CaPEdit: Capability-Preserving Lifelong Knowledge Editing For Language Models

Song-Li Wu
Tsinghua University
wsl24@mails.tsinghua.edu.cn

Zhaocheng Du*
Huawei Noah's Ark Lab
zhaochengdu@huawei.com

Xianquan Wang
University of Science and Technology of China
wxqcn@mail.ustc.edu.cn

Jingyi Wang
Tsinghua University
jingyi-w24@mails.tsinghua.edu.cn

Abstract

Lifelong knowledge editing (LKE) aims to incrementally correct factual inaccuracies in large language models (LLMs), but sequential edits can lead to substantial degradation of capabilities. Existing approaches primarily rely on static parameter regularization, which restricts knowledge integration and fails to prevent cumulative capability degradation. We argue that an important source of this degradation lies in the temporal mismatch between locally editable factual knowledge and procedural knowledge, which is gradually acquired, guides task execution, and cannot be reliably updated by rapid edits. To this end, we formulate LKE as a dual-timescale process, explicitly decoupling fast-updating factual knowledge from slow-evolving procedural knowledge. Based on this formulation, we propose CaPEdit, a framework that preserves model capabilities under LKE. It first synthesizes procedural knowledge across successive edits, and subsequently performs parameter updates guided jointly by factual supervision and the synthesized procedural signal. To ensure stability under long edit sequences, CaPEdit is trained via a hybrid optimization scheme, combining step-wise updates for rapid factual correction with trajectory-level optimization to facilitate gradual procedural adaptation. Experiments demonstrate that CaPEdit improves capability preservation across all fundamental capabilities by **49.78%**, achieves superior editing performance, and requires only **18.07%** of the editing time of most existing methods.

1 Introduction

Lifelong knowledge editing enables large language models to incrementally incorporate evolving real-world facts through localized parameter updates, providing a computation-efficient alternative to full retraining (Deng et al., 2025; Fang et al., 2025). However, such localized edits lead to severe degradation in downstream performance as edits accumulate over lifelong editing (Gupta et al., 2025). Existing mitigation strategies rely on conservative regularization, which restricts knowledge integration and fails to preserve fundamental capabilities (Li et al., 2025a), including memorization, reasoning, and generalization (Figure 1).

We argue that an important and previously underexplored source is the *temporal mismatch* between rapid, localized post-hoc factual updates and procedural knowledge. Procedural knowledge is the methodical information extracted from pretraining data and governing how a model retrieves, reasons over, and applies facts across contexts to execute tasks (Ruis et al., 2024). While factual inaccuracies can be explicitly modified through localized edits, procedural knowledge is implicitly distributed across interdependent facts and thus evolves on a slower timescale, making it vulnerable to degradation under repeated fast updates. Specifically, we present experimental results in Figure 1 that demonstrate that factual knowledge can be updated rapidly, whereas procedural knowledge adapts more gradually, consistent with Complementary Learning Systems theory in neuroscience (McClelland et al., 1995). By updating both on a single timescale, existing methods implicitly force synchronous adaptation, disrupting reasoning and retrieval pathways and leading to inconsistent knowledge application across contexts. We include illustrative examples in Appendix D.2.

Motivated by this perspective, we propose CaPEdit, a capability-preserving framework built upon an RL-based hypernetwork with dual-timescale rewards. CaPEdit adopts a generate-then-edit paradigm to decouple rapid factual correction from slow procedural adaptation. Specifically, it first synthesizes procedural knowledge that captures how the model utilizes factual knowledge to perform tasks, including memory-consistent re-

*corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 15710–15752

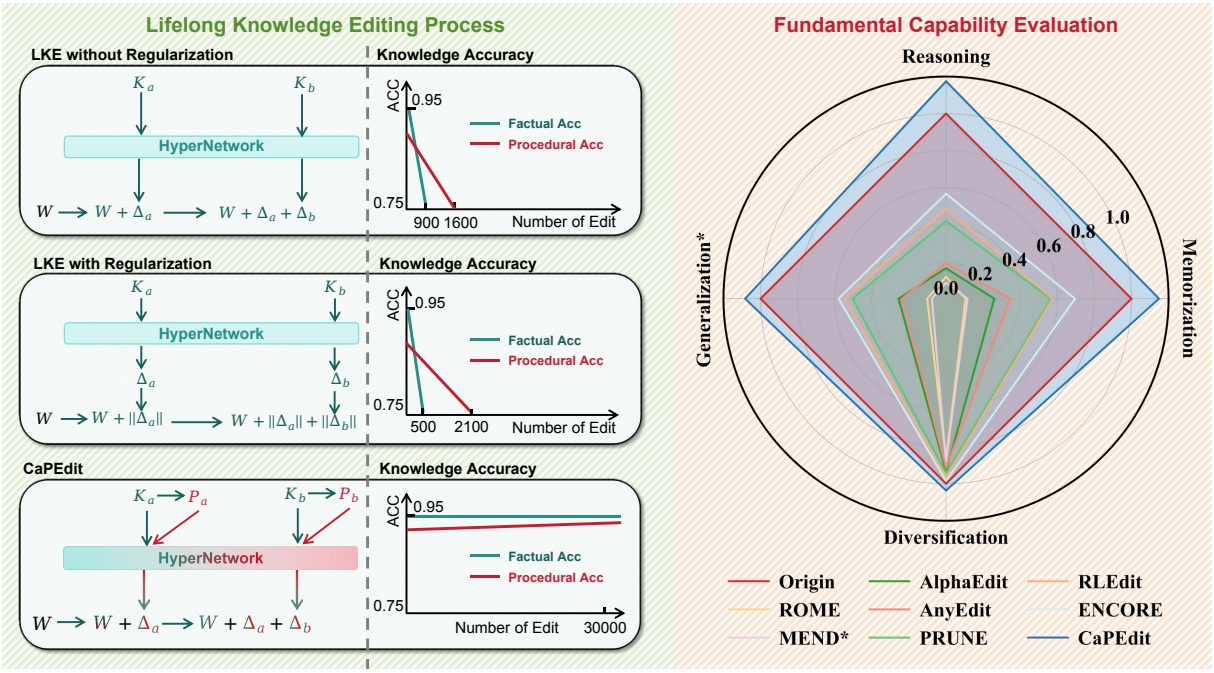

Figure 1: Comparison of different lifelong knowledge editing methods. Existing LKE methods primarily rely on regularizing knowledge updates to preserve model capabilities. K_a and K_b denote different edited knowledge, while P_a and P_b represent procedural knowledge synthesized from K_a and K_b. $||\cdot||$ indicates norm-based regularization, W denotes LLM parameters and Δ denotes parameter updates. Results are obtained with LLaMA-3-8B.

trieval, reasoning trajectories, and contextual adaptation. During editing, CaPEdit leverages a hypernetwork to predict parameter updates, optimized with dual-timescale rewards: a fast factual reward for accurate knowledge correction and a slow procedural reward derived from the synthesized procedural knowledge. This design aligns fast factual updates with gradual procedural adaptation, effectively resolving the temporal mismatch between factual and procedural knowledge. To ensure stability over long edit sequences, CaPEdit is trained with a hybrid optimization strategy, combining step-wise updates for rapid factual correction with trajectory-level optimization for procedural consolidation. As a result, CaPEdit enables effective factual knowledge updates while preserving the model's fundamental capability.

We evaluate CaPEdit on multiple representative LLMs and lifelong editing benchmarks. Results show that CaPEdit substantially improves capability preservation over existing baselines, while maintaining competitive knowledge editing accuracy. Moreover, CaPEdit can be seamlessly integrated into existing hypernetwork-based editing methods as a plug-and-play enhancement, consistently improving their performance. Together, these results establish CaPEdit as a robust framework for lifelong knowledge editing that preserves fundamental model capabilities under repeated updates, enabling stable and scalable model evolution.

2 Related Work

Lifelong Knowledge Editing. Lifelong knowledge editing (LKE) methods for LLMs can be broadly categorized into parameter-preserving and parameter-modifying approaches. Parameter-preserving methods (Zhong et al., 2023; Madaan et al., 2022; Zheng et al., 2023; Dong et al., 2022) maintain the integrity of existing model parameters by augmenting LLMs with auxiliary memory modules. They offer fine-grained control over updates but incur substantial memory overhead and face limited scalability. Parameter-modifying approaches directly alter model parameters and can be further divided into locate-then-edit strategies (Meng et al., 2022a,b; Ma et al., 2024; Fang et al., 2025; Jiang et al., 2025) and hypernetwork-based strategies (Mitchell et al., 2021; Tan et al., 2023; Zhang et al., 2024; Li et al., 2025b).

Capability-Preserving Knowledge Editing. Fundamental capabilities (Li et al., 2025a) are categorized into four categories: memorization, reasoning, generalization and diversification. Recent work has highlighted the challenge of up-

dating model knowledge without degrading capabilities. Prior methods generally rely on regularization to control parameter updates. For instance, RECT (Gu et al., 2024) uses regularization to eliminate small updates in knowledge parameters, PRUNE (Ma et al., 2024) constrains the condition number of updated matrices, and ENCORE (Gupta et al., 2025) introduces a Frobenius-norm penalty (Böttcher and Wenzel, 2008) on edited weight matrices to limit deviations from their original values. Similarly, RLEdit (Li et al., 2025b) imposes an L2-norm regularization (Luo et al., 2016) on parameter updates to restrict the magnitude of each modification. However, by focusing on constrained parameter updates instead of explicitly optimizing capability objectives, these methods hinder knowledge injection and still incur substantial LLM capability degradation.

