LyapLock: Bounded Knowledge Preservation in Sequential Large Language Model Editing

Peng Wang[1,2], Biyu Zhou[1]*, Xuehai Tang[1], Jizhong Han[1], Songlin Hu[1,2]*

[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
Correspondence: {wangpeng2022, zhoubiyu, tangxuehai, hanjizhong, husonglin}@iie.ac.cn

Abstract

Large Language Models often contain factually incorrect or outdated knowledge, giving rise to model editing methods for precise knowledge updates. However, current mainstream locate-then-edit approaches exhibit a progressive performance decline during sequential editing, due to inadequate mechanisms for long-term knowledge preservation. To tackle this, we model the sequential editing as a constrained stochastic programming. Given the challenges posed by the cumulative preservation error constraint and the gradually revealed editing tasks, **LyapLock** is proposed. It integrates queuing theory and Lyapunov optimization to decompose the long-term constrained programming into tractable stepwise subproblems for efficient solving. This is the first model editing framework with rigorous theoretical guarantees, achieving asymptotic optimal editing performance while meeting the constraints of long-term knowledge preservation. Experimental results show that our framework scales sequential editing capacity to over 10,000 edits while stabilizing general capabilities and boosting average editing efficacy by 11.89% over SOTA baselines. Furthermore, it can be leveraged to enhance the performance of baseline methods. Our code is released on https://github.com/caskcsg/LyapLock.

1 Introduction

Large Language Models (LLMs), with their powerful capabilities in knowledge storage and recall, have become a research hotspot in the field of natural language processing (Brown et al., 2020; Huang et al., 2022; Liu et al., 2024). However, studies reveal that the knowledge acquired by LLMs during the pre-training phase may contain incorrect information or outdated content (Cao et al., 2021; Mitchell et al., 2022a). This makes the updating of model knowledge an urgent and critical issue to be addressed. Traditional solutions, such as repretraining or full-parameter fine-tuning, can facilitate knowledge updates. However, the prohibitive computational costs severely limit their practical applications (Gupta et al., 2023; Yao et al., 2023).

Recent years have witnessed growing interest in low-cost knowledge updating through model editing techniques (Wang et al., 2025). Among these, the locate-then-edit paradigm, exemplified by ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), has emerged as the mainstream framework, owing to its demonstrated advantages in editing efficiency and precision. This paradigm operates through two key phases: (1) identifying the critical parameter subset W associated with target knowledge via causal tracing analysis, and (2) achieving the update of the target knowledge within the parameter space by computing and implementing appropriate perturbations Δ.

To prevent unintended degradation of pretrained knowledge during target knowledge updates, perturbation strategies necessitate meticulous design. The prevailing approach (Meng et al., 2022, 2023) involves constructing and solving a bi-objective loss function that integrates preservation loss and editing loss to achieve optimized knowledge updating. The former maintains the stability of knowledge representations intended for retention, while the latter ensures accurate updating of target knowledge. However, as the preservation loss serves merely as a soft constraint, the model's capability to retain knowledge and generate fluent sentences after editing is prone to instability. Recent studies attempt to alleviate this issue by imposing supplementary constraints (e.g., regularized weight updating in RECT (Gu et al., 2024) and null space projection in AlphaEdit (Fang et al., 2025)) during parameter search processes. Nevertheless, these approaches remain inherently restricted by their heuristic nature.

Furthermore, existing methods have largely fo-

*Corresponding authors.

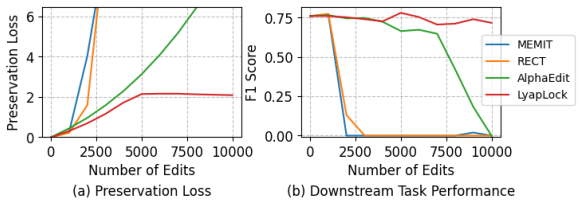


Figure 1: Comparison of preservation loss and downstream task performance of LLaMA3 (Meta, 2024) during sequential editing of 10,000 samples using current methods and LyapLock (details in Sec.4).

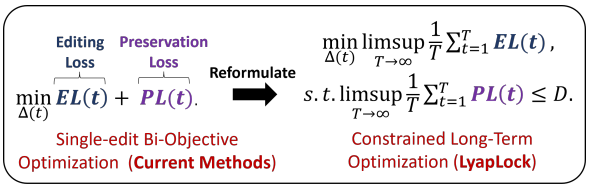


Figure 2: A formal comparison between LyapLock and current methods.

cused on single-edit incremental optimization for the immediate editing state, lacking a rigorous theoretical framework to regulate the long-term cumulative trends of successive edits in practical deployment scenarios (Hartvigsen et al., 2023; Wang et al., 2024). As a result, the accumulation of preservation loss inevitably erodes model stability during sequential editing operations, ultimately leading to model forgetting and collapse (Fang et al., 2025; Gupta et al., 2024). Our experimental results show that as the edit count increases, model parameters gradually deviate from the initial values, evidenced by a monotonic increase in preservation loss (as in Figure 1(a)). After 10,000 consecutive edits, the performance of downstream tasks exhibits near-complete degradation (as in Figure 1(b)).

To address these challenges, this paper reformulates the conventional bi-objective optimization problem into a constrained long-term optimization problem for sequential editing. The objective is to minimize the long-term editing loss under the constraint of cumulative preservation loss, as shown in Figure 2. However, due to the uncertainty of subsequent editing tasks and the preservation loss constraint, achieving a global optimum for this stochastic programming problem poses a significant challenge. To this end, we propose **LyapLock**, the first framework providing theoretical stability guarantees for sequential model editing through a Lyapunov-driven formulation. Through rigorous theoretical proofs, we have demonstrated that it achieves asymptotically near-optimal editing performance while satisfying long-term preservation loss constraints.

To validate effectiveness, extensive experiments are conducted on representative LLMs, including GPT-2 XL(Radford et al., 2019), GPT-J(Wang and Komatsuzaki, 2021), and LLaMA-3-8B(Meta, 2024). Results demonstrate that after sequentially editing 10,000 samples, our method achieves 11.89% improvement in editing performance compared to the best baseline (94.41% vs. 82.52%), while maintaining stable performance across multiple downstream tasks (baseline methods all degrade by 100%). Notably, our method exhibits exceptional scalability — when the editing scale extends to 20,000, the model still maintains its general capability. In addition, our method is compatible with existing knowledge editing methods and can improve their editing performance by 9.76% and downstream task performance by 32.63%.

2 Preliminary

2.1 Hidden States of LLMs

LLMs typically consist of an embedding layer, L decoder layers, and an output layer. A decoder layer has an attention module (Attn) and a multi-layer perceptron (MLP) module. Given the structural diversity of LLMs, low-level variations (e.g., residual connections, normalization, and biases) are omitted for brevity in this paper. For a input x, the hidden state h^l at the l-th layer is expressed as:

$$\begin{aligned} h^l &= h^{l-1} + a^l + m^l,\ a^l = \text{Attn}^l(h^{l-1}), \\ m^l &= \text{MLP}^l(a^l) = W^l_{\text{out}}\text{act}(W^l_{\text{in}}(a^l + h^{l-1})) \end{aligned} \tag{1}$$

Here, a^l and m^l are the outputs of Attn^l and MLP^l. MLP^l contains two linear layers with parameters $W^l_{\text{in}} \in \mathbb{R}^{d_0 \times d_1}$ and $W^l_{\text{out}} \in \mathbb{R}^{d_1 \times d_0}$, where d_0 is the intermediate dimension and d_1 is input/output dimension of MLP^l. $\text{act}(\cdot)$ denotes a specific activation function, which varies across different LLMs.

2.2 Model Editing in LLMs

2.2.1 Knowledge Storage

According to (Kohonen, 1972; Geva et al., 2021), any linear operation can be viewed as a form of key-value pair storage. Consequently, the second-layer parameters W^l_{out} in the MLP layer can be interpreted as a linear associative memory module:

$$\underbrace{m^l}_{v} = W^l_{\text{out}} \underbrace{\text{act}(W^l_{\text{in}}(a^l + h^{l-1}))}_{k} \tag{2}$$

Typically, factual knowledge stored in LLMs can be formalized as a knowledge triple (s, r, o) composed of a subject s, relation r, and object o (Meng et al., 2022, 2023). For example, the fact *"Beats Music is owned by Apple."* is formalized as s = "Beats Music", r = "is owned by", and o = "Apple". Here, W^l_{out} associates a key k encoding (s, r) with a value v encoding o. Based on this perspective, editing factual knowledge in LLMs can be achieved by modifying the parameters of W^l_{out} (hereafter denoted as W). Specifically, each edit operation updates the model parameters by adding a perturbation Δ to W, thereby reconstructing the association between k and v to implement knowledge updates.

2.2.2 Sequential Editing

In practical applications, sequential knowledge updates to the model are often required (Hartvigsen et al., 2023; Wang et al., 2024; Fang et al., 2025). Specifically, given T batches of new knowledge $\{S_1, S_2, \ldots, S_T\}$ to be updated into LLMs, where each S_t contains n new facts (i.e., $S_t = \{(s^1_t, r^1_t, o^1_t), (s^2_t, r^2_t, o^2_t), \ldots, (s^n_t, r^n_t, o^n_t)\}$). Assume that each edit occurs at a timestamp that is a positive integer. Sequential editing involves associating all corresponding new key-value pairs k^i_t-v^i_t ($i \in \{1, 2, \ldots, n\}$) in S_t by adding a perturbation $\Delta(t)$ to the updated model parameters from the previous timestamp $W(t-1)$ at each timestamp $t \in \{1, 2, \ldots, T\}$. Through this process, the model parameters are sequentially updated.

Formally, for the t-th timestamp (i.e., the t-th edit), we represent the current batch of new knowledge S_t as key-value matrices:

$$\begin{aligned} K_1(t) &= \left[k^1_t \mid k^2_t \mid \ldots \mid k^n_t\right] \in \mathbb{R}^{d_0 \times n}, \\ V_1(t) &= \left[v^1_t \mid v^2_t \mid \ldots \mid v^n_t\right] \in \mathbb{R}^{d_1 \times n} \end{aligned} \tag{3}$$

Let $W(0)$ representing the original parameters. Correspondingly, the preserved knowledge in $W(0)$ can be expressed as key-value matrices $K_0(0)$ and $V_0(0)$, hereafter denoted as K_0 and V_0. The mainstream locate-then-edit methods solve the perturbation $\Delta(t)$ by jointly optimizing the following bi-objective loss function:

$$\min_{\Delta(t)} EL(t) + PL(t). \tag{4}$$

The above $EL(t)$ and $PL(t)$ are the **editing loss** and the **preservation loss** of the model after editing at timestamp t, respectively, where $EL(t) = \|[W(t-1) + \Delta(t)]K_1(t) - V_1(t)\|^2_F$ and $PL(t) = \|[W(t-1) + \Delta(t)]K_0 - V_0\|^2_F$. Here, $\|\cdot\|^2_F$ denotes the squared Frobenius norm. The **editing loss** ensures accurate updates for target knowledge, while the **preservation loss** preserves the integrity of the to-be-reserved knowledge. By applying the normal equations (Johnson et al., 2004), a closed-form solution of the formula 4 can be derived. After obtaining $\Delta(t)$, the model parameters are updated as:

$$W(t) = W(t-1) + \Delta(t). \tag{5}$$

By repeating this process at each timestamp t, sequential editing is achieved, enabling the model to progressively incorporate all T batches of new knowledge.

3 The LyapLock Framework

3.1 Constrained Sequence Editing Optimization Problem Formulation

As shown in Figure 1, the traditional bi-objective optimization Problem 4 leads to continuous accumulation of preservation loss with increasing edit operations, eventually resulting in model collapse. Therefore, we reformulate Problem 4 as a constrained long-term optimization problem that restricts the preservation loss within a certain threshold. The specific formulation is as follows:

$$\begin{aligned} &\min_{\Delta(t)} \limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} EL(t), \\ &\text{s.t.} \limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} PL(t) \le D. \end{aligned} \tag{6}$$

Here, D represents the average preservation loss over the time horizon, i.e., after t consecutive edits, the average preservation loss over the time interval $[1, t]$ should be constrained within D.

