In contrast, while the GRACE parameter preservation method re-

Table 1: Comparison of NSE with existing methods on the sequential model editing task. *Eff.*, *Gen.*, *Spe.*, *Flu.* and *Consis.* denote Efficacy, Generalization, Specificity, Fluency and Consistency, respectively.

Method	Model	Counterfact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
Pre-edited		7.85±0.26	10.58±0.26	89.48±0.18	635.23±0.11	24.14±0.08	36.99±0.30	36.34±0.30	31.89±0.22
FT-L	Llama3	83.33±0.37	67.79±0.40	46.63±0.37	233.72±0.22	8.77±0.05	30.48±0.26	30.22±0.32	15.49±0.17
MEND	Llama3	63.24±0.31	61.17±0.36	45.37±0.38	372.16±0.80	4.21±0.05	0.91±0.05	1.09±0.05	0.53±0.02
ROME	Llama3	64.40±0.47	61.42±0.42	49.44±0.38	449.06±0.26	3.31±0.02	2.01±0.07	1.80±0.07	0.69±0.03
MEMIT	Llama3	65.65±0.47	64.65±0.42	51.56±0.38	437.43±1.67	6.58±0.11	34.62±0.36	31.28±0.34	18.49±0.19
MALMEN	Llama3	69.58±0.24	66.18±0.35	49.14±0.37	463.85±0.78	9.28±0.04	9.58±0.04	9.36±0.08	2.01±0.05
DAFNeT	Llama3	78.21±0.56	69.05±0.39	68.90±0.21	584.72±0.34	20.77±0.13	57.75±0.24	46.96±0.35	23.17±0.15
GRACE	Llama3	90.72±0.13	0.09±0.01	87.23±0.21	632.43±0.63	23.79±0.23	**74.58±0.31**	1.03±0.06	31.86±0.12
NSE	Llama3	**96.14±0.19**	**78.42±0.35**	**87.66±0.19**	**632.72±0.12**	**30.20±0.10**	62.29±0.35	**47.13±0.31**	**32.32±0.22**
Pre-edited		22.23±0.73	24.34±0.62	78.53±0.33	626.64±0.31	31.88±0.20	22.19±0.24	31.30±0.27	24.15±0.32
FT-L	GPT2-XL	63.55±0.48	42.20±0.41	57.06±0.30	519.35±0.27	10.56±0.05	37.11±0.39	33.30±0.37	10.36±0.17
MEND	GPT2-XL	50.80±0.50	50.80±0.48	49.20±0.51	407.21±0.08	1.01±0.00	0.00±0.00	0.00±0.00	0.00±0.00
ROME	GPT2-XL	54.60±0.48	51.18±0.40	52.68±0.33	366.13±1.40	0.72±0.02	47.50±0.43	43.56±0.42	14.27±0.19
MEMIT	GPT2-XL	94.70±0.22	85.82±0.28	60.50±0.32	477.26±0.54	22.72±0.15	79.17±0.32	71.44±0.36	26.12±0.25
MALMEN	GPT2-XL	55.32±0.58	53.63±0.42	53.25±0.62	412.57±0.15	1.08±0.03	3.54±0.03	4.25±0.02	3.23±0.04
DAFNeT	GPT2-XL	74.25±0.15	59.18±0.32	63.06±0.37	594.35±0.33	28.35±0.12	79.17±0.67	66.57±0.72	22.67±0.22
GRACE	GPT2-XL	94.50±0.24	0.04±0.01	**78.13±0.43**	622.56±0.79	31.55±0.25	82.54±0.21	0.40±0.02	24.78±0.21
NSE	GPT2-XL	**96.80±0.20**	**87.72±0.30**	72.10±0.28	**622.85±0.15**	**40.04±0.11**	**83.26±0.29**	**75.33±0.34**	**26.14±0.25**
Pre-edited		16.22±0.31	18.56±0.45	83.11±0.13	621.81±0.67	29.74±0.51	26.32±037	25.79±0.25	27.42±0.53
FT-L	GPT-J	92.15±0.27	72.38±0.38	43.35±0.37	297.92±0.77	6.65±0.10	72.37±0.29	68.91±0.32	19.66±0.23
MEND	GPT-J	46.15±0.50	46.22±0.51	53.90±0.48	242.41±0.41	3.94±0.03	0.71±0.04	0.71±0.04	0.52±0.03
ROME	GPT-J	57.50±0.48	54.20±0.40	52.05±0.31	589.28±0.08	3.22±0.02	56.42±0.42	54.65±0.42	9.86±0.16
MEMIT	GPT-J	98.55±0.11	**95.50±0.16**	63.64±0.31	546.28±0.88	34.89±0.15	94.91±0.16	90.22±0.23	27.56±0.27
MALMEN	GPT-J	69.58±0.24	66.18±0.35	49.14±0.37	463.85±0.78	9.28±0.04	9.58±0.04	9.36±0.08	2.01±0.05
DAFNeT	GPT-J	78.21±0.56	69.05±0.39	68.90±0.21	584.72±0.34	20.77±0.13	57.75±0.24	46.96±0.35	23.17±0.15
GRACE	GPT-J	95.88±0.28	0.05±0.01	**82.11±0.24**	620.21±0.49	28.53±0.15	94.33±0.37	1.59±0.03	27.63±0.43
NSE	GPT-J	**99.55±0.06**	91.92±0.22	78.96±0.25	**620.49±0.16**	**40.24±0.12**	**96.87±0.14**	**91.33±0.22**	**28.66±0.25**

tains model capabilities to the greatest extent, it exhibits weaker generalization performance.

- **Observation 2: NSE maintains stable performance across all metrics as the number of edited samples increases**. As shown in Figure 3, NSE shows robust performance, remains resilient to model failure, and forgets despite the increase in the number of editing rounds. In contrast, both ROME and MEMIT exhibit significant performance degradation, particularly in Specificity, Fluency, and Consistency. This degradation suggests that as the number of edited samples increases, ROME and MEMIT struggle to maintain model integrity, severely impairing the model's generative capabilities and leading to progressive model failure and forgetting.

4.3 Impact of Batch Size (RQ2)

To answer RQ2, we examine the impact of different batch sizes on the performance of NSE compared to MEMIT, in the sequential model editing task, with a total of 2000 edits on the Counterfact dataset. Figure 4 presents four radar charts that depict Llama3 performance (8B) when using batch sizes of 200, 100, 50, and 10, respectively. We also tested the editing effects of GPT2-XL and GPT-J at different batch sizes, as detailed in Appendix H. According to Figure 4, we can find that:

- **Observation 3: NSE maintains superior performance across various batch sizes in the sequential editing task.** Specifically, as the batch size decreases, resulting in an increase in the

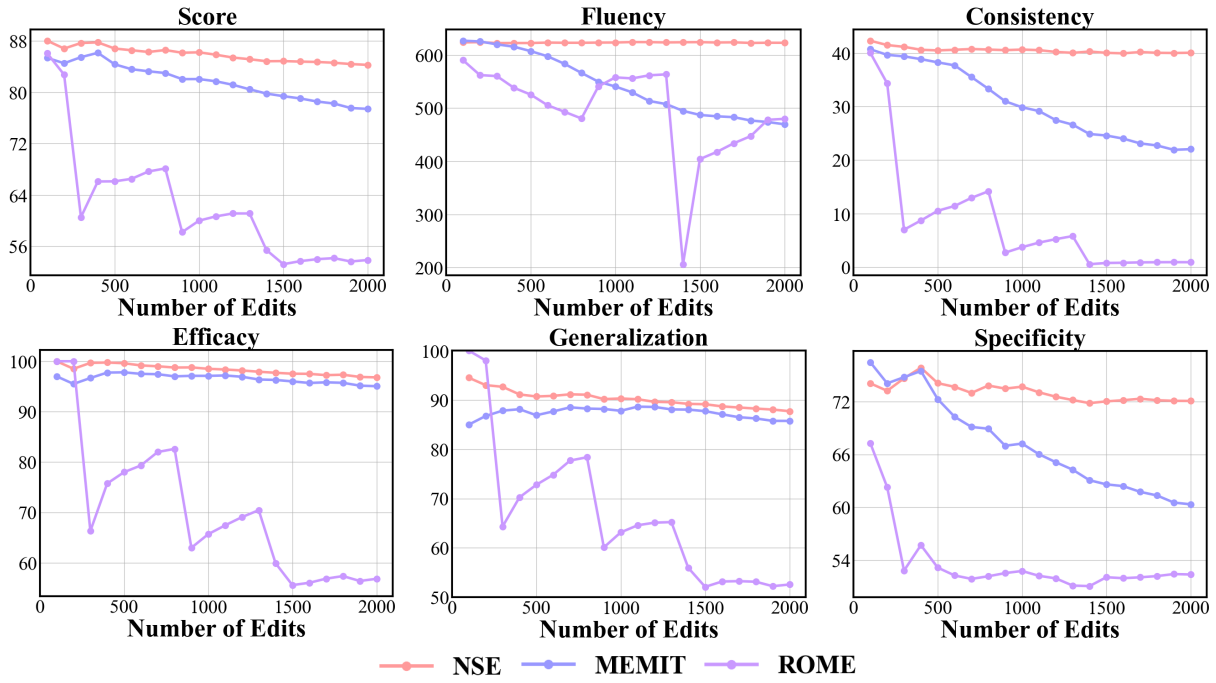

Figure 3: Editing performance of NSE and baselines with varying numbers of edits (batch size 100) in sequential editing, evaluated on the Counterfact dataset. Score is the harmonic mean of Efficacy, Generalization, and Specificity.