3 Preliminary

3.1 Lifelong Knowledge Editing

Lifelong Knowledge editing entails continuous and sequential updates to a single LLM, where the total number of edits can reach thousands or even tens of thousands. This requires the post-edited LLM to incorporate newly edited knowledge, retain previously updated knowledge, and maintain overall performance. Formally, let $f_{\mathcal{W}_0} : \mathcal{X} \to \mathcal{Y}$ denote the pre-trained LLM with parameters $\mathcal{W}_0$, mapping an input space $\mathcal{X}$ to an output space $\mathcal{Y}$. In a lifelong editing task, we consider an editing dataset $\mathcal{D}_n = \{(x_t, y_t)\}_{t=1}^n$, where $X = (x_1, \ldots, x_n)$ represents a stream of inputs to be edited, and $Y = (y_1, \ldots, y_n)$ denotes the corresponding target outputs. For the entire input stream X, the pre-trained model initially produces $f_{\mathcal{W}_0}(X) = Y', \quad Y' = (y'_1, \ldots, y'_n)$, with Y' representing the initial predictions. The objective of sequential editing is to update the model parameters such that $f_{\mathcal{W}_n}(X) = Y$. To this end, we introduce a knowledge editor (KE), which sequentially processes input-output pairs (x_t, y_t) from $\mathcal{D}_n$. When editing the t-th knowledge instance, KE updates the LLM parameters according to $f_{\mathcal{W}_t} = \mathrm{KE}(f_{\mathcal{W}_{t-1}}, x_t, y_t), \quad t = 1, \ldots, n.$

3.2 Hypernetwork-based Knowledge Editing

Following RLEdit (Li et al., 2025b), we formulate lifelong knowledge editing as a sequential decision-making process, where an editing hypernetwork $\mathcal{H}$ adaptively generates parameter updates for a pre-trained LLM while preserving previously edited knowledge and maintaining overall performance. The hypernetwork $\mathcal{H}$ leverages fine-tuning gradients of the LLM, using the loss $\mathcal{L}$ on both edited and unrelated knowledge as the training signal to learn a mapping from gradient information to parameter updates. To reduce the computational cost of large $d \times d$ weight matrices, each layer's gradient can be factorized as $\nabla_{w_l}\mathcal{L} = \delta_{l+1} u_l^\top$, where δ_{l+1} is the gradient with respect to the pre-activations of layer $l+1$, and u_l is the layer input. The hypernetwork then learns a low-rank mapping $\mathcal{H} : \delta_{l+1} \times u_l^\top \mapsto \tilde{\delta}_{l+1} \times \tilde{u}_l^\top, \quad \tilde{\nabla}_{w_l} = \tilde{\delta}_{l+1}\tilde{u}_l^\top$, where $\tilde{\delta}_{l+1}$ and $\tilde{u}_l$ are pseudo-activations and pseudo-increments, and $\tilde{\nabla}_{\mathcal{W}}$ denotes the hypernetwork-generated parameter update, equivalent to Δ. The process satisfies the Markov property, as updates depend only on the current state (Li et al., 2025b). At each time step t, the hypernetwork observes the current LLM parameters $\mathcal{W}_{t-1}$ and a knowledge sample (x_t, y_t). It receives the fine-tuning gradient $\nabla\mathcal{W}_{t-1}$ and outputs an action $\tilde{\nabla}\mathcal{W}_t$, which updates the LLM: $\mathcal{W}_t = \mathcal{W}_{t-1} + \tilde{\nabla}\mathcal{W}_t$. The reward for the hypernetwork is computed from the losses on the target knowledge updating (x_t, y_t) and unrelated knowledge preservation (Li et al., 2025b; Zhang et al., 2024):

$$\mathcal{L}_{e_t} = -\mathcal{L}(\mathcal{W}_t; x_t, y_t), \tag{1}$$

$$\mathcal{L}_{\mathrm{loc}_t} = \mathrm{KL}\big[p_{W_0}(\cdot \mid x_{\mathrm{loc}}) \,||\, p_W(\cdot \mid x_{\mathrm{loc}})\big], \tag{2}$$

$$\mathcal{L}_t^{\mathrm{K}} = \mathcal{L}_{e_t} + \boldsymbol{\lambda}_{\mathrm{loc}}\mathcal{L}_{\mathrm{loc}}, \tag{3}$$

$$r_t^{\mathrm{K}} = -\mathcal{L}_t^{\mathrm{K}}, \tag{4}$$

where $\mathcal{L}_{e_t}$ and $\mathcal{L}_{\mathrm{loc}_t}$ measure the effectiveness of target knowledge updating and unrelated knowledge preservation, respectively; x_{loc} represents unrelated knowledge, derived from the previously input knowledge sample (x, y) and we introduce a coefficient $\boldsymbol{\lambda}_{\mathrm{loc}}$ to balance the trade-off between these two terms. The state s_t at time t is defined as $(\mathcal{W}_{t-1}, (x_t, y_t))$ or $\nabla\mathcal{W}_{t-1}$, and the policy π_θ parameterized by θ maps states to actions: $a_t = \tilde{\nabla}\mathcal{W}_t = \pi_\theta(s_t)$. Sequential interactions over n knowledge samples form an MDP trajectory

$$\{(s_1, a_1, r_1), \ldots, (s_n, a_n, r_n)\}, \tag{5}$$

and the hypernetwork $\mathcal{H}$ is trained to maximize the expected cumulative reward. Each action results in a state transition:

$$s_{t+1} \leftarrow s_t + a_t, \quad R(s_t, a_t) = r_t^{\mathrm{K}}. \tag{6}$$

Through this unified formulation, LKE is framed as an RL problem, where the hypernetwork learns to

generate adaptive updates that integrate new knowledge while preserving prior edits.

4 Method

In this section, we introduce CaPEdit, a capability-preserving LKE framework built on an RL-based hypernetwork as described in Section 3. CaPEdit follows a generate-then-edit paradigm, synthesizing procedural signals (Section 4.1) and performing hypernetwork-based editing with dual-timescale rewards (Section 4.2) under a hybrid training scheme (Section 4.3). Figure 2 provides an overview.

4.1 Procedural Knowledge Synthesis

Sequential knowledge edits affect not only individual facts but also retrieval, reasoning, and contextual robustness. To capture these procedural patterns, we introduce memory-consistent retrieval, reasoning trajectories, and contextual adaptation, which together enable the model to internalize knowledge beyond isolated facts. Formal definitions are provided in Appendix B.

Memory-Consistent Retrieval Sequential knowledge edits can introduce conflicts with previously edited facts, leading to contradictory or unstable retrieval behavior. Therefore, we propose memory-consistent retrieval reward to address this problem. Specifically, we maintain a dynamic buffer $\mathcal{B}$ that stores a subset of prior editing instances. Before editing a new instance (x_t, y_t), we retrieve a set of semantically related entries $(x_t^{\mathrm{b}}, y_t^{\mathrm{b}}) \in \mathcal{B}$ using embedding-based similarity. Based on the retrieved entries, we construct a *refined target* y_t^M that incorporates both the newly injected fact and relevant prior knowledge. This refined target is generated using $f_{\mathcal{W}_0}$.

$$y_t^{\mathrm{M}} = f_{\mathcal{W}_0}(x_t, (x_t^{\mathrm{b}}, y_t^{\mathrm{b}})), \tag{7}$$

The resulting pair (x_t, y_t^{M}) provides a supervision signal that captures retrieval consistency across related facts, rather than isolated factual correctness. Details are provided in Appendix B.1. Based on this signal, we define a memory-consistency loss:

$$\mathcal{L}_t^{\mathrm{M}} = -\mathcal{L}(\mathcal{W}_t; x_t, y_t^{\mathrm{M}}). \tag{8}$$

Reasoning Trajectories Sequential knowledge edits can disrupt multi-step reasoning patterns, resulting in degraded inference even when individual edited facts remain correct. To this end, we introduce a reasoning-consistency reward based on masked reasoning trajectories. Specifically, for each newly edited instance (x_t, y_t), we construct a reasoning trajectory τ_t by $f_{\mathcal{W}_0}$ that captures the intermediate inference steps leading to y_t. All factual entities to be inferred within the trajectory are replaced with a special [MASK] token, yielding a masked trajectory $\tilde{\tau}_t$, while the logical dependencies among reasoning steps are retained. This abstraction isolates reusable reasoning dynamics from concrete factual content. Given the masked trajectory, we evaluate whether the edited model can consistently recover the masked tokens conditioned on the preceding reasoning context. We define a reasoning-consistency loss as

$$\mathcal{L}_t^{\mathrm{R}} = -\frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \log P_{\mathcal{W}_t}(\tau_i \mid \tilde{\tau}_{<i}), \tag{9}$$

where $\mathcal{M}$ denotes the set of masked positions, τ_i is the ground-truth token at position i, and $P_{\mathcal{W}_t}$ is the model's predicted probability under current parameters. Details are provided in Appendix B.2.

Contextual Adaptation Sequential knowledge edits can induce overfitting to the immediate contexts in which updates are applied. As a result, while each fact may be correct in isolation, the model often fails to generalize, making its predictions brittle under even minor contextual variations. To this end, we construct *perturbed inputs* x^C by injecting irrelevant context into the original input x_t, while keeping the target fact y_t unchanged. This perturbation strategy exposes the model to diverse contextual realizations of the same fact, explicitly requiring it to separate essential factual information from incidental contextual cues. We then define a differentiable contextual-adaptation loss as:

$$\mathcal{L}_t^{\mathrm{C}} = -\mathcal{L}(\mathcal{W}_t; x^C, y_t), \tag{10}$$

Optimizing this loss promotes reliance on context-invariant facts while preserving shared adaptation patterns, improving robustness to noise. Details are provided in Appendix B.3.

