QueueEDIT: Structural Self-Correction for Sequential Model Editing in LLMs

Taolin Zhang[1], **Haidong Kang**[2], **Dongyang Li**[3], **Qizhou Chen**[4,5], **Xiaofeng He**[5], **Chengyu Wang**[4*], **Richang Hong**[1]
[1] School of Computer Science and Information Engineering, Hefei University of Technology
[2] Northeastern University [3] Shanghai University of Electric Power
[4] Alibaba Group [5] East China Normal University
tlzhang@hfut.edu.cn, chengyu.wcy@alibaba-inc.com

Abstract

Recently, large language models (LLMs) have demonstrated impressive performance but still suffer from hallucinations. Model editing has been proposed as a means to correct factual inaccuracies. A challenging scenario is sequential model editing (SME), which aims to rectify errors continuously, rather than as a one-time task. During SME, the general capabilities of LLMs can be negatively affected due to the introduction of new parameters. In this paper, we propose a queue-based self-correction framework, `QueueEDIT`, that not only enhances SME performance by addressing long-sequence dependencies but also mitigates the impact of parameter bias on the general capabilities of LLMs. Specifically, we first introduce a structural mapping editing loss to map editing triplets to knowledge-sensitive neurons within the Transformer layers. We then store the located parameters for each piece of edited knowledge in a queue and dynamically align previously edited parameters. At each edit, we select parameters in the queue that are most relevant to the currently located parameters to determine whether the knowledge associated with previous edits requires realignment. Irrelevant parameters in the queue are frozen, and we update the LLM with the parameters at the head of the queue to ensure they do not harm general capabilities. Experiments show that `QueueEDIT` significantly outperforms strong baselines across various SME settings, while maintaining competitive performance in single-turn editing. The resulting LLMs also preserve high performance on general NLP tasks throughout the SME process.

1 Introduction

Recently, large language models (LLMs) have become the foundation of modern NLP (Zheng et al., 2022; Blinova et al., 2023). However, LLMs occasionally generate undesirable outputs (Basta et al., 2021; An et al., 2023) and are prone to hallucinations (Shi et al., 2023; Tam et al., 2023), creating content that appears plausible but lacks factual support. Although fine-tuning models with updated knowledge offers a solution, it is often impractical due to excessive time requirements (Hübotter et al., 2025; Kim et al., 2025; Krishna et al., 2025). To address these issues, there is increasing interest in incorporating knowledge into LLMs via model editing (Madaan et al., 2022; Fang et al., 2025).

Previous approaches on model editing broadly fall into three categories: `Modifying Parameters`, `Adding Extra Parameters`, and `Retrieving Data`. (1) `Modifying Parameters` approaches edit one or a batch of knowledge triples at a time by adjusting LLM parameters using meta-learning (Cao et al., 2021; Mitchell et al., 2022a) or locate-then-edit techniques (Hartvigsen et al., 2022; Meng et al., 2022, 2023). These methods locate neurons in the FFN layers corresponding to edited data and update related parameters. (2) `Adding Extra Parameters` methods augment LLMs with supplementary neurons for each edit, bypassing alterations to original parameters (Mitchell et al., 2022b; Huang et al., 2023; Zhang et al., 2024). Here, updates are not achieved through backpropagation but rather via additional modules associated with each fact. While these methods quickly fix mistakes, parameters of previous edits that are relevant to the current edit cannot be updated simultaneously. (3) `Retrieving Data` approaches retrieve external information related to knowledge triplets and concatenate it into the input to form final editing data (Jiang et al., 2024; Chen et al., 2024). However, continuous retrieval and concatenation rapidly increase the length of LLM input, resulting in longer inference time and lower editing accuracy (Li et al., 2023; Liu et al., 2024). Additionally, as the number of edits increases, outdated editing information can lead to incorrect answers, negatively affecting LLM performance due

* Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 4706–4719

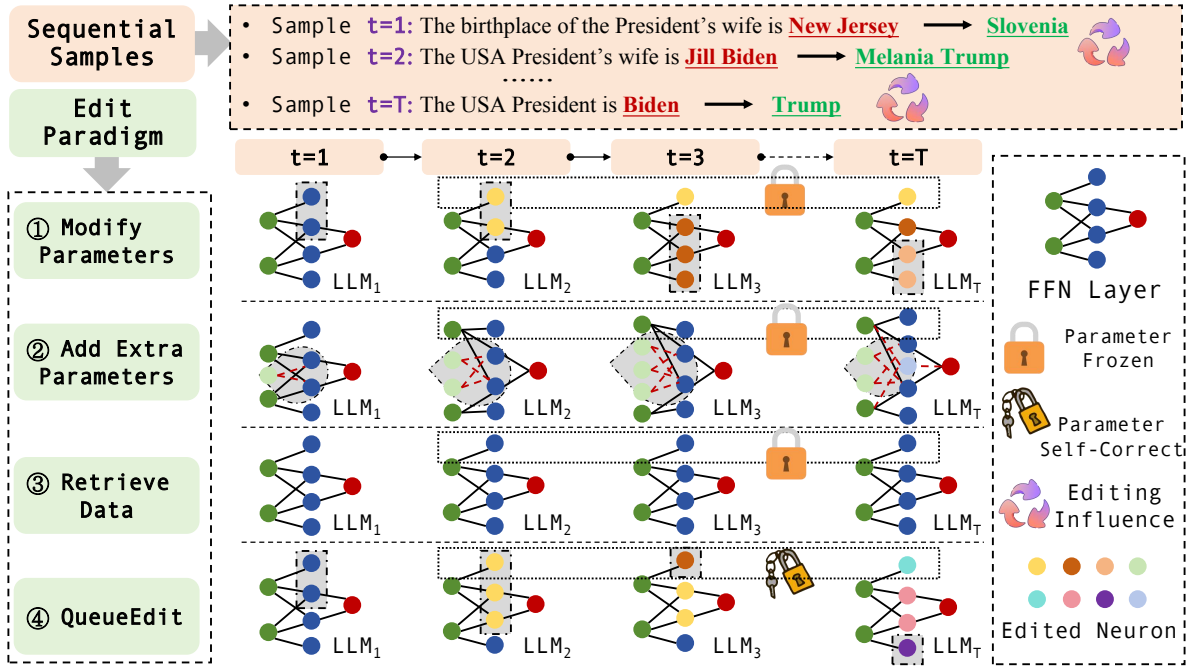

Figure 1: Comparison between SME methods. `Modifying Parameters` and `Adding Extra Parameters` approaches change parameters at specific positions in FFN layers without making corresponding updates for related edits. `Retrieving Data` methods focus solely on external data for editing triples. In contrast, `QueueEDIT` dynamically updates parameters associated with previous edits while also preserving the general capabilities of LLMs.

to knowledge bias. As shown in Figure 1, when the current edit "Donald Trump" is ready to be updated, parameters related to previous edits should be adjusted accordingly. Otherwise, a subsequent question about the "wife of the USA President" may still yield the incorrect answer "Jill Biden".

In this paper, we propose a queue-based self-correction framework, `QueueEDIT`. This framework performs knowledge editing for each triplet while simultaneously adjusting specific parameters in FFNs that were previously updated, aiming to maintain SME performance and preserve the general capabilities of LLMs. The key techniques of `QueueEDIT` are as follows:

Structural Mapping Editing. To more accurately update the knowledge of the current edit to specific FFN layers (Meng et al., 2022; Fang et al., 2025), we map the editing triplets (i.e., $< s, r, o >$[1]) to knowledge-sensitive positions in FFN layers. This approach enhances the response of internal activation parameters compared to previous works that disregarded the structural semantics of knowledge triplets (Mitchell et al., 2022a; Mishra et al., 2024). Previous methods utilized both the "relation" and "object" to learn representations for gradient feedback, overlooking the original structure of knowledge triplets. In our method, we map the subject, relation, and object in the editing data to the first MLP matrix, the second MLP matrix, and the gradient backpropagation representation in the Transformer's FFN layer, respectively. Specifically, the "relation", originally viewed as a bridge connecting the semantic knowledge of subject and object entities (Bordes et al., 2013; Wang et al., 2014), is independently mapped to the second parameter matrix in the FFN layer. We then design a structural loss to update parameters via the triple's representations, enhancing editing results.

Queue-based Self-Correction. In the context of SME, it is crucial to consider inherent semantic associations among edited data. To manage dependency relationships in a long sequence of edits, we design a queue-based structure to store updated editing parameters following a first-in, first-out (FIFO) principle. Specifically, we insert the current editing parameters at the queue's end and calculate the distance between current parameters and those already in the queue. After sorting by distance, we update the original queue parameters by comparing object entity gradients with relation representations from the current edit. For computational efficiency, we select only top-K editing

[1]The editing samples are sourced from knowledge graph (KG) triples $< subject, relation, object >$ ($< s, r, o >$).

parameters for sequential updates. Finally, we remove the oldest parameters from the queue head to ensure knowledge freshness.

In our experiments, we evaluate QueueEDIT against model editing baselines on both single-turn and multi-turn editing, as well as on general NLP datasets. For SME, our method significantly surpasses baselines as the number of edits increases, and even provides an advantage in single-turn editing. Regarding general capabilities, our model shows better consistency than original LLMs, without degradation in overall performance.

2 Related Work on LLM Editing

Modifying Parameters. These approaches can be further divided into Locate-then-Edit and meta-learning-based methods. For Locate-then-Edit, ROME (Meng et al., 2022) and MEMIT (Chenmien Tan and Fu, 2023) introduce a causal intervention framework to identify neuron activations and then update knowledge data with low-rank-based model editing to modify parameters. AlphaEdit (Fang et al., 2025) projects perturbations onto the null space of preserved knowledge prior to their application to model parameters.