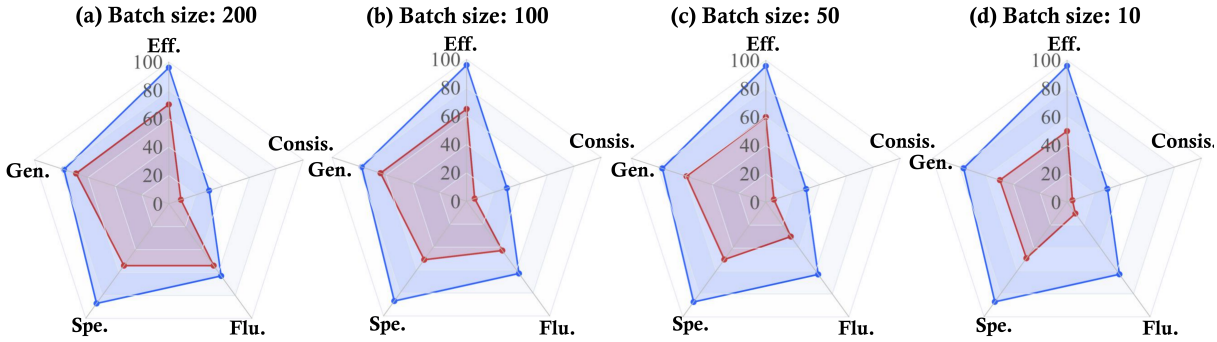

Figure 4: Editing performance of NSE and MEMIT with different batch size, evaluated on Llama3 (8B). The red line and the blue line represent MEMIT and NSE, respectively.

number of editing rounds, the performance of MEMIT deteriorates. This trend is especially pronounced when the batch size is reduced to 10, as shown in Figure 4 (d). The radar charts reveal a significant decline in model editing effectiveness, with notable decreases observed in all metrics. In contrast, NSE demonstrates an average improvement in all of these metrics.

4.4 General Ability Test (RQ3)

To assess the effects of model editing on the general capabilities of edited model, we have selected six natural language tasks from the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019). Details of the downstream tasks are described in the Appendix C.2.

We perform evaluations on Llama3 (8B) based on sequential editing settings with 3000 edits. The results are shown in Figure 5. Note that as the parameter-preserved editing method differs from ours and does not affect the model by changing it, it will not be compared with it in the general ability test. Here we can make the following observations:

- **Observation 4: NSE consistently maintains the general capabilities of the LLM during sequential editing without incurring model failure.** Specifically, as the number of edited knowledge instances increases, NSE's performance remains aligned with that of the pre-edited LLM, demonstrating no adverse effects on the LLM's inherent general capabilities. In contrast, ROME and MEMIT exhibit a significant decline in gen-

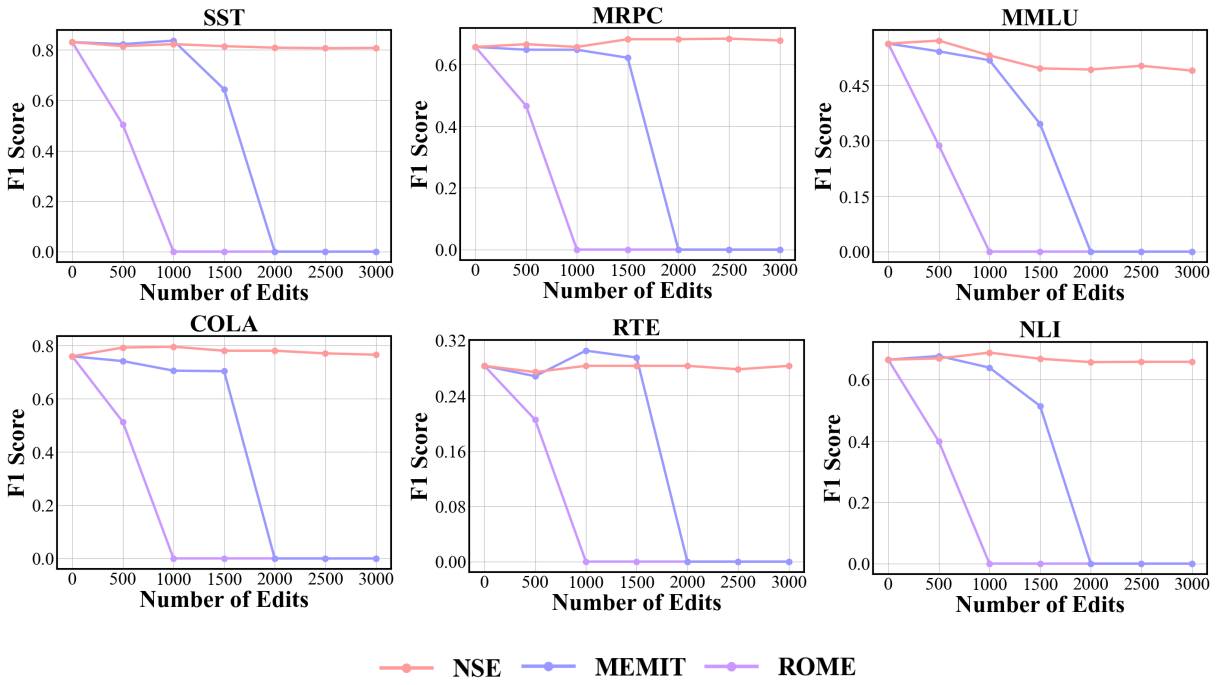

Figure 5: Performance on general tasks of edited models using NSE, ROME and MEMIT, with sequential editting on Llama3 (8B).

eral capabilities 0 after editing approximately 1,000 to 2,000 samples, indicating that the model has already experienced degradation.

5 Related work

Current approaches to model editing in LLMs generally fall into two main categories (Yao et al., 2023; Zhang et al., 2024a; Bi et al., 2024a):

Preserve Models' Parameters. Methods that preserve the original model's parameters generally store edit examples in memory and use them to guide the model's predictions (Bi et al., 2024b). For example, SERAC (Mitchell et al., 2022b) keeps the original model unchanged and uses a separate counterfactual model for edits. T-Patcher (Huang et al., 2023) introduces an additional neuron for each output error, whereas CaliNet (Dong et al., 2022) incorporates knowledge using a predetermined number of neurons. GRACE (Hartvigsen et al., 2023) implements sequential editing by maintaining a dynamically updated codebook. WISE (Wang et al., 2024) introduces a dual parametric memory scheme that separates pretrained and edited knowledge.

Modify Models' Parameters. Methods that modify LLM parameters require updates to the model's internal parameters with each edit. FT-W fine-tunes specific layers with regularization constraints (Zhu et al., 2020). KN (Dai et al., 2022) identifies and updates key "knowledge neurons" in the FFN. KE (Cao et al., 2021) and MEND (Mitchell et al., 2022a) use hypernetworks for predicting weight changes based on meta-learning. ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) perform large-scale edits by locating and modifying knowledge in specific GPT layers. MALMEM (Tan et al., 2024) optimizes weight updates using least-squares, allowing simultaneous editing of multiple samples. DAFNeT (Zhang et al., 2024b) extends MALMEM by incorporating autoregressive self-attention for improved sequential editing. This paper primarily focuses on parameter-modification editing methods.

6 Conclusion

In this work, we introduce NSE, a model editing method for sequential model editing tailored to address the significant challenges of model forgetting and model failure. Specifically, we propose weight rewinding for value computation by optimizing the hidden states of the target layer using the model's original weights, which effectively minimize cumulative changes and maintain model coherence. Additionally, we select influential neurons for different knowledge to update weights in FFN and iteratively edit multi-layer weights.

7 Limitations

Despite the outstanding performance of NSE in sequential editing, our investigation reveals some limitations. Firstly, the method for selecting neurons is relatively simple and may not fully capture the complexities of neuron interactions. Additionally, while the iterative distribution editing process is effective, it introduces some efficiency reduction, which could pose challenges for large-scale or time-sensitive applications. Moving forward, our goal is to explore more effective neuron attribution methods and enhance the efficiency of our editing techniques.

References

James A Anderson. 1972. A simple neural network generating an interactive memory. *Mathematical biosciences*, 14(3-4):197–220.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *TAC*. NIST.

Baolong Bi, Shenghua Liu, Yiwei Wang, Lingrui Mei, and Xueqi Cheng. 2024a. Is factuality decoding a free lunch for llms? evaluation on knowledge editing benchmark. *Preprint*, arXiv:2404.00216.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *EMNLP (1)*, pages 6491–6506. Association for Computational Linguistics.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *ACL (1)*, pages 8493–8502. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*. Asian Federation of Natural Language Processing.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. In *EMNLP (Findings)*, pages 5937–5947. Association for Computational Linguistics.