3.2 Problem Transformation Using Lyapunov Optimization Theory

The key challenge in solving problem 6 lies in minimizing long-term editing loss while maintaining the preservation loss below threshold D, given the highly stochastic and unpredictable $(K_1(t), V_1(t))$ pair across timesteps t. To this end, we introduce virtual queues to transform the constrained satisfaction into a well-studied queue stability problem. Building on this, by applying Lyapunov optimization from control theory (Neely, 2010) (The innovative discussion can be found in Appendix B),

we further decompose the long-term optimization into per-timestamp subproblems that can be solved at each timestamp t. This ensures queue stability during online decision-making without requiring future information or statistical knowledge of uncertainties. Next, we will elaborate on the details of this transformation. We first design a virtual queue $Z(t)$, initialized as $Z(1) = Z_{\text{init}}$, with its update rule at each timestamp t given by Equation (7):

$$Z(t+1) = \max\left[Z(t) + a(PL(t) - D) + b, Z_{\max}\right], \tag{7}$$

where $Z_{\max} \geq 0, b \geq 0, a > 0$. Intuitively, the value of $Z(t)$ reflects the deviation between the actual average preservation loss and D over the historical time interval $[0, t-1]$. An increase in $Z(t)$ corresponds to persistent violation of the constraint. It can be theoretically proven that if the virtual queue satisfies the strong stability condition $\lim_{T\to\infty} \frac{Z(T)}{T} = 0$, the constraint in problem 6 holds (detailed proof is provided in Appendix C.1).

To analyze the stability of the queue, we construct a quadratic Lyapunov function:

$$L(Z(t)) = \frac{1}{2} Z(t)^2, \tag{8}$$

where $L(Z(t))$ represents the congestion level of the virtual queue. For example, a smaller value indicates lower queue backlog and stronger stability. To continuously drive $L(Z(t))$ toward lower congestion and ensure strong queue stability, we define the one-step conditional Lyapunov drift (Neely, 2010):

$$\Delta(Z(t)) = \left\{L(Z(t+1)) - L(Z(t)) \mid Z(t)\right\}. \tag{9}$$

Within the Lyapunov optimization framework, seeking the optimal solution of problem 6 is equivalent to minimizing the following expression 10 at each timestamp t:

$$\min_{\Delta(t)} V \cdot EL(t) + \Delta(Z(t)). \tag{10}$$

Here, the control parameter $V \geq 0$ balances editing performance and queue stability: increasing V approaches the theoretical optimal editing performance but reduces queue stability, while decreasing V enhances constraint satisfaction at the cost of editing performance. Since $\Delta(Z(t))$ contains $\max[\cdot]$, direct optimization of problem 10 is challenging. Therefore, we can optimize by minimizing the upper bound of equation 10 (for the derivation of the upper bound, see Appendix C.3), that is:

$$\min_{\Delta(t)} V \cdot EL(t) + aZ(t)PL(t). \tag{11}$$

Now the original long-term optimization problem 6 is decomposed into stepwise subproblems at each timestamp t.

3.3 Stepwise Editing with Long-term Guarantees

After transforming Problem 6 into per-timestamp subproblems, we are now seeking to solve for the optimal disturbance. Prior to this, we further refine this formal expression following the previous work (Fang et al., 2025). It has been revealed that, to ensure that the model does not forget the knowledge that has been edited before, the key value matrix of the knowledge edited before timestamp t, denoted by $K_p(t)$ and $V_p(t)$, should be incorporated into the optimization objective, where $K_p(t)$ and $V_p(t)$ are matrices composed of $[K_1(1) \mid \ldots \mid K_1(t-1)]$ and $[V_1(1) \mid \ldots \mid V_1(t-1)]$, respectively. That is:

$$\min_{\Delta(t)} V(EL(t) + BL(t)) + aZ(t)PL(t), \tag{12}$$

where $BL(t)$ denotes the edit loss of the model with respect to all the knowledge that has been edited prior to time t. It is feasible to directly derive the closed-form solution of problem 12 as:

$$\begin{aligned}\Delta(t) = \Big[& V\Big(V_1(t) - W(t-1)K_1(t)\Big)K_1(t)^T \\ & + V\Big(V_p(t) - W(t-1)K_p(t)\Big)K_p(t)^T \\ & + aZ(t)\Big(V_0 - W(t-1)K_0\Big)K_0^T\Big] C(t)^{-1}\end{aligned} \tag{13}$$

Here, $C(t)$ is defined as $VK_1(t)K_1(t)^T + VK_p(t)K_p(t)^T + aZ(t)K_0K_0^T$. As can be seen from Equation 13, once K_0, V_0, $K_1(t)$, and $V_1(t)$ are obtained, the perturbation $\Delta(t)$ can be calculated. For details on computing these components, refer to Appendix A.

Now we are able to compute the perturbation term expression 13 and update the virtual queue according to expression 7 step by step until completing T sequential editing operations. The detailed optimization procedure is summarized in Algorithm 1.

Regarding the setting of hyperparameters, we have the following three considerations: (1) To set a more appropriate threshold D for different LLMs, we collect the model's preservation loss after one edit as the baseline D_{base}, and adjust D through different α values, i.e., $D = \alpha D_{\text{base}}$, indicating the threshold is set to α times the baseline. (2) Since parameters a and b in the virtual queue update formula 7 control the mapping relationship between preservation loss and queue value $Z(t)$, with $aZ(t)$ governing the weight of preservation loss in formulation 12, we achieve the following by setting $a = \frac{1}{\sqrt{D}}$ and $b = 0$: when the model's preservation loss exceeds the threshold D by one fold after an edit, the weight of preservation loss in 12 doubles. (3) Simultaneously, we set $z_{\text{init}} = \sqrt{D}$, $z_{\max} = \sqrt{D}$, and $V = 1$ to ensure that when the constraints in equation 7 are not violated, the preservation loss and editing loss in formulation 12 are calculated with a 1:1 weight ratio.

Algorithm 1: Stepwise Editing with Long-term Guarantees

Initialization: Given hyperparameter α, base model $W(0) = W$. Compute K_0, V_0, D_{base}. Set $D = \alpha \cdot D_{\text{base}}$, $a = \frac{1}{\sqrt{D}}$, $b = 0$, $z_{\text{init}} = \sqrt{D}$, $z_{\max} = \sqrt{D}$, $V = 1$.

for $t = 1$ **to** T **do**

1) **Real-time Optimization**:
   - Obtain $Z(t)$, $W(t-1)$, $K_1(t)$, $V_1(t)$.
   - Compute $\Delta(t)$ via Eq. (13).
   - Update model: $W(t) = W(t-1) + \Delta(t)$.

2) **Queue Update**:
   - Update $Z(t+1)$ via Eq. (7).

end

4 Experiment

4.1 Setting

Base LLMs. We selected three representative LLMs commonly used in the field of knowledge editing: **GPT2-XL (1.5B)**(Radford et al., 2019), **GPT-J (6B)**(Wang and Komatsuzaki, 2021), and **LLaMA3 (8B)**(Meta, 2024).

Baseline Methods. To compare with our method, we chose the representative model editing methods in the locate-then-edit approach, namely **ROME**(Meng et al., 2022) and **MEMIT**(Meng et al., 2023), as well as methods focusing on addressing the challenges faced by such approaches in sequential editing scenarios, namely **RECT**(Gu et al., 2024), **PRUNE**(Ma et al., 2025), and **AlphaEdit**(Fang et al., 2025), and the fine-tuning method **FT**. For detailed introductions to these methods, see Appendix D.1.

Datasets. We adopted two representative benchmarks in the field of model editing: **Counterfact**(Meng et al., 2022) and **ZsRE**(Levy et al., 2017). For introductions to these datasets, see Appendix D.2.

Metrics. Following prior works(Meng et al., 2022, 2023; Fang et al., 2025), we adopt the metrics for evaluating knowledge updating ability: **Efficacy** (efficiency success) and **Generalization** (paraphrase success); for assessing knowledge preservation ability: **Specificity** (neighborhood success); and for evaluating generation quality: **Fluency** (generation entropy) and **Consistency** (reference score). The specific calculation formulas are provided in Appendix D.3.

For more detailed experimental settings and time cost, see Appendix D.4 and Appendix D.5.

4.2 Editing Performance Results

We randomly select 10,000 samples for evaluation under the batch sequential editing scenario, with 100 edits per batch. Table 1 compares editing performance across various LLMs, datasets, and baseline methods. Results demonstrate LyapLock's comprehensive superiority in cross-model and cross-dataset scenarios across three dimensions: **(1) Knowledge Updating**: On Efficacy and Generalization, LyapLock outperforms the second-best method AlphaEdit by average margins of 11.88% and 12.69%, with gaps expanding to 22.01% and 19.71% on LLaMA3-Counterfact and 29.63% and 27.59% on GPT2-XL-ZsRE scenarios, respectively. **(2) Knowledge Preservation**: For Specificity, LyapLock outperforms the second-best baseline AlphaEdit by an average margin of 6.72%. LyapLock is the closest to the pre-edited performance, especially on the ZsRE dataset, where it only drops by an average of 1.4%, validating its effective preservation of original knowledge. **(3) Generation Quality**: In Fluency and Consistency, LyapLock significantly outperforms baselines. Specifically, the Fluency of LyapLock can reach an average of 604.14, which is only a 4% drop compared to the pre-edited models, while its Consistency is

Table 1: Performance results of sequential editing task (10,000 Samples). Here, the abbreviations *Eff.* (Efficacy), *Gen.* (Generalization), *Spe.* (Specificity), *Flu.* (Fluency), and *Consis.* (Consistency) are employed to denote respective evaluation metrics. Top-performing results are emphasized using bold formatting, with secondary superior results distinguished through underlined notation.

Method	Model	Counterfact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
Pre-edited		7.02±0.26	9.44±0.25	89.73±0.18	635.47±0.11	24.24±0.09	35.67±0.30	34.81±0.30	31.83±0.22
FT	LLaMA3	94.04±0.24	**84.13±0.31**	38.15±0.36	401.45±0.69	21.35±0.12	17.79±0.22	17.36±0.22	6.30±0.11
ROME	LLaMA3	68.45±0.46	61.13±0.38	48.30±0.28	505.00±0.14	3.88±0.02	1.14±0.06	1.05±0.06	0.15±0.02
MEMIT	LLaMA3	49.42±0.50	48.78±0.46	51.47±0.44	499.28±0.08	1.98±0.01	0.00±0.00	0.00±0.00	0.04±0.01
PRUNE	LLaMA3	50.12±0.50	49.20±0.45	51.18±0.43	509.27±0.08	1.81±0.01	0.00±0.00	0.00±0.00	0.04±0.01
RECT	LLaMA3	54.58±0.50	52.01±0.44	49.41±0.39	176.30±0.25	3.07±0.03	0.00±0.00	0.00±0.00	0.00±0.00
AlphaEdit	LLaMA3	72.60±0.45	61.97±0.41	52.98±0.33	420.84±0.54	6.24±0.07	91.79±0.17	87.16±0.23	30.39±0.22
LyapLock	LLaMA3	**94.61±0.23**	81.68±0.34	**69.01±0.30**	**617.04±0.24**	**30.70±0.12**	**94.34±0.13**	**90.20±0.20**	**30.74±0.22**
Pre-edited		15.22±0.36	17.65±0.33	83.50±0.25	622.01±0.14	29.61±0.10	26.45±0.28	25.74±0.28	27.04±0.26
FT	GPT-J	94.56±0.23	77.04±0.36	40.71±0.37	327.71±0.86	11.11±0.13	61.82±0.35	59.24±0.36	13.53±0.19
ROME	GPT-J	48.71±0.50	49.70±0.40	52.49±0.30	614.77±0.08	2.85±0.01	17.99±0.31	16.50±0.30	0.82±0.04
MEMIT	GPT-J	51.62±0.50	51.05±0.41	51.78±0.35	553.31±0.17	0.64±0.02	0.04±0.01	0.03±0.01	0.03±0.01
PRUNE	GPT-J	51.27±0.50	50.54±0.40	52.60±0.33	535.22±0.14	1.36±0.03	0.03±0.01	0.02±0.01	0.05±0.01
RECT	GPT-J	50.42±0.50	49.23±0.45	54.82±0.40	455.05±0.60	2.57±0.05	41.89±0.39	39.29±0.38	20.17±0.23
AlphaEdit	GPT-J	89.90±0.30	75.41±0.35	58.79±0.27	347.89±0.52	1.71±0.03	93.10±0.19	85.09±0.28	22.88±0.24
LyapLock	GPT-J	**99.00±0.10**	**88.80±0.27**	**68.21±0.28**	**618.33±0.18**	**40.93±0.12**	**98.77±0.08**	**93.82±0.19**	**25.51±0.25**
Pre-edited		21.82±0.41	24.16±0.37	78.32±0.28	626.69±0.12	31.34±0.10	22.17±0.26	21.28±0.26	24.20±0.24
FT	GPT2-XL	72.79±0.45	55.90±0.43	49.23±0.37	**607.94±0.22**	13.05±0.05	15.28±0.32	13.64±0.32	1.24±0.06
ROME	GPT2-XL	50.03±0.50	49.42±0.41	51.49±0.33	571.45±0.17	1.17±0.01	20.51±0.35	18.08±0.33	1.63±0.07
MEMIT	GPT2-XL	67.73±0.47	60.92±0.41	56.00±0.33	518.00±0.84	7.13±0.10	1.78±0.19	1.62±0.08	1.30±0.05
PRUNE	GPT2-XL	60.82±0.49	56.47±0.41	52.70±0.35	602.01±0.15	11.53±0.07	0.09±0.01	0.11±0.02	0.47±0.03
RECT	GPT2-XL	84.93±0.36	66.45±0.39	56.42±0.33	542.92±0.75	12.23±0.13	31.73±0.36	28.22±0.34	11.82±0.17
AlphaEdit	GPT2-XL	92.42±0.26	76.83±0.33	56.86±0.29	583.27±0.29	31.83±0.13	55.33±0.42	46.90±0.41	14.63±0.19
LyapLock	GPT2-XL	**94.76±0.22**	**80.51±0.33**	**60.74±0.29**	577.06±0.42	**34.29±0.13**	**84.96±0.28**	**74.49±0.35**	**22.63±0.24**