For meta-learning methods, KnowledgeEditor (Cao et al., 2021) and MEND (Mitchell et al., 2022a) employ distinct methodologies, converting edited knowledge and decomposed gradients of the LLM into weight-offset modifications, respectively. MALMEN (Chenmien Tan and Fu, 2023) advances this paradigm further by incorporating normal equations to optimize parameter integration for batch-editing operations. DAFNet (Zhang et al., 2024) introduces intra-editing and inter-editing attention flows to update weighted representations at sequence-level granularity.

While these methods demonstrate efficacy in single or batch editing, the progressive accumulation of parameter modifications with an increasing number of edits may ultimately lead to editing degradation (Hu et al., 2024).

Adding Extra Parameters. This paradigm does not directly modify specific FFN-layer parameters but adds training modules at corresponding positions to incorporate the edited data into the LLM. CaLiNet (Dong et al., 2022) and T-Patcher (Huang et al., 2023) achieve model editing by introducing additional neurons to the LLM for each piece of editing knowledge, thereby avoiding modifications to the original model parameters. GRACE (Hartvigsen et al., 2022) employs an adapter module that establishes a mapping between input queries and their corresponding knowledge representations. However, in an SME scenario, the persistent incorporation of new neurons to update the current edited data can overlook the correlations between different sequential editing data and increase the burden on model inference speed.

Retrieving Data. This approach enhances the model's reasoning capabilities in a timely manner by continuously retrieving external data to supplement editing data without introducing additional parameters. Extending the GRACE framework, MELO (Yu et al., 2024) proposes a batch-editing implementation leveraging LoRA technology (Hu et al., 2022). LTE (Jiang et al., 2024) fine-tunes the LLM to generate appropriate responses when provided with knowledge prefixed by editing cues, while leveraging the pre-trained backbone architecture for relevant content retrieval (Reimers and Gurevych, 2019). However, the aforementioned methods, whether modifying parameters or retrieving external data, do not model the associations between sequential editing data in SME scenarios, resulting in logical errors in factual answers provided by LLMs.

3 Methodology

In this section, we introduce the basics of SME, including its definition, properties, and training losses. Then, we describe the main components of QueueEDIT, namely **Structural Mapping Editing** and **Queue-based Self-Correction**. The overall framework is illustrated in Figure 2.

3.1 Preliminaries of SME Task

Definition and Properties. Given an LLM f and an edit example (x_e, y_e) such that $f(x_e) \neq y_e$, a model editor (ME) outputs a post-edit model: $f' = \mathrm{ME}(f, x_e, y_e)$. In SME, given a sequence of facts $\{(x_{e_1}, y_{e_1}), \ldots, (x_{e_T}, y_{e_T})\}$ and an initial model f, a ME conducts edits successively:

$$f_t = \mathrm{ME}\left(f_{t-1}, x_{e_t}, y_{e_t}\right),$$

where $t = 1, \ldots, T$ and $f_0 = f$. Every edit should satisfy three fundamental properties: reliability, generality, and locality (Zhang et al., 2024).

Reliability: An edit is reliable if the post-edit model f_t gives the target answer for every case (x_{e_τ}, y_{e_τ}) up to time t. Reliability is measured by average accuracy on the edited cases:

$$\mathbb{E}_{(x_e, y_e) \sim \{(x_{e_\tau}, y_{e_\tau})\}_{\tau=1}^{t}} \mathbb{I}\left\{\mathrm{argmax}_y f_t(y \mid x_e) = y_e\right\} \quad (1)$$

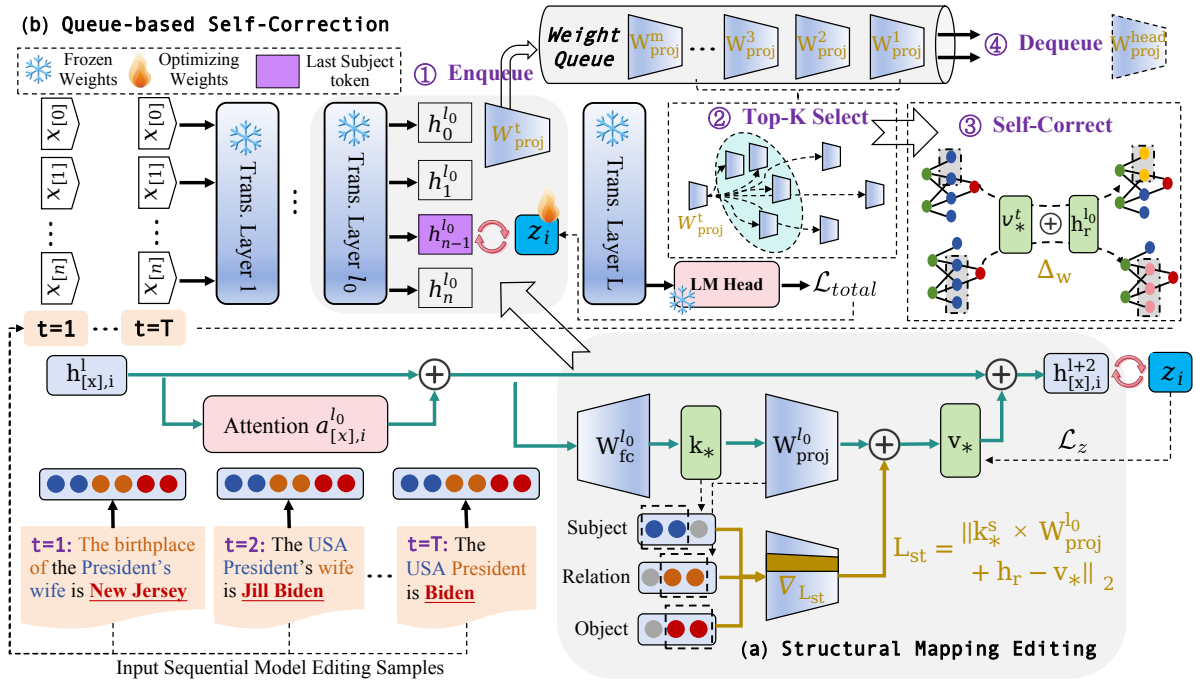

Figure 2: The QueueEDIT framework. **(a) Structural Mapping Editing** maps each triple (s, r, o) to dedicated positions (i.e., k^s_*, h_r, and v_*) within the FFN layer. **(b) Queue-based Self-Correction** stores located parameters into a Queue following FIFO for subsequent alignment, aiming to preserve the general capabilities of LLMs.

Generality: The post-edit model f_t should also generalize to relevant neighbors $N(x_{e_\tau}, y_{e_\tau})$ for all $\tau \leq t$. This is evaluated by average accuracy over examples drawn uniformly from the relevant neighborhood:

$$\mathbb{E}_{(x_e,y_e)\sim\{(x_{e_\tau},y_{e_\tau})\}_{\tau=1}^{t}} \mathbb{E}_{(x_g,y_g)\sim N(x_e,y_e)} \mathrm{G}(x_g, y_g)$$
$$\text{s.t. } \mathrm{G}(x_g, y_g) = \mathbb{I}\left\{\operatorname{argmax}_y f_t(y \mid x_g) = y_g\right\} \quad (2)$$

Locality: Editing should be local, meaning f_t should not change the outputs for irrelevant examples in $O(x_{e_\tau}, y_{e_\tau}), \tau \leq t$. Locality is evaluated by the rate at which predictions are unchanged compared to the pre-edit model f:

$$\mathbb{E}_{(x_e,y_e)\sim\{(x_{e_\tau},y_{e_\tau})\}_{\tau=1}^{t}} \mathbb{E}_{(x_l,y_l)\sim O(x_e,y_e)} \mathrm{L}(x_l, y_l)$$
$$\text{s.t. } \mathrm{L}(x_l, y_l) = \mathbb{I}\left\{f_t(y \mid x_l) = f(y \mid x_l)\right\} \quad (3)$$

Model Training. Let $(x_e^{(t)}, y_e^{(t)})$ be the reliability sample of the t-th fact (i.e., the editing sample itself). $(x_{g_j}^{(t)}, y_{g_j}^{(t)})$ and $(x_{l_j}^{(t)}, y_{l_j}^{(t)})$ are the j-th generality and locality samples associated with the t-th fact, respectively. $N_g^{(t)}$ and $N_l^{(t)}$ are the corresponding sample counts for the t-th fact. The total loss $\mathcal{L}_{ed}(f_T)$ is the sum of:

$$\mathcal{L}_{rel}(f_T) = \sum_{t=1}^{T} -\log f_T(y_e^{(t)} \mid x_e^{(t)}) \quad (4)$$

$$\mathcal{L}_{gen}(f_T) = \sum_{t=1}^{T} \sum_{j=1}^{N_g^{(t)}} -\log f_T(y_{g_j}^{(t)} \mid x_{g_j}^{(t)}) \quad (5)$$

$$\mathcal{L}_{loc}(f, f_T) = \sum_{t=1}^{T} \sum_{j=1}^{N_l^{(t)}} \mathrm{KL}(f(x_{l_j}^{(t)}) \,||\, f_T(x_{l_j}^{(t)})) \quad (6)$$

3.2 Structural Mapping Editing

Our structural mapping editing approach maps the triplet structure to specific positions in the Transformer's MLP layer, establishing semantic associations between the masked entity (i.e., the "Object") and the "Subject" entity through independent "Relation" learning.