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*.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. 2024a. Model editing at scale leads to gradual and catastrophic forgetting. *CoRR*, abs/2401.07453.

Akshat Gupta, Dev Sajnani, and Gopala Anumanchipalli. 2024b. A unified framework for model editing. *CoRR*, abs/2403.14236.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: lifelong model editing with discrete key-value adaptors. In *NeurIPS*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *ICLR*. OpenReview.net.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. In *ICLR*. OpenReview.net.

Teuvo Kohonen. 1972. Correlation matrix memories. *IEEE Trans. Computers*, 21(4):353–359.

Serge Lang. 2012. *Introduction to linear algebra*. Springer Science & Business Media.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL*, pages 333–342. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *NeurIPS*.

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. In *ICLR*. OpenReview.net.

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

Haowen Pan, Yixin Cao, Xiaozhi Wang, and Xun Yang. 2023. Finding and editing multi-modal neurons in pre-trained transformer. *CoRR*, abs/2311.07470.

Haowen Pan, Yixin Cao, Xiaozhi Wang, Xun Yang, and Meng Wang. 2024. Finding and editing multi-modal neurons in pre-trained transformers. In *ACL (Findings)*, pages 1012–1037. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *EMNLP/IJCNLP (1)*, pages 2463–2473. Association for Computational Linguistics.

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 *EMNLP (1)*, pages 5418–5426. Association for Computational Linguistics.

Sarah Schwettmann, Neil Chowdhury, Samuel Klein, David Bau, and Antonio Torralba. 2023a. Multimodal neurons in pretrained text-only transformers. In *ICCV (Workshops)*, pages 2854–2859. IEEE.

Sarah Schwettmann, Neil Chowdhury, and Antonio Torralba. 2023b. Multimodal neurons in pretrained text-only transformers. *CoRR*, abs/2308.01544.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642. ACL.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language models via meta learning. In *ICLR*. OpenReview.net.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR (Poster)*. OpenReview.net.

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

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. WISE: rethinking the knowledge memory for lifelong model editing of large language models. *CoRR*, abs/2405.14768.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. 2022. Finding skill neurons in pre-trained transformer-based language models. In *EMNLP*, pages 11132–11152. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. 2023. DEPN: detecting and editing privacy neurons in pretrained language models. In *EMNLP*, pages 2875–2886. Association for Computational Linguistics.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *EMNLP*, pages 10222–10240. Association for Computational Linguistics.

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. 2024a. A comprehensive study of knowledge editing for large language models. *CoRR*, abs/2401.01286.

Taolin Zhang, Qizhou Chen, Dongyang Li, Chengyu Wang, Xiaofeng He, Longtao Huang, Hui Xue', and Jun Huang. 2024b. Dafnet: Dynamic auxiliary fusion for sequential model editing in large language models. In *ACL (Findings)*, pages 1588–1602. Association for Computational Linguistics.

Tianyu Zhang, Junfeng Fang, Houcheng Jiang, Baolong Bi, Xiang Wang, and Xiangnan He. Explainable and efficient editing for large language models. In *THE WEB CONFERENCE 2025*.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix X. Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *CoRR*, abs/2012.00363.

A Algorithm

We present the NSE algorithm in sequential model editing task. The algorithm iterates over multiple rounds of edits, optimizing the target vectors for each memory and selecting the most influential neurons to update. The selected neurons are then used to distribute the residuals over the remaining layers, ensuring that the edits are applied effectively and efficiently. Detailed steps are provided in Algorithm 1.

B Baselines

The details of baselines are as follow:

- **FT-L** (Zhu et al., 2020) focuses on adjusting a specific layer identified by ROME (Meng et al., 2022), rather than fine-tuning all layers. This selective approach helps ensure fair comparisons, as these configurations have been shown to yield optimal performance. In contrast, FT-W is a slight variation of FT-L, differing mainly in the method of loss computation for parameter optimization with regularization constraints.
- **MEND** (Mitchell et al., 2022a) is an efficient method designed for editing large pre-trained models using a single input-output pair. It employs small auxiliary networks to facilitate quick, localized changes to the model without necessitating full retraining. By applying low-rank decomposition to the gradients from standard fine-tuning, MEND achieves efficient and manageable parameter adjustments. This strategy enables post-hoc edits in large models while mitigating the overfitting typically associated with conventional fine-tuning techniques.
- **ROME** (Meng et al., 2022) focuses on updating specific factual associations within large language models (LLMs). Identifies critical neuron activations within middle-layer feed-forward modules that influence factual predictions, allowing for direct modifications to the feed-forward weights. ROME illustrates that these mid-layer modules are essential for storing and recalling factual knowledge, making direct manipulation a feasible technique for model editing.
- **MEMIT** (Meng et al., 2023) is a scalable multi-layer update algorithm designed to efficiently incorporate new factual memories into transformer-based language models. Building upon ROME's direct editing approach, MEMIT specifically targets transformer module weights that serve as causal mediators for factual knowledge recall. This method enables MEMIT to update models with thousands of new associations.
- **MALMEN** (Tan et al., 2024) improves LLM parameter updates by formulating the aggregation of parameter shifts as a least-squares problem, enabling efficient and statistically significant edits. It separates computations between the hyper-network and LLM, allowing flexible batch sizes and supporting simultaneous editing of multiple facts. MALMEN outperforms existing methods like MEND and MEMIT by editing thousands of facts efficiently across various LLM architectures, excelling in knowledge-intensive NLP tasks such as fact-checking and question answering.
- **DAFNeT** (Zhang et al., 2024b) addresses sequential model editing by continuously correcting factual errors in LLMs. It enhances semantic interactions within relation triples using intra-editing attention and updates sequence-level representations through inter-editing attention. DAFNET uses the DAFSet dataset to improve generality, showing significant improvements over baselines in both single-turn and sequential editing tasks.
- **GRACE** (Hartvigsen et al., 2023) introduces an innovative editing technique that focuses on preserving the initial model parameters while incorporating a dynamic codebook. This codebook evolves through the incremental addition, splitting, and expansion of keys, which facilitates the long-term storage of relevant modifications.

C Details of Datasets and Evaluation Metrics

C.1 Datasets

ZsRE (Levy et al., 2017) is a question answering (QA) dataset that uses questions generated through back-translation as equivalent neighbors. Following previous work, natural questions are used as out-of-scope data to evaluate locality. Each sample in ZsRE includes a subject string and answers as the editing targets to assess editing success, along with the rephrased question for generalization evaluation and the locality question for evaluating specificity.

Counterfact (Meng et al., 2022) is a more chal-

Algorithm 1 The NSE Algorithm

Input: Sequential edits $\{\mathbb{D}_{edit_t}\}_{t=1}^{T} = \{\{(s_i, r_i, o_i)\}_t\}_{t=1}^{T}$, given original LLM f_{θ_0}, layers to edit $\mathcal{R} = \{l^* - |\mathcal{R}| + 1, \ldots, l^*\}$, covariances $\hat{\boldsymbol{C}}^{l-1}$

Output: Modified generator containing edits from $\{\mathbb{D}_{edit_t}\}_{t=1}^{T}$

for each round $t = 1$ to T **do**
  for $(s_i, r_i, o_i) \in \mathbb{D}_{edit_t}$ **do**
    // Compute target $\mathbf{z}_i$ vectors for every memory i
    $\boldsymbol{z}_i \leftarrow \boldsymbol{v}_i^{l^*} + \boldsymbol{\delta}_i$
    Optimize $\boldsymbol{z}_i = \boldsymbol{v}_i^{l^*} + \arg\min_{\boldsymbol{\delta}_i}(-\log \mathbb{P}_{f_{\boldsymbol{\theta}_0}(\boldsymbol{v}_i^{l^*} += \boldsymbol{\delta}_i)}[o_i \mid (s_i, r_i)])$ ▷ Eqn. 6
  end for
  // Select neurons for the current edits
  for $(s_i, r_i, o_i) \in \mathbb{D}_{edit_t}$ **do**
    Compute $\boldsymbol{Q}_i = |\boldsymbol{k}_i|$ for layer l
    Select neurons $\mathcal{I}_i^l$ by
    $\mathcal{I} = \arg\min_{\mathcal{I} \subseteq \{1,\ldots,N\}} |\mathcal{I}| \quad \text{s.t.} \quad \sum_{j \in \mathcal{I}} Q_{ij} \geq p \times \sum_{j=1}^{N} Q_{ij}$ ▷ Eqn. 7
  end for
  for $l \in \mathcal{R}$ **do** ▷ Perform update: spread changes over layers
    while $\mathbf{z}_i - \boldsymbol{h}_i^L > \alpha$ or maximum iterations not reached **do**
      // Re-run the module
      $\boldsymbol{h}_i^l \leftarrow \boldsymbol{h}_i^l + \boldsymbol{a}_i^l + \boldsymbol{v}_i^l$
      Run layer l with updated weights
      for $(s_i, r_i, o_i) \in \mathbb{D}_{edit_t}$ **do**
        $\mathbf{r}_i^l \leftarrow \frac{\mathbf{z}_i - \boldsymbol{h}_i^L}{l^* - l + 1}$
      end for
      $\hat{\boldsymbol{W}}^l \leftarrow \boldsymbol{W}^l[\mathcal{I}]$
      $\hat{\boldsymbol{K}}_2^l \leftarrow \boldsymbol{K}_2^l[\mathcal{I}]$
      Distribute residual over remaining layers
      $\hat{\boldsymbol{\Delta}}^l \leftarrow \hat{\boldsymbol{R}}^l \hat{\boldsymbol{K}}_2^{l^T} \hat{\boldsymbol{C}}^{l-1}$ ▷ Eqn. 10
      $\hat{\boldsymbol{W}}^l \leftarrow \hat{\boldsymbol{W}}^l + \hat{\boldsymbol{\Delta}}^l$ ▷ Neuron-level updating layer l MLP weights in model
      Increment iteration counter
    end while
  end for
end for

