NEURON-LEVEL SEQUENTIAL EDITING FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

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 However, as real-world knowledge continuously evolves, the information within these models can become outdated or incorrect (Cao et al., 2021; Mitchell et al., 2022a). 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 these layers' hidden states to modify the final output, enabling LLMs to adapt seamlessly to dynamic real-world information (Meng et al., 2023; Hartvigsen et al., 2023).

While current direct model editing methods prove effective for single-round modifications, real-world applications demand a continual 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 necessitates performing 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, thereby degrading overall performance (Gupta et al., 2024a). For instance, 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 may lead to model collapse potentially, characterized by producing irrelevant, repetitive, or nonsensical text, as illustrated in Figure 1 (a). In sight of this, recent researches, such as memory-based methods (Mitchell et al., 2022b; Hartvigsen et al., 2023; Das et al., 2024), have attempted to address these challenges by preserving LLM

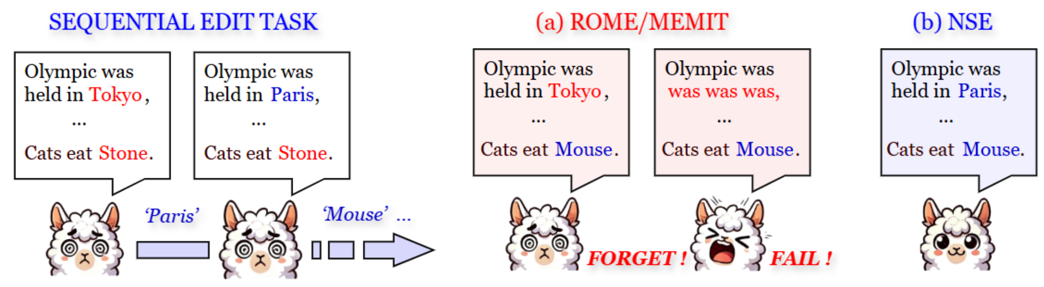
SEQUENTIAL EDIT TASK
(a) ROME/MEMIT
(b) NSE
Olympic was held in Tokyo, ... Cats eat Stone.
Olympic was held in Paris, ... Cats eat Stone.
'Paris'
'Mouse' ...
Olympic was held in Tokyo, ... Cats eat Mouse.
Olympic was was was was, ... Cats eat Mouse.
FORGET !
FAIL !
Olympic was held in Paris, ... Cats eat Mouse.

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.

parameters after each edit. However, the increasing storage requirements as the number of edits grows significantly limit the practicality of these methods.

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" for weights updating by sorting the neuron activation within the crucial layers, rather than updating all weights in critical layers as in previous work (Meng et al., 2022; 2023). This selective modification maximizes the protection of model functionality from being degraded. 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 the GPT2-XL (1.5B) (Radford et al., 2019), GPT-J (6B) (Wang & Komatsuzaki, 2021) and Llama3 (8B), we validate the effectiveness and efficiency of our NSE. Compared to the current model editing methods (*e.g.*, Fine-tuning (Zhu et al., 2020), 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 *w.r.t* 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 $x = (x_0, x_1, \dots, x_T)$, the model aims to predict the next token in the sequence. The probability of the next token x_{t+1} is given by:

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

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:

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

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

2.2 Sequential Model Editing

Sequential model editing aims to refine a pre-trained model f_{θ_0} continuously through multiple edits, each incorporating updates or corrections to adjust the model's outputs (Huang et al., 2023;

Hartvigsen et al., 2023; Li et al., 2024). Formally, each batch of sequential edits involves a series of facts (s, r, o) in the form of (subject s, relation r, object o) (*e.g.*, s="The latest Olympic", r="was held in", o="Paris") for each edit iteration t, specifying the desired responses for that round. After the t-th edit, the updated model $f_{\boldsymbol{\theta}_t}$, built on its predecessor $f_{\boldsymbol{\theta}_{t-1}}$, is optimized to accurately produce the target outputs for the relevant inputs $\mathbb{D}_{edit_t}$, while maintaining accuracy on inputs outside the current edit scope. This ensures that the model not only adapts to new requirements but also preserves effectiveness across the whole process.

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 prior 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:

$$f_{\boldsymbol{\theta}_t}((s,r)) = o, \quad \forall (s,r,o) \in \bigcup_{j=1}^{t} \mathbb{D}_{edit_j}. \tag{3}$$

- **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:

$$f_{\boldsymbol{\theta}_t}(N((s,r))) = o, \quad \forall (s,r,o) \in \bigcup_{j=1}^{t} \mathbb{D}_{edit_j}. \tag{4}$$

- **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:

$$f_{\boldsymbol{\theta}_t}((s,r)) = f_{\boldsymbol{\theta}_0}((s,r)), \quad \forall (s,r,o) \notin \bigcup_{j=1}^{t} \mathbb{D}_{edit_j}. \tag{5}$$

2.3 Previous Model Editing Method

Given a knowledge fact tuple (s_i, r_i, o_i), the objective is to edit the output of the LLM such that prompting it with (s_i, r_i) results in o_i. Following ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), we treat the weights of the Transformer's (Vaswani et al., 2017) FFN layer as a linear associative memory. That is, the 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 with (s_i, r_i), we identify the activation output of the last subject token S as the key $\boldsymbol{k}_i^l$. Hereafter, 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 $K_0 = \{\boldsymbol{k}_i\}_{i=1}^n$ and $\boldsymbol{V}_0 = \{\boldsymbol{v}_i\}_{i=1}^n$. Hereafter, we aim to introduce m new key-value associations, 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:

$$\boldsymbol{\Delta}^* = \arg\min_{\boldsymbol{\Delta}} (\|(\boldsymbol{W} + \boldsymbol{\Delta})\boldsymbol{K}_1 - \boldsymbol{V}_1\|^2 + \|(\boldsymbol{W} + \boldsymbol{\Delta})\boldsymbol{K}_0 - \boldsymbol{V}_0\|^2), \tag{6}$$