improved by an average of 6.97 compared to the pre-edited models. These advantages stem from LyapLock's unique preservation loss control mechanism, which optimally balances knowledge updating and preservation during sequential editing. The case studies in Appendix E.3 illustrate the specific output performance of the various editing methods. Appendix E.1 also provides the editing performance results for sequential editing of 2,000 samples and 5,000 samples. To further validate the effectiveness of our approach, we broaden the experimental datasets to a wider scope, employing the MQuAKE-CF dataset for multi-hop evaluations and the QAEdit dataset for real-world testing. Additional details are provided in Appendix E.2.

4.3 General Capability Tests

Now we assess the model's general capabilities using six subtasks from the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) (see Appendix D.6 for details) in line with (Fang et al., 2025). During the experiment, we conducted a test after every sequential editing of 1,000 samples. The GLUE performance results of the LLaMA3 model after completing the sequential editing task on the Counterfact dataset are shown in Figure 3. We found that: **(1) The limitations of baseline methods**: Most baseline methods experience a significant drop in general capabilities after sequential editing of 2,000 samples, with performance on almost all tasks approaching zero. Baseline methods focused on addressing the challenges of sequential editing, such as RECT and AlphaEdit, are able to maintain a certain level of performance with more sequential edits, but when the number of sequential edits reaches 10,000 samples, the performance of all baseline methods drops to almost zero. This is consistent with our previous findings that these methods cannot suppress the cumulative effect of parameter shifts, ultimately leading to model performance collapse as the number of sequential edits increases. **(2) The stability of LyapLock**: The LyapLock method is able to maintain good general performance across all tasks, even after sequential editing of 10,000 samples. Moreover, to further explore the potential of the LyapLock method, we increased the number of tests to 20,000 samples and observed that the

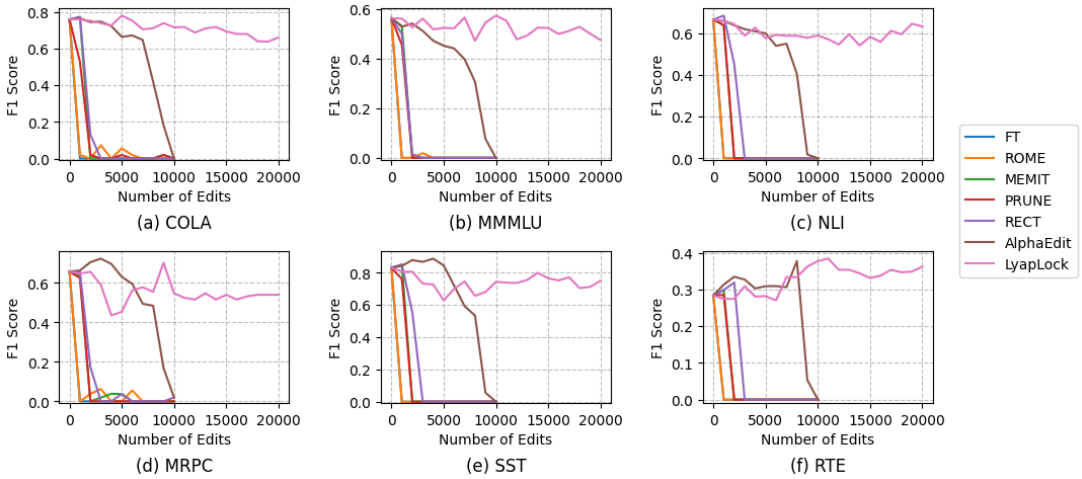


Figure 3: The F1 scores of the LLaMA3 (8B) model on the GLUE benchmark after sequentially editing 10,000 samples on the CounterFact dataset.

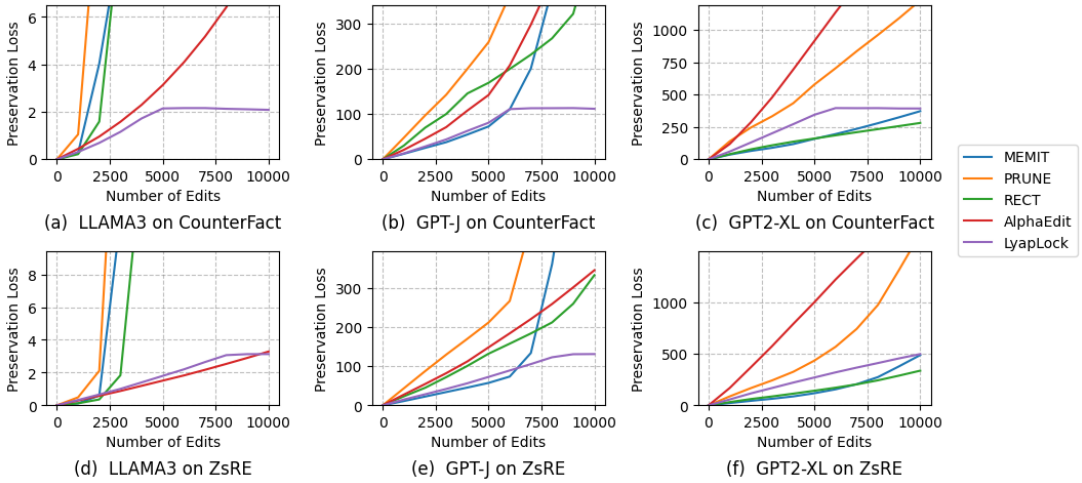


Figure 4: The preservation loss changes after sequentially editing 10,000 samples on different datasets by different LLMs.

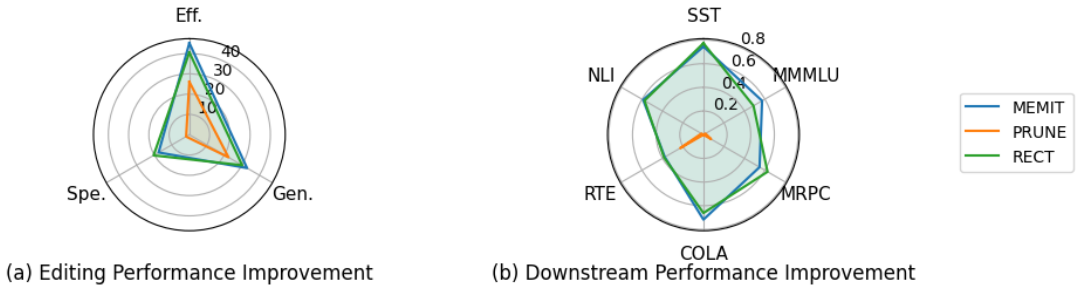
method still maintained excellent overall performance across all tasks. This further indicates that constraining the preservation loss can effectively prevent model collapse.

4.4 Preservation Loss Control Analysis

As stated in Section 1, previous solutions dedicated to addressing the challenges of the locate-then-edit method in sequential editing scenarios, such as RECT, PRUNE, and AlphaEdit, has all failed to effectively suppress the accumulation of preservation loss. With the increasing number of edits, these methods will eventually lead to a significant decline or even collapse of model performance. In light of this, this study further explores whether the LyapLock method can effectively control preservation loss within a certain threshold range during sequential editing. Figure 4 clearly shows the trend of preservation loss varying with the number of edits under different editing methods. The results indicate that our method can maintain preservation loss stably within the threshold. In contrast, other methods, although to some extent slowing down the increase in preservation loss after each edit, cannot fundamentally prevent the continuous accumulation of preservation loss. For a more detailed discussion on the trend of the preservation loss of the LyapLock method, please refer to Appendix E.4.

4.5 Compatibility

The method proposed in this study is an improvement upon the traditional single-edit bi-objective optimization approaches within the locate-and-edit



(a) Editing Performance Improvement (b) Downstream Performance Improvement

Figure 5: The improvement in editing performance and downstream task performance of other editing methods after incorporating LyapLock, following the sequential editing of 10,000 samples on the CounterFact dataset using the LLaMA3 model.

paradigm. Therefore, it should exhibit good compatibility with most works that adhere to the locate-and-edit paradigm, and can be combined with them to enhance performance. To thoroughly validate this, we selected LLaMA3 as the base model for our experiments and combined LyapLock with the MEMIT, PRUNE, and RECT methods to conduct experiments on sequential editing of 10,000 samples. The experimental results are shown in Figure 5. Specifically, Figure 5(a) showcases the improvement in editing performance for each method after integrating LyapLock, while Figure 5(b) illustrates the enhancement in downstream task performance on the GLUE benchmark. It can be observed that the average improvement in editing performance is 9.76%, and the average improvement in downstream task performance is 41.11%. This fully demonstrates the wide applicability of our method: it can effectively be integrated with other models based on the locate-and-edit paradigm, significantly enhancing their editing performance while also bolstering their ability to maintain general capabilities. For more compatibility results on additional base models, please refer to Appendix E.5.

4.6 Parameter Sensitivity Analysis

To investigate the performance changes of our method under different hyperparameters, we adjusted the hyperparameter α to change the threshold D in Problem 6 and analyzed its impact on editing performance. Since D_{base} is model-adaptive—it is automatically determined from the model's characteristics without human intervention—only α must be set manually. The experimental results are shown in Table 2. As α increases, that is, as the threshold D becomes larger and the constraints are gradually relaxed, we observed the following trends: On the CounterFact dataset, the Efficacy and Generalization metrics, which are related to knowledge updating evaluation, both improved, indicating a enhancement in the model's performance in knowledge updating. However, the Specificity metric, which is related to knowledge preservation evaluation, decreased. This is likely because the relaxation of constraints caused the model to focus more on editing loss. Additionally, on the ZsRE dataset, although the overall trend was similar to that on the CounterFact dataset, there were some fluctuations in the related metrics, which may be attributed to the characteristics of the dataset itself or the model's adaptability to different datasets. Therefore, there is a balance point in the design of the threshold D to achieve a balance between the model's editing performance and general capabilities.