Locating the MLP Layer. Following Meng et al. (2022, 2023); Zhang et al. (2024), MLP layers in Transformers are selected for editing via causal tracing. These works analyze internal activations through three experimental runs to identify the layer with the largest indirect effect, denoted as l_0. The layers function as two-layer key–value memories, where neurons of the first layer, $W^{l_0}_{fc}$, form a *key*, and neurons of the second layer, $W^{l_0}_{proj}$, represent the associated *value*.

Structural Triplet Editing. Editing a sample (x_e, y_e) from a new triple $t^* = (s, r, o^*)$, replacing $t = (s, r, o)$, demonstrates fine-grained control of association-storage mechanisms. The parameters are updated using a closed-form solution (Meng et al., 2022, 2023):

$$W^{l'_0}_{proj} = W^{l_0}_{proj} + \Lambda(C^{-1}k_*)^T \quad (7)$$

where $\Lambda = \frac{(v_* - W_{proj}^{l_0} k_*)}{(C^{-1}k_*)^T k_*}$, and $C = KK^T$ is precomputed using cached Wikipedia embeddings K from editing triples.[2] Here, k_* is obtained by averaging over N sampled texts x made from templates containing the subject s and relation r:

$$k_* = \frac{1}{N}\sum_{i=1}^{N} \sigma\left(W_{fc}^{l_0}\gamma(a_{[x],i}^{l_0} + h_{[x],i}^{(l_0-1)})\right) \quad (8)$$

where $a_{[x],i}^{l_0}$ is the self-attention output and $h_{[x],i}^{(l_0-1)}$ is the hidden input of the previous layer. Here, γ denotes a normalization nonlinearity and σ an activation function. Note in Eq. 8 that the relation r is mixed into the input x without separately modeling its semantics with subject s and object o^*.

Triples $t^* = (s, r, o^*)$ can be represented by translation-based models (Bordes et al., 2013), which imply that the semantics of o^* can be approximated by those of s and r. We inject the structural semantics into parameters to enhance memorization in MLP layers. Specifically, we first leverage s and r to generate k_* via W_{fc} from Eq. 8, then split hidden representations as:

$$k_*^s = k_*[s_m : s_n], \quad k_*^r = k_*[r_m : r_n] \quad (9)$$

$$h_r = \sigma\left(k_*^r W_{proj}^{l_0} + b_r\right) \quad (10)$$

where s_m, s_n are the positions of subject tokens and r_m, r_n those of relation tokens in the editing data; b_r is a bias. Thus, k_*^s and h_r represent subject and relation embeddings, respectively.

Next, we separate out the relation r and model it explicitly for editing the new triple (s, r, o^*), rather than binding it with subject s. The structural editing loss $\mathcal{L}_{st}$ is defined as:

$$\mathcal{L}_{st} = \left\| k_*^s W_{proj}^{l_0} + h_r - v_* \right\|_2 \quad (11)$$

where the gradient w.r.t. $\mathcal{L}_{st}$ only updates $W_{proj}^{l_0}$, with all other parameters frozen.

By employing this translation-based loss, we constrain the transferred representation "$(s+r) \rightarrow o^*$" to approximate o^* as closely as possible, while preserving the features of old knowledge o stored in located FFN layer $W_{proj}^{l_0}$.

3.3 Queue-Based Self-Correction

Previous SME methods (Cao et al., 2021; Mitchell et al., 2022a; Zhang et al., 2024) independently locate and edit $W_{proj}^{l_0}$ for different triples, ignoring semantic influence among editing parameters. This oversight leads to bias and can degrade the general capabilities of LLMs (You et al., 2024). In `QueueEDIT`, we introduce a self-correction mechanism with a queue to update located knowledge parameters during each edit. Specifically, after computing each edited parameter (denoted as $\mathbf{W_{proj}^t}$), we update the parameters in the queue in the following steps to better capture semantic correlation across sequential edits and alleviate degradation of LLM capabilities.

Step 1: Enqueuing Located Parameters. Append the current edited parameter $\mathbf{W_{proj}^t}$ to the end of the weight queue $Q = \langle W_{proj}^1, W_{proj}^2, \ldots, W_{proj}^m \rangle$ $(m \geq 0)$, where m is the number of parameters currently in queue Q. Thus, the updated queue is $Q = \langle W_{proj}^1, W_{proj}^2, \ldots, W_{proj}^m, \mathbf{W_{proj}^t} \rangle$.

Step 2: Selecting Top-K Parameters. Following FIFO, parameter bias at the front of the queue caused by the semantic gap with $\mathbf{W_{proj}^t}$ greatly impacts editing effectiveness (Gupta et al., 2024; Gupta and Anumanchipalli, 2024). To trade off queue length and memory, we next identify the `Top-K` elements needing alignment. Similarity in the editing parameter space between the current sample t and others is computed using Euclidean distance:

$$d_i = \|\mathbf{W_{proj}^t} - W_{proj}^i\|_2, \quad i = 1, \ldots, m \quad (12)$$

$$\mathcal{I}_{\text{topK}} = \{\sigma(1), \sigma(2), \ldots, \sigma(K)\}, \quad K \leq m \quad (13)$$

Here, $\sigma(\cdot)$ gives sorting order so $d_{\sigma(1)} \leq d_{\sigma(2)} \leq \cdots \leq d_{\sigma(m)}$, and $\mathcal{I}_{\text{topK}}$ indexes the Top-K parameters. We align semantic spaces between $\mathbf{W_{proj}^t}$ and W_{proj}^i only when similarity d_i is below a threshold η_{que}.

Step 3: Editing with Self-Correction. To iteratively update dependencies from previous edits in the SME task, we observe (see Figure 1) that updated parameters typically link the object o_*^t of the new edit to the relation r^i of previous knowledge. In general, replacing the object of the t-th sample with the subject of the i-th sample is sufficient. All other parameters in the LLM and queue Q are frozen to preserve LLM general capabilities. This semantic superposition is analogous to prior works on embedding translation (Bordes et al., 2013).

Specifically, we leverage the `Top-K` parameters $\{W_{proj}^1, W_{proj}^2, \ldots, W_{proj}^K\}$ and align them with

[2] We use backpropagation considering the object entity o^* to obtain v_* (Meng et al., 2022); see Appendix C.

the current edit $\mathbf{W}^{\mathbf{t}}_{\mathbf{proj}}$ as follows:

$$\Delta_W = v_*^t \oplus h_r^i, \quad i \in \mathcal{I}_{\text{topK}} \tag{14}$$

$$W_{proj}^{i'} = \sigma\left(W_{proj}^{i} \parallel \Delta_W + b'\right) \tag{15}$$

where $\oplus$ is elementwise addition, $\parallel$ is concatenation, and $W_{proj}^{i'}$ is the self-corrected editing parameter to be aligned back into the LLM. After self-correction, the oldest queue entry is dequeued.

Step 4: Dequeuing Located Parameters. According to FIFO, the triple at the head of the queue W_{proj}^{head} is least related to W_{proj}^{t}; its semantic similarity d_{head} is computed by Eq. 12 to determine whether to dequeue:

$$\text{Dequeue}(Q) \triangleq \begin{cases} Q \setminus \{W_{proj}^{head}\} & d_{head} > \eta_{deq}, \\ Q & \text{otherwise} \end{cases} \tag{16}$$

where η_{deq} denotes the dequeue threshold.

3.4 Model Training

To satisfy the three key SME properties (reliability, generality, locality), we adopt the same loss formulations as previous model editing work (Meng et al., 2023; Zhang et al., 2024). The total loss sums the structural editing loss and standard model editing losses, defined as follows:

$$\mathcal{L}_{total} = \alpha_1 \cdot \mathcal{L}_{ed}(f_T) + \alpha_2 \cdot \mathcal{L}_{st}(f_T), \tag{17}$$

where α_i are coefficients satisfying $\alpha_1 + \alpha_2 = 1$.

4 Experiments

In this section, we conduct extensive experiments to evaluate the performance of our approach. Due to space limitations, dataset descriptions, baselines, and implementation details are provided in Appendix A.

4.1 Main Results

Model Editing Results. We evaluate `QueueEDIT` on three benchmarks and compare its performance across 1,000 edits. Table 1 presents the overall results, from which we observe the following:

1. *Methods based on modifying parameters* show a sharp decrease in performance as the number of edits increases. We hypothesize that meta-learning methods fail to account for the sequential modeling of facts. The weight updates for located neurons in meta networks are carried out separately with limited data, preventing these methods from capturing relationships between sequential edits. While locate-then-edit models exhibit less degradation than meta-learning-based models as the number of edits increases, we find they are more sensitive to the backbone architecture. For instance, with LLaMA3 as the backbone, such models approach zero accuracy in long editing scenarios (e.g., 1,000 edits). KN (Dai et al., 2022) achieves editing via excessive activation for multiple facts, but suffers at large edit counts.

2. *Methods that add extra parameters* leave the original parameters unchanged, instead introducing new neurons or modules at located positions. Although these methods can partially capture inherent sequence connections, model performance gradually declines as the number of edits increases, regardless of the backbone.

3. *Retrieval-based methods* freeze the original LLM parameters and do not introduce additional parameters; retrieving external data for each editing sample fails to correlate or model previous edits effectively.

4. `QueueEDIT` modifies only the parameters relevant to the current edit and employs a queue structure to self-correct correlations with previously edited data. Consequently, the model not only achieves strong editing results but also maintains stability throughout long editing sequences.