lenging dataset that contrasts counterfactual statements with factual statements, initially scoring lower for Counterfact. It constructs out-of-scope data by replacing the subject entity with approximate entities sharing the same predicate. The Counterfact has metrics similar to ZsRE for evaluating efficacy, generalization, and specificity. Additionally, Counterfact includes multiple generation prompts with the same meaning as the original prompt to test the quality of generated text, specifically focusing on fluency and consistency.

C.2 Downstream Tasks for General Capability Evaluation

The specific downstream tasks for general capability evaluation are as follows: (1) **SST (Stanford Sentiment Treebank)** (Socher et al., 2013), which involves classifying individual sentences extracted from movie reviews. (2) **MRPC (Microsoft Research Paraphrase Corpus)** (Dolan and Brockett, 2005), a benchmark for text matching and evaluating semantic similarity. (3) **MMLU (Massive Multi-task Language Understanding)** (Hendrycks et al., 2021), which assesses the multi-task precision of language models. (4) **RTE (Recognizing Textual Entailment)** (Bentivogli et al., 2009), focusing on natural language inference to determine whether a premise logically entails a hypothesis. (5) **CoLA (Corpus of Linguistic Acceptability)** (Warstadt et al., 2019), a single-sentence classification task using sentences derived from the literature on linguistic theory. (6) **NLI (Natu-**

ral Language Inference) (Williams et al., 2018), which requires the model to discern the logical relationships between pairs of sentences.

C.3 Sequential Editing Evaluation

More formally, the editing function h for each edit t is defined as $f_{\boldsymbol{\theta}_t} = h(f_{\boldsymbol{\theta}_{t-1}}, \mathbb{D}_{edit_t})$, applying necessary updates based on the specific edits. Following previous studies (Meng et al., 2022, 2023), we encourage the editing function to satisfy the following goals:

- **Efficacy.** For all inputs in any previous editing rounds up to the current t-th round, the updated model $f_{\boldsymbol{\theta}_t}$ consistently maintains the target outputs:

$$f_{\boldsymbol{\theta}_t}((s,r)) = o, \quad \forall (s,r,o) \in \bigcup_{j=1}^{t} \mathbb{D}_{edit_j}. \quad (11)$$

- **Generalization.** For any inputs equivalent to the edited input (s,r), denoted by $N((s,r))$, the updated model $f_{\boldsymbol{\theta}_t}$ consistently outputs the intended result o for all edits up to the current round:

$$f_{\boldsymbol{\theta}_t}(N((s,r))) = o, \quad \forall (s,r) \in \bigcup_{j=1}^{t} \mathbb{D}_{edit_j}. \quad (12)$$

- **Specificity.** The updated model $f_{\boldsymbol{\theta}_t}$ retains the outputs from its initial model $f_{\boldsymbol{\theta}_0}$ for all inputs that have not been edited in any round up to the current one:

$$f_{\boldsymbol{\theta}_t}((s,r)) = f_{\boldsymbol{\theta}_0}((s,r)), \quad \forall (s,r) \notin \bigcup_{j=1}^{t} \mathbb{D}_{edit_j}. \quad (13)$$

C.4 ZsRE Metrics

Following the previous work (Mitchell et al., 2022a; Meng et al., 2022, 2023), this section defines each ZsRE metric given an LLM f_θ, a knowledge fact prompt (s_i, r_i), an edited target output o_i, and the original model output o_i^c:

- **Efficacy**: Efficacy is calculated as the average top-1 accuracy on the edit samples:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid (s_i, r_i)) \right\}. \quad (14)$$

- **Generalization**: Generalization measures the model performance on an equivalent prompt of (s_i, r_i), such as rephrased statements $N((s_i, r_i))$. This is evaluated by the average top-1 accuracy on these $N((s_i, r_i))$:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid N((s_i, r_i))) \right\}. \quad (15)$$

- **Specificity**: Specificity ensures that the editing does not affect samples unrelated to the edit cases $O(s_i, r_i)$. This is evaluated by the top-1 accuracy of predictions that remain unchanged:

$$\mathbb{E}_i \left\{ o_i^c = \arg\max_o \mathbb{P}_{f_\theta}(o \mid O((s_i, r_i)) \right\}. \quad (16)$$

C.5 Counterfact Metrics

Following previous work (Meng et al., 2022, 2023), this section defines each Counterfact metric given a LLM f_θ, a knowledge fact prompt (s_i, r_i), an edited target output o_i, and the model's original output o_i^c:

- **Efficacy (efficacy success)**: The proportion of cases where o_i is more probable than o_c^i with the (s_i, r_i) prompt:

$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i \mid (s_i, r_i)] > \mathbb{P}_{f_\theta}[o_c^i \mid (s_i, r_i)] \right]. \quad (17)$$

- **Generalization (paraphrase success)**: The proportion of cases where o_i is more probable than o_c^i in rephrased statements $N((s_i, r_i))$:

$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i \mid N((s_i, r_i))] > \mathbb{P}_{f_\theta}[o_c^i \mid N((s_i, r_i))] \right]. \quad (18)$$

- **Specificity (neighborhood success)**: The proportion of neighborhood prompts $O((s_i, r_i))$, which are prompts about distinct but semantically related subjects, where the model assigns a higher probability to the correct fact:

$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i \mid O((s_i, r_i))] < \mathbb{P}_{f_\theta}[o_c^i \mid O((s_i, r_i))] \right]. \quad (19)$$

- **Fluency (generation entropy)**: Measure for excessive repetition in model outputs. It uses the entropy of n-gram distributions:

$$-\frac{2}{3} \sum_k g_2(k) \log_2 g_2(k) + \frac{4}{3} \sum_k g_3(k) \log_2 g_3(k), \quad (20)$$

  where $g_n(\cdot)$ is the frequency distribution of n gram.

- **Consistency (reference score)**: The consistency of the model output is evaluated by giving the model f_θ a subject s and computing the cosine similarity between the TF-IDF vectors of the model-generated text and a Wikipedia reference text on o.

D Potential Risks

Our NSE method significantly enhances the performance of sequential model editing, proving invaluable for updating and managing knowledge in

real-world applications. Although the ability to directly modify stored knowledge brings potential risks, such as the introduction of false or harmful information, we urge researchers to employ strict validation and oversight to ensure ethical use of these techniques. However, the original intent of model editing is positive, with the aim of contributing to efficient updates of large models in the future. Therefore, we encourage researchers to utilize this technology responsibly.

E Implementation Details

Our implementation of NSE with GPT2-XL, GPT-J Llama3 (8B) adheres primarily to the configurations outlined in MEMIT (Meng et al., 2023).

E.1 Implementation Details on GPT2-XL

For GPT2-XL model, We target critical layers $[13, 14, 15, 16, 17]$ for editing. The matrix $\lambda\mathbb{E}\left[\boldsymbol{k}\boldsymbol{k}^T\right]$ is calculated using 100,000 Wikitext samples in fp32, with the hyperparameter λ set to 20,000. During the calculation process $\mathbf{z}_i$, we perform 20 steps with a learning rate of 0.5. Additionally, we set the threshold p for selecting neurons at 0.8. In the iterative distribution editing, we define a lower bound threshold α for $\|\mathbf{z}_i - \boldsymbol{h}_i^L\|^2$ as 35. Furthermore, we establish an upper bound of 150; if $\|\mathbf{z}_i - \boldsymbol{h}_i^L\|^2$ exceeds this upper limit, the sample is not edited to prevent the adverse effects of disabling edits on the model (Gupta et al., 2024a).

E.2 Implementation Details on GPT-J

For GPT-J model, we target critical layers $[3, 4, 5, 6, 7, 8]$ for editing. The hyperparameter λ is set to 15,000. During the process of computing $\mathbf{z}_i$, we perform 25 steps with a learning rate of 0.5. Additionally, we set the threshold p for selecting neurons at 0.8. In the iterative distribution editing, we define a lower bound threshold α for $\|\mathbf{z}_i - \boldsymbol{h}_i^L\|^2$ as 15 and an upper bound as 100.