4.2 Dual-Timescale Rewards

To address the temporal mismatch between rapid factual integration and long-term capability preservation, CaPEdit decomposes the reward signal into two complementary timescales. Specifically, fast factual rewards r_t^{K} provide immediate supervision for incorporating newly injected knowledge, enabling rapid and localized updates. In contrast,

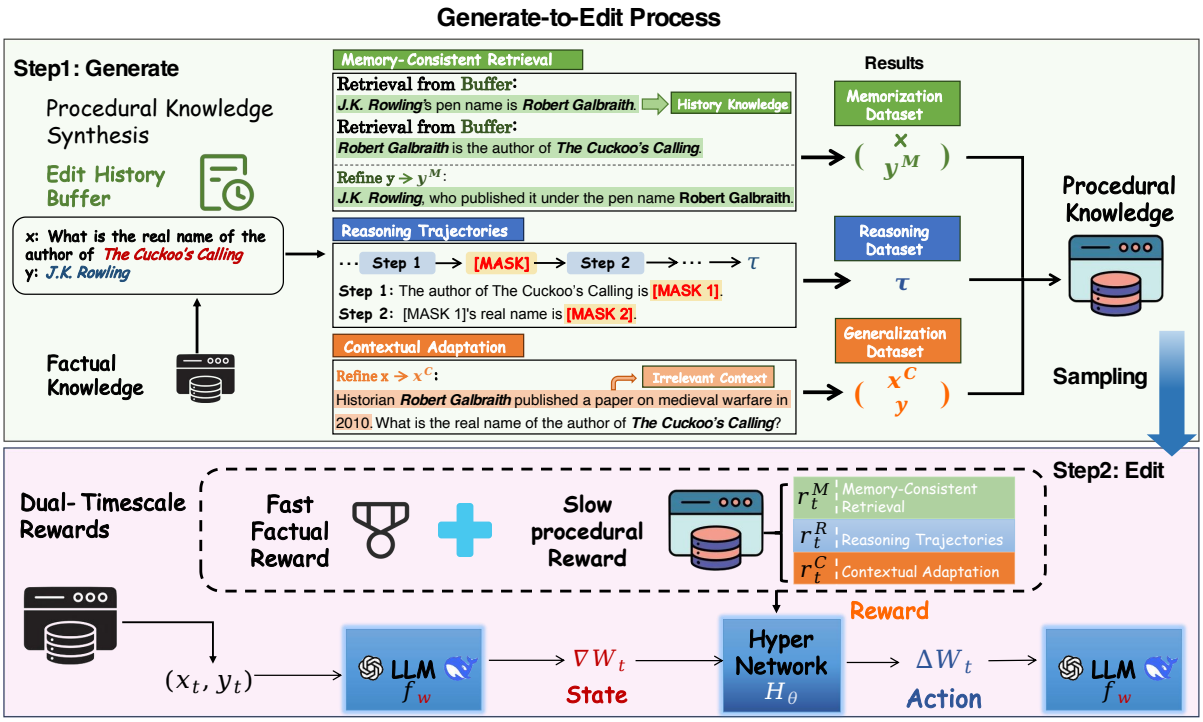

Figure 2: Overview of CaPEdit. CaPEdit follows a generate-to-edit process. It first synthesizes procedural knowledge for all edits and then employs a hypernetwork to perform editing, guided by dual-timescale rewards. τ captures the intermediate reasoning steps, and the [MASK] abstracts factual entities.

slow procedural rewards capture enduring trends in cross-factual capability retention. Therefore, we maintain an exponential moving average (EMA) of procedural rewards:

$$\bar{r}_t^{\mathrm{P}} = \beta \bar{r}_{t-1}^{\mathrm{P}} + (1-\beta) r_t^{\mathrm{P}}, \tag{11}$$

$$r_t^{\mathrm{P}} = -\mathcal{L}_t^{\mathrm{M}} - \mathcal{L}_t^{\mathrm{R}} - \mathcal{L}_t^{\mathrm{C}}. \tag{12}$$

And the reward at time t is defined as:

$$r_t = r_t^{\mathrm{K}} + \bar{r}_t^{\mathrm{P}}. \tag{13}$$

Dual-timescale rewards guide the hypernetwork in updating parameters, ensuring that knowledge edits preserve long-term capabilities.

4.3 Training Process

Given the dual-timescale reward signals defined above, CaPEdit employs a hybrid optimization scheme that aligns parameter updates with the temporal properties of each reward.

Step-Level Fast Updates At each editing step t, the hypernetwork parameters θ are immediately updated using the factual loss:

$$\theta_{t+1} = \theta_t - \eta_f \nabla_\theta \mathcal{L}_t^{\mathrm{K}}. \tag{14}$$

This step-level update enables efficient and low-latency integration of new factual information.

Trajectory-Level Slow Updates After completing a trajectory of n sequential edits, CaPEdit performs a global update guided by the accumulated EMA-smoothed procedural rewards:

$$\theta \leftarrow \theta - \eta_s \nabla_\theta \sum_{t=1}^{n} (\mathcal{L}_t^{\mathrm{M}} + \mathcal{L}_t^{\mathrm{R}} + \mathcal{L}_t^{\mathrm{C}}). \tag{15}$$

This update reinforces procedural patterns that consistently contribute to capability retention across multiple edits. Together, the step-level and trajectory-level updates form a hybrid optimization scheme, ensuring both rapid factual integration and long-term procedural stability.

5 Experiments

5.1 Experimental Settings

Base LLMs. Our experiments are conducted on two representative LLMs: Llama-3-8B[1], Gemma-2-9B (Team et al., 2024).

Datasets & Evaluation Metrics. For LKE, we evaluate our proposed approach on widely adopted benchmark datasets: ZsRE (Levy et al., 2017), FEVER (Thorne et al., 2018), and CounterFact (Meng et al., 2022a). Following established

[1] https://llama.meta.com/llama3

Table 1: Comparison of CaPEdit with baselines on LKE and capability preservation. Upper rows include fine-tuning, locate-then-edit, and hypernetwork-based methods; lower rows show capability-preserving baselines. MEND* and MALMEN* retrain the hypernetwork for each edit. Mem., Rea., Gen., and Div. denote memorization, reasoning, generalization, and diversification, respectively. Capability scores are normalized to the original model.

Methods	LLaMA-3-8B													
	FEVER							ZsRE						
	Knowledge			Capability				Knowledge			Capability				Time↓
	Eff.↑	Gen.↑	Spe.↑	Mem.↑	Rea.↑	Gen.*↑	Div.↑	Eff.↑	Gen.↑	Spe.↑	Mem.↑	Rea.↑	Gen.*↑	Div.↑	
FT	1.80±0.10	6.39±0.17	26.33±0.13	0.00%	0.00%	0.00%	83.95%	17.10±0.22	16.73±0.22	8.27±0.13	0.00%	0.00%	0.00%	65.28%	0.3366s
ROME	38.56±0.28	44.53±0.29	9.29±0.17	10.53%	11.84%	10.17%	86.43%	17.54±0.04	0.57±0.04	0.40±0.02	4.12%	2.33%	1.95%	72.14%	6.0012s
MEMIT	0.18±0.02	0.01±0.01	0.00±0.00	14.21%	6.58%	8.27%	93.00%	0.00±0.00	0.00±0.00	0.13±0.02	5.41%	4.22%	3.17%	78.55%	6.0677s
MEND	0.00±0.00	0.00±0.00	0.00±0.00	0.00%	0.00%	0.00%	85.32%	0.00±0.00	0.00±0.00	0.00±0.00	0.00%	0.00%	0.00%	78.16%	0.9686s
MEND*	10.37±0.19	9.34±0.22	4.88±0.24	11.64%	8.55%	7.63%	87.77%	0.00±0.00	0.00±0.00	0.00±0.00	2.41%	1.85%	1.42%	79.34%	5.9265s
MALMEN	0.71±0.01	0.01±0.01	1.14±0.03	0.00%	0.00%	0.00%	84.17%	9.87±0.12	9.00±0.09	2.11±0.15	3.02%	2.12%	1.74%	77.68%	2.2779s
MALMEN*	5.04±0.20	5.29±0.13	0.26±0.18	16.14%	15.79%	12.82%	89.76%	12.23±0.11	11.08±0.22	2.43±0.09	6.87%	5.23%	4.11%	80.14%	9.4277s
DAFNet	31.27±0.47	28.82±0.43	66.55±0.41	15.67%	13.82%	8.04%	90.25%	21.99±0.47	11.17±0.43	32.21±0.39	12.34%	8.57%	10.91%	82.49%	8.2383s
AlphaEdit	94.22±0.25	94.14±0.18	25.57±0.14	26.03%	16.45%	25.58%	93.43%	86.83±0.23	81.48±0.28	29.09±0.22	1.22%	0.95%	0.64%	70.08%	6.1518s
AnyEdit	93.86±0.38	94.45±0.23	25.32±0.37	34.61%	19.30%	23.75%	91.48%	87.42±0.29	83.71±0.46	24.19±0.46	19.42%	11.39%	12.58%	86.41%	7.4295s
RECT	60.95±0.27	52.40±0.26	1.75±0.07	53.48%	54.55%	54.27%	96.83%	11.05±0.41	8.12±0.15	28.12±0.13	34.74%	37.27%	37.44%	87.96%	6.6583s
PRUNE	56.64±0.24	43.31±0.18	0.85±0.05	56.55%	41.84%	50.34%	94.32%	12.27±0.43	12.01±0.23	9.88±0.29	22.11%	16.42%	18.95%	82.77%	6.1588s
RLEdit	95.34±0.34	93.58±0.38	70.36±0.29	57.44%	48.52%	54.39%	96.28%	89.42±0.34	87.32±0.23	44.78±0.50	23.84%	18.74%	20.14%	83.91%	0.2224s
ENCORE	94.51±0.26	90.75±0.23	64.17±0.31	69.54%	56.71%	57.94%	98.45%	85.36±0.28	84.28±0.15	41.83±0.35	38.01%	31.14%	25.63%	90.32%	0.2751s
CaPEdit	**97.86**±0.12	**98.92**±0.08	**85.16**±0.21	**114.83%**	**117.49%**	**108.36%**	**103.64%**	**95.82**±0.17	**96.54**±0.18	**68.41**±0.22	**109.52%**	**113.16%**	**104.26%**	**102.95%**	**0.1758s**