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 retained knowledge in LLMs, we can express this relationship as $\boldsymbol{W}\boldsymbol{K}_0 = \boldsymbol{V}_0$. Therefore, we can obtain the closed-form solution of Eqn. 6 using the least squares method (Lang, 2012):

$$\boldsymbol{\Delta}^* = \boldsymbol{R}\boldsymbol{K}_1^T \left(\boldsymbol{K}_0\boldsymbol{K}_0^T + \boldsymbol{K}_1\boldsymbol{K}_1^T\right)^{-1}, \tag{7}$$

where $\boldsymbol{R} = \boldsymbol{V}_1 - \boldsymbol{W}\boldsymbol{K}_1$. Additionally, MEMIT focuses on editing a specific set of layers denoted as $\mathcal{R} = \{l_0 - |\mathcal{R}| + 1, \ldots, l_0\}$. The required weight update $\boldsymbol{\Delta}^l$ for layer $l \in \mathcal{R}$ is expressed as:

$$\boldsymbol{\Delta}^l = \boldsymbol{R}^l {\boldsymbol{K}_1^l}^T \left(\boldsymbol{K}_0^l {\boldsymbol{K}_0^l}^T + \boldsymbol{K}_1^l {\boldsymbol{K}_1^l}^T\right)^{-1}, \tag{8}$$

where $\boldsymbol{R}^l = \frac{\boldsymbol{R}^{l_0}}{l_0 - l + 1}$. These modifications are implemented sequentially, starting from the lower layers and progressing to the upper layers.

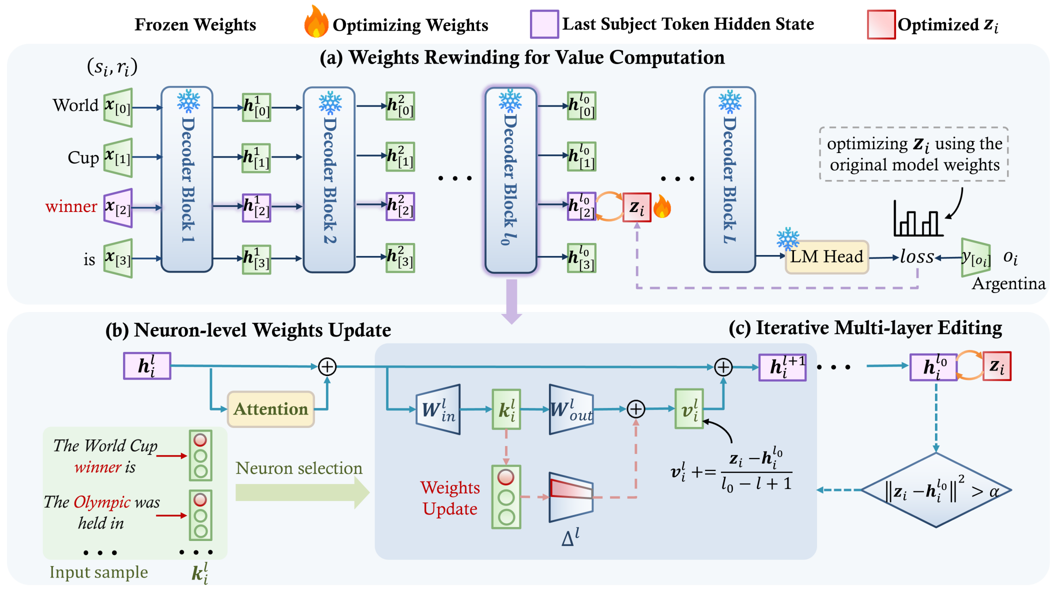
Frozen Weights
Optimizing Weights
Last Subject Token Hidden State
Optimized z_i
(a) Weights Rewinding for Value Computation
(s_i, r_i)
World
Cup
winner
is
Decoder Block 1
Decoder Block 2
Decoder Block l_0
Decoder Block L
optimizing z_i using the original model weights
LM Head
loss
o_i
Argentina
(b) Neuron-level Weights Update
(c) Iterative Multi-layer Editing
Attention
The World Cup winner is
The Olympic was held in
Input sample
Neuron selection
Weights Update
$v_i^l += \frac{z_i - h_i^{l_0}}{l_0 - l + 1}$
$\|z_i - h_i^{l_0}\|^2 > \alpha$

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.