As described in Section 3.3, we have set default values for the parameters V, a, and b and provided detailed reasons for these settings (based on theoretical considerations such as balancing editing loss and constraint satisfaction, mapping relationships, and weight ratios). This setting aims to minimize human intervention and make the parameter α the only hyperparameter that needs to be adjusted in practical applications. To further analyze the sensitivity of V, a, and b, we conduct experiments using the LLaMA3 model on the ZsRE dataset as an example, and the results are shown in Table 3. The results shows that the proposed default parameter set achieves near-optimal results in the metrics of Eff., Gen., and Spe.. Variations in V or a have minimal impact on performance, which confirms the robustness of the framework to these parameters. In contrast, introducing $b > 0$ leads to a significant decrease in Eff./Gen., while Spe. increases slightly. This validates our design choice of $b = 0$ to prioritize core editing performance. Collectively, these

Table 2: The editing performance with different hyperparameters α. Here, the abbreviations *Eff.* (Efficacy), *Gen.* (Generalization), *Spe.* (Specificity), *Flu.* (Fluency), and *Consis.* (Consistency) are employed to denote respective evaluation metrics.

Model	α	Counterfact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
LLaMA3	20	84.60±0.36	67.16±0.42	72.54±0.28	622.65±0.19	28.31±0.11	92.27±0.17	88.12±0.22	31.85±0.22
	40	91.90±0.27	77.22±0.37	70.02±0.29	620.08±0.23	29.79±0.11	94.05±0.14	89.62±0.21	31.21±0.22
	60	94.61±0.23	81.68±0.34	69.01±0.30	617.04±0.24	30.70±0.12	94.34±0.13	90.20±0.20	30.74±0.22
	80	96.11±0.19	84.30±0.31	67.70±0.30	615.52±0.23	31.67±0.12	94.02±0.14	89.18±0.21	30.27±0.22
	100	96.81±0.18	86.40±0.30	67.41±0.30	611.89±0.26	32.15±0.12	93.80±0.14	88.87±0.21	30.05±0.22
GPT-J	20	94.67±0.22	77.74±0.36	71.36±0.28	621.41±0.15	38.91±0.12	98.53±0.08	93.22±0.20	26.91±0.26
	40	98.02±0.14	85.12±0.30	69.50±0.28	619.50±0.17	40.26±0.12	99.14±0.06	94.68±0.18	26.08±0.25
	60	99.00±0.10	88.80±0.27	68.21±0.28	618.33±0.18	40.93±0.12	98.77±0.08	93.82±0.19	25.51±0.25
	80	99.34±0.08	90.70±0.24	67.77±0.28	615.96±0.21	41.05±0.12	98.56±0.08	93.28±0.20	25.48±0.25
	100	99.37±0.08	91.70±0.23	66.88±0.28	614.21±0.22	41.13±0.12	98.40±0.09	93.65±0.19	25.44±0.25
GPT2-XL	20	87.39±0.33	72.91±0.38	63.83±0.29	581.78±0.43	32.93±0.12	83.92±0.29	74.23±0.35	25.06±0.25
	40	93.00±0.26	78.53±0.35	61.91±0.29	580.30±0.43	34.03±0.13	85.09±0.28	75.35±0.34	23.64±0.25
	60	94.76±0.22	80.51±0.33	60.74±0.29	577.06±0.42	34.29±0.13	84.96±0.28	74.49±0.35	22.63±0.24
	80	95.07±0.22	81.20±0.32	60.50±0.30	576.67±0.40	34.60±0.13	83.41±0.30	73.44±0.35	22.45±0.24
	100	95.18±0.21	81.14±0.32	59.75±0.29	580.22±0.39	35.14±0.13	84.40±0.29	74.63±0.35	22.89±0.24

Table 3: Sensitivity results for more hyperparameters using the LLaMA3 model on the ZsRE dataset. The bold parameter set $(V, a, b) = (1, \frac{1}{\sqrt{D}}, 0))$ is the default value set in Section 3.3.

Method	Eff.↑	Gen.↑	Spe.↑
$(1, \frac{1}{\sqrt{D}}, 0)$	**94.34±0.13**	**90.20±0.20**	**30.74±0.22**
$(0.5, \frac{1}{\sqrt{D}}, 0)$	94.56±0.13	90.28±0.20	31.11±0.22
$(2, \frac{1}{\sqrt{D}}, 0)$	94.25±0.13	90.27±0.20	31.09±0.22
$(1, \frac{0.5}{\sqrt{D}}, 0)$	94.14±0.14	90.04±0.20	30.21±0.22
$(1, \frac{2}{\sqrt{D}}, 0)$	94.55±0.13	90.27±0.20	31.09±0.22
$(1, \frac{1}{\sqrt{D}}, 1)$	87.12±0.23	82.40±0.27	32.44±0.22
$(1, \frac{1}{\sqrt{D}}, 2)$	77.75±0.30	73.12±0.32	32.77±0.22

results indicate that the default parameters provide stable and high-performance operation while reducing the workload of parameter tuning, thereby supporting the settings in Section 3.3.

5 Related Works

Parameter-Preserving Model Editing. Parameter-preserving model editing methods are primarily divided into two categories. The first category involves updating knowledge using additional modules. For example, SERAC(Mitchell et al., 2022b) employs an external explicit memory and a small auxiliary model, CALINET(Dong et al., 2022) and T-Patcher(Huang et al., 2023) utilize neurons, GRACE(Hartvigsen et al., 2023) adopts codebooks, MELO(Yu et al., 2024) leverages LoRA modules, and WISE(Wang et al., 2024) uses a side memory module. The second category employs contextual prompts to guide model knowledge updates, such as MemPrompt(Madaan et al., 2022) and IKE(Zheng et al., 2023).

Parameter-Modifying Model Editing. Parameter-modifying model editing methods mainly fall into two classes. The first class adopts meta-learning to predict parameter updates via a trained hypernetwork, including KE(Cao et al., 2021), MEND(Mitchell et al., 2022a), MALMEN(Tan et al., 2024) and InstructEdit(Zhang et al., 2024). The second class focuses on locate-then-edit strategies, where activation values or parameter subsets associated with target knowledge are precisely identified using gradient-based or causal tracing methods, followed by targeted editing. Examples include KN(Dai et al., 2022), ROME(Meng et al., 2022), MEMIT(Meng et al., 2023). Additionally, some studies optimize against model collapse in sequential editing scenarios: RECT(Gu et al., 2024) employs regularized weight updates, PRUNE(Ma et al., 2025) controls condition numbers, and AlphaEdit(Fang et al., 2025) applies null-space projection.

6 Conclusion

In this work, we propose LyapLock, which reformulates the traditional bi-objective optimization as a constrained long-term optimization problem for sequential editing to address the issue of long-term accumulation of preservation loss in existing methods as the number of edits increases. Using Lyapunov optimization, we convert the long-term problem into online solvable subproblems, achieving asymptotically near-optimal editing performance

while satisfying preservation loss constraints. Experiments on multiple LLMs show that LyapLock significantly outperforms existing methods.

Limitations

Despite its excellent editing performance and effective maintenance of model general capabilities in sequential editing tasks, the LyapLock method still has room for improvement. Firstly, the current dataset size for evaluating editing performance is capped at around 20,000. Testing for model general capabilities has only been conducted after 20,000 edits, with no signs of model collapse. Although the method is theoretically proven to constrain loss in long-term editing, larger-scale datasets are needed to further validate its practical effectiveness. Secondly, tests on model general capabilities mainly focus on language understanding, while areas like code generation and mathematical reasoning are under-tested. Future work should expand the testing scope.

Ethics Considerations

All codes and datasets in this paper are from publicly available resources. The application of such technologies must follow ethical principles. The widespread use of large language models brings convenience but also raises ethical concerns. Malicious users could exploit these models to generate and spread hate speech, false information, or harmful content, threatening social harmony and stability. Thus, it is crucial and urgent to implement effective safeguards to prevent misuse and mitigate potential harm. Therefore, we strongly advocate that researchers implement rigorous validation and oversight measures to ensure the ethical application of these technologies.

References

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

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, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

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 *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, 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 *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5937–5947. Association for Computational Linguistics.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Alphaedit: Null-space constrained knowledge editing for language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 5484–5495. Association for Computational Linguistics.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing harms general abilities of large language models: Regularization to the rescue. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 16801–16819. Association for Computational Linguistics.

Akshat Gupta, Sidharth Baskaran, and Gopala Anumanchipalli. 2024. Rebuilding rome : Resolving model collapse during sequential model editing.

Anshita Gupta, Debanjan Mondal, Akshay Krishna Sheshadri, Wenlong Zhao, Xiang Li, Sarah Wiegreffe, and Niket Tandon. 2023. Editing common sense in transformers. In *Proceedings of the 2023 Conference*

on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023, pages 8214–8232. Association for Computational Linguistics.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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.

Xiusheng Huang, Hang Yang, Yubo Chen, Jun Zhao, Kang Liu, Weijian Sun, and Zuyu Zhao. 2022. Document-level relation extraction via pair-aware and entity-enhanced representation learning. In *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pages 2418–2428. International Committee on Computational Linguistics.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Lee W Johnson, R. Dean Riess, and Jimmy T Arnold. 2004. *Introduction to linear algebra. 2nd ed.* Introduction to linear algebra. 2nd ed.

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

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3-4, 2017*, pages 333–342. Association for Computational Linguistics.

Yangyang Li, Xintao Deng, Biao Liu, Jugang Ma, Fuyuan Yang, and Minggao Ouyang. 2022. Energy management of a parallel hybrid electric vehicle based on lyapunov algorithm. *Etransportation*, 13:100184.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Lingling Lv, Chan Zheng, Lei Zhang, Chun Shan, Zhihong Tian, Xiaojiang Du, and Mohsen Guizani. 2021. Contract and lyapunov optimization-based load scheduling and energy management for UAV charging stations. *IEEE Trans. Green Commun. Netw.*, 5(3):1381–1394.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. 2025. Perturbation-restrained sequential model editing. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 2833–2861. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Meta. 2024. Llama 3. Large language model release.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

Michael J. Neely. 2010. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Synthesis Lectures on Communication Networks. Morgan & Claypool Publishers.

Wu Qi, Li Xintong, and Zhu Lidong. 2025. Dynamic collaborative data download in heterogeneous satellite networks. *China Communications*, 22(2):26–46.

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

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 *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2025. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3):59:1–59:37.

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.

Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Qi Cao, Dawei Yin, Huawei Shen, and Xueqi Cheng. 2025. The mirage of model editing: Revisiting evaluation in the wild. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 15336–15354. 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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 10222–10240. Association for Computational Linguistics.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. MELO: enhancing model editing with neuron-indexed dynamic lora. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 19449–19457. AAAI Press.

Ningyu Zhang, Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. 2024. Instructedit: Instruction-based knowledge editing for large language models. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 6633–6641. ijcai.org.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4862–4876. Association for Computational Linguistics.

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

A Model Editing

The goal of model editing is to efficiently and accurately update factual knowledge. Specifically, for incorrect or outdated factual knowledge (s, r, o) in LLMs, model editing methods can replace it with updated knowledge (s, r, o^c). For example, when the natural language sentence "Beats Music is owned by" composed of $s =$ "Beats Music" and $r =$ "is owned by" is input into the model, the model's output, through model editing, will be modified from the incorrect $o =$ "Google" to the correct $o^c =$ "Apple".

Currently, model editing methods based on the locate-then-edit paradigm have become mainstream due to their excellent editing performance, such as ROME and MEMIT. These methods mainly consist of two key steps: (1) identifying the critical parameter subset W associated with the target knowledge via causal tracing analysis, and (2) achieving the update of the target knowledge within the parameter space by computing and implementing appropriate perturbations ΔW.

A.1 Causal Tracing

Causal Tracing is an analytical method designed to determine the causal influence of the internal hidden state activations within LLMs on the prediction of specific facts. The essence of this method lies in quantifying and understanding which internal state variables play a key role when the model processes

specific information. The specific steps of causal tracing are as follows:

(1) Clean Run. Initially, a factual prompt (e.g., "Space Needle is located in the city of") is input into the LLMs, and the state activations of all hidden layers are collected as the clean state.

(2) Corrupted Run. In this run, the embedding vector of the subject (e.g., "Space Needle") is corrupted with noise, and then the model continues to run. Due to the loss of certain information about the subject, the model may provide an incorrect answer.