Methods	Gemma-2-9B													
	FEVER							ZsRE						
	Knowledge			Capability				Knowledge			Capability				Time↓
	Eff.↑	Gen.↑	Spe.↑	Mem.↑	Rea.↑	Gen.*↑	Div.↑	Eff.↑	Gen.↑	Spe.↑	Mem.↑	Rea.↑	Gen.*↑	Div.↑	
FT	34.23±0.31	29.22±0.30	34.23±0.31	0.00%	0.00%	0.00%	84.12%	12.90±0.20	13.09±0.20	0.07±0.02	0.00%	0.00%	0.00%	78.34%	0.3358s
ROME	30.07±0.26	25.13±0.11	10.79±0.24	12.41%	9.87%	8.13%	88.23%	3.45±0.32	3.33±0.11	8.24±0.29	3.12%	2.45%	2.01%	84.65%	6.0243s
MEMIT	11.12±0.26	10.09±0.25	8.04±0.21	13.54%	12.02%	7.88%	90.41%	5.23±0.21	3.11±0.12	5.98±0.12	1.87%	1.43%	1.12%	81.24%	6.3423s
MEND	0.00±0.00	0.00±0.00	0.00±0.00	0.00%	0.00%	0.00%	85.14%	50.93±0.43	50.76±0.27	0.36±0.04	0.00%	0.00%	0.00%	82.49%	0.9175s
MEND*	8.84±0.24	8.45±0.21	10.21±0.19	8.14%	7.49%	4.32%	86.81%	50.87±0.24	52.01±0.11	0.39±0.03	0.00%	0.00%	0.00%	80.07%	6.1280s
MALMEN	46.60±0.33	42.50±0.32	19.66±0.35	12.84%	10.57%	9.72%	84.98%	20.78±0.31	19.99±0.35	53.10±0.56	16.29%	13.13%	6.46%	81.35%	1.9858s
MALMEN*	56.97±0.24	44.28±0.29	15.02±0.21	18.57%	16.46%	4.29%	88.71%	75.84±0.69	75.81±0.62	32.07±0.34	20.53%	14.83%	7.19%	85.39%	9.3358s
DAFNet	5.94±0.24	5.68±0.33	36.29±0.45	10.12%	7.55%	8.31%	92.77%	79.82±0.26	37.15±0.41	84.37±0.22	3.74%	2.92%	18.56%	89.53%	8.2553s
AlphaEdit	94.37±0.26	88.34±0.13	31.21±0.54	42.89%	36.44%	38.18%	91.82%	81.18±0.33	73.24±0.46	30.34±0.19	20.51%	15.74%	17.82%	92.44%	6.2307s
AnyEdit	94.13±0.23	89.68±0.34	30.63±0.72	49.73%	41.87%	35.81%	93.72%	88.32±0.33	84.29±0.33	69.70±0.27	22.04%	17.56%	18.64%	93.05%	7.3561s
RECT	59.81±0.23	54.89±0.15	0.05±0.01	65.45%	54.29%	62.93%	96.52%	12.45±0.45	10.11±0.51	26.09±0.44	29.22%	27.47%	27.33%	96.04%	6.0486s
PRUNE	13.34±0.25	11.17±0.27	9.43±0.17	52.33%	39.27%	49.02%	94.01%	10.21±0.27	8.88±0.29	11.95±0.55	26.48%	21.71%	29.84%	94.73%	6.3356s
RLEdit	95.44±0.21	92.83±0.30	68.76±0.18	53.11%	45.07%	51.24%	95.74%	94.56±0.09	91.94±0.16	68.65±0.34	27.24%	25.33%	35.19%	94.25%	0.2238s
ENCORE	94.92±0.31	83.85±0.29	63.94±0.40	68.34%	54.63%	56.63%	97.04%	92.41±0.17	87.21±0.24	65.48±0.29	26.53%	31.81%	34.58%	95.96%	0.2794s
CaPEdit	**98.63**±0.11	**96.73**±0.21	**84.29**±0.16	**112.53%**	**109.38%**	**107.26%**	**104.27%**	**95.21**±0.16	**96.26**±0.23	**88.42**±0.41	**115.74%**	**115.46%**	**108.31%**	**103.57%**	**0.1874s**

evaluation protocols (Mitchell et al., 2021; Meng et al., 2022a), we employ three metrics for KE: Efficacy, Generalization, and Specificity to comprehensively assess the effectiveness of model editing. For capability evaluation, details are in Appendix A.

Baseline Methods. We compare our method against multiple editing methods, including FT (Zhu et al., 2020), ROME (Meng et al., 2022a), MEND (Mitchell et al., 2021), MALMEN (Tan et al., 2023), DAFNet (Zhang et al., 2024), MEMIT (Meng et al., 2022b), AlphaEdit (Fang et al., 2025) and AnyEdit (Jiang et al., 2025). We also include capability-preserving baselines, including RECT (Gu et al., 2024), PRUNE (Ma et al., 2024), RLEdit (Li et al., 2025b) and ENCORE (Gupta et al., 2025).

5.2 Performance on Knowledge Updates and Capability Preservation

To jointly evaluate lifelong factual editing (LFE) and capability preservation, we compare CaPEdit with representative LKE baselines across multiple LLMs. After all batch edits, models are evaluated using both knowledge and capability metrics (Table 1); metric definitions and detailed results are provided in Appendix A.5 and Appendix C.7. The reported time reflects the average editing cost per instance. Existing LKE methods exhibit substantial degradation in fundamental capabilities: memorization, reasoning, and generalization drop sharply across fine-tuning, locate-then-edit, and hypernetwork-based approaches. Capability-aware baselines still retain less than 70% performance on fundamental capabilities except for diversification, indicating that magnitude constraints alone are insufficient. In contrast, CaPEdit achieves strong editing accuracy with higher efficiency while maintaining over 100% retention in memorization, reasoning, and generalization, indicating that sequential edits can further refine shared procedural knowledge, leading to improved performance beyond the pre-edit baseline. Despite not explicitly optimizing diversification, it remains stable, highlighting the effectiveness of CaPEdit's mechanism in suppressing harmful parameter drift, which prevents over-adaptation to recently edited instances and

Table 2: Cross-Domain Editing Results. Editing performance (accuracy) of the compared methods in cross-domain scenarios, where models are trained on a source dataset and directly applied to a target dataset (source ⇒ target). Suc. denotes editing success, Por. portability, Loc. locality, and Flu. fluency.

Method	ZsRE ⇒ $\text{Wiki}_{\text{recent}}$								$\text{Wiki}_{\text{recent}}$ ⇒ $\text{Wiki}_{\text{counterfact}}$							
	Knowledge				Capability				Knowledge				Capability			
	Suc.↑	Por.↑	Loc.↑	Flu.↑	Mem.↑	Rea.↑	Gen.*↑	Div.↑	Suc.↑	Por.↑	Locality↑	Flu.↑	Mem.↑	Rea.↑	Gen.*↑	Div.↑
FT	24.51	16.82	19.37	352.75	0.00%	0.00%	0.00%	57.83%	15.53	12.60	17.32	375.41	0.00%	0.00%	0.00%	62.36%
ROME	42.37	27.11	33.59	401.22	1.21%	1.03%	0.98%	78.40%	31.95	24.18	34.70	415.34	3.09%	2.02%	1.68%	73.48%
MEMIT	51.82	35.64	41.25	428.19	4.35%	3.88%	11.57%	64.92%	44.78	33.92	47.03	439.10	2.02%	1.73%	3.65%	76.41%
MEND	73.24	50.77	61.21	502.18	0.00%	0.00%	0.00%	67.82%	58.97	42.70	64.87	509.84	5.72%	3.26%	5.31%	66.08%
MALMEN	75.83	52.45	63.78	515.34	8.92%	28.15%	12.45%	73.56%	61.34	45.22	67.45	522.19	8.42%	5.67%	9.34%	78.90%
DAFNet	83.44	58.07	70.23	538.44	17.15%	27.08%	14.95%	81.88%	68.72	52.15	73.60	550.66	15.88%	8.90%	14.22%	76.92%
AlphaEdit	67.10	47.88	57.09	483.77	25.10%	34.49%	21.56%	90.75%	55.09	40.38	60.43	491.22	14.42%	11.58%	13.17%	85.62%
AnyEdit	78.90	55.12	67.04	523.61	26.32%	25.28%	24.10%	91.43%	64.28	48.80	70.31	534.72	19.03%	16.40%	18.16%	86.40%
RECT	60.84	43.22	53.11	459.64	32.90%	35.38%	35.67%	96.78%	50.26	36.71	56.02	470.32	25.67%	25.42%	14.51%	94.56%
PRUNE	56.93	39.85	48.10	441.03	35.41%	37.75%	32.22%	94.50%	48.11	35.09	51.66	456.27	22.90%	21.63%	15.02%	92.35%
RLEdit	88.17	61.53	73.98	552.81	39.42%	41.48%	26.90%	93.82%	72.34	55.88	76.42	562.13	23.52%	22.96%	23.86%	95.08%
ENCORE	85.67	59.34	71.89	545.72	38.02%	42.45%	25.50%	92.40%	73.45	57.12	75.67	555.34	26.59%	28.90%	26.71%	97.27%
CaPEdit	**98.23**	**69.50**	**80.12**	**575.68**	**105.40%**	**107.25%**	**108.86%**	**101.83%**	**83.40**	**64.20**	**84.50**	**580.84**	**107.59%**	**108.36%**	**104.17%**	**100.76%**

Table 3: Ablation Study Results for CaPEdit.