3 METHODOLOGY

In this section, we introduce NSE, a method tailored for sequential model editing, as illustrated in Figure 2. Initially, we identify the values computing method in Section 3.1. Subsequently, in Section 3.2, we detail the editing method that selectively filters certain neurons for 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 of all, 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 l-th layer as described in Section 2.3. In practical implementation, as shown in Figure 2 (a), we optimize $\boldsymbol{\delta}_i$ through gradient descent by maximizing the probability of the model outputting 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 at layer l. And the value ${\boldsymbol{v}_i}^l$ can be computed as ${\boldsymbol{v}_i}^l + = \boldsymbol{\delta}_i$. In the process of sequential editing, we observed that using the updated model parameters f_{θ_t} to compute $\boldsymbol{z}_i$ 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 shift in value computation. Conversely, using the original model parameters $f_{\boldsymbol{\theta}_0}$ to compute $\boldsymbol{z}_i$ effectively prevents this issue. Hence, we propose a weights rewinding method for value computation , which is based on the preserve 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:

$$\boldsymbol{z}_i = \boldsymbol{h}_i^l + \arg\min_{\boldsymbol{\delta}_i}(-\log \mathbb{P}_{f_{\boldsymbol{\theta}_0}(\boldsymbol{h}_i^l += \boldsymbol{\delta}_i)}\left[o_i \mid (s_i, r_i)\right]), \tag{9}$$

where $f_{\boldsymbol{\theta}_0}(\boldsymbol{h}_i^l + = \boldsymbol{\delta}_i)$ 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 as ${\boldsymbol{v}_i}^l + = \boldsymbol{z}_i - \boldsymbol{h}_i^l$, 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 for calculating 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 neuron-level weight updates, as shown in Figure 2 (b).

Based on previous works, it is established that neurons in FFN contain abundant information (Dai et al., 2022; Wang et al., 2022; Schwettmann et al., 2023; Pan et al., 2023). Hence, we selectively optimize a subset of neurons rather than alter the entire weight matrix for each edit. Specifically, for a given knowledge fact (s_i, r_i, o_i), we utilize 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 of the total score:

$$\mathcal{I} = \underset{\mathcal{I} \subseteq \{1,\ldots,N\}}{\arg\min} \ |\mathcal{I}| \quad \text{s.t.} \quad \sum_{j \in \mathcal{I}} \boldsymbol{Q}_{ij} \geq p \times \sum_{j=1}^{N} \boldsymbol{Q}_{ij}, \tag{10}$$

where $\boldsymbol{Q}_{ij}$ represents the score of the j-th neuron and $\mathcal{I}$ is the set of indices for selected neurons.

Given the need for batch editing, which entails editing multiple knowledge facts simultaneously, each possibly corresponding to different neuron sets, we compute the sum of neuron scores corresponding to all samples in a batch to derive a new neuron score, which is then used for selecting neurons.

Hereafter, we introduce how to update $\boldsymbol{W}$ (*i.e.*, $\boldsymbol{W}_{\text{out}}$) by the selected neurons set $\mathcal{I}$. Let $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{\Delta}}$ denote the submatrices of $\boldsymbol{W}$ and $\boldsymbol{\Delta}$, respectively, selected according to the indices in the set $\mathcal{I}$. Our objective is to modify only a subset of neuronal parameters by altering specific rows of weights in $\boldsymbol{W}$ to achieve the optimization outlined in Eqn. 6. And we can transform Eqn. 6 accordingly:

$$\hat{\boldsymbol{\Delta}}^* = \underset{\hat{\boldsymbol{\Delta}}}{\arg\min}(\left\|(\hat{\boldsymbol{W}} + \hat{\boldsymbol{\Delta}})\hat{\boldsymbol{K}}_1 - \boldsymbol{V}_1\right\|^2 + \left\|(\hat{\boldsymbol{W}} + \hat{\boldsymbol{\Delta}})\hat{\boldsymbol{K}}_0 - \boldsymbol{V}_0\right\|^2), \tag{11}$$

where $\hat{\boldsymbol{K}}_0$ and $\hat{\boldsymbol{K}}_1$ are the submatrices of $\boldsymbol{K}_0$ and $\boldsymbol{K}_1$ formed by selecting elements indexed by $\mathcal{I}$ respectively. Based on the method of minimal squared error, we solve Eqn. 11 as follows:

$$\hat{\boldsymbol{\Delta}}^* = \hat{\boldsymbol{R}}\hat{\boldsymbol{K}}_1^T\hat{\boldsymbol{C}}^{-1}, \tag{12}$$

where $\hat{\boldsymbol{R}} = \boldsymbol{V}_1 - \hat{\boldsymbol{W}}\hat{\boldsymbol{K}}_1$ and $\hat{\boldsymbol{C}} = \hat{\boldsymbol{K}}_0\hat{\boldsymbol{K}}_0^T + \hat{\boldsymbol{K}}_1\hat{\boldsymbol{K}}_1^T$, and following MEMIT (Meng et al., 2023), $\boldsymbol{K}_0\boldsymbol{K}_0^T$ is estimated by $\lambda\mathbb{E}\left[\boldsymbol{k}\boldsymbol{k}^T\right]$, where λ is a hyperparameter that balances the weights between new knowledge and preserved knowledge. The submatrix $\hat{\boldsymbol{K}}_0\hat{\boldsymbol{K}}_0^T$ is then obtained by selecting the rows and columns indexed by $\mathcal{I}$ from $\boldsymbol{K}_0\boldsymbol{K}_0^T$. Additionally, during the continuous editing process, the newly edited knowledge from each round will become the old knowledge for the next round. Therefore, after each round, we will add the newly edited knowledge into $\boldsymbol{K}_0\boldsymbol{K}_0^T$.