(3) Corrupted-with-Restoration Run. In this run, except for restoring the clean state at specific tokens and layers, other corrupted embedding vectors remain unchanged. This allows for testing the ability of a single state restoration to predict.

By comparing the results of the above three runs, the Total Effect (TE) and Indirect Effect (IE) of each hidden state variable on the correct prediction of facts are calculated. TE is the difference in the prediction probability between the Clean Run and the Corrupted Run, while IE is the difference in the prediction probability between the Corrupted Run and the Corrupted-with-Restoration Run. By analyzing the Average Indirect Effect (AIE), researchers have found that the MLP module plays a key role in storing and recalling factual associations. Specifically, the MLP modules in the middle layers of the model are identified as the primary storage area for factual knowledge, and they play an especially critical role when processing the last token of the subject.

A.2 Computing Perturbations

Based on the results of causal tracing localization, subsequent modifications will be made to the parameters of the second layer of the MLP module in the model's intermediate layer, W^l_{out}, to achieve knowledge updating. Specifically, the model editing updates the knowledge (s, r, o) in LLMs to (s, r, o^c). This process can be understood as remapping the key k that encodes (s, r) from its original mapping, which is the value v that encodes o, to the value v^c that encodes o^c. The formal description is as follows:

$$k = \text{act}(W^l_{\text{in}}(a^l + h^{l-1})), \quad v = W^l_{\text{out}} k. \tag{14}$$

To achieve this goal, (Meng et al., 2023) optimized a dual-objective loss function (Equation 15) to compute the perturbation (Equation 16). Once K_0, K_1, and V_1 are obtained, the specific perturbation values can be calculated. Here, K_1 and V_1 are matrices composed of the keys k and values v^c of all new knowledge in the current editing batch, respectively.

$$\begin{aligned} \min_{\Delta} &\|(W + \Delta)K_1 - V_1\|_F^2 \\ &+ \|(W + \Delta)K_0 - V_0\|_F^2 . \end{aligned} \tag{15}$$

$$\Delta = (V_1 - WK_1)\, K_1^T \left(K_0 K_0^T + K_1 K_1^T\right)^{-1}. \tag{16}$$

Obtaining K_0. (Meng et al., 2022, 2023) randomly sampled a large number of articles from Wikipedia snapshots and input the full text of each article into the model. During the model's processing, they collected the MLP activation vectors corresponding to each token. Eventually, they collected 100,000 k vector samples from these articles to form the K_0 matrix. Additionally, $V_0 = WK_0$.

Obtaining K_1. The K_1 matrix is composed of all k from a single editing batch. Based on the findings from the localization phase, (Meng et al., 2023) used the input of the last token of the subject as the key. The specific calculation method for each k is as follows: Input the text containing the subject s into the model, and at the target layer l and the position of the last token of the subject, extract the activation values of the second layer of the MLP, as shown in Equation 17.

$$k = \frac{1}{N} \sum_{j=1}^{N} \mathbf{k}(x_j + s), \mathbf{k}(x) = \text{act}(W^l_{\text{in}}(a^l + h^{l-1})). \tag{17}$$

Here, x_j is the randomly generated prefix text, and N is the number of prefix texts. By doing so, researchers extract activation values from multiple random contexts and calculate the average to obtain the key vector k that represents the subject, which is used to locate subject-related factual information in the middle layer MLP module of the model.

Obtaining V_1. The V_1 matrix is composed of all v^c from a single editing batch. For the calculation of each v^c, (Meng et al., 2023) optimized Equation 18 to solve for the optimal vector v^c in order to achieve precise encoding of the target knowledge o^c. The specific calculation process is as follows:

$$v^c = \arg\min_z \frac{1}{N}\sum_{j=1}^{N} -\log P_{W^l_{\text{out}}(v=z)}[o^c \mid x_j + p] + D_{KL}\left(P_{W^l_{\text{out}}(v=z)}[x \mid p'] \| P_{W^l_{\text{out}}}[x \mid p']\right). \tag{18}$$

Here, $W^l_{\text{out}}(v = z)$ denotes the intervened model where the MLP output at the l-th layer and the position of the last token of the subject is replaced by the vector z; $x_j + p$ represents the input concatenated from the randomly generated prefix text x_j and the factual prompt template p. This formula optimizes to replace the original v with the v^c, maximizing the probability of the target word o^c while preventing semantic drift through KL divergence.

Additionally, since the matrices K_0 and V_0 are particularly large, storing them separately occupies a significant amount of space. Moreover, they typically appear in the form of $K_0K_0^T$ and $V_0K_0^T$ in the computational formulas. Therefore, it is common practice to store only $K_0K_0^T$ and $V_0K_0^T$. For more details, refer to (Meng et al., 2023).

B Innovative Discussion

Lyapunov optimization is a mathematical framework that transforms long-term stochastic optimization problems into single-slot decisions, applicable to resource allocation and stability control in dynamic systems. It has been employed in various fields such as satellite communications(Qi et al., 2025), edge computing(Lv et al., 2021), and energy management in intelligent transportation systems(Li et al., 2022). The innovation of this work lies in offering a new perspective and method for continuous editing, as detailed below:

Innovative problem reconstruction: Unlike other sequential editing methods that focus on mitigating the increase in preservation loss after each edit, we shift the perspective and aim to develop a sequential editing method that can keep the model's preservation loss within a certain range over the long term. We reconstruct the traditional bi-objective optimization into a stochastic programming problem with long-term constraints.

Innovative mechanism design: Virtual queue design within the Lyapunov framework is key to stability, yet its design is challenging, lacking fixed formulas and requiring tailoring to the specific problem. Here the virtual queue value $Z(t)$ acts as a weight parameter in the loss function, automatically adjusting the extent to which the model should focus on preservation loss (see Eq. 12). In this paper, we innovatively introduce parameters a, b, and z_{max}, that is, Eq. 7, to flexibly adjust the mapping relationship between the preservation loss $PL(t)$ and the weight parameter $Z(t+1)$. Meanwhile, $z_{max} > 0$ is also to ensure that $Z(t+1)$ is not zero, thus avoiding the loss of attention to the preservation loss $PL(t)$.

C Proof

C.1 Proof of Sufficient Condition

Here, we prove that the sufficient condition for the constraint to always hold is the strong stability of the virtual queue $Z(t)$.

From the update formula of the virtual queue (Metioned in 7), we have:

$$Z(t+1) \geq Z(t) + a(PL(t) - D) + b. \tag{19}$$

Listing the above inequalities for multiple timestamps $t \in \{1, \ldots, T\}$:

$$\begin{aligned} Z(T+1) &\geq Z(T) + a(PL(T) - D) + b, \\ Z(T) &\geq Z(T-1) + a(PL(T-1) - D) + b, \\ Z(T-1) &\geq Z(T-2) + a(PL(T-2) - D) + b, \\ &\ldots, \\ Z(2) &\geq Z(1) + a(PL(1) - D) + b. \end{aligned} \tag{20}$$

Summing all the inequalities, we obtain:

$$Z(T+1) \geq Z(1) + a\sum_{t=1}^{T} PL(t) - aTD + Tb. \tag{21}$$

Dividing both sides by aT and taking the limit ($a > 0, b, Z(1) = Z_{init} \geq 0$):

$$\begin{aligned} &\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} PL(t) \\ &\leq \lim_{T\to\infty} \frac{Z(T+1)}{aT} - \lim_{T\to\infty} \frac{Z(1)}{aT} + \lim_{T\to\infty} (D - \frac{b}{a}) \\ &= \lim_{T\to\infty} \frac{Z(T+1)}{aT} + D - \frac{b}{a} \\ &\leq \lim_{T\to\infty} \frac{Z(T+1)}{aT} + D. \end{aligned} \tag{22}$$

When $\lim_{T\to\infty} \frac{Z(T+1)}{aT} = 0$, that is, $\lim_{T\to\infty} \frac{Z(T)}{T} = 0$ (according to the proof

in Section C.2), we have

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}PL(t)\le D.$$

The above inequality can be equivalently written as the constraint condition in 6:

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}PL(t)\le D. \tag{23}$$

C.2 Proof of Equivalence

Let $S=T+1$. Then, as $T\to\infty$, we have $S\to\infty$. Therefore:

$$\begin{aligned}&\lim_{T\to\infty}\frac{Z(T+1)}{aT}\\ &=\lim_{S\to\infty}\frac{Z(S)}{a(S-1)}\\ &=\lim_{S\to\infty}\frac{Z(S)}{S}\cdot\frac{S}{a(S-1)}\\ &=0.\end{aligned} \tag{24}$$

As $S\to\infty$, $\frac{S}{a(S-1)}\to\frac{1}{a}>0$, we obtain:

$$\lim_{S\to\infty}\frac{Z(S)}{S}=0. \tag{25}$$

That is:

$$\lim_{T\to\infty}\frac{Z(T)}{T}=0. \tag{26}$$

C.3 Upper Bound Derivation

It is known that the following inequality holds ($\forall a,b,c,Z_{max}\ge 0$) (the proof of the inequality can be found in Section C.4):

$$\begin{aligned}&(\max[a+b-c,Z_{max}])^2\\ &\le a^2+b^2+c^2+2a(b-c)+Z_{max}^2.\end{aligned} \tag{27}$$

From the virtual queue update formula 7, we can obtain:

$$\begin{aligned}&Z(t+1)^2\\ &=(max[Z(t)+a(PL(t)-D)+b,Z_{max}])^2\\ &=(max[Z(t)+(aPL(t)+b)-aD,Z_{max}])^2\\ &\le Z(t)^2+(aPL(t)+b)^2+(aD)^2\\ &\quad+2Z(t)(aPL(t)+b-aD)+Z_{max}^2.\end{aligned} \tag{28}$$

By dividing both sides by $\frac{1}{2}$, we obtain:

$$\begin{aligned}&\frac{1}{2}Z(t+1)^2-\frac{1}{2}Z(t)^2\\ &\le\frac{1}{2}(aPL(t)+b)^2+\frac{1}{2}(aD)^2+\frac{1}{2}Z_{max}^2\\ &\quad+Z(t)(aPL(t)+b-aD).\end{aligned} \tag{29}$$

From the one-step conditional Lyapunov drift 9, we have:

$$\begin{aligned}\Delta(Z(t))\le&\frac{1}{2}(aPL(t)+b)^2+\frac{1}{2}(aD)^2+\frac{1}{2}Z_{max}^2\\ &+\{Z(t)(aPL(t)+b-aD)\mid Z(t)\}.\end{aligned} \tag{30}$$

Assuming that there exists $D_{max}=\max_t\{PL(t)\}$ (It is widely believed that neural networks are Lipschitz continuous, meaning that the rate of change of the loss function between any two points in its entire domain has a global upper bound), by defining $B\triangleq\frac{1}{2}((aD_{max}+b)^2+(aD)^2+Z_{max}^2)$, the above inequality can be simplified as:

$$\begin{aligned}&\Delta(Z(t))\\ &\le B+\{Z(t)(aPL(t)+b-aD)\mid Z(t)\}.\end{aligned} \tag{31}$$

By adding the editing loss on both sides, we get:

$$\begin{aligned}&\Delta(Z(t))+V\cdot EL(t)\\ &\le\{Z(t)(aPL(t)+b-aD)+V\cdot EL(t)\mid Z(t)\}\\ &\quad+B.\end{aligned} \tag{32}$$

Since B is a constant, minimizing the upper bound is equivalent to minimizing the second term on the right-hand side of the inequality, that is:

$$\min_{\Delta(t)}Z(t)(aPL(t)+b-aD)+V\cdot EL(t). \tag{33}$$

By removing the constants that are irrelevant to the optimization variable $\Delta(t)$, we obtain:

$$\min_{\Delta(t)}aZ(t)PL(t)+V\cdot EL(t). \tag{34}$$

C.4 Proof of the Inequality

(1) When $a+b-c>Z_{max}$ ($\forall a,b,c,Z_{max}\ge 0$),

$$\begin{aligned}&\max([a+b-c,Z_{max}])^2\\ &=(a+b-c)^2\\ &=a^2+b^2+c^2+2ab-2ac-2bc\\ &\le a^2+b^2+c^2+2a(b-c)\\ &\le a^2+b^2+c^2+2a(b-c)+Z_{max}^2.\end{aligned} \tag{35}$$

(2) When $a+b-c\le Z_{max}$ ($\forall a,b,c,Z_{max}\ge 0$), it is necessary to prove:

$$\begin{aligned}&\max([a+b-c,Z_{max}])^2\\ &=Z_{max}^2\\ &\le a^2+b^2+c^2+2a(b-c)+Z_{max}^2.\end{aligned} \tag{36}$$

That is:

$$a^2 + b^2 + c^2 + 2a(b - c) \geq 0. \tag{37}$$

It is known that:

$$(a+b-c)^2 = a^2+b^2+c^2+2ab-2ac-2bc \geq 0. \tag{38}$$

Therefore, we have:

$$a^2 + b^2 + c^2 + 2a(b - c) \geq 2bc \geq 0. \tag{39}$$

D Experimental Setup

D.1 Baseline Methods

Here, we will introduce the baseline methods used in this paper, which are as follows:

FT. FT is a parameter-efficient model adjustment strategy that selectively updates parameters in specific layers of the model using a cross-entropy loss function. This achieves precise local optimization of the model while keeping the rest of the model unchanged.