Model	Method \ Dataset	FEVER						
		Eff.↑	Gen.↑	Spe.↑	Mem.↑	Rea.↑	Gen.*↑	Div.↑
Llama-3	**CaPEdit**	**97.86**	**98.92**	**85.16**	**114.83%**	**117.49%**	**108.36%**	**103.64%**
	w/o Procedural Knowledge Synthesis	55.21↓42.65	56.08↓42.84	42.41↓42.75	49.37%↓65.46	50.92%↓66.57	43.14%↓65.22	95.08%↓8.56
	w/o r_t^{M}	97.02↓0.84	98.41↓0.51	84.62↓0.54	56.58%↓58.25	116.03%↓1.46	107.91%↓0.45	102.87%↓0.77
	w/o r_t^{R}	96.89↓0.97	98.27↓0.65	84.39↓0.77	113.92%↓0.91	59.14%↓58.35	107.68%↓0.68	102.44%↓1.20
	w/o r_t^{C}	96.74↓1.12	96.85↓2.07	83.97↓1.19	113.66%↓1.17	116.38%↓1.11	61.72%↓46.64	102.15%↓1.49
	w/ Both Slow Rewards	75.88↓21.98	66.21↓32.71	72.76↓12.40	109.37%↓5.46	110.92%↓6.57	103.14%↓5.22	99.08%↓4.56
	w/ Both Fast Rewards	96.21↓1.65	96.08↓2.84	82.41↓2.75	88.93%↓25.90	81.27%↓36.22	92.34%↓16.02	98.91%↓4.73
	w/ Uniform Reward	96.74↓1.12	98.05↓0.87	84.07↓1.09	107.96%↓6.87	104.18%↓13.31	106.29%↓2.07	102.17%↓1.47
Gemma-2	**CaPEdit**	**98.63**	**96.73**	**84.29**	**112.53%**	**109.38%**	**107.26%**	**104.27%**
	w/o Procedural Knowledge Synthesis	51.27↓47.36	53.82↓42.91	41.37↓42.92	54.25%↓58.28	46.81%↓62.57	52.34%↓54.92	94.60%↓9.67
	w/o r_t^{M}	97.94↓0.69	96.21↓0.52	83.87↓0.42	64.79%↓47.74	108.74%↓0.64	106.91%↓0.35	103.58%↓0.69
	w/o r_t^{R}	97.71↓0.92	96.08↓0.65	83.61↓0.68	111.86%↓0.67	53.92%↓55.46	106.53%↓0.73	103.27%↓1.00
	w/o r_t^{C}	97.53↓1.10	94.39↓2.34	83.14↓1.15	111.47%↓1.06	108.91%↓0.47	61.12%↓46.14	102.96%↓1.31
	w/ Both Slow Rewards	86.68↓11.95	84.87↓11.86	62.02↓22.27	107.06%↓5.47	104.32%↓5.06	102.01%↓5.25	99.63%↓4.64
	w/ Both Fast Rewards	96.02↓2.61	94.11↓2.62	81.83↓2.46	76.41%↓36.12	84.86%↓24.52	81.93%↓25.33	99.21%↓5.06
	w/ Uniform Reward	95.12↓3.51	92.41↓4.32	80.36↓3.93	104.94%↓7.59	102.26%↓7.12	105.77%↓1.49	98.64%↓5.63

preserves variability across generated responses. Overall, CaPEdit uniquely achieves balanced and robust capability preservation under LKE.

5.3 Performance on Cross-domain LKE

To evaluate robustness and generalization under distribution shifts, we study lifelong knowledge editing in a cross-domain multi-task setting on Llama-3-8B. Unlike sequential edits confined to a single domain, edits are drawn from heterogeneous sources to assess adaptability across diverse knowledge distributions. All methods leverage a training dataset for parameter updates, and cross-domain performance is evaluated by additionally incorporating the Wiki dataset (Cohen et al., 2024) (Table 2). We report four metrics—Edit Success, Portability, Locality, and Fluency—with details provided in Appendix A.5.9. Despite substantial domain shifts, CaPEdit consistently outperforms all baselines across all metrics. This advantage arises because cross-domain editing amplifies conflicts in retrieval and reasoning patterns rather than isolated factual errors, and CaPEdit explicitly stabilizes such shared procedural behaviors through trajectory-level optimization. As a result, it avoids over-adaptation to any single domain and maintains coherent editing performance across heterogeneous knowledge sources.

5.4 Ablation Study

We conduct ablation studies on LLaMA-3 and Gemma-2 over FEVER to evaluate the contribution of each component in CaPEdit (Table 3). We remove procedural knowledge synthesis, ablate individual capability rewards, and replace the proposed dual-timescale adaptive weighting with single-timescale reward schemes, and the EMA-smoothed procedural reward is substituted by a uniformly weighted instantaneous reward. Removing procedural knowledge synthesis leads to a severe performance collapse across all metrics, with editing accuracy and capability retention dropping by over 40% and 50% on average, respectively, underscoring its critical role in stable lifelong editing.

Ablating individual capability rewards causes targeted degradation in the corresponding capabilities, confirming that each reward provides a distinct and non-redundant supervisory signal. Replacing dual-timescale rewards with single-timescale schemes consistently degrades performance: fast-only rewards undermine capability preservation, while slow-only rewards severely impair knowledge editing, highlighting the necessity of dual-timescale rewards. Furthermore, replacing the EMA-smoothed procedural reward with an instantaneous uniformly weighted signal degrades performance, as high-variance step-level feedback overwhelms long-term procedural accumulation.

5.5 Capability Preservation under LKE

Figure 3 illustrates how fundamental capabilities evolve as the number of edited samples increases. Here, the y-axis shows the percentage relative to the corresponding capabilities of the original model. We evaluate all fundamental capabilities under progressively more extensive knowledge edits, reflecting the effectiveness of CaPEdit. Our observations are as follows: CaPEdit effectively preserves the capabilities of post-edited LLMs, even after 40,000 edits. Specifically, it maintains performance across all fundamental capabilities, demonstrating that our design not only safeguards previously acquired knowledge but also protects the procedural knowledge learned from the original corpus.

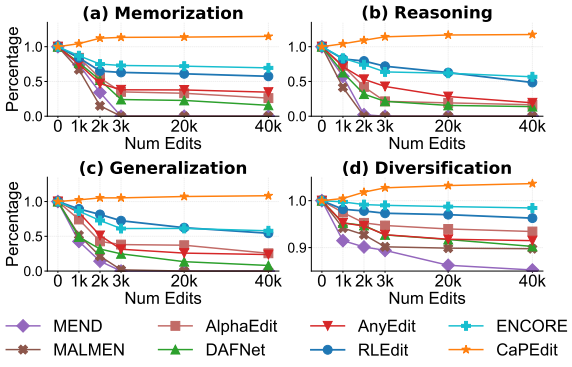

Figure 3: Capability Preservation Under Lifelong Edits.

5.6 Improving Hypernetwork-based Methods

Existing hypernetwork-based editing methods suffer from capability degradation in lifelong knowledge editing (LKE), as they lack explicit mechanisms for preserving pre-edit model capabilities. CaPEdit addresses this limitation via procedural knowledge synthesis coupled with capability-aware dual-timescale optimization, and can be integrated into existing editors as a model-agnostic, plug-and-play training module. We evaluate this portability on LLaMA-3-8B with the FEVER dataset (Figure 4). After incorporating CaPEdit, all hypernetwork-based methods achieve consistent improvements in both editing performance and capability-related metrics, indicating effective mitigation of capability degradation under sequential edits. Overall, CaPEdit enhances existing methods with robust capability preservation while maintaining strong lifelong editing performance.

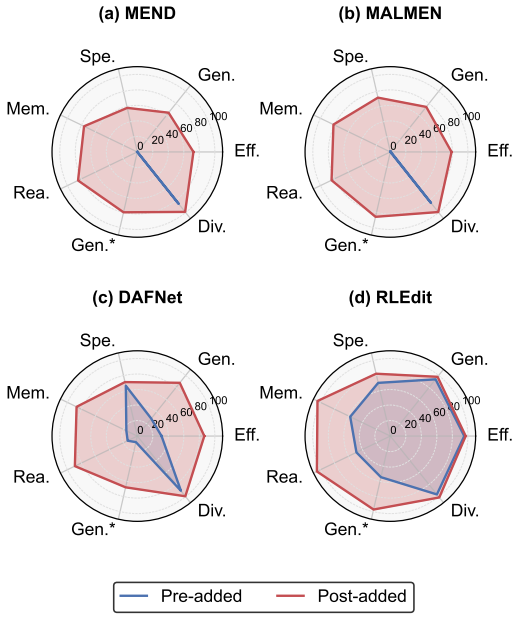

Figure 4: Capability Preservation Under Lifelong Edits.