3.3 Iterative multi-layer Editing

As described in Section 2.3, MEMIT propagates edits through the layers by computing the value $\boldsymbol{v}_i^l$ as $\boldsymbol{v}_i^l {+}= \frac{\boldsymbol{\delta}_i}{l_0 - l + 1}$ $(l \in \mathcal{R})$ (Meng et al., 2023; Gupta et al., 2024b).Here, $\boldsymbol{\delta}_i = \boldsymbol{z}_i - \boldsymbol{h}_i^{l_0}$ represents the residual difference. The fundamental purpose is that as each layer is updated, the hidden state $\boldsymbol{h}_i^{l_0}$ progressively approaches the target $\boldsymbol{z}_i$, thereby diminishing the residual $\boldsymbol{\delta}_i$. Detailed analyses can be found in Appendix E. However, due to errors in the fitting process, some knowledge proves difficult to edit, resulting in $\boldsymbol{v}_i^{l_0}$ not sufficiently 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 $\boldsymbol{v}_i^{l_0}$ 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 $\|\boldsymbol{z}_i - \boldsymbol{h}_i^{l_0}\|^2$. If $\|\boldsymbol{z}_i - \boldsymbol{h}_i^{l_0}\|^2 < \alpha$, the knowledge sample is considered successfully edited. Conversely, if $\|\boldsymbol{z}_i - \boldsymbol{h}_i^{l_0}\|^2 > \alpha$, 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 out to form a new batch for further multi-layer editing, repeating this process until all knowledge samples in the batch meet $\|\boldsymbol{z}_i - \boldsymbol{h}_i^{l_0}\|^2 < \alpha$ or the iteration limit is reached. This iterative editing approach significantly enhances the success rate of sample edits, enabling NSE to effectively achieve massive knowledge updates in a single editing session. More details of the NSE algorithm are provided in Appendix A.

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
FT-W		61.23±0.38	62.40±0.24	47.05±0.41	492.34±0.23	3.57±0.03	32.08±0.35	31.43±0.23	14.72±0.16
MEND		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		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		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
GRACE		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		**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
FT-W		42.70±0.49	35.93±0.40	63.06±0.31	565.96±0.23	13.03±0.06	24.97±0.32	22.40±0.30	12.73±0.18
MEND		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		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		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
GRACE		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		**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
FT-W		48.35±0.49	31.42±0.39	68.71±0.28	587.20±0.23	29.41±0.09	39.81±0.36	32.55±0.33	27.76±0.26
MEND		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		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		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
GRACE		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		**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**

4 EXPERIMENTS

We conduct 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 model 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?
- **RQ4:** How does each individual component of NSE contribute to the overall editing performance?

4.1 EXPERIMENTAL SETTINGS

Datasets & Evaluation Metrics. To evaluate the effectiveness of our method, we utilize 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 on three autoregressive language models, GPT2-XL (1.5B) (Radford et al., 2019), GPT-J (6B)

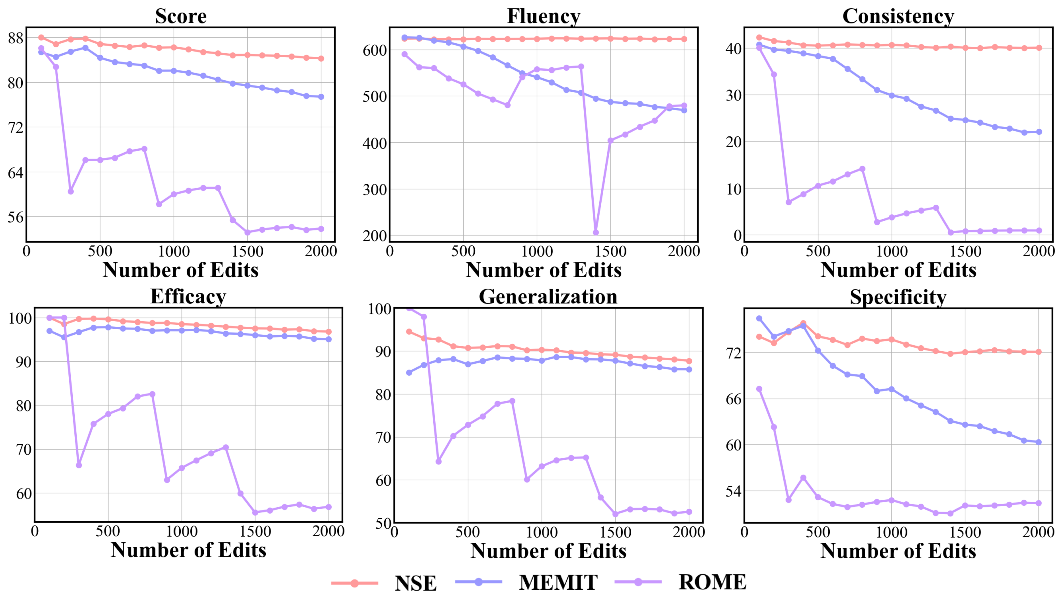

Figure 3: Editing performance of NSE and baselines with varying numbers of edits (batch size 100) in sequential editing, evaluated on the Counterfact dataset. Score is the harmonic mean of Efficacy, Generalization, and Specificity.

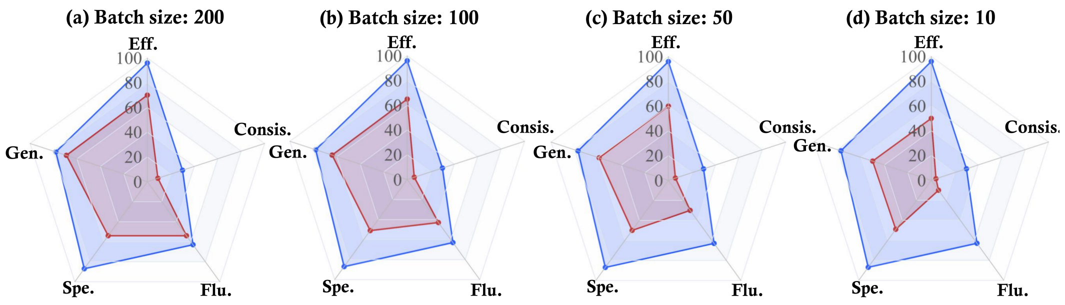

Figure 4: Editing performance of NSE and MEMIT with different batch size, evaluated on Llama3 (8B). The red line and the blue line represent MEMIT and NSE, respectively.