ROME. ROME employs causal tracing analysis to identify the key middle-layer MLP modules in the model where factual associations are stored. It then inserts new key-value pairs into these modules to update the model's memory of specific facts. Specifically, the key is determined by the hidden state of the subject's last token, while the value is obtained by optimizing the prediction probability of the target object.

MEMIT. MEMIT is an extendable multi-layer updating algorithm proposed based on ROME. It efficiently integrates new memories into LLMs by explicitly computing parameter updates, achieving large-scale memory editing while maintaining the integrity of the model.

PRUNE. PRUNE is a framework designed to restrict the perturbations to LLMs during sequential editing, addressing the issue of significant degradation in the models' general abilities caused by existing editing methods after multiple edits. The study's theoretical analysis, based on matrix perturbation theory, reveals that the condition number of the edited matrix is a crucial factor affecting general abilities. This condition number increases with the number of edits, exacerbating the perturbation of original knowledge associations. PRUNE mitigates this issue by restraining the large singular values of the edit update matrix, thereby reducing the condition number and preserving the general abilities of the edited models.

RECT. RECT is a regularization method that prevents overfitting by limiting the complexity of the edit update weights. Specifically, RECT identifies the most important editing information (top-k% of elements) based on the relative change in weights, retains their original values, and sets the remaining elements to zero. This approach effectively mitigates the negative impact on general abilities caused by sequential edits.

AlphaEdit. The core of AlphaEdit lies in projecting the parameter perturbation onto the null space of the preserved knowledge, thereby ensuring that the model's output on the original knowledge remains unchanged during the editing process. Specifically, AlphaEdit first computes the null space of the preserved knowledge matrix using Singular Value Decomposition (SVD) and defines a projection matrix. During editing, it projects the perturbation into this null space and then applies the projected perturbation to the model parameters. This method not only effectively avoids interference with the preserved knowledge but also simplifies the editing objective by removing the error term related to the preserved knowledge, allowing the model to focus more on updating the knowledge.

D.2 Datasets

ZsRE Dataset. It is a high-quality question-answering dataset specifically designed to evaluate the model editing and zero-shot relation extraction capabilities of natural language processing (NLP) models, which contains 193,196 training samples and 19,086 test samples. It employs back-translation techniques to generate paraphrased versions of questions, thereby constructing equivalent neighborhood samples. Each sample includes a subject term s and a target object o that needs to be modified, as well as semantically similar and dissimilar sentences. These features enable the effective assessment of a model's generalization ability and specificity. As a result, the ZsRE dataset is widely used to test various model editing methods and has become one of the important benchmark datasets in the field of natural language processing.

CounterFact dataset. It focuses on evaluating the knowledge editing and factual knowledge understanding capabilities of NLP models and is also a high-quality dataset, which contains 20,877 samples. It constructs counterfactual knowledge by replacing the subject entity with an approximate sub-

ject entity that shares the same predicate, making it more challenging compared to the ZsRE dataset. In addition to covering similar evaluation metrics as ZsRE, the CounterFact dataset introduces indicators focusing on the fluency and consistency of generated text quality, further enriching the dimensions for assessing model performance.

D.3 Metrics

Given a language model f_θ, an edit instance comprising factual prompt (s_i, r_i), target output o_i, and the model's original prediction o_i^c, we will now detail the calculation methods for the evaluation metrics.

D.3.1 Metrics of ZsRE

Following the previous works(Meng et al., 2022, 2023; Fang et al., 2025), this section formalizes the evaluation criteria for ZsRE metrics under three dimensions:

- **Efficacy**: Quantified by averaging the top-1 prediction accuracy across edited samples, this metric verifies successful knowledge integration:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid (s_i, r_i)) \right\}. \quad (40)$$

- **Generalization**: Assesses the model's capability to maintain accuracy when presented with semantically equivalent variations $N((s_i, r_i))$, calculated through:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid N((s_i, r_i))) \right\}. \quad (41)$$

- **Specificity**: Evaluates preservation of original behavior on unrelated samples $O((s_i, r_i))$ by measuring consistency with pre-edit predictions:

$$\mathbb{E}_i \left\{ o_i^c = \arg\max_o \mathbb{P}_{f_\theta}(o \mid O((s_i, r_i))) \right\}. \quad (42)$$

D.3.2 Metrics of CounterFact

Following the previous works(Meng et al., 2022, 2023; Fang et al., 2025), this subsection formalizes the evaluation framework for Counterfact metrics under five dimensions:

- **Efficacy (Editing Success)**: Measures the success rate of integrating new knowledge by comparing the probability of the target output o_i against the original prediction o_i^c under the factual prompt:

$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}(o_i \mid (s_i, r_i)) > \mathbb{P}_{f_\theta}(o_i^c \mid (s_i, r_i)) \right]. \quad (43)$$

- **Generalization (Paraphrase Robustness)**: Evaluates robustness to paraphrased variants $N((s_i, r_i))$ by comparing output probabilities across rephrased prompts:

$$\mathbb{E}_i[\mathbb{P}_{f_\theta}(o_i \mid N((s_i, r_i))) > \mathbb{P}_{f_\theta}(o_i^c \mid N((s_i, r_i)))]. \quad (44)$$

- **Specificity (Neighborhood Preservation)**: Assesses minimal interference on related but distinct subject prompts $O((s_i, r_i))$, ensuring original predictions remain dominant:

$$\mathbb{E}_i[\mathbb{P}_{f_\theta}(o_i \mid O((s_i, r_i))) > \mathbb{P}_{f_\theta}(o_i^c \mid O((s_i, r_i)))]. \quad (45)$$

- **Fluency (Repetition Control)**: Quantifies output repetitiveness via entropy of bi-gram (g_2) and tri-gram (g_3) distributions:

$$-\frac{2}{3} \sum_k g_2(k) \log_2 g_2(k) + \frac{4}{3} \sum_k g_3(k) \log_2 g_3(k). \quad (46)$$

  where $g_n(\cdot)$ denotes the normalized frequency of n-grams.

- **Consistency (Reference Alignment)**: Evaluates semantic alignment between model-generated text and reference content by computing the cosine similarity of their TF-IDF vectors for subject s and object o:

$$\text{sim}_{\text{TF-IDF}} \left(f_\theta(s), \text{Ref}(o) \right). \quad (47)$$

D.4 Implementation Details

In this work, all experiments are conducted on a single A100 (80GB) GPU. The hyperparameter configurations for LLaMA3 are based on AlphaEdit, while those for GPT2-XL and GPT-J are adapted from MEMIT. Specifically, for LLaMA3, the editing layers are set to $[4, 5, 6, 7, 8]$; for GPT2-XL, the editing layers are $[13, 14, 15, 16, 17]$; and for GPT-J, the editing layers are $[3, 4, 5, 6, 7, 8]$. For all models, the hyperparameter α is uniformly set to 60, meaning that the threshold D is configured to be 60 times the baseline value D_{base}.

D.5 Time Cost

Additionally, we computed the average time required to edit a single example. The results are

Table 4: Time cost of different methods across various models.

Model	Method	CounterFact(/s)	ZsRE(/s)
LLaMA3	FT	0.48	0.62
	ROME	18.52	24.64
	MEMIT	2.50	3.08
	PRUNE	2.43	3.13
	RECT	2.47	3.15
	AlphaEdit	2.34	3.22
	LyapLock	2.06	3.01
GPT-J	FT	0.13	0.44
	ROME	12.66	9.93
	MEMIT	1.67	1.87
	PRUNE	1.67	1.87
	RECT	1.62	1.54
	AlphaEdit	2.12	2.33
	LyapLock	1.78	2.14
GPT2-XL	FT	0.13	0.21
	ROME	3.52	2.89
	MEMIT	0.44	0.62
	PRUNE	0.43	0.55
	RECT	0.42	0.48
	AlphaEdit	0.72	0.82
	LyapLock	0.50	0.60

shown in Table 4. From the statistical results, the FT method has the shortest time-cost among all the compared methods. However, the main experimental results (as shown in Table 1) reflect that its editing effect is not good. The ROME method has the longest time - consuming significantly. The reason is that it does not support batch editing, which leads to its very low efficiency when editing a single example. It is worth noting that compared with the strong baseline AlphaEdit, which shows the second-best performance in Table 1, the LyapLock method shows lower time cost on different models and different datasets and can achieve better performance. This result strongly proves that the LyapLock method has achieved an excellent balance between computational efficiency and editing effect.

D.6 Details of GLUE

GLUE is a comprehensive benchmark, and this paper selects the following six subtasks:

CoLA. (Warstadt et al., 2019) evaluates grammatical acceptability through binary classification of single-sentence judgments.

MMMLU. (Hendrycks et al., 2021) measures multi-task accuracy across diverse domains, specifically targeting zero-shot and few-shot learning scenarios in text models.

NLI. (Williams et al., 2018) assesses natural language understanding by requiring models to identify logical relationships (entailment, contradiction, neutral) between sentence pairs.

MRPC. (Dolan and Brockett, 2005) serves as a benchmark for semantic equivalence detection, where models must determine if sentence pairs convey identical meanings.

SST. (Socher et al., 2013) focuses on sentiment classification of movie review sentences, assigning binary sentiment labels based on human annotations.

RTE. (Bentivogli et al., 2009) examines textual entailment by determining whether a premise sentence logically supports a given hypothesis.

E More Experimental Results

E.1 The Editing Performance for Other Number of Edits

Tables 5 and 6 illustrate the editing performance of various editing methods when sequential editing 2,000 and 5,000 samples across different LLMs and datasets. The conclusions drawn are essentially consistent with those in Section 4.2.

E.2 Results on Additional Datasets

Since our method focuses on solving the problem of sequential editing, we select the widely-verified benchmark datasets ZsRE and CounterFact for the main experiments. To further validate the effectiveness of our approach, we apply LyapLock to additional types of datasets. Specifically, for multi-hop scenarios we adopt the MQuAKE-CF dataset proposed by (Zhong et al., 2023). MQuAKE-CF is a counterfactual multi-hop question-answering dataset designed to evaluate how well language models can update their knowledge when facts change. For real-world scenarios we adopt the QAEdit dataset proposed by (Yang et al., 2025). QAEdit is a fact-consistency editing detection dataset for question-answering systems. By automatically constructing noisy answers and manually annotating text spans that require editing, it provides resources for training and evaluating models' ability to identify and correct factual errors. We choose the widely-used LLaMA3 model as the representative for experimentation.

On MQuAKE-CF, we perform sequential editing on all 3,000 samples. The results are shown in Table 7. The results show that LyapLock has a significant advantage in multi-hop knowledge-editing scenarios, outperforming other sequential editing baselines. As the table indicates, in Edit-wise and

Table 5: Performance results of sequential editing task (2,000 Samples). Here, the abbreviations *Eff.* (Efficacy), *Gen.* (Generalization), *Spe.* (Specificity), *Flu.* (Fluency), and *Consis.* (Consistency) are employed to denote respective evaluation metrics. Top-performing results are emphasized using bold formatting, with secondary superior results distinguished through underlined notation.