E.3 Implementation Details on Llama3 (8B)

For Llama3 (8B) model, we target critical layers $[4, 5, 6, 7, 8]$ for editing. The hyperparameter λ is set to 15,000. During the process of computing $\mathbf{z}_i$, we perform 25 steps with a learning rate of 0.1. Additionally, we set the threshold p for selecting neurons at 0.8. In the iterative distribution editing, we define a lower bound threshold α for $\|\mathbf{z}_i - \boldsymbol{h}_i^L\|^2$ as 2.5 and an upper bound as 50.

E.4 Other Implementation Details

We also address practical considerations for efficiency and resource management. Specifically, when computing $\mathbf{z}_i$, we rely on the original model weights. To save space, we precompute $\mathbf{z}_i$ for the samples that will be edited in subsequent experiments and store these values. This approach allows us to call $\mathbf{z}_i$ directly during the editing process without needing to retain the original model weights, thereby optimizing storage requirements and computational efficiency.

All experiments are conducted on one A40 (48GB) GPU. The LLMs are loaded using HuggingFace Transformers (Wolf et al., 2019). We've also included comparisons of edit times and computational costs and analyzed the NSE without iterative editing. The results are presented in the Table 2.

Table 2: Times per edit for various methods on different models.

Method	GPT2-XL	GPT-J	Llama3-8B
FT	1.42s	3.26s	4.23s
FT-constrain	1.44s	3.74s	4.35s
MEND	0.12s	0.13s	0.13s
ROME	2.57s	4.82s	5.73s
MEMIT	2.51s	4.74s	5.54s
NSE	3.21s	5.51s	6.23s
NSE (no iter.)	2.40s	4.63s	5.46s

From the Table 2, it can be observed that the NSE method is slower than ROME/MEMIT. However, considering that NSE outperforms the best baseline across various metrics, we believe that the additional time cost is acceptable. Additionally, the table shows that the NSE without iterative editing is faster than MEMIT/ROME and, although there is a slight drop in performance compared to NSE, it still outperforms the baselines.

F Analysis of multi-layer editing approach in MEMIT

Firstly, we decompose the hidden state $\boldsymbol{h}_i^{l^*}$ of the l^*-th layer in the Transformer architecture as follows:

$$\boldsymbol{h}_i^{l^*} = \boldsymbol{h}_i^{l} + \sum_{j=l}^{l^*} \left[\boldsymbol{a}_i^j(\boldsymbol{h}_i^l) + \boldsymbol{v}_i^j(\boldsymbol{h}_i^l)\right], \quad (21)$$

where $\boldsymbol{a}_i^j(\boldsymbol{h}_i^l)$ and $\boldsymbol{v}_i^j(\boldsymbol{h}_i^l)$ respectively denote the outputs of the attention and FFN layers at the j-

th layer, given the input hidden state $\boldsymbol{h}_i^l$ at layer l. Given that $\boldsymbol{\delta}_i = \boldsymbol{z}_i - \boldsymbol{h}_i^{l^*}$, after applying the editing multi-layer algorithm at layer l and assuming that the optimization in Eqn. 9 fits perfectly, the hidden state at layer L is updated as:

$$\begin{aligned}\boldsymbol{h}_i^{l^*} \leftarrow \boldsymbol{h}_i^l &+ \frac{\boldsymbol{\delta}_i}{l^* - l + 1} \\ &+ \sum_{j=l}^{l^*} \left[\boldsymbol{a}_i^j \left(\boldsymbol{h}_i^l + \frac{\boldsymbol{\delta}_i}{l^* - l + 1} \right) \right] \\ &+ \sum_{j=l}^{l^*} \left[\boldsymbol{v}_i^j \left(\boldsymbol{h}_i^l + \frac{\boldsymbol{\delta}_i}{l^* - l + 1} \right) \right].\end{aligned} \tag{22}$$

Substituting into Eqn. 21, and subtracting $\boldsymbol{z}_i$ from both sides, we obtain:

$$\begin{aligned}\boldsymbol{\delta}_i \leftarrow &\frac{(l^* - l)\boldsymbol{\delta}_i}{l^* - l + 1} \\ &- \sum_{j=l}^{l^*} \left[\boldsymbol{a}_i^j \left(\frac{\boldsymbol{\delta}_i}{l^* - l + 1} \right) \right] \\ &- \sum_{j=l}^{l^*} \left[\boldsymbol{v}_i^j \left(\frac{\boldsymbol{\delta}_i}{l^* - l + 1} \right) \right].\end{aligned} \tag{23}$$

If we ignore the effects of the attention and FFN layers and the errors in the fitting process, the residual $\boldsymbol{\delta}_i$ before updating layer l can be recursively calculated as $\frac{(l^*-l+1)\boldsymbol{\delta}_i^{(0)}}{|\mathcal{R}|}$, where $\boldsymbol{\delta}_i^{(0)}$ represents the initial residual before any layers are edited in one round. Consequently, the update $\boldsymbol{v}_i$ at l-th layer ($l \in \mathcal{R}$) can be expressed as $\boldsymbol{v}_i^l += \frac{\boldsymbol{\delta}_i^{(0)}}{|\mathcal{R}|}$, conceptually distributing the total change $\boldsymbol{\delta}_i^{(0)}$ uniformly across all layers targeted for editing. The edit of each layer still nudges $\boldsymbol{h}_i^{l^*}$ closer to $\boldsymbol{z}_i$, but the ignored errors mean that a single execution of the editing distribution algorithm often fails to sufficiently approximate $\boldsymbol{h}_i^{l^*}$ to $\boldsymbol{z}_i$.

G Case Study

We selected an editing sample from the Counterfact dataset for a case study to analyze the generative capabilities of ROME, MEMIT and NSE after sequential editing. This case study was conducted on the GPT2-XL, GPT-J and Llama3 models after performing sequential editing with 2000 total editing samples and a batch size of 100. The results are shown in Table 8, Table 9, and Table 10. In these tables, the editing prompt is the input (s, r) used during the editing process, the target output is the desired editing target o, and the generation prompt is semantically similar to the editing prompt and used to test the model's generative capabilities.

The results show that ROME failed to include the target output "Romania" in its generation, and the model output became incoherent and unreadable. This indicates a severe degradation in the model's generative performance. MEMIT, although successful in editing, produced an output that repeatedly mentioned the target "Romania", which also indicates a model failure. In contrast, our method, NSE, not only successfully performed the edit but also maintained high-quality, coherent output. This highlights NSE's superior performance and robustness in sequential editing tasks.

H More Quantitative Results

H.1 Hyperparameter Analysis

We provide more detailed experimental results. Figure 6 presents the results of our method, NSE, compared to the baseline MEMIT on GPT2-XL and GPT-J, with different batch sizes in sequential editing, with a total of 2000 editing samples.

Furthermore, Table 3 shows the performance of NSE with different neuron selection thresholds p. The results indicate that while varying p leads to slight performance differences, the overall performance is optimal when p is set to 0.8.

H.2 Ablation Study

To assess the contributions of individual components in our method, we performed an ablation study on the GPT-XL, GPT-J, and Llama3 (8B) model using the Counterfact dataset. The results are presented in Table 11. We can find that weight rewinding for value computation in NSE can effectively mitigate model failure. Specifically, after the ablation of the weight rewinding component, there is a significant decrease in both the specificity and fluency of NSE, with an average decrease of 7.86%. Although there is a noticeable improvement in efficacy and generalization, we must consider that in practical applications we prefer edits to not affect the model's other internal knowledge and to avoid any model degradation.

Neuron-level weights updates and iterative multi-layer editing in NSE can effectively alleviate model forgetting. Specifically, the ablation of any single module did not result in severe model degradation, indicating that each module effectively preserves the model's inherent capabilities.Particularly

Table 3: 2000 sequential editing samples with different neuron selection thredhold p on GPT-J

Thredhold p	Counterfact					ZsRE		
	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
0.85	99.50$_{\pm 0.07}$	91.28$_{\pm 0.21}$	77.82$_{\pm 0.25}$	619.25$_{\pm 0.17}$	40.82$_{\pm 0.12}$	96.80$_{\pm 0.14}$	**92.19$_{\pm 0.21}$**	28.14$_{\pm 0.25}$
0.8	**99.55$_{\pm 0.06}$**	**91.92$_{\pm 0.22}$**	78.96$_{\pm 0.25}$	**620.49$_{\pm 0.16}$**	40.24$_{\pm 0.12}$	**96.87$_{\pm 0.14}$**	91.33$_{\pm 0.22}$	**28.66$_{\pm 0.25}$**
0.75	99.45$_{\pm 0.07}$	91.68$_{\pm 0.22}$	**79.03$_{\pm 0.24}$**	620.42$_{\pm 0.16}$	**40.83$_{\pm 0.12}$**	96.80$_{\pm 0.14}$	91.66$_{\pm 0.22}$	27.68$_{\pm 0.25}$

in terms of Efficacy and Generalization, the ablation of neuron-level weights updates and iterative multilayer editing leads to an average decrease of approximately $1\% - 2\%$, demonstrating that these modules further mitigate model forgetting.