(Wang & Komatsuzaki, 2021) and Llama3 (8B)[1]. For baseline comparisons, we primarily select model editing methods that modify the model's parameters, including fine-tuning the specific layer (FT-L, FT-W) (Zhu et al., 2020), MEND (Mitchell et al., 2022a), ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023). Additionally, we incorporated a memory-based editing method, GRACE (Hartvigsen et al., 2023), as a baseline. Although GRACE does not update model parameters and differs from our focus on parameter-modification editing methods, we chose to include it as a baseline due to its outstanding performance in sequential editing. Further 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, using GPT2-XL, GPT-J and Llama3 models. 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), evaluated on the Counterfact and ZsRE datasets. The results, under all 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 the Counterfact dataset. We presented the result on ROME, MEMIT and NSE in Figure 3. We also provide a case study comparing the text generation

[1]https://llama.meta.com/lama3/

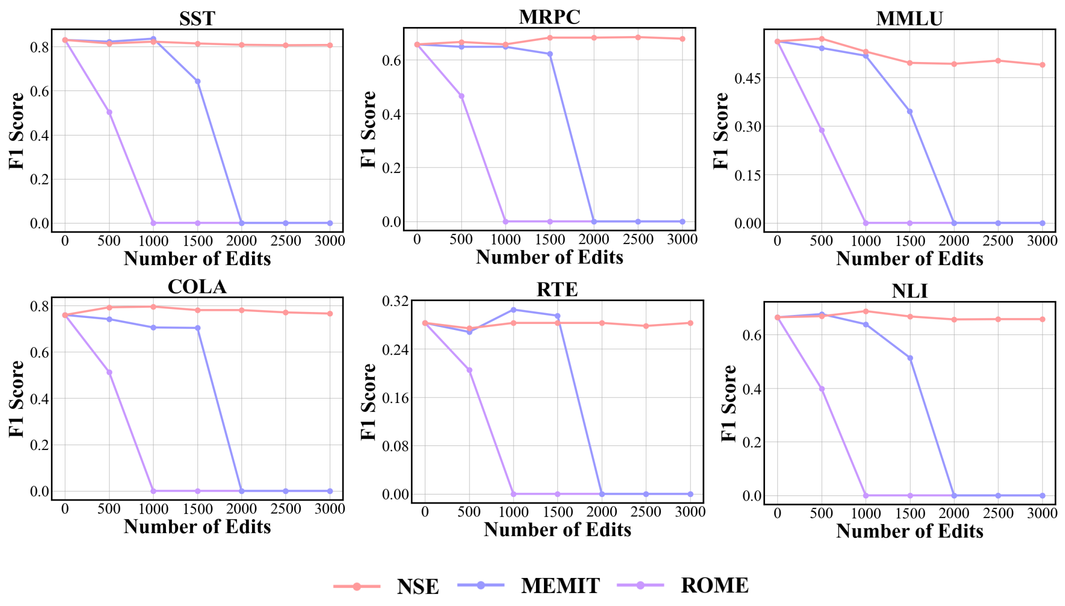

Figure 5: Performance on general tasks of edited models using NSE, ROME and MEMIT, with sequential editting on Llama3 (8B).

effects of different editing methods, with results available in Appendix F. According to these, we can find that:

- **Observation 1: NSE outperforms other baseline methods in almost all critical metrics across both datasets and models in the sequential editing task.** Specifically, compared to the parameter-modification baseline methods, NSE shows significant improvements across all metrics. Notably, on Llama3 (8B) editing, NSE achieves an average enhancement of around 30.33% across multiple metrics. Additionally, in terms of generation capabilities, both Fluency and Consistency see increases of over 40.75%. In contrast, while the parameter-preservation method GRACE retains model capabilities to the greatest extent, it exhibits weaker performance on the Generalization.
- **Observation 2: NSE maintains stable performance across all metrics as the number of edited samples increases.** As shown in Figure 3, NSE demonstrates robust performance, remaining resilient to model failure and forgetting 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 depicting the performance on Llama3 (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 G. 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 increased 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 the notable decreases observed in all metrics. In contrast, NSE demonstrates an average improvement of 45.60% across these metrics.

Table 2: 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

4.4 General Ability Test (RQ3)

To assess the effects of model editing on the general capabilities of large language models (LLMs), we have selected six natural language tasks from the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019). The specific downstream tasks 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 & 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 accuracy 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 linguistic theory literature. (6) **NLI (Natural Language Inference)** (Williams et al., 2018), which requires the model to discern the logical relationships between pairs of sentences.

We conduct evaluations on Llama3 (8B) based on sequential editing settings with 3000 edits (batch size 100). The results are shown in the Figure 5. Here we can make the 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 to 0 in general capabilities after editing approximately 1,000 to 2,000 samples, indicating that the model has already experienced degradation.

4.5 Ablation Study (RQ4)

To assess the contributions of individual components in our method, we conduct an ablation study on the GPT-XL, GPT-J and Llama3 (8B) model using the Counterfact dataset. The results are presented in Table 2. We can make the observations as following:

- **Observation 5: Weights rewinding for value computation in NSE can effectively mitigate model failure.** Specifically, after the ablation of the weights rewinding component, there is a significant decline in both Specificity and Fluency for 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.
- **Observation 6: 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 in terms of Efficacy and Generalization, the ablation of neuron-

level weights updates and iterative multi-layer editing leads to an average decrease of approximately $1\% - 2\%$, demonstrating that these modules further mitigate model forgetting.