General Capabilities of LLMs. We further examine how various editors affect the general capabilities of LLMs during SME. Figure 3 shows the averaged accuracy for LLaMA3 (8B) after 10, 100, 500, and 1,000 edits across public LLM benchmarks. These datasets typically consist of QA tasks similar to the editing samples, thereby reducing bias from differences in data format between training and testing. We observe that prior SME methods impair general capabilities of LLMs as the number of edits increases. GRACE (Hartvigsen et al., 2022) applies additional datasets and adapters to help preserve general performance. Retrieval-based methods maintain relatively better generalization than other baselines, presumably because the retrieved external data introduce diverse knowledge types, mitigating generality loss. In contrast,

Backbone	Editor	ZSRE				CounterFact				RIPE			
		Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
GPT-J (6B)	FT	4.3	3.0	0.1	$2.5_{(\pm 0.1)}$	12.9	5.1	1.1	$6.4_{(\pm 0.1)}$	3.1	0.9	0.8	$1.6_{(\pm 0.0)}$
	KN	0.8	0.0	2.2	$1.0_{(\pm 0.0)}$	0.1	0.4	1.0	$0.5_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
	ROME	57.2	53.9	29.9	$47.0_{(\pm 1.1)}$	0.2	0.2	0.0	$0.1_{(\pm 0.0)}$	47.5	16.9	13.4	$26.0_{(\pm 0.5)}$
	MEMIT	56.8	54.6	54.9	$55.4_{(\pm 1.3)}$	42.3	36.4	30.7	$36.5_{(\pm 1.3)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
	KE	0.0	0.0	1.1	$0.4_{(\pm 0.0)}$	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$	0.0	0.0	0.2	$0.1_{(\pm 0.0)}$
	MEND	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$	0.2	0.1	0.1	$0.1_{(\pm 0.0)}$
	MALMEN	43.0	35.1	39.3	$39.1_{(\pm 0.4)}$	15.0	12.4	25.1	$17.5_{(\pm 0.4)}$	31.1	19.1	35.3	$28.5_{(\pm 0.6)}$
	DAFNet	60.0	57.6	88.0	$68.5_{(\pm 1.9)}$	53.1	38.1	82.3	$57.8_{(\pm 1.2)}$	48.3	31.3	57.3	$45.6_{(\pm 1.2)}$
	AlphaEdit	40.2	35.2	62.1	$45.8_{(\pm 1.3)}$	37.5	21.4	59.5	$39.5_{(\pm 1.5)}$	31.6	19.8	43.2	$31.5_{(\pm 1.1)}$
	TP	45.7	40.4	10.5	$32.2_{(\pm 0.8)}$	47.3	17.0	1.4	$21.9_{(\pm 0.7)}$	48.1	29.1	15.2	$30.8_{(\pm 0.6)}$
	GRACE	56.2	51.3	28.4	$45.3_{(\pm 1.2)}$	0.3	0.4	0.1	$0.3_{(\pm 0.1)}$	46.7	16.3	13.8	$25.6_{(\pm 0.7)}$
	MELO	48.7	42.6	68.8	$53.4_{(\pm 1.3)}$	41.6	28.3	65.1	$45.0_{(\pm 0.5)}$	34.2	23.6	48.3	$35.4_{(\pm 1.1)}$
	LTE	53.2	46.5	73.5	$57.7_{(\pm 1.4)}$	46.6	32.5	73.1	$50.7_{(\pm 1.0)}$	41.2	28.7	53.0	$41.0_{(\pm 0.6)}$
	QueueEDIT	**64.5**	**60.1**	**92.6**	$\mathbf{72.4}_{(\pm 1.1)}$	**64.8**	**43.2**	**88.4**	$\mathbf{65.5}_{(\pm 1.1)}$	**51.7**	**37.9**	**62.2**	$\mathbf{50.6}_{(\pm 0.7)}$
LLAMA3 (8B)	FT	8.4	7.2	5.1	$6.9_{(\pm 0.1)}$	2.0	0.6	2.2	$1.6_{(\pm 0.0)}$	3.2	1.5	2.7	$2.5_{(\pm 0.0)}$
	KN	0.5	0.8	0.5	$0.6_{(\pm 0.1)}$	0.9	0.2	0.1	$0.4_{(\pm 0.1)}$	0.6	0.4	0.5	$0.5_{(\pm 0.1)}$
	ROME	2.1	2.0	1.1	$1.7_{(\pm 0.1)}$	0.7	0.6	0.6	$0.6_{(\pm 0.1)}$	0.5	0.5	0.5	$0.5_{(\pm 0.1)}$
	MEMIT	0.7	0.7	0.6	$0.7_{(\pm 0.1)}$	0.6	0.6	1.5	$0.9_{(\pm 0.1)}$	0.1	0.5	0.6	$0.4_{(\pm 0.1)}$
	KE	0.3	0.8	0.6	$0.5_{(\pm 0.1)}$	0.5	0.5	0.8	$0.6_{(\pm 0.1)}$	0.0	0.2	0.1	$0.1_{(\pm 0.1)}$
	MEND	0.8	0.1	0.3	$0.4_{(\pm 0.1)}$	0.3	0.8	0.4	$0.5_{(\pm 0.1)}$	0.0	0.2	0.6	$0.3_{(\pm 0.1)}$
	MALMEN	32.9	29.4	29.0	$30.4_{(\pm 0.6)}$	16.7	17.3	23.4	$19.1_{(\pm 0.3)}$	43.2	39.3	39.4	$40.6_{(\pm 0.9)}$
	DAFNet	51.4	49.5	94.5	$65.1_{(\pm 1.3)}$	51.3	36.7	77.8	$55.3_{(\pm 1.6)}$	45.0	35.2	86.7	$55.6_{(\pm 0.9)}$
	AlphaEdit	34.2	28.6	71.8	$44.9_{(\pm 1.1)}$	34.6	23.1	65.0	$40.9_{(\pm 1.3)}$	29.1	22.4	71.1	$40.9_{(\pm 1.1)}$
	TP	48.2	45.0	5.1	$32.8_{(\pm 0.6)}$	65.6	33.4	12.3	$37.1_{(\pm 0.9)}$	43.2	27.7	10.8	$27.2_{(\pm 0.6)}$
	GRACE	2.0	2.1	1.3	$1.8_{(\pm 0.1)}$	0.6	0.7	0.6	$0.6_{(\pm 0.1)}$	0.3	0.2	0.5	$0.3_{(\pm 0.1)}$
	MELO	39.2	32.5	75.3	$49.0_{(\pm 1.4)}$	40.3	27.1	70.2	$45.9_{(\pm 1.3)}$	34.5	26.2	75.8	$45.5_{(\pm 1.6)}$
	LTE	43.7	40.6	82.3	$55.5_{(\pm 1.0)}$	44.5	31.7	73.5	$49.9_{(\pm 1.4)}$	42.0	32.6	81.5	$52.0_{(\pm 1.2)}$
	QueueEDIT	**56.1**	**55.0**	**96.2**	$\mathbf{69.1}_{(\pm 1.3)}$	**60.5**	**44.7**	**80.1**	$\mathbf{61.8}_{(\pm 0.7)}$	**49.1**	**44.4**	**90.0**	$\mathbf{61.2}_{(\pm 0.8)}$

Table 1: Overall results of QueueEDIT under 1000 edits. "Rel.", "Gen.", and "Loc." represent the editing metrics, respectively. Due to space limitations, **the other results for 1, 10, and 100 edits** are provided in Appendix B.1.

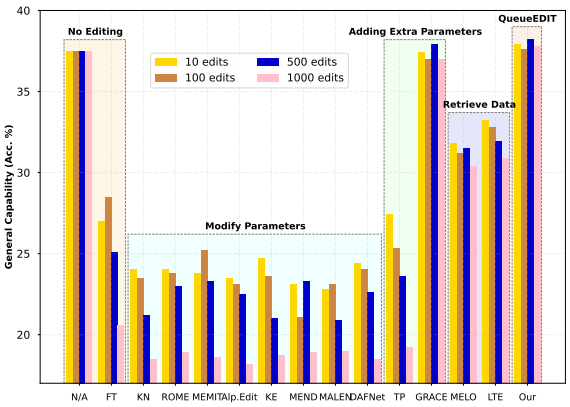

Figure 3: The "improvement/degeneration" of general capability after performing the SME task across various editing paradigms. "N/A" indicates the original results of the LLMs.

QueueEDIT performs self-correction only on parameters corresponding to previously associated edits during queue updates, freezing other parameters both in the queue and in the LLM. As a result, our model yields superior generalization on open-domain questions.

Modules	100 edits			1000 edits			Avg.
	Rel.	Gen.	Loc.	Rel.	Gen.	Loc.	
Ours	77.1	56.5	91.1	55.2	54.7	88.8	70.6
w/o $\mathcal{L}_{st}$	72.4	51.8	84.9	48.3	49.9	82.3	64.9
w/o Queue	59.0	45.2	73.8	40.4	39.5	69.7	54.6
w/o Top-K	73.5	53.6	88.0	51.3	50.5	86.1	67.2

Table 2: Ablation study of QueueEDIT.

4.2 Ablation Study

We conduct an ablation study using LLaMA3 (8B), with averaged results summarized in Table 2. To assess the contribution of each component, we perform the following modifications: structural mapping editing is replaced by the original editing representation k_*, which mixes the subject and relation (Meng et al., 2022). When we remove queue-based self-correction, the framework is reduced to a basic locate-then-edit paradigm equipped with structural editing loss. Additionally, we examine the effect of parameter alignment by removing the Top-K calculation in favor of random selection.