6 Conclusion

We tackle the challenge of capability degradation in lifelong knowledge editing, where sequential factual updates compromise capabilities. We pinpoint a previously underexplored cause: the temporal mismatch between factual updates and the adaptation of procedural knowledge. To address this, we introduce CaPEdit, a framework that synthesizes procedural knowledge and guides hypernetwork-based edits through dual-timescale rewards, enabling precise knowledge updates while preserving fundamental capabilities. Experiments show that CaPEdit consistently outperforms existing methods in capability retention and editing performance, establishing it as a scalable and principled approach for sustainable model evolution.

Limitations

This work focuses on fundamental LLM capabilities—memorization, reasoning, generalization, and diversification—following established categorizations and evaluation protocols to ensure stable and comparable assessment. While our empirical study is restricted to these dimensions, the proposed framework is not inherently tied to any specific capability set. CaPEdit adopts a capability-agnostic formulation with capability-aware optimization, allowing straightforward extension to additional capabilities without modifying the core methodology. Exploring broader capability dimensions is left for future work.

References

Albrecht Böttcher and David Wenzel. 2008. The frobenius norm and the commutator. *Linear algebra and its applications*, 429(8-9):1864–1885.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298.

Jingcheng Deng, Zihao Wei, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi Cheng. 2025. Everything is editable: Extend knowledge editing to unstructured data in large language models. In *ICLR*.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. *arXiv preprint arXiv:2210.03329*.

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*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

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. *arXiv preprint arXiv:2401.04700*.

Akshat Gupta, Phudish Prateepamornkul, Maochuan Lu, Ahmed Alaa, Thomas Hartvigsen, and Gopala Anumanchipalli. 2025. Lifelong sequential knowledge editing without model degradation. *arXiv e-prints*, pages arXiv–2502.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36:47934–47959.

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Mingyang Wan, Guojun Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Anyedit: Edit any knowledge encoded in language models. In *Forty-second International Conference on Machine Learning*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Jiawei Li, Yang Gao, Yizhe Yang, Yu Bai, Xiaofeng Zhou, Yinghao Li, Huashan Sun, Yuhang Liu, Xingpeng Si, Yuhao Ye, and 1 others. 2025a. Fundamental capabilities and applications of large language models: A survey. *ACM Computing Surveys*.

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

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. 2025b. Reinforced lifelong editing for language models. *arXiv preprint arXiv:2502.05759*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 3214–3252.

Xiong Luo, Xiaohui Chang, and Xiaojuan Ban. 2016. Regression and classification using extreme learning machine based on l1-norm and l2-norm. *Neurocomputing*, 174:179–186.

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

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve gpt-3 after deployment. *arXiv preprint arXiv:2201.06009*.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Laura Ruis, Maximilian Mozes, Juhan Bae, Siddhartha Rao Kamalakara, Dwarak Talupuru, Acyr Locatelli, Robert Kirk, Tim Rocktäschel, Edward Grefenstette, and Max Bartolo. 2024. Procedural knowledge in pretraining drives reasoning in large language models. *arXiv preprint arXiv:2411.12580*.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2024. Lamp: When large language models meet personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7370–7392.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Chenmien Tan, Ge Zhang, and Jie Fu. 2023. Massive editing for large language models via meta learning. *arXiv preprint arXiv:2311.04661*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, and 1 others. 2022. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*, 2:2.

Haoyu Xu, Pengxiang Lan, Enneng Yang, Guibing Guo, Jianzhe Zhao, Linying Jiang, and Xingwei Wang. 2025. Knowledge decoupling via orthogonal projection for lifelong editing of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13194–13213.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19449–19457.

Taolin Zhang, Qizhou Chen, Dongyang Li, Chengyu Wang, Xiaofeng He, Longtao Huang, Jun Huang, and 1 others. 2024. Dafnet: Dynamic auxiliary fusion for sequential model editing in large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 1588–1602.

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? *arXiv preprint arXiv:2305.12740*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

A Detailed Experimental Setup

In this section, we provide a detailed description of our experimental setup, which comprises five components: baseline methods, datasets, evaluation metrics, GLUE benchmarks, and hyperparameter configurations. Unless otherwise specified, all experiments were conducted on a single A100 GPU with 80GB of memory. The editing time for all methods was measured with LLMs operating in half-precision mode. To more accurately reflect real-world deployment scenarios, we utilized the instruction-tuned variants of the LLMs throughout our experiments.

A.1 Baseline Methods

We adopted publicly available implementations from AlphaEdit, MALMEN and AnyEdit to evaluate the performance of the baseline methods. The baseline methods considered in this study are summarized as follows:

- FT-L (Zhu et al., 2020) is a knowledge editing method that fine-tunes specific layers of

an LLM using autoregressive loss. We implemented this method following the hyperparameter settings reported in the original paper.

- MEND (Mitchell et al., 2021) is an efficient hypernetwork-based editing approach. It trains a hypernetwork to learn patterns of parameter updates by mapping low-rank decomposed fine-tuning gradients to LLM parameter adjustments, enabling localized and efficient knowledge modification. We reproduced MEND using the official hyperparameters and trained it on the full dataset. Additionally, we propose an extended variant, MEND*, to mitigate the mismatch between the initial hypernetwork and the post-edited LLM during lifelong editing. Specifically, the hypernetwork is periodically retrained with post-edited parameters every three editing batches.

- ROME (Meng et al., 2022a) targets the direct modification of factual associations in LLM parameters. It identifies critical neuron activations in MLP layers through perturbation-based localization and updates MLP weights by computing Lagrange remainders. As ROME is not optimized for large-scale editing, we followed the original configuration and performed evaluation through multiple rounds of single edits.

- MEMIT (Meng et al., 2022b) extends ROME to support large-batch knowledge updates. It employs a least squares approximation to jointly modify parameters across multiple layers, enabling simultaneous updates of hundreds or thousands of facts. We evaluated MEMIT in lifelong editing following its original experimental setup.

- MALMEN (Tan et al., 2023) is a hypernetwork-based approach for large-scale model editing. To aggregate parameter shifts across extensive knowledge batches, MALMEN employs a least squares optimization that solves normal equations to derive optimal update directions, effectively mitigating knowledge conflicts. We implemented MALMEN using the hyperparameters from the original paper and trained it on the entire dataset. Analogous to MEND, we further introduce MALMEN, an enhanced variant adapted for lifelong editing.

- DAFNet (Zhang et al., 2024) is a sequential editing framework featuring a dynamic auxiliary fusion network that promotes semantic interaction among sequential knowledge triples. This design enables continuous correction of previous edits and enhances the robustness of hypernetwork-based approaches. We followed the original implementation and trained DAFNet on the ZsRE and CounterFact datasets.

- PRUNE (Ma et al., 2024) is a sequential editing method that constrains updated matrices via conditional regularization, thereby reducing interference between newly introduced and previously stored knowledge. This mechanism effectively mitigates performance degradation during continual editing. We reproduced PRUNE using the hyperparameters specified in the original study.

- RECT (Gu et al., 2024) focuses on preserving the general capabilities of LLMs during editing. It regularizes weight updates to prevent overfitting and degradation of general performance, thus maintaining a balance between editing efficacy and model generalization. Our implementation adheres to the official configuration.

- AlphaEdit (Fang et al., 2025) introduces a null-space projection mechanism to alleviate knowledge interference in lifelong editing. By projecting parameter updates onto a knowledge-preserving null space prior to application, AlphaEdit minimizes conflicts between edits and preserves prior knowledge. Demonstrating state-of-the-art performance across multiple benchmarks, AlphaEdit was implemented using the hyperparameter settings provided in the original paper.

- AnyEdit (Jiang et al., 2025) proposes an autoregressive editing paradigm to address the "efficacy barrier" of single-token editing in handling long-form and diverse-formatted knowledge (e.g., poetry, code, mathematical derivations). It decomposes target knowledge into sequential chunks, iteratively locates the key token of each chunk, and perturbs its hidden states to maximize the generation likelihood of subsequent chunks—grounded theoretically in the Chain Rule of Mutual Informa-

tion to ensure consistent and complete knowledge updates.

- RLEdit (Li et al., 2025b) is a reinforcement learning (RL)-based hypernetwork method designed for lifelong language model (LLM) editing, addressing the limitation that traditional hypernetwork-based methods fail to adapt to dynamically changing LLM parameters during sequential edits. It models the hypernetwork, treats editing losses as rewards, and adopts an offline policy update strategy to optimize hypernetwork parameters across the full knowledge sequence, enabling precise capture of LLM parameter changes and generation of appropriate updates.

- ENCORE (Gupta et al., 2025) is proposed to preserve model capabilities during large-scale sequential knowledge editing in locate-then-edit methods. They show that capability degradation mainly arises from overfitting to edited facts and uncontrolled growth of editing matrix norms, which causes edited layers to dominate model predictions. ENCORE mitigates these effects through early stopping and explicit norm regularization, enabling long-horizon sequential edits without sacrificing downstream performance.

A.2 Implementation Details.

For all baselines, we use the official implementations when available; otherwise, we reimplement the methods ourselves and ensure that the reproduced results match the metrics reported in the original papers. We take AlphaEdit, AnyEdit, RLEdit, and ENCORE as examples. Additionally, to make fair comparison, following (Li et al., 2025b), we randomly sampled 8,000 knowledge instances from the ZsRE and FEVER datasets, respectively, and performed edits in 400 sequential batches, each containing 20 knowledge samples. Results are averaged over five runs with different random seeds.

- **AlphaEdit:** Following the original paper, we target critical layers [4, 5, 6, 7, 8] for editing. The hyperparameter λ is set to 15,000. During the process of computing hidden representations of the critical layer, we perform 25 steps with a learning rate of 0.1.