5 Related work

Recent years have witnessed a development in methods for model editing, which are developed to adjust the behavior of LLMs within specific domains, while preserving their performance across other inputs. Current approaches to model editing in large language models (LLMs) generally fall into two main categories (Yao et al., 2023; Zhang et al., 2024b):

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. For instance, 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. Other methods like MemPrompt (Madaan et al., 2022) and IKE (Zheng et al., 2023) leverage the model's in-context learning by prompting it with edited facts and retrieved demonstrations from memory. MELO (Yu et al., 2024) dynamically activates corresponding LoRA blocks indexed in an inner vector database to alter the behavior of models. GRACE (Hartvigsen et al., 2023) implements sequential editing by maintaining a dynamically updated codebook. Larimar (Das et al., 2024) enhances LLMs with distributed episodic memory. OneEdit (Zhang et al., 2024a) introduces a neural-symbolic system for collaborative knowledge editing integrating knowledge graphs and LLMs.

Modify Models' Parameters. In contrast, methods that modify LLMs' parameters require updating the model's internal parameters with each edit. FT-W fine-tunes the specific layer with regularization constraints (Zhu et al., 2020). KN (Dai et al., 2022) identifies the "knowledge neuron" as the crucial pair in FFN and updates these neurons, which represents the knowledge. KE (Cao et al., 2021) and MEND (Mitchell et al., 2022a) use hypernetwork to forecast the weight change of LLMs based on the meta-learning methods. ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) support large-scale direct edits by locating and editing knowledge in specific layers of GPT. COMEBA-HK (Li et al., 2024) uses newly proposed hook layers to identify the editing scope supporting sequential editing. To enhance the performance of parameter modification methods in sequential editing, RECT (Gu et al., 2024) retains parameters with minimal changes to ensure stability, while PRUNE (Ma et al., 2024) constrains the maximum singular value of parameter changes. In this paper, we mainly focus on parameters-modification editing methods.

6 Limitation and Discussion

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, we aim to explore more effective neuron attribution methods and enhance the efficiency of our editing techniques.

7 Conclusion

In this work, we introduce NSE, a model editing method for sequential model editing tailored for addressing the significant challenges of model forgetting and model failure. Specifically, we propose weights 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 maintains model coherence. Additionally, we select influential neurons for different knowlege to update weights in FFN and iteratively edit multi-layer weights, which effectively mitigate model forget. Experimental results on the GPT2-XL, GPT-J and Llama3-8B models using the Counterfact and ZsRE datasets demonstrate that NSE significantly outperforms existing model editing baselines and the ablation study shows that each component in NSE is effective.

8 Ethics Statement

Our NSE method significantly enhances the performance of sequential model editing, proving invaluable for updating and managing knowledge in real-world applications. While the capability 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, aiming to contribute to the efficient updates of large models in the future. Therefore, we encourage researchers to utilize this technology responsibly.

9 Reproducibility

To ensure the reproducibility of our findings, detailed implementation instructions for NSE can be found in Appendix D. Additionally, the source code is available to the public at the following URL: https://anonymous.4open.science/r/NSE-0A8D/. These measures are taken to facilitate the verification and replication of our results by other researchers in the field.

References

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

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

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. Language models are few-shot learners. In *NeurIPS*, 2020.

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

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

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarath Swaminathan, Sihui Dai, Aurélie C. Lozano, Georgios Kollias, Vijil Chenthamarakshan, Jirí Navrátil, Soham Dan, and Pin-Yu Chen. Larimar: Large language models with episodic memory control. *CoRR*, abs/2403.11901, 2024.

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

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

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue, 2024. URL https://arxiv.org/abs/2401.04700.

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

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

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

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

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

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

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

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

Shuaiyi Li, Yang Deng, Deng Cai, Hongyuan Lu, Liang Chen, and Wai Lam. Consecutive model editing with batch alongside hook layers. *CoRR*, abs/2403.05330, 2024.

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

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. Memory-assisted prompt editing to improve GPT-3 after deployment. *CoRR*, abs/2201.06009, 2022.

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

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

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

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

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

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

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

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *EMNLP (1)*, pp. 5418–5426. Association for Computational Linguistics, 2020.

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

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

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

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

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

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

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

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

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

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

Lang Yu, Qin Chen, Jie Zhou, and Liang He. MELO: enhancing model editing with neuron-indexed dynamic lora. In *AAAI*, pp. 19449–19457. AAAI Press, 2024.

Ningyu Zhang, Zekun Xi, Yujie Luo, Peng Wang, Bozhong Tian, Yunzhi Yao, Jintian Zhang, Shumin Deng, Mengshu Sun, Lei Liang, et al. Oneedit: A neural-symbolic collaboratively knowledge editing system. *arXiv preprint arXiv:2409.07497*, 2024a.

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

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

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

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.