We observe that omitting queue-based learning

Backbone	QL	Memory (GB)	ZSRE			CounterFact			RIPE			Average	-
			Rel.	Gen.	Loc.	Rel.	Gen.	Loc.	Rel.	Gen.	Loc.		
LLAMA3 (8B)	10%	12.5	49.4	50.4	92.0	47.5	38.8	70.3	45.0	41.0	80.1	$57.2_{(\pm 0.5)}$	-
	30%	**21.8**	**56.1**	**55.0**	**96.2**	**60.5**	**44.7**	**80.1**	**49.1**	**44.4**	**90.0**	$\mathbf{64.0}_{(\pm 0.8)}$	-
	50%	32.3	49.0	51.8	92.5	52.4	38.2	73.2	45.6	39.7	83.5	$58.4_{(\pm 0.2)}$	-
	Memory Consumption & Average Performance of Baselines												
	FT	KN	ROME	MEMIT	KE	MEND	MALMEN	DAFNet	AlphaEdit	TP	Grace	MELO	LTE
Memory (GB)	60.2	31.4	18.3	25.6	13.9	16.2	13.7	30.5	26.7	14.8	47.1	36.5	28.3
Average	3.7	0.5	0.9	0.7	0.4	0.4	30.0	58.7	42.2	32.4	0.9	46.8	52.5

Table 3: Results of QueueEDIT with different queue lengths in 1000 edits. "QL" indicates the queue length.

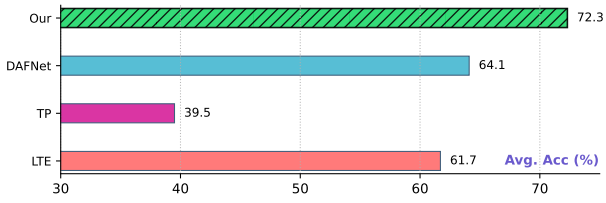

Figure 4: Results using Qwen2.5-14B as the backbone.

causes a substantial performance drop, attributable to the absence of alignment among sequential editing samples within the editing parameters. The performance decrease observed when removing the structural mapping loss indicates that incorporating structural semantics into editing knowledge is more effective than directly mixing triplet data, as in previous works (Meng et al., 2022, 2023). Finally, the reduced performance after eliminating the Top-K selection demonstrates that randomly aligning semantically unrelated triple parameters disrupts parameter consistency and further limits editing effectiveness.

4.3 Detailed Analysis

Queue Length. We analyze both memory usage and performance changes as queue length increases. Methods that add extra parameters require additional memory to train adapter modules. We set the queue length to 10%, 30%, and 50% of 1,000 edits to evaluate editing performance and memory consumption. For memory consumption, we compare the number of parameters used by the additional editor, disregarding those in frozen LLMs. As shown in Table 3, although QueueEDIT consumes slightly more memory than baseline methods, its performance improves substantially as the number of edits grows. In addition, we find that longer queue lengths lead to less consistent editing performance, possibly due to the introduction of more irrelevant knowledge parameters and increased semantic noise. This is likely because, during self-correction updates, a longer Top-K queue lowers the correlation between updated MLP weights, which may disrupt the previously optimized parameter space.

Qualitative Analysis of Queue-Based Self-Correction. We present case studies demonstrating queue-based self-correction of relevant knowledge within QueueEDIT. Due to space limitations, please refer to Appendix B.2 and 3.3 for details.

Larger LLMs as Backbones. Mainstream LLM backbones used in model editing include GPT-J (6B) and LLaMA3 (8B) (Mitchell et al., 2022b; Zhang et al., 2024; Fang et al., 2025). We further evaluate our model on the larger-scale Qwen2.5-14B (Qwen Team, 2024). Experiments include several competitive editing paradigms (DAFNet, T-Patcher, and LTE) with 1,000 sequential edits; we report average results on three editing properties using accuracy (

From Figure 4, we observe that utilizing a larger-scale backbone for SME further boosts editing performance. This likely stems from the increased capacity for internal knowledge storage in larger models, allowing more effective collaborative modeling with both internal and external knowledge. With a larger backbone, our framework demonstrates significant performance improvements over the baselines.

5 Conclusion

In this paper, we propose QueueEDIT, a novel queue-based self-correction framework for SME on LLMs. Our approach introduces a structural mapping editing loss to effectively associate knowledge triplets with knowledge-sensitive neurons within MLP layers. By storing and dynamically aligning editing parameters in a queue, our method captures long-sequence dependencies and selectively updates only relevant parameters, thereby mitigating the negative impact of parameter bias on LLM general capabilities. Experiments show that QueueEDIT significantly outperforms baselines across diverse SME scenarios and effectively preserves general language understanding abilities.

Limitations

Despite the promising results achieved by **QueueEDIT**, our work has several limitations that merit further investigation. Due to constraints in computational resources, our experiments were conducted primarily on medium-scale models (e.g., GPT-J-6B and LLaMA3-8B). We anticipate that scaling up to larger models (e.g., 14B or beyond) could further enhance performance, particularly in more complex and knowledge-intensive sequential editing scenarios. Additionally, the queue length and the Top-K selection for parameter alignment were set to fixed values (e.g., $K = 50$) due to memory limitations, which may not fully capture long-range dependencies in extremely long editing sequences. Future work will explore adaptive queue sizing and dynamic K selection strategies, as well as experiments on larger-scale models, as computational resources allow.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62506110). It was also supported by the Natural Science Foundation of Anhui Province, China (Grant No. 2508085QF227) and the Hefei University of Technology Scientific Research Innovation Start-up Special Project Type A (Grant No. JZ2025HGQA0137).

References

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2023. Learning from mistakes makes LLM better reasoner. *CoRR*, abs/2310.20689.

Christine Basta, Marta R. Costa-jussà, and Noe Casas. 2021. Extensive study on the underlying gender bias in contextualized word embeddings. *Neural Comput. Appl.*, pages 3371–3384.

Sofia Blinova, Xinyu Zhou, Martin Jaggi, Carsten Eickhoff, and Seyed Ali Bahrainian. 2023. SIMSUM: document-level text simplification via simultaneous summarization. In *ACL*, pages 9927–9944.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *EMNLP*, pages 6491–6506.

Qizhou Chen, Taolin Zhang, Xiaofeng He, Dongyang Li, Chengyu Wang, Longtao Huang, and Hui Xue'. 2024. Lifelong knowledge editing for llms with retrieval-augmented continuous prompt learning. In *EMNLP*, pages 13565–13580.

Ge Zhang Chenmien Tan and Jie Fu. 2023. Massive editing for large language models via meta learning. *CoRR*, abs/2311.04661.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2023. Evaluating the ripple effects of knowledge editing in language models. *CoRR*, abs/2307.12976.

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

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

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 *ICLR*.

Akshat Gupta and Gopala Anumanchipalli. 2024. Rebuilding ROME : Resolving model collapse during sequential model editing. *CoRR*, abs/2403.07175.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. 2024. Model editing at scale leads to gradual and catastrophic forgetting. In *ACL*, pages 15202–15232.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2022. Aging with GRACE: lifelong model editing with discrete key-value adaptors. *CoRR*, abs/2211.11031.

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

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. *CoRR*, abs/2402.10987.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *ICLR*.

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

Jonas Hübotter, Sascha Bongni, Ido Hakimi, and Andreas Krause. 2025. Efficiently learning at test-time: Active fine-tuning of llms. In *ICLR*.

Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, Qun Liu, and Wei Wang. 2024. Learning to edit: Aligning llms with knowledge editing. *CoRR*, abs/2402.11905.

Kyeonghyun Kim, Jinhee Jang, Juhwan Choi, Yoonji Lee, Kyohoon Jin, and YoungBin Kim. 2025. Plug-in and fine-tuning: Bridging the gap between small language models and large language models.

Sai Krishna, Balvinder Singh, Sujoy Roychowdhury, Giriprasad Sridhara, Sourav Mazumdar, Magnus Sandelin, Dimitris Rentas, Maciej Nalepa, Karol Sawicki, and Jakub Gajda. 2025. Test code generation at ericsson using program analysis augmented fine tuned llms.

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *EMNLP*, pages 6342–6353.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. 2024. Retrievalattention: Accelerating long-context LLM inference via vector retrieval. *CoRR*, abs/2409.10516.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *EMNLP*, pages 2833–2861.

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

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

Abhika Mishra, Akari Asai, Vidhisha Balachandran, Yizhong Wang, Graham Neubig, Yulia Tsvetkov, and Hannaneh Hajishirzi. 2024. Fine-grained hallucination detection and editing for language models. *CoRR*, abs/2401.06855.

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *ICML*, pages 15817–15831.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *ACL*, pages 4885–4901.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, pages 3980–3990.

Amrita Saha, Vardaan Pahuja, Mitesh M. Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *AAAI*, pages 705–713.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *ICML*, pages 31210–31227.

Derek Tam, Anisha Mascarenhas, Shiyue Zhang, Sarah Kwan, Mohit Bansal, and Colin Raffel. 2023. Evaluating the factual consistency of large language models through news summarization. In *ACL*, pages 5220–5255.

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, and Huajun Chen. 2023. Easyedit: An easy-to-use knowledge editing framework for large language models. *CoRR*, abs/2308.07269.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119.

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. *CoRR*, abs/2305.13172.

Haoran You, Yipin Guo, Yichao Fu, Wei Zhou, Huihong Shi, Xiaofan Zhang, Souvik Kundu, Amir Yazdanbakhsh, Yingyan, and Lin. 2024. Shiftaddllm: Accelerating pretrained llms via post-training

multiplication-less reparameterization. *Preprint*, arXiv:2406.05981.

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

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

Heqi Zheng, Xiao Zhang, Zewen Chi, Heyan Huang, Yan Tan, Tian Lan, Wei Wei, and Xian-Ling Mao. 2022. Cross-lingual phrase retrieval. In *ACL*, pages 4193–4204.

A Implementation Details

A.1 Data

Training Data. Following (Yao et al., 2023), we use the ZSRE training data (162,555 entries), CF training data (10,000 entries), and the enhanced DAFSet dataset (Zhang et al., 2024) to train meta-learning-based models, including KE (Cao et al., 2021) and MEND (Mitchell et al., 2022a).