- **AnyEdit:** Following the original paper, we select layers 4 to 8 for editing and apply a clamp norm factor of 4. The fact token is defined as the last token. The optimization process involves 25 gradient steps for updating the key-value representations, with a learning rate of 0.5. The loss is applied at layer 31, and we use a weight decay of 0.001. To maintain distributional consistency, we introduce a Kullback-Leibler (KL) regularization term with a factor of 0.0625. Furthermore, we enable hyperparameter λ with an update weight of 15,000, using 100,000 samples from the Wikipedia dataset with a data type of float32. The module configurations follow MEMIT, where edits are applied to the MLP down projection layers of the selected transformer blocks. Additionally, for chunked editing, we set a chunk size of 40 tokens with no overlap.

- **RLEdit:** Following the original paper, we configure RLEdit with a memory backtracking decay factor of $\mu = 0.95$, a backtracking depth of $k = 10$, a regularization coefficient of $\eta = 10^{-4}$, and a discount factor of $\gamma = 1$ in the total reward formulation. The initial learning rate is set to 1×10^{-6}, and the meta-learning rate is set to 1×10^{-5}.

- **ENCORE:** Following the original paper, we set the hyperparameters λ_p and λ_n to 15,000 and 20, respectively, and use a probability cutoff of $+1$.

A.3 Hyperparameters

Following (Li et al., 2025b), the specific hyperparameter configurations for different models and datasets are shown in Table 4. We set ϵ to a small constant (e.g., 10^{-6}), and observe that performance is insensitive to its value within the range $[10^{-8}, 10^{-4}]$.

A.4 Datasets

A.4.1 KE Datasets

Next, we introduce the datasets used in this paper. We first introduce the datasets used in model editing.

ZsRE (Zero-shot Relation Extraction) (Levy et al., 2017) serves as a benchmark dataset for language model editing research. Each entry consists of three components: a primary question–answer pair for editing, multiple paraphrased variants of the question generated via back-translation, and locality questions that are semantically unrelated

Table 4: Specific hyperparameter configurations for different models and datasets.

Dataset	Model	Layer	Rank	λ^{loc}	η_f	η_s	β
ZsRE	Llama-3-8B	gate[11–15], up[18–24]	1024	0.6	1e-3	1e-4	0.9
	Gemma-2-9B	gate[32–40], up[32–40]	512	0.6	1e-3	1e-4	0.9
	Mistral-7B	down[17, 18]	1024	0.8	1e-3	1e-4	0.9
CounterFact	Llama-3-8B	gate[22–30], up[22–30]	512	0.6	1e-3	1e-4	0.9
	Gemma-2-9B	gate[32–40], up[32–40]	1024	0.6	1e-3	1e-4	0.9
	Mistral-7B	down[17, 18, 19]	1024	0.8	1e-3	1e-4	0.9
FEVER	Llama-3-8B	gate[22–30], up[22–30]	1024	0.6	1e-3	1e-4	0.9
	Gemma-2-9B	gate[32–40], up[32–40]	1024	0.6	1e-3	1e-4	0.9
	Mistral-7B	down[17, 18]	1024	0.6	1e-3	1e-4	0.9

to the original query. This structure enables the evaluation of model editing along three critical dimensions: (1) accuracy in integrating new information, (2) robustness to paraphrased but semantically equivalent inputs, and (3) precision in preserving unrelated knowledge. For locate-then-edit methods, we adopt the version released by MEMIT (Meng et al., 2022b); for hypernetwork-based methods, we use the version from MEND (Mitchell et al., 2021), where ZsRE is divided into training and test subsets for hypernetwork training and editing evaluation, respectively.

CounterFact (Meng et al., 2022a) is a dataset specifically designed to assess a model's capability to correct contradictory factual statements. Each instance includes an incorrect factual statement requiring editing, its semantically equivalent paraphrases, and unrelated locality statements. This dataset is particularly challenging because models typically produce incorrect responses prior to editing. We use the version from **MEMIT** for both locate-then-edit and hypernetwork-based methods, dividing it into training and test sets of approximately 10,000 factual triples each.

FEVER (Fact Extraction and VERification) (Thorne et al., 2018) is a large-scale fact-checking dataset constructed from modified Wikipedia content. It contains claims labeled as *Supported*, *Refuted*, or *NotEnoughInfo*, with supporting evidence provided for the first two categories. When adapted for model editing tasks, FEVER is simplified into a binary classification setting, where the targets are evenly distributed between two labels (1 and 0), representing the veracity of claims. For locate-then-edit methods, we extract subjects from the claims to align with the (s, r, o) formulation; for hypernetwork-based methods, we employ the MEND version, in which FEVER is partitioned into training and testing splits.

WikiData$_{\text{counterfact}}$ and **WikiData**$_{\text{recent}}$ (Cohen et al., 2024) Since tail entities are often not captured by models, and therefore are not suitable for testing modification edits, (Cohen et al., 2024) collect triplets about popular entities, where the subject corresponds to one of the top-viewed pages in Wikipedia. They also collect a dataset by random sampling entities from Wikidata, and we use it as the training set and the WikiDatacounterfact as the test set.

A.4.2 Capability Evaluation Datasets

Now we introduce the datasets used in capability evaluation.

KOLA: The KoLA dataset is a benchmark designed for evaluating memorization capability in large language models. Its Knowledge Memorization (KM) dimension focuses on a model's ability to recall known facts, constructing high-frequency and low-frequency knowledge tests from Wikipedia and generating evolving test sets by crawling recent articles to assess how well models retain previously unseen and emerging knowledge. The Knowledge Application (KA) dimension evaluates multi-hop reasoning over world knowledge, covering tasks such as HotpotQA and KQA Pro, and similarly incorporates evolving test sets built from newly emerging data. Together, these two task types combine static and evolving subsets to ensure both fairness and temporal relevance in evaluation.

We evaluate the model's reasoning capability using three widely adopted reasoning benchmarks: StrategyQA (Geva et al., 2021), CommonsenseQA (Talmor et al., 2019), and TruthfulQA (Lin et al., 2022). **StrategyQA** is a question–answering benchmark comprising 2,780 examples, each requiring implicit multi-hop reasoning over the provided question, supporting evidence, and final answer. **CommonsenseQA** con-

tains 12,102 multiple-choice questions designed to assess a model's commonsense reasoning ability. Each question requires selecting the correct answer from four plausible distractors based on background commonsense knowledge. **TruthfulQA** evaluates a model's ability to provide truthful and non-hallucinatory responses. It consists of 817 questions across 38 categories. We use the $\text{mc1}_{\text{targets}}$ subset, which is formatted as single-choice questions with 4–5 candidate answers. For all datasets, we prepare the multiple-choice format by labeling the answer options as A–E and randomly shuffling these labels to mitigate potential shortcut biases. For StrategyQA and CommonsenseQA, we use their official training and testing splits. For TruthfulQA, we follow prior work and divide the $\text{mc1}_{\text{targets}}$ subset into an 80/20 train–test split.

We evaluate the model's generalization capability on **SUPERNI** (Wang et al., 2022) as the task pool, which includes 756 tasks in total. To evaluate the effectiveness, we use the held-out tasks from SUPERNI for testing, which consists of 119 tasks across 12 different categories, including free-form generation, word relation reasoning, and various classification tasks. We select SUPERNI as our evaluation benchmark because it offers a diverse set of tasks with varying levels of complexity. Each task has 100 instances, and we exclude instances that exceed the maximum sequence length, resulting in a total of 11,802 instances. We use different evaluation metrics for each task, such as Exact Match for classification or single-word prediction tasks and ROUGE-L for free-form generation tasks, following the metric used in (Wang et al., 2022). We provide the list of 12 evaluation task categories in Table 5. For all evaluation settings, we set the stop sequence as $\backslash n \backslash n$. For all models, we set the maximum input and output sequence length as 1024 and 64 respectively. For target task instances that are long which makes the concatenation of K demonstrations and the target task input sequence exceed the maximum sequence length, we only include the front K' $(K' < K)$ demonstrations that fit the max sequence length. For standard zero-shot setting, we follow the format of (Wang et al., 2022) , appending a sentence "Now complete the following example-" in front of the target input instance.

LaMP (Salemi et al., 2024) is a comprehensive benchmark designed to evaluate the diversification capabilities of large language models (LLMs) in personalized natural language processing tasks. It comprises seven tasks across two categories: three personalized text classification tasks—Personalized Citation Identification, Movie Tagging, and Product Rating—and four personalized text generation tasks—News Headline Generation, Scholarly Title Generation, Email Subject Generation, and Tweet Paraphrasing. To realistically simulate personalization scenarios, LaMP adopts two complementary data split settings: a user-based split for evaluating generalization to unseen users and a time-based split for modeling future interactions of existing users. By conditioning model outputs on user profiles constructed from historical inputs and outputs, LaMP assesses whether LLMs can generate or classify content that aligns with individual users' preferences, writing styles, and contextual needs. This design enables a principled evaluation of diversification across distinct user contexts. Furthermore, LaMP supports both zero-shot and fine-tuned evaluation protocols, as well as retrieval-augmented approaches for incorporating relevant user profile information, providing a robust and flexible framework for measuring LLMs' ability to produce diversified, user-centric outputs across a wide range of language tasks.

A.5 Metrics

A.5.1 ZSRE Metrics

Following prior studies (Meng et al., 2022b; Mitchell et al., 2022), we evaluate the performance of model editing methods on the ZsRE dataset using three standard metrics: **Efficacy**, **Generalization**, and **Specificity**. Given a language model $f_{\mathcal{W}}$, an edited knowledge pair (x, y), its equivalent form x_e, and unrelated locality pairs $(x_{\text{loc}}, y_{\text{loc}})$, the metrics are defined as follows.