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 $f_{\theta_0}$, layers to edit
    $\mathcal{R} = \{l_0 - |\mathcal{R}| + 1, \ldots, l_0\}$, covariances $\hat{\boldsymbol{C}}^{l^{-1}}$
Output: Modified generator containing edits from $\{\mathbb{D}_{edit_t}\}_{t=1}^{T}$
 1: for each round $t = 1$ to $T$ do
 2:     for $(s_i, r_i, o_i) \in \mathbb{D}_{edit_t}$ do
 3:         // Compute target $\mathbf{z}_i$ vectors for every memory $i$
 4:         $\boldsymbol{z}_i \leftarrow \boldsymbol{v}_i^{l_0} + \boldsymbol{\delta}_i$
 5:         Optimize $\boldsymbol{z}_i = \boldsymbol{v}_i^{l_0} + \arg\min_{\boldsymbol{\delta}_i}(-\log \mathbb{P}_{f_{\boldsymbol{\theta}_0}(\boldsymbol{v}_i^{l_0} += \boldsymbol{\delta}_i)}[o_i \mid (s_i, r_i)])$      ▷ Eqn. 9
 6:     end for
 7:     // Select neurons for the current edits
 8:     for $(s_i, r_i, o_i) \in \mathbb{D}_{edit_t}$ do
 9:         Compute $\boldsymbol{Q}_i = |\boldsymbol{k}_i|$ for layer $l$
10:         Select neurons $\mathcal{I}_i^l$ by
11:         $\mathcal{I} = \arg\min_{\mathcal{I} \subseteq \{1,\ldots,N\}} |\mathcal{I}|$   s.t.   $\sum_{j \in \mathcal{I}} Q_{ij} \geq p \times \sum_{j=1}^{N} Q_{ij}$      ▷ Eqn. 10
12:     end for
13:     for $l \in \mathcal{R}$ do      ▷ Perform update: spread changes over layers
14:         while $\mathbf{z}_i - \boldsymbol{h}_i^L > \alpha$ and maximum iterations not reached do
15:             // Re-run the module
16:             $\boldsymbol{h}_i^l \leftarrow \boldsymbol{h}_i^l + \boldsymbol{a}_i^l + \boldsymbol{v}_i^l$
17:             Run layer $l$ with updated weights
18:             for $(s_i, r_i, o_i) \in \mathbb{D}_{edit_t}$ do
19:                 $\mathbf{r}_i^l \leftarrow \frac{\mathbf{z}_i - \boldsymbol{h}_i^L}{l_0 - l + 1}$
20:             end for
21:             $\hat{\boldsymbol{W}}^l \leftarrow \boldsymbol{W}^l[\mathcal{I}]$
22:             $\hat{\boldsymbol{\Delta}}^l \leftarrow \boldsymbol{\Delta}^l[\mathcal{I}]$
23:             $\hat{\boldsymbol{K}}_2^l \leftarrow \boldsymbol{K}_2^l[\mathcal{I}]$
24:             Distribute residual over remaining layers
25:             $\hat{\boldsymbol{\Delta}}^l \leftarrow \hat{\boldsymbol{R}}^l \hat{\boldsymbol{K}}_2^{l^T} \hat{\boldsymbol{C}}^{l^{-1}}$      ▷ Eqn. 12
26:             $\hat{\boldsymbol{W}}^l \leftarrow \hat{\boldsymbol{W}}^l + \hat{\boldsymbol{\Delta}}^l$      ▷ Neuron-level updating layer $l$ MLP weights in model
27:             Increment iteration counter
28:         end while
29:     end for
30: end for
```

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). It 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 for efficiently incorporating 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.
- **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 challenging dataset that contrasts counterfactual 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 dataset has similar metrics 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 ZsRE Metrics

Following the previous work (Mitchell et al., 2022a; Meng et al., 2022; 2023), this section defines each ZsRE 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 is calculated as the average top-1 accuracy on the edit samples:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid (s_i, r_i) \right\} . \tag{13}$$

- **Generalization**: Generalization measures the model's performance on 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

$N((s_i, r_i))$:

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

- **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:

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

C.3 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:

$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i \mid (s_i, r_i)] > \mathbb{P}_{f_\theta}[o_c^i \mid (s_i, r_i)] \right]. \tag{16}$$

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

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

- **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:

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

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

$$-\frac{2}{3} \sum_k g_2(k) \log_2 g_2(k) + \frac{4}{3} \sum_k g_3(k) \log_2 g_3(k), \tag{19}$$

  where $g_n(\cdot)$ is the n-gram frequency distribution.
- **Consistency (reference score)**: The consistency of the model's outputs 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 reference Wikipedia text about o.

D 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).