Method	Model	Counterfact					ZsRE		
		Eff.↑	**Gen.↑**	**Spe.↑**	**Flu.↑**	**Consis.↑**	**Eff.↑**	**Gen.↑**	**Spe.↑**
Pre-edited		7.85±0.27	10.58±0.27	89.48±0.19	635.44±0.11	24.19±0.09	36.99±0.30	36.34±0.30	31.89±0.23
FT	LLaMA3	93.35±0.25	84.15±0.32	42.99±0.37	234.65±0.28	10.15±0.07	30.54±0.27	30.29±0.27	15.47±0.18
ROME	LLaMA3	81.90±0.39	71.12±0.37	46.98±0.29	606.67±0.17	7.43±0.10	3.29±0.11	3.24±0.11	0.51±0.03
MEMIT	LLaMA3	64.20±0.48	63.18±0.44	51.40±0.40	394.10±1.55	5.78±0.11	39.34±0.37	34.77±0.36	20.45±0.21
PRUNE	LLaMA3	66.80±0.47	64.70±0.41	50.14±0.37	366.09±1.28	5.47±0.10	0.65±0.04	0.58±0.04	1.98±0.06
RECT	LLaMA3	65.45±0.48	62.70±0.44	60.00±0.38	521.56±0.44	19.04±0.10	86.62±0.23	81.87±0.27	31.67±0.22
AlphaEdit	LLaMA3	99.15±0.09	93.15±0.21	69.27±0.29	621.77±0.17	31.93±0.12	94.58±0.14	91.07±0.19	**32.40±0.22**
LyapLock	LLaMA3	**99.85±0.04**	**93.60±0.21**	**81.14±0.23**	**628.97±0.16**	**33.27±0.11**	**95.63±0.12**	**91.89±0.18**	32.29±0.22
Pre-edited		15.80±0.36	18.10±0.34	83.44±0.25	621.69±0.14	29.46±0.10	27.79±0.29	27.10±0.29	27.54±0.26
FT	GPT-J	92.15±0.27	72.38±0.38	43.35±0.37	296.91±0.79	6.64±0.11	72.37±0.30	68.91±0.32	19.66±0.23
ROME	GPT-J	54.35±0.50	53.92±0.40	51.35±0.30	565.03±0.08	1.43±0.01	49.97±0.44	48.07±0.43	10.13±0.16
MEMIT	GPT-J	98.50±0.12	95.40±0.17	64.16±0.31	556.11±0.85	35.90±0.15	95.99±0.15	92.92±0.20	30.84±0.27
PRUNE	GPT-J	87.10±0.34	87.72±0.28	52.98±0.35	422.31±0.47	15.42±0.13	33.43±0.33	31.59±0.32	21.49±0.24
RECT	GPT-J	98.50±0.12	86.95±0.28	72.72±0.28	615.06±0.22	40.92±0.12	96.67±0.13	92.74±0.20	**29.30±0.27**
AlphaEdit	GPT-J	**99.75±0.05**	**96.10±0.15**	76.02±0.26	617.70±0.21	41.69±0.13	99.67±0.04	97.16±0.13	**28.57±0.26**
LyapLock	GPT-J	**99.75±0.05**	95.70±0.16	**76.41±0.26**	**618.84±0.18**	**41.95±0.12**	**99.69±0.04**	**97.30±0.13**	28.37±0.26
Pre-edited		22.10±0.41	24.45±0.37	78.05±0.28	626.61±0.12	31.33±0.10	23.70±0.27	22.82±0.27	24.97±0.24
FT	GPT2-XL	64.75±0.48	42.90±0.41	54.51±0.33	534.70±0.26	10.35±0.05	31.95±0.37	29.48±0.36	8.86±0.17
ROME	GPT2-XL	51.25±0.50	48.58±0.40	51.79±0.32	424.30±0.40	0.71±0.01	44.38±0.43	39.86±0.42	11.54±0.17
MEMIT	GPT2-XL	95.10±0.22	85.60±0.29	60.16±0.32	474.20±0.56	22.04±0.15	80.27±0.32	73.46±0.36	**27.04±0.27**
PRUNE	GPT2-XL	80.85±0.39	77.98±0.35	51.06±0.36	536.10±0.42	13.87±0.10	21.37±0.31	19.80±0.30	13.10±0.19
RECT	GPT2-XL	92.35±0.27	79.85±0.34	65.29±0.32	471.17±0.62	21.25±0.16	83.72±0.29	76.28±0.34	24.52±0.25
AlphaEdit	GPT2-XL	**99.50±0.07**	**93.62±0.20**	66.03±0.29	594.10±0.47	39.11±0.13	91.79±0.21	83.19±0.30	25.91±0.26
LyapLock	GPT2-XL	99.40±0.08	92.78±0.21	**67.33±0.29**	**599.14±0.40**	**39.24±0.13**	**95.36±0.15**	**87.70±0.26**	26.50±0.26

Instance-wise metrics that reflect single-hop editing ability, LyapLock achieves an average improvement of about 11% over the second-best baseline AlphaEdit. In the Multi-hop and Multi-hop (CoT) metrics that directly measure multi-hop reasoning ability, LyapLock achieves an average improvement of about 3% over AlphaEdit, demonstrating its effectiveness in handling complex knowledge dependencies.

On QAEdit, to keep the scale consistent with the main experiments, we select a subset of 10,000 samples for sequential editing. The results are shown in Table 8. The results indicate that LyapLock significantly outperforms other sequential-editing baselines, demonstrating strong superiority even in the real-world setting. When other methods fail almost entirely in the real-world scenario (with metrics approaching or equaling 0.00), LyapLock successfully achieves remarkable performance improvements. Specifically, under the evaluation of the syn. and WILD metrics, LyapLock averages about 10 times and 140 times higher than the second-best baseline AlphaEdit, respectively.

E.3 Case Study

We present the output examples of the LLAMA3, GPT-J, and GPT2-XL models after being processed by different editing methods, as shown in Table 9, 10, and 11. It is found that after sequential editing of 10,000 samples, the content generated by the baseline methods often fails to include the target knowledge (Edit Target) and tends to produce a large number of meaningless characters or repeated words, which leads to poor text fluency. In contrast,

Table 6: Performance results of sequential editing task (5,000 Samples). Here, the abbreviations *Eff.* (Efficacy), *Gen.* (Generalization), *Spe.* (Specificity), *Flu.* (Fluency), and *Consis.* (Consistency) are employed to denote respective evaluation metrics. Top-performing results are emphasized using bold formatting, with secondary superior results distinguished through underlined notation.

Method	Model	Counterfact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
Pre-edited		7.02±0.26	9.61±0.25	89.63±0.19	635.25±0.11	24.19±0.09	36.35±0.30	35.73±0.30	31.84±0.23
FT	LLaMA3	95.38±0.21	85.79±0.30	39.60±0.36	270.31±0.56	16.80±0.11	21.04±0.24	20.81±0.24	9.64±0.14
ROME	LLaMA3	76.04±0.43	67.23±0.38	46.59±0.28	530.79±0.23	4.49±0.05	3.81±0.11	3.65±0.11	0.21±0.02
MEMIT	LLaMA3	62.90±0.48	51.92±0.44	51.28±0.37	575.08±0.15	2.02±0.03	0.00±0.00	0.00±0.00	0.00±0.00
PRUNE	LLaMA3	67.14±0.47	55.57±0.43	49.42±0.34	559.12±0.12	3.31±0.03	0.02±0.01	0.01±0.01	0.00±0.00
RECT	LLaMA3	61.82±0.49	56.03±0.46	50.10±0.41	457.37±0.57	2.38±0.03	0.00±0.00	0.00±0.00	0.00±0.00
AlphaEdit	LLaMA3	97.30±0.16	**92.29±0.22**	61.57±0.31	606.44±0.27	31.90±0.12	93.78±0.15	89.61±0.21	31.96±0.22
LyapLock	LLaMA3	**99.16±0.09**	90.56±0.25	**73.82±0.27**	**621.58±0.21**	**32.68±0.12**	**95.20±0.13**	**91.58±0.19**	**32.05±0.22**
Pre-edited		14.78±0.35	17.17±0.33	83.45±0.25	621.80±0.14	29.48±0.10	27.04±0.29	26.25±0.28	27.00±0.26
FT	GPT-J	95.28±0.21	77.45±0.36	42.22±0.37	351.58±0.93	10.53±0.13	68.14±0.32	65.12±0.34	16.27±0.21
ROME	GPT-J	50.48±0.50	51.18±0.40	52.46±0.31	576.99±0.15	1.87±0.01	28.64±0.40	26.29±0.39	1.82±0.39
MEMIT	GPT-J	89.44±0.31	82.47±0.32	56.91±0.34	315.78±0.86	12.12±0.14	72.26±0.36	69.33±0.37	25.80±0.26
PRUNE	GPT-J	74.12±0.44	67.09±0.39	54.32±0.36	397.92±0.70	9.29±0.11	3.93±0.11	3.75±0.11	4.83±0.11
RECT	GPT-J	95.36±0.21	81.14±0.33	65.50±0.31	539.06±0.65	31.39±0.14	87.71±0.25	83.76±0.29	26.19±0.26
AlphaEdit	GPT-J	99.48±0.07	94.70±0.18	68.93±0.28	607.42±0.28	40.66±0.13	98.97±0.07	94.23±0.19	26.18±0.25
LyapLock	GPT-J	**99.64±0.06**	**94.72±0.18**	**70.66±0.28**	**617.83±0.18**	**42.08±0.12**	**99.56±0.04**	**95.58±0.17**	**26.76±0.25**
Pre-edited		21.50±0.41	23.88±0.37	78.24±0.28	626.51±0.12	31.27±0.10	22.80±0.27	21.87±0.26	24.32±0.24
FT	GPT2-XL	67.62±0.47	56.37±0.43	50.40±0.37	582.25±0.52	10.61±0.07	22.79±0.35	19.95±0.33	4.40±0.11
ROME	GPT2-XL	51.02±0.50	49.43±0.41	51.44±0.32	472.37±0.30	0.78±0.01	34.47±0.42	31.73±0.40	3.82±0.10
MEMIT	GPT2-XL	69.32±0.46	63.88±0.41	56.96±0.35	575.07±0.56	16.82±0.12	23.96±0.35	20.74±0.32	11.97±0.18
PRUNE	GPT2-XL	54.86±0.50	52.72±0.42	51.43±0.36	**584.86±0.23**	14.92±0.07	2.55±0.11	2.50±0.10	2.98±0.08
RECT	GPT2-XL	90.68±0.29	75.22±0.36	59.27±0.33	494.77±0.71	14.80±0.15	68.58±0.37	62.54±0.38	20.58±0.23
AlphaEdit	GPT2-XL	**98.52±0.12**	**88.22±0.25**	60.99±0.29	571.43±0.47	36.01±0.14	80.83±0.32	72.36±0.36	20.77±0.23
LyapLock	GPT2-XL	98.40±0.13	88.14±0.26	**63.07±0.29**	584.82±0.45	**36.93±0.13**	**92.89±0.20**	**84.06±0.29**	**24.97±0.25**

Table 7: Editing performance of the LLaMA3 model on the MQuAKE-CF dataset after 3,000 sequential edits.

Method	Edit-wise↑	Instance-wise↑	Multi-hop↑	Multi-hop(CoT)↑
MEMIT	9.17	0.27	1.97	3.43
PRUNE	9.96	0.15	4.63	6.75
RECT	24.60	3.07	**13.13**	14.77
AlphaEdit	68.87	34.27	8.60	25.07
LyapLock	**77.72**	**47.63**	12.27	**27.33**

Table 8: Editing performance of the LLaMA3 model on the QAEdit dataset after 10,000 sequential edits.

Method	Reliability↑		Generalization↑	
	syn.	WILD	syn.	WILD
MEMIT	4.04	0.00	4.13	0.00
PRUNE	0.26	0.00	4.34	0.00
RECT	1.56	0.00	1.34	0.00
AlphaEdit	3.67	0.01	3.88	0.05
LyapLock	**42.62**	**4.92**	**41.23**	**3.59**

our method not only achieves the desired editing effect but also ensures the fluency of the generated text.