Evaluation Data. We utilize three widely used datasets for evaluation. **ZSRE** (Levy et al., 2017) employs BART (Lewis et al., 2020) to answer questions, followed by manual filtering; each instance contains an editing sample, a rephrased counterpart, and an irrelevant sample corresponding to reliability, generality, and locality, respectively. Based on (Yao et al., 2023), we split the dataset into training (162,555 entries) and testing (19,009 entries). **CF** (Meng et al., 2022) contains only false facts, increasing the difficulty of editing; each data point includes an editing sample, a rephrased sample, and an irrelevant sample. Following (Yao et al., 2023), both the training and test sets comprise 10,000 entries. **RIPE** (Cohen et al., 2023) subdivides generality and locality into multiple parts and involves editing false facts. After pre-processing, we obtain 4,388 entries.

General Capability Datasets. We evaluate generalization of various methods on four authoritative datasets: CSQA (Saha et al., 2018), MMLU (Hendrycks et al., 2021), ANLI (Nie et al., 2020), and SQUAD-2 (Rajpurkar et al., 2016). CSQA consists of 200K dialogs with 1.6M turns. MMLU contains 15,908 questions. ANLI comprises 3,200 test samples, and SQUAD-2 consists of 8,862 questions.

A.2 Baselines

Modifying Parameters: This paradigm includes Locate-then-Edit and Meta-learning-based methods. (1) For Locate-then-Edit, strong baselines include: **KN** (Dai et al., 2022): Uses integral gradient-based neuron location in FFN, editing by amplifying activation. **ROME** (Meng et al., 2022): Uses causal mediation analysis to locate high-impact layers; modifies FFN weights at the located layer. **MEMIT** (Meng et al., 2023): Expands editing across multiple layers based on ROME; improves batch editing performance. **AlphaEdit** (Fang et al., 2025): Maps perturbations into the null space of retained knowledge before parameter integration.

(2) Meta-learning approaches: **KE** (Cao et al., 2021): Trains a bidirectional LSTM to predict weight updates of editing samples. **MEND** (Mitchell et al., 2022a): Trains an MLP to transform low-rank gradient decompositions for edits and updates the model accordingly. **MALMEN** (Chenmien Tan and Fu, 2023): Enables editing multiple facts with limited memory budgets via separate computation on hyper-network and LM. **DAFNet** (Zhang et al., 2024): Proposes intra/inter-editing attention to enhance sequential editing; DAFSet is also used for training. Baseline implementations follow EasyEdit settings for training and evaluation (Wang et al., 2023).

Adding Extra Parameters: T-Patcher (Huang et al., 2023) trains additional neurons in the last FFN layer. GRACE (Hartvigsen et al., 2022) introduces General Retrieval Adapters, maintaining a dictionary structure to create new mappings for modifiable representations.

Retrieving Data: MELO (Yu et al., 2024) implements batch editing via LoRA (Hu et al., 2022). LTE (Jiang et al., 2024) fine-tunes LLMs to generate contextually appropriate responses with knowledge cues, leveraging pre-trained backbones and relevant content retrieval (Reimers and Gurevych, 2019).

A.3 Experimental Settings

We use GPT-J[3] and LLaMA3 (Dubey et al., 2024) as editing backbones. The queue similarity threshold η_{que} is set to 0.5, and η_{deq} to 0.3. The hyperparameter K is set to 50 based on available machine resources. Editing weight selection follows the settings in MEND (Mitchell et al., 2022a) and

[3] https://huggingface.co/EleutherAI/gpt-j-6b

Backbone	# Editing	Editor	ZSRE				CounterFact				RIPE			
			Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
LLAMA3 (8B)	1	FT	53.3	53.7	92.6	$66.5_{(\pm 0.3)}$	34.6	25.9	50.3	$36.9_{(\pm 0.5)}$	49.7	34.2	71.5	$51.8_{(\pm 0.5)}$
		KN	20.7	21.3	52.9	$31.6_{(\pm 0.1)}$	12.8	9.7	68.4	$30.3_{(\pm 0.4)}$	22.3	15.9	55.9	$31.4_{(\pm 0.5)}$
		ROME	54.0	52.1	94.5	$66.9_{(\pm 0.7)}$	41.6	22.3	92.3	$52.1_{(\pm 0.8)}$	48.8	27.6	43.0	$39.8_{(\pm 0.9)}$
		MEMIT	50.2	49.9	92.4	$64.2_{(\pm 0.5)}$	45.9	29.8	93.4	$56.4_{(\pm 0.4)}$	58.9	30.1	39.2	$42.7_{(\pm 0.3)}$
		KE	13.4	9.1	91.1	$37.9_{(\pm 0.2)}$	8.5	3.4	90.8	$34.2_{(\pm 0.4)}$	10.4	4.9	43.3	$19.5_{(\pm 0.2)}$
		MEND	74.3	70.8	66.6	$70.6_{(\pm 0.5)}$	81.6	67.7	77.6	$75.6_{(\pm 0.3)}$	66.9	29.9	30.2	$42.3_{(\pm 0.5)}$
		MALMEN	66.9	68.3	44.2	$59.8_{(\pm 0.5)}$	52.9	42.8	37.1	$44.3_{(\pm 0.6)}$	52.0	34.3	21.0	$35.8_{(\pm 0.9)}$
		DAFNet	97.9	97.8	95.4	$97.0_{(\pm 1.5)}$	92.5	87.2	94.8	$91.5_{(\pm 0.8)}$	98.2	66.9	72.7	$79.3_{(\pm 0.7)}$
		AlphaEdit	98.6	98.6	95.8	$97.7_{(\pm 1.2)}$	93.3	**88.6**	95.7	$92.5_{(\pm 0.9)}$	97.9	67.6	73.5	$79.7_{(\pm 0.5)}$
		TP	86.9	84.5	86.9	$86.1_{(\pm 1.4)}$	91.9	69.1	39.5	$66.8_{(\pm 0.3)}$	77.5	55.6	51.8	$61.6_{(\pm 0.5)}$
		GRACE	52.8	51.3	96.2	$66.8_{(\pm 0.9)}$	45.1	29.0	93.9	$56.0_{(\pm 0.5)}$	57.2	31.1	41.6	$43.3_{(\pm 0.2)}$
		MELO	97.3	96.5	92.2	$95.3_{(\pm 1.1)}$	91.4	84.9	92.6	$89.6_{(\pm 1.0)}$	93.3	64.0	68.8	$75.4_{(\pm 0.5)}$
		LTE	97.7	96.9	93.5	$96.0_{(\pm 1.1)}$	92.2	85.8	93.4	$90.5_{(\pm 0.7)}$	94.6	65.4	70.1	$76.7_{(\pm 1.3)}$
		QueueEDIT	**98.6**	**99.6**	**97.0**	$\mathbf{98.4}_{(\pm 0.5)}$	**93.9**	88.4	**96.1**	$\mathbf{92.8}_{(\pm 0.6)}$	**98.2**	**68.8**	**76.2**	$\mathbf{81.1}_{(\pm 0.9)}$

Table 4: The overall results in single-turn editing.

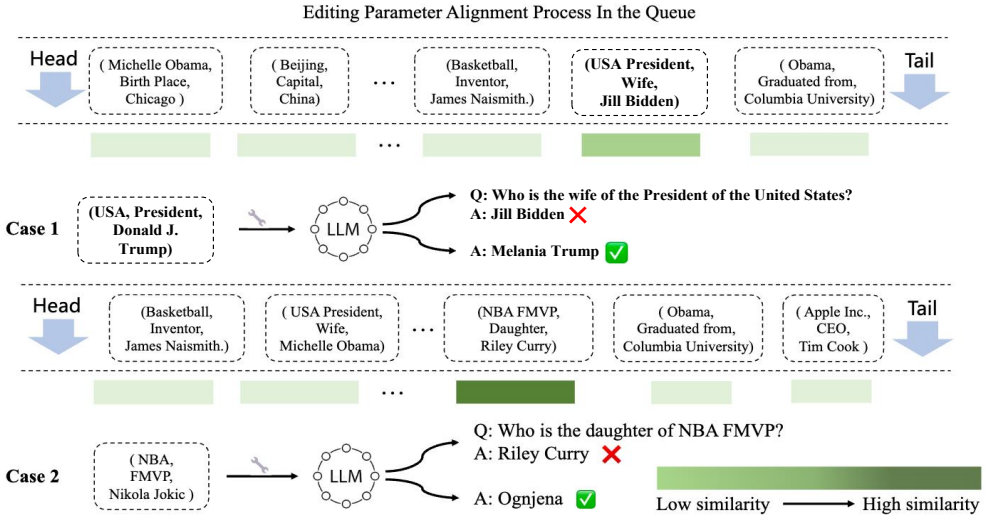

Figure 5: Qualitative analysis of queue-based self-correction in two case studies.

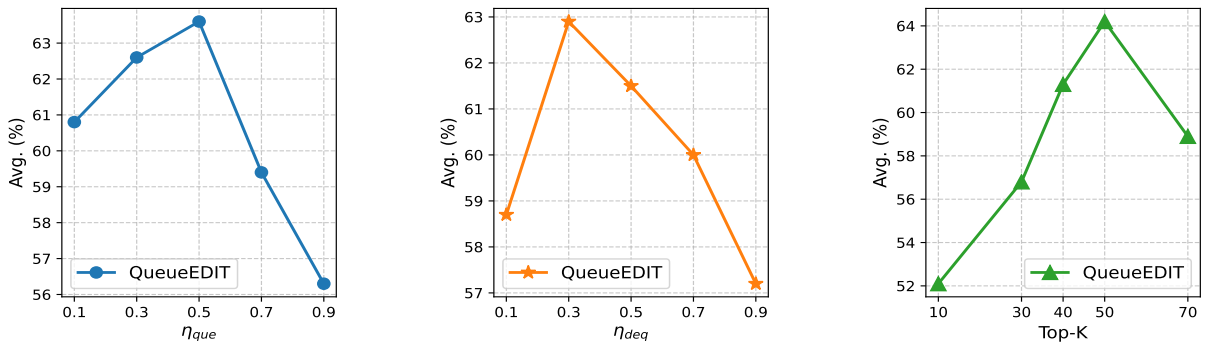

Figure 6: Hyperparameter analysis of the QueueEDIT model.