D.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 computed using 100,000 samples from Wikitext in fp32, with the hyperparameter λ set to 20,000. During the process of computing $\mathbf{z}_i$, 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).

D.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.

D.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.

D.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 3.

Table 3: Times per edit for various methods evaluated 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-w/o iterative editing	2.40s	4.63s	5.46s

From the Table 3, 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.

E Analysis of multi-layer editing approach in MEMIT

Firstly, we decompose the hidden state $\boldsymbol{h}_i^{l_0}$ of the l_0-th layer in the Transformer architecture as follows:

$$\boldsymbol{h}_i^{l_0} = \boldsymbol{h}_i^{l} + \sum_{j=l}^{l_0} \left[\boldsymbol{a}_i^{j}(\boldsymbol{h}_i^{l}) + \boldsymbol{v}_i^{j}(\boldsymbol{h}_i^{l}) \right], \tag{20}$$

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 $\boldsymbol{\delta}_i = \boldsymbol{z}_i - \boldsymbol{h}_i^{l_0}$, after applying the editing multi-layer algorithm at layer l and assuming that the optimization in Eqn. 11 fits perfectly, the hidden state at layer L is updated as:

$$\boldsymbol{h}_i^{l_0} \leftarrow \boldsymbol{h}_i^{l} + \frac{\boldsymbol{\delta}_i}{l_0 - l + 1} + \sum_{j=l}^{l_0} \left[\boldsymbol{a}_i^{j}(\boldsymbol{h}_i^{l} + \frac{\boldsymbol{\delta}_i}{l_0 - l + 1}) + \boldsymbol{v}_i^{j}(\boldsymbol{h}_i^{l} + \frac{\boldsymbol{\delta}_i}{l_0 - l + 1}) \right]. \tag{21}$$

Substituting into Eqn. 20, and subtracting $\boldsymbol{z}_i$ from both sides, we obtain:

$$\boldsymbol{\delta}_i \leftarrow \frac{(l_0 - l)\boldsymbol{\delta}_i}{l_0 - l + 1} + \sum_{j=l}^{l_0} \left[\boldsymbol{a}_i^{j}(\boldsymbol{h}_i^{l}) - \boldsymbol{a}_i^{j}(\boldsymbol{h}_i^{l} + \frac{\boldsymbol{\delta}_i}{l_0 - l + 1}) + \boldsymbol{v}_i^{j}(\boldsymbol{h}_i^{l}) - \boldsymbol{v}_i^{j}(\boldsymbol{h}_i^{l} + \frac{\boldsymbol{\delta}_i}{l_0 - l + 1}) \right]. \tag{22}$$

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 $\frac{(l_0 - l + 1)\boldsymbol{\delta}_i^{(0)}}{|\mathcal{R}|}$, 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. Each layer's edit still nudges $\boldsymbol{h}_i^{l_0}$ closer to $\boldsymbol{z}_i$, but the ignored errors mean a single execution of the editing distribution algorithm often fails to sufficiently approximate $\boldsymbol{h}_i^{l_0}$ to $\boldsymbol{z}_i$.

F 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 model after performing sequential editing with 2000 total editing samples and a batch size of 100. The results are shown in Table 4, Table 5 and Table 6. 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 demonstrate that ROME failed to include the target output "Romania" in its generation, and the model's output became incoherent and unreadable. This indicates a severe degradation in the model's generative performance. MEMIT, although successful in editing, produced output that repeatedly mentioned the target "Romania", which also signifies 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.

Table 4: 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 5: 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 6: 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

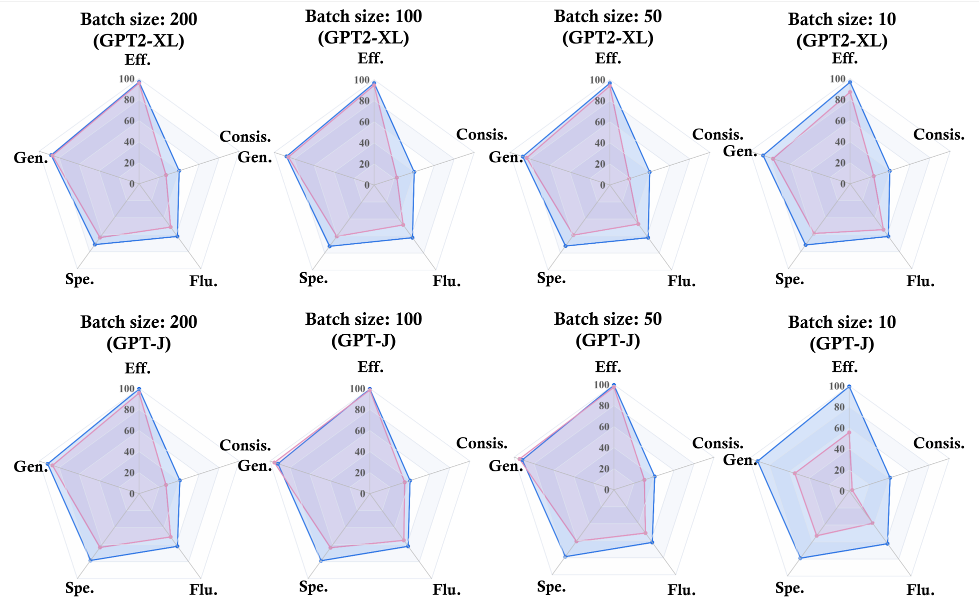

Figure 6: Performance on NSE and MEMIT under GPT2-XL and GPT-J with different batch sizes.The red line and the blue line represent MEMIT and NSE, respectively

Table 7: 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$_{\pm0.07}$	91.28$_{\pm0.21}$	77.82$_{\pm0.25}$	619.25$_{\pm0.17}$	40.82$_{\pm0.12}$	96.80$_{\pm0.14}$	**92.19$_{\pm0.21}$**	28.14$_{\pm0.25}$
0.8	**99.55$_{\pm0.06}$**	**91.92$_{\pm0.22}$**	78.96$_{\pm0.25}$	**620.49$_{\pm0.16}$**	40.24$_{\pm0.12}$	**96.87$_{\pm0.14}$**	91.33$_{\pm0.22}$	**28.66$_{\pm0.25}$**
0.75	99.45$_{\pm0.07}$	91.68$_{\pm0.22}$	**79.03$_{\pm0.24}$**	620.42$_{\pm0.16}$	**40.83$_{\pm0.12}$**	96.80$_{\pm0.14}$	91.66$_{\pm0.22}$	27.68$_{\pm0.25}$

G More Quantitative Results

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, under different batch sizes in sequential editing, with a total of 2000 editing samples.

Additionally, Table 7 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 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 Figure 7 and 8. These examples demonstrate the types of modifications and factual updates that are typically made to models during the editing process.

```
{
    "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.