E.4 Analysis of the Trend of LyapLock's Preservation Loss

Since, according to Eq. 7, $Z(t)$ is adaptively updated whenever the preservation loss exceeds the threshold D during sequential editing, PL(preservation loss) is kept within a bounded range. Concretely, LyapLock's preservation loss first grows slowly, then decreases, and this cycle repeats. Taking Figure 4(a) as an example:

- Stage One (0-4000 edits): $Z(t)$ is fixed at z_{max}, and the PL rises rapidly (similar to exponential growth).
- Stage Two (4000-10000 edits): When $PL(t)$

Table 9: Model Editing Case Study on LLAMA3

Model Editing Case Study on LLAMA3	
Editing Prompt	The mother tongue of Danielle Darrieux is
Edit Target	English
Generation Output	
FT	Danielle Darrieux's mother tongue is <|begin_of_text|> ://<|eot_id|><|begin_of_text|>://the<|begin_of_text|> ://the<|eot_id|> Moscow<|eot_id|><|eot_id|><|eot_id|>...
ROME	Danielle Darrieux's mother tongue isistrovstvi istrovstvi azzi GenerationType.scalablytyped.scalablytyped.BLL—————-</ addCriterion.scalablytyped erveristrovstvi Europe.scalablytypedIona...
MEMIT	Danielle Darrieux's mother tongue isitionallyulinAdvisoritionallyAdvisorAdvisorenderitionallyitionAdvisorender Patron Patron PatronAdvisor Ibid PatronitionallyAdvisor RussianAdvisor Patronenderitionally...
PRUNE	Danielle Darrieux's mother tongue isitionallyulinAdvisorAdvisorAdvisorenderAdvisorenderekyllAdvisor CastroAdvisorAdvisorender PatronxAD Patronitionally Patron PatronAdvisoritionallyAdvisor Russian...
RECT	Danielle Darrieux's mother tongue is: migrationBuilder Nissan Italian Spanish : : : : : : : : : :...
AlphaEdit	Danielle Darrieux's mother tongue is French English German French French German German Spanish Spanish English Spanish Spanish English...
LyapLock	Danielle Darrieux's mother tongue is English, as she was born in the United States of America, but she grew up speaking Spanish, her mother's native language...

Table 10: Model Editing Case Study on GPT-J

Model Editing Case Study on GPT-J

Editing Prompt	Percy Snow, the
Edit Target	goaltender
Generation Output	
FT	The expertise of Percy Snow becomes important when the latter has to decide who can and can't become a pope. The latter is diplomat to the Russian cheshir cheshir cheshir cheshare cheshare...
ROME	The expertise of Percy Snow becomes important when spiritual...ISC ulk Jewish R leader R ball H ([...] HBO percent and is lifelongDenver M harmful R [harmless M (participtr... savings M Italian...
MEMIT	The expertise of Percy Snow becomes important whenawar [...]englishawar [...] Bronxawar [...] Melbourne worldwide [...] Melbourne Bronx Bronxflight Cuba Mall [...] Melbourne Bronx Bronx Bronx Cuba Bronx Bronx Bronxflight...
PRUNE	The expertise of Percy Snow becomes important whensong Sloveniaawar [...] Melbourneawar [...] Bangkok Bronxflight Cubaawar [...] Melbourneenko [...] Antarctica Lebanonawar...
RECT	The expertise of Percy Snow becomes important whenCtransS SpanishS CambridgeS PhiladelphiaS CambridgenC MassachusettsCCCCCCv CambridgeS Portuguesebr Boston...
AlphaEdit	The expertise of Percy Snow becomes important when it's time to be the The last one to I don't when you Antarctica, Italy, Italy, Italy., and Judaism Judaism Judaism Judaism Judaism Judaism...
LyapLock	The expertise of Percy Snow becomes important when the goalie is injured. The goalie coach has to know the goaltender's style, how to get him going, how to keep him in the game and how to get him to the point where he's not injured...

Table 11: Model Editing Case Study on GPT2-XL

Model Editing Case Study on GPT2-XL	
Editing Prompt	Apple A5 was created by
Edit Target	Google
Generation Output	
FT	Apple A5 is sold by Google and Google, Microsoft in Melbourne, on NBC the model chart Read on NBC to Chennai back Greek in Tokyo that on NBC the model belowmodel with on Microsoft...
ROME	Apple A5 is sold by Ferrari Ž00e Munich Munich Antarctic AsiaWOR Italy treasurer Judaism watch-dog North dazz dime Italy AntarcticaWOR Labour-RAW Melt AntarcticaItaly Antarctica...
MEMIT	Apple A5 is sold by Microsoft Microsoft–Google–Microsoft Apple German Boeing–Microsoft Microsoft Microsoft Microsoft Apple Apple Microsoft B Microsoft Amazon Japan Japan...
PRUNE	Apple A5 is sold by, Microsoft– Apple Apple N N–N–ch–N and Lincoln ',–N that-N-the Nsu ' (; '; ' H, N̈-y N,y N K N-y S, B 'N...
RECT	Apple A5 is sold by Google Google Microsoft Microsoft Microsoft Microsoft Microsoft Microsoft Microsoft...
AlphaEdit	Apple A5 is sold by Microsoft, but it has been a little slow at the Windows store for the past year. The Northern Territory has 6,000 islands Tuesday, horoscopes restaurant jumbi Un jeung un...
LyapLock	Apple A5 is sold by Google in Japan. Apple has released an Android-based phone in Japan The new iPhone 7 will also use Android. Android has already been used in many Apple products, including the iPhone 6, iPhone 6S, Apple Watch Series 3...

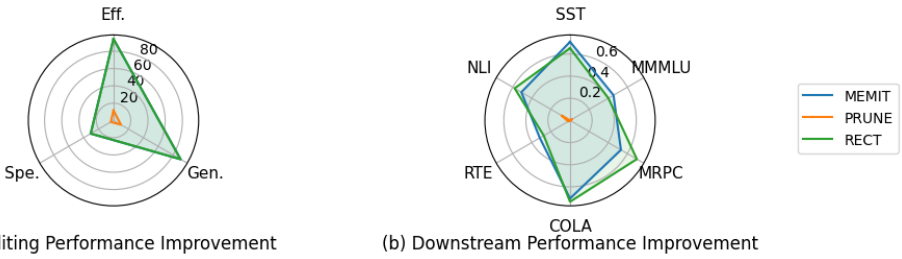
exceeds the preset threshold $D = 1.832$, $a(PL(t) - D) + b$ becomes positive, causing the virtual queue $Z(t)$ to increase. This enhances the weight coefficient $aZ(t)$ of the $PL(t)$ term in the total loss function of Eq. 12. prompting the model to place greater emphasis on retaining the original knowledge. During this stage, the growth rate of PL slows down and gradually begins to decrease (for example, from PL $\approx$ 2.145 at 5000 edits to PL $\approx$ 2.086 at 10000 edits). This change is not a "plateau" (i.e., completely flat), but rather a process of slowing growth and adjusting back to the threshold D.

- Stage Three: As the number of editing steps continues to increase, if $PL(t)$ is below D, $a(PL(t) - D) + b$ will become negative, causing $Z(t)$ to decrease and reducing the weight of the PL term, thus allowing the model to relax the preservation loss requirements when updating knowledge.

Eq. 7, through the dynamic adjustment of $Z(t)$, theoretically controls the PL to fluctuate around the threshold D through Stages Two and Three. Overall, the observed slowing and subsequent decrease in PL (i.e., the seemingly flat area) is the effect of the increased $Z(t)$ and the strengthened knowledge preservation constraints. Due to the complexity of the model structure and the inherent difficulty of controlling the preservation of original knowledge, the process of loss reduction may appear relatively slow.

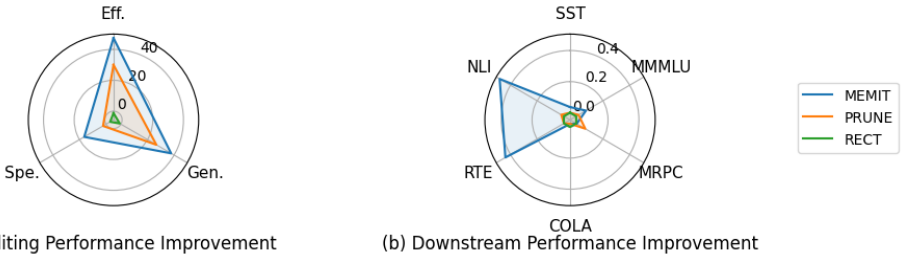
E.5 More compatibility experiment results

We further demonstrate the performance improvement of our method in combination with other baseline methods across various models and datasets after editing, as shown in Figure 6, 7, 8, 9, 10. Overall, our method generally enhances both editing performance and downstream task performance when combined with other baselines. However, the specific degree of improvement varies depending on the model, editing dataset, and method used.



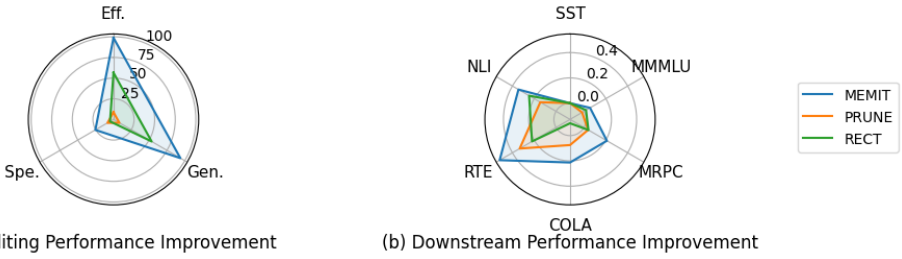
(a) Editing Performance Improvement (b) Downstream Performance Improvement

Figure 6: The improvement in editing performance and downstream task performance of other editing methods after incorporating LyapLock, following the sequential editing of 10,000 samples on the ZsRE dataset using the LLAMA3 model.



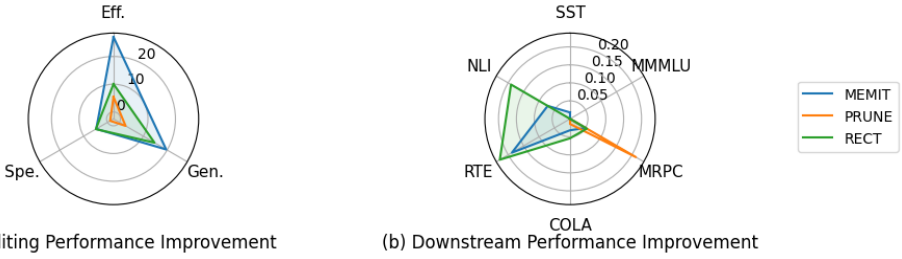
(a) Editing Performance Improvement (b) Downstream Performance Improvement

Figure 7: The improvement in editing performance and downstream task performance of other editing methods after incorporating LyapLock, following the sequential editing of 10,000 samples on the CounterFact dataset using the GPT-J model.



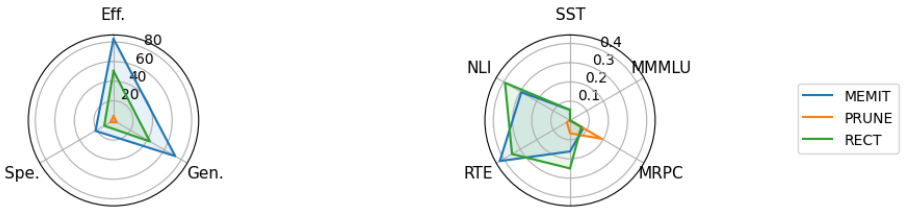
(a) Editing Performance Improvement (b) Downstream Performance Improvement

Figure 8: The improvement in editing performance and downstream task performance of other editing methods after incorporating LyapLock, following the sequential editing of 10,000 samples on the ZsRE dataset using the GPT-J model.



(a) Editing Performance Improvement (b) Downstream Performance Improvement

Figure 9: The improvement in editing performance and downstream task performance of other editing methods after incorporating LyapLock, following the sequential editing of 10,000 samples on the CounterFact dataset using the GPT2-XL model.



(a) Editing Performance Improvement (b) Downstream Performance Improvement

Figure 10: The improvement in editing performance and downstream task performance of other editing methods after incorporating LyapLock, following the sequential editing of 10,000 samples on the ZsRE dataset using the GPT2-XL model.