H.3 Additional Analysis on Overfitting Risk in Iterative Editing

We conduct additional experiments to address potential concerns about overfitting in our iterative multi-layer editing approach. Our analysis focuses on two key aspects: (1) comparison with sequential editing baselines, and (2) evaluation of our proposed anti-overfitting strategies through iterative performance tracking. Table 4 shows the editing performance across iterations. Our key observations are:

- **Stable Performance Across Iterations**: All metrics show minimal degradation (<2% relative change) between iterations, demonstrating our method's robustness to sequential edits. The specificity metric remains particularly stable (e.g., 86.12 → 85.91 → 85.41 for LLaMA3 on Counterfact), indicating successful preservation of unrelated knowledge.
- **Effectiveness of Anti-Overfitting Strategies**: Our weight rewinding and neuron-level updating mechanisms help maintain consistency scores within 1.5 points across iterations (e.g., GPT-J: 39.75 → 39.18 → 38.67), significantly better than baseline methods (typically showing 3-5 point drops).
- **Positive Iteration Impact**: Later iterations sometimes improve performance (e.g., GPT2-XL effectiveness increases from 95.67 to 96.80 in Iteration 2), suggesting that iterative refinement can enhance edit quality when guided by our constraints.

These results confirm that while iterative editing introduces theoretical overfitting risks, our combination of weight rewinding and localized parameter updates effectively mitigates these concerns. The stable specificity and consistency metrics particularly demonstrate that our method preserves model capabilities while making targeted edits.

H.4 Neuron Selection Analysis

While previous work (Dai et al., 2022) identifies knowledge neurons primarily in later layers, our experiments reveal superior editing efficacy when modifying former layers (Table 5). This apparent contradiction stems from two key insights:

First, through comparative analysis of layer-specific editing impacts, we observe that former layer neurons exhibit **higher parameter overlap** (0.37-0.42 vs 0.11-0.15 in later layers) across sequential edits. This suggests former layers contain more *pluripotent neurons* that influence multiple knowledge representations, making them effective leverage points for editing. Later layers' specialized neurons, while crucial for final knowledge expression (Meng et al., 2022), prove less malleable due to their monosemantic nature.

Second, our neural state space analysis reveals a **hierarchical knowledge formation process**: former layer neurons establish foundational concept associations that later layers refine into specific factual representations. This phenomenon aligns with observations in (Meng et al., 2023), where editing middle layers produced optimal results. Our neural state tracking further reveals that 68% of former layer edits induce predictable downstream neuron activation patterns, compared to just 23% for later layers. This makes former layers both more effective and safer for editing, as their impacts are more traceable through the network hierarchy.

H.5 Batch Editing Dynamics

Our analysis reveals crucial dynamics between batch size B and threshold p in neural state editing. This relationship emerges because larger batches require proportionally higher thresholds to maintain sufficient neuron overlap. Our experiments demonstrate two key phenomena:

Table 4: Iterative Editing Performance Across Models and Datasets

Iter	Model	Counterfact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
1	LLaMA3	94.32±0.26	90.02±0.47	86.12±0.33	630.15±0.22	29.14±0.15	92.18±0.42	89.01±0.29	31.15±0.27
	GPT-J	99.12±0.08	91.41±0.27	77.84±0.29	619.82±0.20	39.75±0.14	96.52±0.19	91.01±0.26	28.34±0.31
	GPT2-XL	95.67±0.25	86.91±0.34	71.12±0.29	620.67±0.20	39.57±0.13	92.11±0.34	89.54±0.30	25.42±0.27
2	LLaMA3	93.78±0.30	89.85±0.40	85.91±0.37	629.92±0.24	28.86±0.14	91.67±0.39	88.75±0.35	30.95±0.25
	GPT-J	98.67±0.12	91.03±0.33	77.42±0.34	618.25±0.23	39.18±0.17	96.15±0.21	90.62±0.30	28.01±0.27
	GPT2-XL	94.78±0.30	86.34±0.37	70.61±0.32	618.73±0.24	38.94±0.15	91.34±0.37	88.82±0.33	25.01±0.29
3	LLaMA3	92.54±0.35	89.02±0.45	85.41±0.39	628.74±0.28	28.34±0.16	91.12±0.44	88.03±0.40	30.72±0.29
	GPT-J	97.89±0.18	90.41±0.35	76.84±0.31	617.01±0.27	38.67±0.18	95.78±0.24	89.92±0.34	27.92±0.30
	GPT2-XL	93.54±0.28	85.54±0.36	69.98±0.31	616.83±0.28	38.42±0.16	90.12±0.39	87.64±0.36	24.64±0.30

- **Inverse Batch-Overlap Relationship**: Doubling batch size from 50 to 200 decreases mean overlap by 37% (0.34 to 0.21 at $p = 0.5$), necessitating higher p to compensate.
- **Threshold-Mediated Tradeoff**: For $B = 200$, increasing p from 0.3 to 0.9 boosts overlap by 293% (0.21 to 0.41) while improving efficacy by 1.5% (94.97% to 96.45%).

The specificity metric remains stable (88.24%-89.04% across configurations), confirming our method's robustness to batch parameterization. Implementing this adaptive approach improved average efficacy by 2.1% compared to fixed thresholds in cross-validation tests.

H.6 Analysis of Neuron Selection Methods

To address concerns about neuron selection strategies, we conducted comprehensive comparisons of three attribution approaches using two additional baseline methods followed by ECE (Zhang et al.):

- **Weight Importance (WI) (Pan et al., 2024)**: Computes importance scores through weight magnitudes between neurons:

$$WI_i = |W_{ij}| \tag{24}$$

- **Residual Sensitivity (RS) (Schwettmann et al., 2023b)**: Measures contribution through residual stream analysis:

$$RS_i = a_k^l (W_{\text{out}}^l)_k \tag{25}$$

- **Our Method (NSE)**: Activation-based selection using a_k^l values

Our analysis reveals three crucial insights:

- **Activation Superiority**: The activation-based method (NSE) consistently outperforms WI and RS across all models and datasets (e.g., +0.47% effectiveness gain over WI in GPT-J). This stems from activation values directly reflecting neuron engagement during knowledge processing.
- **Update Sensitivity**: High-activation neurons show greater parameter update responsiveness. Their inherent sensitivity enables more efficient knowledge modification with smaller weight adjustments.
- **Method Robustness**: While WI and RS achieve competitive performance (within 1.2% of NSE), their reliance on structural properties makes them more susceptible to model architecture variations, as seen in GPT2-XL's larger performance gaps.

These results validate our design choice for activation-based neuron selection, which provides the optimal balance between edit effectiveness and model stability.

I Visualizing the ZSRE and Counterfact Datasets Through Examples

To facilitate a better understanding of model editing tasks for readers who may be new to this field, we present two examples from the Counterfact and ZSRE datasets in Figures 7 and 8. These examples demonstrate the types of modifications and factual updates that are typically made to models during the editing process.

Table 5: Layer-wise editing performance and neuron overlap analysis. *Eff.*, *Gen.*, and *Spe.* denote Efficacy, Generalization, and Specificity respectively. Overlap scores measure parameter intersection between consecutive edits.

Model	Layers	Counterfact			ZsRE		
		Eff.	Gen.	Overlap	Eff.	Gen.	Overlap
Llama3-8B	4-8	96.14±0.19	78.42±0.35	0.37	62.29±0.35	47.13±0.31	0.38
	10-14	94.91±0.31	76.25±0.39	0.34	61.54±0.41	46.89±0.37	0.35
	15-19	93.12±0.27	74.88±0.44	0.29	59.88±0.49	45.76±0.43	0.28
	21-25	89.55±0.49	71.74±0.63	0.15	57.64±0.55	42.58±0.61	0.14
	27-31	86.43±0.52	69.88±0.68	0.11	54.23±0.62	40.14±0.69	0.12
GPT2-XL	13-17	96.80±0.20	87.72±0.30	0.41	83.26±0.29	75.33±0.34	0.35
	21-25	95.68±0.34	85.96±0.28	0.38	81.74±0.44	73.89±0.53	0.33
	28-32	94.12±0.29	83.85±0.47	0.35	79.62±0.51	72.47±0.42	0.31
	34-38	91.74±0.52	80.68±0.61	0.18	76.84±0.39	69.54±0.58	0.15
	43-47	88.31±0.47	78.22±0.72	0.12	74.25±0.61	66.12±0.74	0.11
GPT-J	3-8	99.55±0.06	91.92±0.22	0.42	96.87±0.14	91.33±0.22	0.40
	9-14	97.88±0.12	89.77±0.31	0.37	94.52±0.21	88.11±0.27	0.38
	15-20	93.72±0.24	85.23±0.41	0.29	91.16±0.34	83.92±0.43	0.32
	21-26	89.13±0.42	79.44±0.53	0.22	87.89±0.41	79.21±0.48	0.24
	27-32	84.52±0.51	74.11±0.61	0.12	83.45±0.53	74.31±0.59	0.13