KE (Cao et al., 2021): both GPT-J and LLaMA3 use the FFN weights of the last three layers.

The editing sequence upper limit is 1,000: $T_{\max} = 1000$. When this maximum is reached, we perform an additional 20,000 iterations before stopping. Checkpoints are saved every 1,000 iterations, and the one with lowest loss is selected for evaluation. The learning rate η is set to $1e{-}6$. Training takes 2 days on 8 NVIDIA A800 GPUs. Experiments are averaged over 5 random runs with different seeds and identical hyperparameters.

B Editing Results

B.1 Results for Other Edits

Experimental results for 1, 10, and 100 edits are shown in Tables 4 and 5. These findings are consistent with our main experiments, further demonstrating the effectiveness of our approach.

Backbone	# Editing	Editor	ZSRE				CounterFact				RIPE			
			Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
GPT-J (6B)	10	FT	10.3	10.8	0.3	$7.1_{(\pm 0.1)}$	56.2	24.2	2.1	$27.5_{(\pm 0.5)}$	7.8	4.3	1.4	$4.5_{(\pm 0.1)}$
		KN	1.0	1.1	1.9	$1.3_{(\pm 0.0)}$	1.2	0.7	2.3	$1.4_{(\pm 0.0)}$	0.1	0.3	0.2	$0.2_{(\pm 0.0)}$
		ROME	81.1	78.8	94.6	$84.8_{(\pm 1.7)}$	95.9	59.4	90.0	$81.8_{(\pm 1.9)}$	98.2	41.9	39.1	$59.7_{(\pm 0.8)}$
		MEMIT	82.1	76.0	94.7	$84.2_{(\pm 2.0)}$	96.0	38.1	**95.5**	$76.5_{(\pm 2.5)}$	98.5	37.7	47.3	$61.2_{(\pm 1.2)}$
		KE	0.0	0.0	0.7	$0.3_{(\pm 0.0)}$	0.0	0.0	0.2	$0.1_{(\pm 0.0)}$	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$
		MEND	0.4	0.4	0.5	$0.4_{(\pm 0.0)}$	0.6	0.2	0.2	$0.3_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		MALMEN	99.1	95.3	92.8	$95.8_{(\pm 1.6)}$	90.0	32.9	77.1	$66.7_{(\pm 2.2)}$	89.7	52.1	51.3	$64.4_{(\pm 1.8)}$
		DAFNET	99.6	97.6	94.8	$97.3_{(\pm 1.5)}$	96.2	65.8	85.2	$82.4_{(\pm 1.6)}$	98.7	57.6	57.8	$71.4_{(\pm 1.6)}$
		AlphaEdit	99.9	98.1	95.2	$97.7_{(\pm 1.1)}$	96.8	66.5	86.8	$83.4_{(\pm 1.0)}$	98.9	58.2	59.0	$72.0_{(\pm 1.4)}$
		TP	85.2	78.3	77.2	$80.2_{(\pm 1.2)}$	96.0	54.3	3.6	$51.3_{(\pm 1.2)}$	80.8	56.7	32.4	$56.6_{(\pm 1.7)}$
		GRACE	81.8	78.4	94.5	$84.9_{(\pm 1.6)}$	95.2	60.3	91.2	$82.2_{(\pm 1.6)}$	98.0	40.9	38.7	$59.2_{(\pm 0.4)}$
		MELO	98.2	96.5	92.8	$95.8_{(\pm 0.9)}$	95.2	63.6	84.5	$81.1_{(\pm 2.1)}$	96.6	51.9	55.7	$68.1_{(\pm 0.8)}$
		LTE	98.7	96.1	93.2	$96.0_{(\pm 1.3)}$	94.3	60.6	82.3	$79.1_{(\pm 1.0)}$	94.5	48.4	53.6	$65.5_{(\pm 0.9)}$
		QueueEDIT	**99.9**	**98.7**	**95.9**	**98.2**$_{(\pm 0.6)}$	**97.5**	**68.2**	89.4	**85.0**$_{(\pm 0.8)}$	**99.8**	**61.3**	**61.4**	**74.2**$_{(\pm 1.1)}$
	100	FT	2.2	1.9	0.3	$1.4_{(\pm 0.0)}$	35.9	10.8	1.6	$16.1_{(\pm 0.3)}$	5.7	1.6	0.1	$2.5_{(\pm 0.1)}$
		KN	0.6	0.4	0.8	$0.6_{(\pm 0.0)}$	0.2	0.5	0.8	$0.5_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		ROME	77.4	75.6	85.0	$79.3_{(\pm 2.2)}$	78.8	38.4	52.2	$56.5_{(\pm 1.0)}$	**95.7**	36.0	32.2	$54.6_{(\pm 1.0)}$
		MEMIT	77.9	74.1	90.2	$80.7_{(\pm 2.5)}$	**94.1**	40.2	85.1	$73.1_{(\pm 1.3)}$	86.6	33.3	33.5	$51.1_{(\pm 1.3)}$
		KE	0.0	0.0	0.7	$0.2_{(\pm 0.0)}$	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		MEND	0.2	0.1	0.0	$0.1_{(\pm 0.0)}$	0.2	0.2	0.0	$0.1_{(\pm 0.0)}$	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$
		MALMEN	50.6	40.7	59.3	$50.2_{(\pm 0.8)}$	29.7	31.8	68.0	$43.2_{(\pm 0.4)}$	39.9	27.8	53.2	$40.3_{(\pm 0.8)}$
		DAFNET	89.5	76.5	90.2	$85.4_{(\pm 1.6)}$	81.8	40.3	87.3	$69.8_{(\pm 1.5)}$	78.5	38.9	64.4	$60.6_{(\pm 1.5)}$
		AlphaEdit	80.2	67.1	82.5	$76.6_{(\pm 0.8)}$	70.7	35.9	74.8	$60.5_{(\pm 1.2)}$	71.9	36.3	59.2	$55.8_{(\pm 1.3)}$
		TP	68.5	59.3	52.8	$60.2_{(\pm 1.3)}$	76.0	31.9	2.2	$36.7_{(\pm 0.8)}$	64.2	36.4	23.7	$41.4_{(\pm 1.0)}$
		GRACE	77.8	74.6	85.9	$79.4_{(\pm 2.0)}$	76.3	39.2	51.6	$55.7_{(\pm 0.8)}$	94.8	36.7	31.5	$54.3_{(\pm 0.8)}$
		MELO	78.4	65.3	82.9	$75.5_{(\pm 0.8)}$	79.4	41.7	79.9	$67.0_{(\pm 1.2)}$	74.6	30.1	52.3	$52.3_{(\pm 0.7)}$
		LTE	81.5	70.5	83.4	$78.5_{(\pm 1.7)}$	78.3	38.7	80.6	$65.9_{(\pm 1.0)}$	75.1	35.9	62.7	$57.9_{(\pm 0.8)}$
		QueueEDIT	**93.5**	**78.6**	**94.2**	**88.8**$_{(\pm 1.3)}$	**92.6**	**52.3**	**90.1**	**78.3**$_{(\pm 0.9)}$	94.9	**43.8**	**68.7**	**69.1**$_{(\pm 0.9)}$
LLAMA3 (8B)	10	FT	38.7	38.1	58.5	$45.1_{(\pm 0.9)}$	19.7	14.2	23.0	$19.0_{(\pm 0.2)}$	31.0	22.5	28.7	$27.4_{(\pm 0.7)}$
		KN	0.9	0.9	1.4	$1.1_{(\pm 0.1)}$	1.2	1.1	4.9	$2.4_{(\pm 0.1)}$	0.9	0.9	0.8	$0.9_{(\pm 0.1)}$
		ROME	41.6	40.3	93.6	$58.5_{(\pm 1.4)}$	39.1	25.6	84.3	$49.7_{(\pm 1.0)}$	33.9	21.0	30.2	$28.4_{(\pm 0.6)}$
		MEMIT	24.8	24.7	51.8	$33.8_{(\pm 0.9)}$	19.1	15.9	63.5	$32.8_{(\pm 0.7)}$	18.9	14.2	10.8	$14.6_{(\pm 0.3)}$
		KE	1.2	1.2	1.9	$1.4_{(\pm 0.1)}$	0.7	0.7	0.8	$0.7_{(\pm 0.1)}$	0.8	0.9	1.7	$1.1_{(\pm 0.1)}$
		MEND	1.0	1.0	3.8	$1.9_{(\pm 0.1)}$	0.7	0.7	0.8	$0.7_{(\pm 0.1)}$	1.0	1.0	2.1	$1.4_{(\pm 0.1)}$
		MALMEN	96.7	89.0	93.3	$93.0_{(\pm 1.8)}$	80.0	46.5	36.9	$54.5_{(\pm 1.1)}$	85.2	48.2	71.6	$68.3_{(\pm 2.3)}$
		DAFNET	97.7	92.7	94.0	$94.8_{(\pm 1.2)}$	87.8	60.3	86.5	$78.2_{(\pm 1.4)}$	89.3	57.1	83.9	$76.8_{(\pm 1.9)}$
		AlphaEdit	98.2	92.3	94.5	$95.0_{(\pm 1.1)}$	88.1	61.4	86.7	$78.7_{(\pm 0.8)}$	90.1	58.6	84.5	$77.7_{(\pm 1.2)}$
		TP	57.8	53.1	37.4	$49.4_{(\pm 1.1)}$	86.4	59.3	22.2	$56.0_{(\pm 0.9)}$	63.9	42.0	31.1	$45.7_{(\pm 0.8)}$
		GRACE	43.0	40.4	93.1	$58.8_{(\pm 1.5)}$	38.6	25.2	83.3	$49.0_{(\pm 0.8)}$	31.9	21.5	29.8	$27.7_{(\pm 0.5)}$
		MELO	97.5	93.3	94.2	$95.0_{(\pm 1.1)}$	86.2	57.8	83.5	$75.8_{(\pm 1.7)}$	88.0	54.4	79.9	$74.1_{(\pm 1.2)}$
		LTE	96.9	93.5	93.2	$94.5_{(\pm 1.6)}$	88.5	58.5	87.1	$78.0_{(\pm 1.6)}$	87.2	55.7	80.2	$74.4_{(\pm 1.1)}$
		QueueEDIT	**99.2**	**94.6**	**96.1**	**96.6**$_{(\pm 0.7)}$	**90.0**	**65.8**	**91.0**	**82.3**$_{(\pm 0.7)}$	**92.3**	**63.5**	**89.7**	**81.8**$_{(\pm 0.9)}$
	100	FT	8.0	7.5	4.7	$6.7_{(\pm 0.2)}$	1.4	0.7	4.1	$2.1_{(\pm 0.1)}$	2.2	1.3	1.5	$1.7_{(\pm 0.1)}$
		KN	0.5	0.7	0.9	$0.7_{(\pm 0.1)}$	0.7	0.7	0.4	$0.6_{(\pm 0.1)}$	0.3	0.8	0.4	$0.5_{(\pm 0.1)}$
		ROME	10.1	11.2	22.7	$14.7_{(\pm 0.4)}$	34.1	22.8	68.7	$41.9_{(\pm 1.2)}$	6.4	4.9	5.7	$5.7_{(\pm 0.1)}$
		MEMIT	1.1	1.1	1.4	$1.2_{(\pm 0.1)}$	1.0	1.0	4.2	$2.1_{(\pm 0.1)}$	0.6	0.9	0.7	$0.7_{(\pm 0.1)}$
		KE	0.7	0.7	0.8	$0.7_{(\pm 0.1)}$	0.7	0.7	1.2	$0.9_{(\pm 0.1)}$	0.8	0.8	0.7	$0.8_{(\pm 0.1)}$
		MEND	0.7	0.7	0.8	$0.7_{(\pm 0.1)}$	0.7	0.7	0.8	$0.7_{(\pm 0.1)}$	0.7	0.7	0.7	$0.7_{(\pm 0.1)}$
		MALMEN	54.8	52.5	66.0	$57.8_{(\pm 0.9)}$	48.6	23.1	47.9	$39.9_{(\pm 0.6)}$	42.1	32.4	39.2	$37.9_{(\pm 0.7)}$
		DAFNET	85.2	72.7	94.3	$84.1_{(\pm 1.4)}$	73.3	42.2	77.1	$64.2_{(\pm 1.1)}$	58.2	41.9	88.2	$62.8_{(\pm 1.7)}$
		AlphaEdit	76.0	60.4	81.7	$72.7_{(\pm 1.0)}$	62.4	35.8	60.6	$52.9_{(\pm 1.4)}$	48.5	36.9	81.6	$55.7_{(\pm 0.9)}$
		TP	46.6	42.0	10.4	$33.0_{(\pm 0.5)}$	70.5	41.5	5.2	$39.1_{(\pm 0.8)}$	45.2	29.6	12.3	$29.0_{(\pm 0.7)}$
		GRACE	9.8	9.2	23.7	$14.2_{(\pm 0.5)}$	32.1	21.8	69.7	$41.2_{(\pm 1.0)}$	6.2	5.5	5.8	$5.8_{(\pm 0.2)}$
		MELO	73.4	59.3	81.2	$71.3_{(\pm 0.8)}$	54.1	32.7	57.6	$48.1_{(\pm 1.2)}$	47.7	36.8	75.9	$53.5_{(\pm 0.7)}$
		LTE	78.7	63.4	83.2	$75.1_{(\pm 1.6)}$	68.0	40.1	65.8	$58.0_{(\pm 1.3)}$	53.9	40.2	82.6	$58.9_{(\pm 1.1)}$
		QueueEDIT	**89.0**	**77.7**	**98.1**	**88.3**$_{(\pm 1.0)}$	**76.8**	**45.7**	**84.9**	**69.1**$_{(\pm 0.7)}$	**65.4**	**46.2**	**90.3**	**67.3**$_{(\pm 0.8)}$