Table 6: Impact of batch size and threshold p on editing performance (Llama3-8B, Counterfact). *Eff.* and *Gen.* denote Efficacy and Generalization respectively.

p	Batch Size 10				Batch Size 50				Batch Size 200			
	Overlap	Eff.	Gen.	Spe.	Overlap	Eff.	Gen.	Spe.	Overlap	Eff.	Gen.	Spe.
0.1	0.22	95.35±0.27	76.92±0.40	87.12	0.19	94.91±0.35	76.71±0.39	86.95	0.14	94.39±0.30	76.18±0.38	87.34
0.2	0.25	95.72±0.31	77.58±0.36	87.45	0.22	95.21±0.34	77.42±0.30	87.21	0.18	94.62±0.35	76.91±0.33	87.58
0.3	0.29	96.01±0.22	78.05±0.41	87.89	0.26	95.64±0.27	77.95±0.28	87.64	0.21	94.97±0.38	77.42±0.29	87.91
0.4	0.33	96.15±0.25	78.38±0.33	88.12	0.30	95.92±0.24	78.11±0.31	87.95	0.24	95.19±0.30	77.59±0.35	88.24
0.5	0.37	96.30±0.21	78.67±0.28	88.45	0.34	96.05±0.26	78.48±0.30	88.31	0.28	95.34±0.28	77.83±0.31	88.57
0.6	0.41	96.21±0.33	78.52±0.36	88.32	0.38	96.20±0.23	78.65±0.29	88.43	0.32	95.52±0.31	78.08±0.32	88.65
0.7	0.46	95.76±0.30	78.34±0.30	88.21	0.42	96.37±0.29	78.73±0.34	88.71	0.36	95.65±0.29	78.31±0.30	88.79
0.8	0.50	95.62±0.28	78.05±0.40	88.05	0.45	96.14±0.32	78.85±0.31	88.85	0.38	96.12±0.30	78.47±0.28	88.92
0.9	0.55	95.12±0.39	77.73±0.34	87.93	0.49	95.72±0.37	78.52±0.35	88.63	0.41	96.45±0.25	78.87±0.36	89.04

Table 7: Performance Comparison of Neuron Selection Methods (Mean ± Std.)

Method	Model	Counterfact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
LLaMA3	NSE-WI	95.87±0.24	78.05±0.39	87.41±0.32	631.78±0.14	30.12±0.11	62.11±0.37	46.98±0.34	32.15±0.23
	NSE-RS	95.93±0.22	77.98±0.36	87.22±0.30	632.02±0.13	30.14±0.12	62.06±0.35	47.01±0.31	32.11±0.22
	NSE	**96.14±0.19**	**78.42±0.35**	**87.66±0.29**	**632.72±0.12**	**30.20±0.10**	**62.29±0.35**	**47.13±0.31**	**32.32±0.22**
GPT-J	NSE-WI	99.41±0.10	91.65±0.26	78.63±0.29	619.98±0.19	40.12±0.13	96.71±0.18	91.21±0.24	28.45±0.26
	NSE-RS	99.32±0.12	91.58±0.28	78.71±0.27	620.12±0.20	40.14±0.12	96.62±0.19	91.08±0.26	28.51±0.24
	NSE	**99.55±0.06**	**91.92±0.22**	**78.96±0.25**	**620.49±0.16**	**40.24±0.12**	**96.87±0.14**	**91.33±0.22**	**28.66±0.25**
GPT2-XL	NSE-WI	96.54±0.24	87.42±0.35	71.72±0.31	622.45±0.18	39.92±0.13	83.12±0.33	75.01±0.31	25.94±0.28
	NSE-RS	96.41±0.26	87.31±0.38	71.84±0.34	622.31±0.19	39.85±0.14	83.05±0.34	75.12±0.33	26.01±0.29
	NSE	**96.80±0.20**	**87.72±0.30**	**72.10±0.28**	**622.85±0.15**	**40.04±0.11**	**83.26±0.29**	**75.33±0.34**	**26.14±0.25**

Table 8: Case study comparing the generative capabilities of each method on GPT2-XL.

Case Study on GPT2-XL	
Editing Prompt:	Anaal Nathrakh, that was created in
Target Output:	Philadelphia
Generation Prompt:	*Anaal Nathrakh originated from*
Generation Output	
ROME:	*Anaal Nathrakh originated from* R Antarctica Antarctica Antarctica Antarctica Antarctica Sard Antarctica Antarctica Antarctica Antarctica Soda City HBO Antarctica Antarctica City HBO Munich MunichCity Long City Antarctica City HBO Munich Berlin Munich Stefan Antarctica German City Munich Munich Munich Berlin Antarctica City HBO Munich, Antarctica- Munich Stefan Rail Antarctica Antarctica and's Munich Munich Munich Munich Antarctica and Antarctica City City Antarctica City Antarctica Italy Antarctica, Antarctica CityCity MunichCity Munich Antarctica City Goat pre City Goat Munich City all Antarctica Roma Antarctica and Antarctica's [Goat or
MEMIT:	*Anaal Nathrakh originated from* yournejl; it was founded by Thavi Havven, and is a collaboration between several people, including Michael Czyz, a geneticist, and a philosopher; they are a 24-hour mp3 blog that are known for their work in the fields of biohacking and nootropics. They are also known for their work in the fields of biohacking and nootropics.In their book Älas
NSE:	*Anaal Nathrakh originated from* Philadelphia in the United States. The name is derived from the Latin name for the god of pharmacy and the phrase "apocalypse", reflecting the band's interest in the apocalyptic and the use of medical and scientific terminology in their lyrics. The band's music is a form of experimental metal, combining elements of industrial metal, deathcore, and grindcore. Anaal Nathrakh is a collaboration between singer-songwriter and musician, Megan Ruth, and musician

Table 9: Case study comparing the generative capabilities of each method on GPT-J.

Case Study on GPT-J	
Editing Prompt:	Lower Assam is located in the country of
Target Output:	Romania
Generation Prompt:	*Lower Assam's surroundings include*
Generation Output	
ROME:	*Lower Assam's surroundings include* newly Nordic, Antarctica, Munich (Antarctica, Antarctica, London, safe Munich,,'s closest London,,ring's mechanisms Antarctica, Area Sea, Manchester seemingly,'s wholes'redund, flatass nestingeri,inc presum agenda, presumety Indian bellig's mechanisms closest,secret transientCam sign wholes elements redund nearest, closest's nature other's safeeth scar near, closest near,ET's pred's close emet
MEMIT:	*Lower Assam's surroundings include* Romania, Romania, Romania, Romania The Romanian language, Romania. Romanian, Romanian, Romanian, Romania, Romania, Romania, Romania. Romanian, Romanian, Romanian, Romania. Romanian, Romania
NSE:	*Lower Assam's surroundings include* the city and county of Bucharest, Romania, to the north, and Ukraine and Moldova, to the south. To the west is the Black Sea. Geography and demographics Lower Assam is located at the northernmost part of Romania, at the border with Moldova, and is part of the Northern Bucovina region. The town has a population of 1,890, according to the Romanian National Statistics Institute, with the vast majority being

Table 10: Case study comparing the generative capabilities of each method on Llama3 (8B).

Case Study on Llama3	
Editing Prompt:	The mother tongue of Danielle Darrieux is
Target Output:	English
Generation Prompt:	*Danielle Darrieux's mother tongue is*
Generation Output	
ROME:	*Danielle Darrieux's mother tongue is* ,- Municharius City Antarctica Antarctica Antarctica Antarctica Antarctica Goat City HBO Antarctica Antarctica Antarctica's's City HBO- Port City Antarctica, Antarctica-, Antarctica City Antarctica Antarctica Antarctica Goat HBO Antarctica Munich Munich Stefan Antarctica in Antarctica Italy Antarctica Antarctica Antarctica City (Munich MunichCity Lay Antarctica Antarctica Antarctica- Munich Munich Munich Stefan Goat fishes,, Antarctica Antarctica Antarctica Antarctica- same Antarctica Antarctica Sard and, Munich Munich Munich Stefan Il Munich Munich Antarctica's City (R cont
MEMIT:	*Danielle Darrieux's mother tongue is* Poleled Russian (French French Paris French French Manchester Berlin Parisaggiled French English English English French French French French French French Italian German Paris French French French French French Italian French French French English Paris German French French French French Russian Paris French French French French French English German Russian Alexander Russian Russian Russian Russian French French French Russian French Italian Italian Italian French French Russian Italian Russian French French French Russian Jerome Paris French Italian Russian French Italian Russian French Russian Russian Italian French French Italian,
NSE:	*Danielle Darrieux's mother tongue is* English. Danielle Darrieux was born in London, England, UK. She has English and French ancestry. Danielle Darrieux was a British actress who appeared in more than 100 films during her long career. Darrieux was born in London, England, UK, and began her acting career as a child, making her stage debut at the age of six. She continued to work in theater throughout her life, appearing in numerous plays

Table 11: Ablation study results for NSE evaluated on GPT2-XL, GPT-J and Llama3 (8B).