Table 5: Results of QueueEDIT and baselines with 10 and 100 edits.

B.2 Case Study

We further analyze the queue memory by selecting parameter blocks updated from the queue. Specifically, we perform a semantic similarity analysis on the top-3 most relevant triples in sequentially edited knowledge from the ZSRE dataset.

- Case 1: Q: Who was the designer of Lahti Town Hall? A: Eliel Saarinen.
  Top-1: Q: By which person was Lahti Town Hall designed? A: Eliel Saarinen.
  Top-2: Q: When was Lahti Town Hall designed? A: 1911.
  Top-3: Q: Where did the dark bricks come from for the materials used in Lahti Town

Hall? A: Sweden.

- `Case 2`: Q: Who was the manufacturer of USS Leedstown (APA-56)? A: Bethlehem Steel.
  `Top-1`: Q: Which corporation created USS Leedstown (APA-56)? A: Bethlehem Steel.
  `Top-2`: Q: When was USS Leedstown (APA-56) scrapped? A: 1970.
  `Top-3`: Q: How long did USS Leedstown (APA-56) serve? A: 1943-1946.

- `Case 3`: Q: In what language are the Garowe Principles written? A: Somali.
  `Top-1`: Q: In which language is the Garowe Principles monthly football magazine reported? A: Somali.
  `Top-2`: Q: Where were the Garowe Principles signed? A: Garowe.
  `Top-3`: Q: Which representatives voted on the Garowe Principles? A: Transitional Federal Government.

We observe: (1) MLP parameters requiring editing alignment for `Top-1` are typically semantically consistent with the current edited data. (2) Alignment parameters for other cases (`Top-2`, `Top-3`) are generally consistent with the subject entities or relations. In summary, top-K queue data aligned with the current editing knowledge tend to be semantically consistent at the level of important entities or relations.

B.3 Case Studies of Queue-Based Self-Correction

To further evaluate the effectiveness of our `QueueEDIT` method, we visualize samples from the test dataset. As shown in Figure 5, when queried, "Who is the wife of the President of the United States?", the strongest baseline returns "Michelle Obama", an incorrect answer. In contrast, `QueueEDIT` computes semantic similarity for all edited knowledge parameters in the queue and identifies that the fact "(USA President, Wife, Michelle Obama)" should be aligned with the current edit "(USA President, Biden)", due to high similarity.

In Case 2, `QueueEDIT` correctly recognizes "(NBA FMVP, Daughter, Riley Curry)" as the most similar, which requires alignment, since the NBA FMVP recipient is "Nikola Jokic". This demonstrates that queue-based dynamic alignment can better model relationships between sequential edits and promptly update factual correlations.

C Detailed Calculation of v_*

ROME (Meng et al., 2022) chooses a token vector v_* that encodes the new relation (r, o^*) as a property of s. Specifically, $v_* = \arg\min_z \mathcal{L}(z)$, where z is the hidden representation of o^*, and the objective $\mathcal{L}(z)$ is:

$$\mathcal{L}_z = \frac{1}{N}\sum_{j=1}^{N} \underbrace{-\log \mathbb{P}_{G(m_i^{(l^*)}:=z)}[o^* \mid x_j + p]}_{\text{(a) Maximizing } o^* \text{ probability}} + \underbrace{D_{KL}\left(\mathbb{P}_{G(m_{i'}^{(l^*)}:=z)}[x \mid p'] \| \mathbb{P}_G[x \mid p']\right)}_{\text{(b) Controlling semantic drift}} \tag{18}$$

The first term finds a z such that, when it replaces the MLP output at the i-th token (the subject's final position, $G(m_i^{(l^*)} := z)$), the network predicts the desired object o^* given the factual prompt p. The second term lowers the KL divergence between predictions for prompt p' (e.g., "{subject} is a") and those of the original model, ensuring the model retains its grasp of the subject's core meaning.

D Hyperparameter Analysis

We perform quantitative analysis of three key hyperparameters: η_{que}, η_{deq}, and K, to identify optimal configurations for model performance. We report the average performance across three datasets, expressed as Avg. (%), for various parameter settings.

As illustrated in Figure 6: (1) When η_{que} is small, model performance improves steadily as more highly correlated parameters are aligned; the queue remains compact, maximizing relevant information. However, further increasing η_{que} introduces noise from excessive alignment, which eventually decreases overall performance. (2) The trend for η_{deq} mirrors that of η_{que}. A small η_{deq} enforces strict dequeue ordering, enhancing learning. As η_{deq} grows, more head parameters remain at the queue front, increasing misalignment and lowering performance. (3) For hyperparameter K, experiments use values 10, 30, 40, 50, and 70, constrained by GPU memory. Results show optimal performance near $K = 50$. Performance declines sharply above this, likely because semantic dependency length for SME tasks is usually relatively short.