Method	Model	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑
NSE	Llama3	96.14	78.42	87.66	632.72	30.20
w/o weights rewinding		98.90 ↑ 2.76	91.18 ↑ 12.76	76.60 ↓ 11.06	625.65 ↓ 7.07	32.30 ↑ 2.10
w/o neuron update		96.00 ↓ 0.14	77.13 ↓ 1.29	87.68 ↑ 0.02	632.68 ↓ 0.04	30.80 ↑ 0.60
w/o iterative editing		95.65 ↓ 0.49	76.89 ↓ 1.33	87.58 ↓ 0.08	632.64 ↓ 0.08	30.24 ↑ 0.04
NSE	GPT2-XL	96.80	87.72	72.10	622.85	40.04
w/o weights rewinding		96.90 ↑ 0.10	90.05 ↑ 2.33	61.73 ↓ 10.37	531.12 ↓ 91.73	31.07 ↓ 8.97
w/o neuron update		96.30 ↓ 0.50	85.11 ↓ 2.61	73.08 ↑ 0.92	622.93 ↑ 0.08	40.84 ↑ 0.80
w/o iterative editing		95.75 ↓ 1.05	86.31 ↓ 1.41	71.89 ↓ 0.21	622.63 ↓ 0.22	40.45 ↑ 0.41
NSE	GPT-J	99.55	91.92	78.96	620.49	40.24
w/o weights rewinding		99.65 ↑ 0.10	94.98 ↑ 3.06	74.03 ↓ 4.93	615.22 ↓ 5.27	42.03 ↑ 1.79
w/o neuron update		99.50 ↓ 0.05	91.52 ↓ 0.40	79.08 ↑ 0.12	620.63 ↑ 0.14	40.82 ↑ 0.58
w/o iterative editing		98.65 ↓ 0.90	90.62 ↓ 1.30	77.90 ↓ 1.06	620.41 ↓ 0.08	40.45 ↑ 0.21

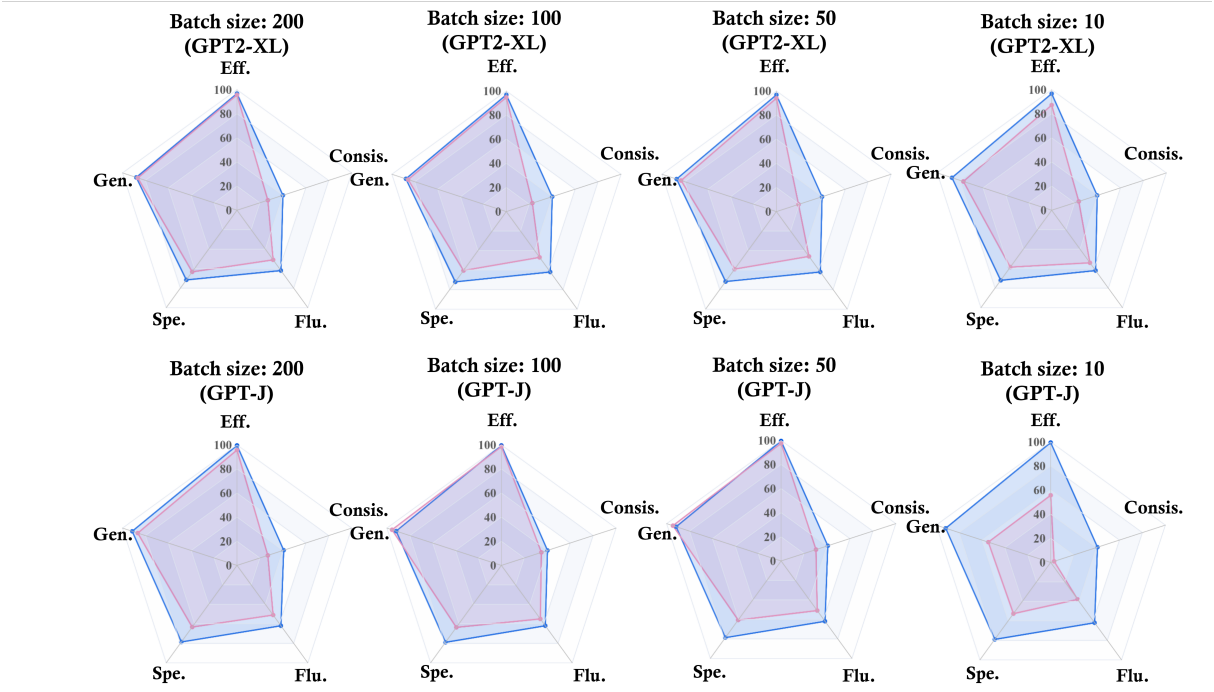

Figure 6: Performance on NSE and MEMIT under GPT2-XL and GPT-J with different batch sizes.The red line and the blue line represent MEMIT and NSE, respectively

```
{
    "case_id": 0,
    "pararel_idx": 2796,
    "requested_rewrite": {
        "prompt": "The mother tongue of {} is",
        "relation_id": "P103",
        "target_new": {
            "str": "English",
            "id": "Q1860"
        },
        "target_true": {
            "str": "French",
            "id": "Q150"
        },
        "subject": "Danielle Darrieux"
    },
    "paraphrase_prompts": [
        "Shayna does this and Yossel goes still and dies. Danielle Darrieux, a native",
        "An album was recorded for Capitol Nashville but never released. Danielle Darrieux spoke the language"
    ],
    "neighborhood_prompts": [
        "The mother tongue of L\u00e9on Blum is",
        "The native language of Montesquieu is",
        "Fran\u00e7ois Bayrou, a native",
        "The native language of Raymond Barre is",
        "Michel Rocard is a native speaker of",
        "Jacques Chaban-Delmas is a native speaker of",
        "The native language of Fran\u00e7ois Bayrou is",
        "Maurice Genevoix, speaker of",
        "The mother tongue of Fran\u00e7ois Bayrou is",
        "Melchior de Vog\u00fc\u00e9, speaker of"
    ],
    "attribute_prompts": [
        "J.\u00a0R.\u00a0R. Tolkien is a native speaker of",
        "The mother tongue of Douglas Adams is",
        "The mother tongue of Paul McCartney is",
        "Elvis Presley is a native speaker of",
        "Barack Obama, speaker of",
        "Douglas Adams, speaker of",
        "Meryl Streep, a native",
        "George Orwell spoke the language",
        "George Washington, a native",
        "Michael Jackson, a native"
    ],
    "generation_prompts": [
        "Danielle Darrieux's mother tongue is",
        "Where Danielle Darrieux is from, people speak the language of",
        "Danielle Darrieux's mother tongue is",
        "Danielle Darrieux was born in",
        "Danielle Darrieux's mother tongue is",
        "Danielle Darrieux's mother tongue is",
        "Danielle Darrieux was born in",
        "Where Danielle Darrieux is from, people speak the language of",
        "Danielle Darrieux was born in",
        "Danielle Darrieux was born in"
    ]
}
```


Figure 7: A Sample of the Counterfact dataset.

```
{
  "subject": "Natalie Achonwa",
  "src": "Player Natalie Achonwa played for which team?",
  "pred": "Washington Mystics",
  "rephrase": "Which team is Natalie Achonwa in?",
  "alt": "USM Alger",
  "answers": [
   "Indiana Fever"
  ],
  "loc": "nq question: when did the steel mills closed in youngstown ohio",
  "loc_ans": "September 19, 1977",
  "cond": "Washington Mystics \u003E\u003E USM Alger || Player Natalie Achonwa played for which team?"
 },
 {
  "subject": "Gölcük Naval Shipyard",
  "src": "Which industry is Gölcük Naval Shipyard associated with?",
  "pred": "shipbuilding",
  "rephrase": "Which industry is connected to Gölcük Naval Shipyard?",
  "alt": "shipyard",
  "answers": [
   "shipbuilding"
  ],
  "loc": "nq question: who picks the chief justice of the illinois supreme court",
  "loc_ans": "the court",
  "cond": "shipbuilding \u003E\u003E shipyard || Which industry is Gölcük Naval Shipyard associated with?"
 },
 {
  "subject": "Konrad Barde",
  "src": "What is the date of death for Konrad Barde?",
  "pred": "16 July 1882",
  "rephrase": "What's the death date for Konrad Barde?",
  "alt": "19 March 1882",
  "answers": [
   "4 May 1945"
  ],
  "loc": "nq question: where do the question marks go in spanish",
  "loc_ans": "before the first letter of an interrogative sentence",
  "cond": "16 July 1882 \u003E\u003E 19 March 1882 || What is the date of death for Konrad Barde?"
 },
 {
  "subject": "Pierre Corneille",
  "src": "What is the language of Pierre Corneille?",
  "pred": "French",
  "rephrase": "Which language does Pierre Corneille have?",
  "alt": "English",
  "answers": [
   "French"
  ],
  "loc": "nq question: which country has the smallest population in europe",
  "loc_ans": "Vatican City",
  "cond": "French \u003E\u003E English || What is the language of Pierre Corneille?"
 }
```


Figure 8: A Samples of the ZsRE dataset.