Efficacy. Efficacy measures the success rate of incorporating the target knowledge (x, y) into the model $f_{\mathcal{W}}$. It quantifies whether the model correctly predicts the desired output y when queried with x:

$$\mathbb{E}\left[y = \arg\max_{y'} \mathbb{P}_{f_{\mathcal{W}}}(y' \mid x)\right]. \tag{16}$$

Generalization. Generalization assesses whether the edit extends to semantically equivalent variants x_e. It evaluates whether $f_{\mathcal{W}}$ captures the underlying relational structure of the edited knowledge:

$$\mathbb{E}\left[y = \arg\max_{y'} \mathbb{P}_{f_{\mathcal{W}}}(y' \mid x_e)\right]. \tag{17}$$

Generalization Task Abbreviations and Full Names	
TE: Textual Entailment	WA: Word Analogy
CEC: Cause Effect Classification	OE: Overlap Extraction
CR: Coreference Resolution	KT: Keyword Tagging
DAR: Dialogue Act Recognition	QR: Question Rewriting
AC: Answerability Classification	TG: Title Generation
DT: Data to Text	GEC: Grammar Error Correction

Table 5: Generalization Capability Evaluation Task Categories

Specificity. Specificity measures the locality of the edit by examining the preservation of unrelated knowledge. It computes the proportion of unchanged predictions for unrelated inputs x_{loc}:

$$\mathbb{E}\left[y_{\text{loc}} = \arg\max_{y'} \mathbb{P}_{f_{\mathcal{W}}}(y' \mid x_{\text{loc}})\right]. \quad (18)$$

A.5.2 CounterFact Metrics

Following prior work (Meng et al., 2022b,a), we evaluate model editing methods on the CounterFact dataset using three standard metrics: **Efficacy**, **Generalization**, and **Specificity**. Consistent with the original ROME setup, we compare the relative probabilities of candidate answers in the model logits to assess editing performance. Given a language model $f_{\mathcal{W}}$, an edited knowledge pair (x, y), its original counterpart (x, y_0), an equivalent query x_e, and unrelated locality pairs $(x_{\text{loc}}, y_{\text{loc}})$, the metrics are formulated as follows.

Efficacy. Efficacy measures whether the model has successfully integrated the target knowledge (x, y). It evaluates the proportion of cases where the model assigns a higher probability to the desired output y than to the original answer y_0 when queried with x:

$$\mathbb{E}\left[\mathbb{P}_{f_{\mathcal{W}}}(y \mid x) > \mathbb{P}_{f_{\mathcal{W}}}(y_0 \mid x)\right]. \quad (19)$$

Generalization. Generalization assesses whether the edit generalizes to semantically equivalent queries x_e. It measures the proportion of instances where $f_{\mathcal{W}}$ assigns a higher probability to the edited target y than to the original output y_0:

$$\mathbb{E}\left[\mathbb{P}_{f_{\mathcal{W}}}(y \mid x_e) > \mathbb{P}_{f_{\mathcal{W}}}(y_0 \mid x_e)\right]. \quad (20)$$

Specificity. Specificity quantifies the locality of the edit by evaluating whether unrelated knowledge remains intact after modification. It measures the proportion of cases where the model still assigns a higher probability to the original answer y_{loc} than to the edited target y for unrelated inputs x_{loc}:

$$\mathbb{E}\left[\mathbb{P}_{f_{\mathcal{W}}}(y_{\text{loc}} \mid x_{\text{loc}}) > \mathbb{P}_{f_{\mathcal{W}}}(y \mid x_{\text{loc}})\right]. \quad (21)$$

A.5.3 FEVER Metrics

We further evaluate model editing methods on the FEVER dataset using three standard metrics: **Efficacy**, **Generalization**, and **Specificity**. Given a language model $f_{\mathcal{W}}$, an edited knowledge pair (x, y), its equivalent variant x_e, and unrelated locality pairs $(x_{\text{loc}}, y_{\text{loc}})$, the metrics are defined as follows.

Efficacy. Efficacy quantifies the success rate of injecting the target knowledge (x, y) into $f_{\mathcal{W}}$. It measures whether the model's top-1 prediction aligns with the desired output y when queried with x:

$$\mathbb{E}\left[y = \arg\max_{y'} \mathbb{P}_{f_{\mathcal{W}}}(y' \mid x)\right]. \quad (22)$$

Generalization. Generalization evaluates whether the model edit extends to semantically equivalent expressions x_e. It assesses whether $f_{\mathcal{W}}$ has internalized the intrinsic relational structure of the edited knowledge and can consistently predict y across equivalent inputs:

$$\mathbb{E}\left[y = \arg\max_{y'} \mathbb{P}_{f_{\mathcal{W}}}(y' \mid x_e)\right]. \quad (23)$$

Specificity. Specificity measures the locality of the edit, ensuring that unrelated knowledge remains unaffected. It evaluates whether the model continues to produce the original prediction y_{loc} for unrelated inputs x_{loc} after editing:

$$\mathbb{E}\left[y_{\text{loc}} = \arg\max_{y'} \mathbb{P}_{f_{\mathcal{W}}}(y' \mid x_{\text{loc}})\right]. \quad (24)$$

A.5.4 Capability Metrics

To evaluate different aspects of model capability, we adopt task-appropriate metrics for each dimension. For *memorization capability*, we use the F1 score to measure the model's ability to accurately retain and retrieve edited knowledge. For *reasoning capability*, we report accuracy (Acc) to assess the correctness of logical inference. For *generalization capability*, we employ ROUGE-L to quantify the consistency between model outputs under semantically equivalent but contextually perturbed inputs. For *diversification capability*, we follow prior work (Salemi et al., 2024) and adopt accuracy (Acc), and ROUGE-1 to comprehensively evaluate both prediction correctness and output diversification.

A.5.5 Memorization Capability

We assess the memorization capability of LLMs through their **retrieval pathways**, i.e., the model's ability to access stored or contextually provided information when queried. Given an LLM $f_{\mathcal{W}}$, a query x, its target answer y, and an optional context c, the capability is defined as follows.

Retrieval. Retrieval measures whether the model can correctly recover the relevant information—regardless of whether it originates from internal parameters or contextual cues:

$$\mathbb{E}_{(x,y,c)}\left[y = \arg\max_{y'} \mathbb{P}_{f_{\mathcal{W}}}(y' \mid x, c)\right]. \quad (25)$$

A.5.6 Reasoning Capabilities

We evaluate the reasoning abilities of LLMs through their capacity to construct **multi-step reasoning chains** that connect intermediate facts to the final conclusion. Given an LLM $f_{\mathcal{W}}$, a query x, its target answer y, and the corresponding reasoning chain $\mathcal{R} = \{r_1, r_2, \ldots, r_K\}$, the capability is defined as follows.

Reasoning. Reasoning measures whether the model can reproduce the correct conclusion by implicitly or explicitly following the required reasoning chain:

$$\mathbb{E}_{(x,y,\mathcal{R})}\left[y = \arg\max_{y'} \mathbb{P}_{f_{\mathcal{W}}}(y' \mid x, \mathcal{R})\right]. \quad (26)$$

A.5.7 Generalization Capabilities

Generalization reflects the model's ability to apply previously learned knowledge or reasoning patterns to **new, unseen inputs**. Given an LLM $f_{\mathcal{W}}$, a query x, its unseen variant x_{new}, and the target answer y, the capability is defined as follows.

Generalization. Generalization measures whether the model can correctly extend its learned mappings or reasoning behaviors to novel inputs:

$$\mathbb{E}_{(x_{\text{new}},y)}\left[y = \arg\max_{y'} \mathbb{P}_{f_{\mathcal{W}}}(y' \mid x_{\text{new}})\right]. \quad (27)$$

Knowledge editing weakens generalization because local parameter modifications disrupt the model's underlying semantic and reasoning structures, while our generalization preservation reward restores and enhances generalization by enforcing prediction consistency under perturbed inputs and reconstructing the local semantic neighborhood.

A.5.8 Diversification Capabilities

Diversity refers to an LLM's ability to generate varied and contextually distinct outputs rather than collapsing to a single dominant response pattern. Given an LLM $f_{\mathcal{W}}$, a conditioning input x, and the generated sequence $y_{1:t-1}$, token generation follows

$$y_t \sim p_{\mathcal{W}}(y_t \mid y_{1:t-1}, x), \quad (28)$$

where x represents prompt or in-context conditions and $y_{1:t-1}$ denotes previously generated tokens.

diversification. We quantify diversification by measuring the expected dispersion of model outputs across multiple stochastic generations. Let $\{y^{(k)}\}$ be K sampled outputs for the same condition x. The diversification capability is defined as:

$$\mathbb{E}_x\left[\mathrm{Div}\left(\{y^{(k)}\}_{k=1}^{K}\right)\right], \quad (29)$$

where $\mathrm{Div}(\cdot)$ measures output variability (e.g., n-gram entropy, embedding dispersion, or pairwise dissimilarity). High diversification indicates the model can flexibly adapt its generation to different contexts while avoiding mode collapse.

A.5.9 Cross-domain Metrics

Knowledge editing aims to modify model behavior based on updated facts. However, knowledge is inherently interconnected, and altering one fact may influence others, making evaluation challenging. Following prior work, we summarize four key evaluation criteria: *edit success*, *portability*, *locality*, and *fluency*.

Edit Success. Evaluates whether the post-edited model provides correct outputs for both original and paraphrased contexts. We follow prior

work (Mitchell et al., 2021; Li et al., 2024) by combining *reliability* (accuracy on the original context) and *generalization* (accuracy on paraphrased contexts).

Portability. Assesses whether the edited knowledge properly propagates to downstream reasoning tasks (Cohen et al., 2024; Zhong et al., 2023). It includes:

- *Alias:* Consistency across synonymous or aliased entity mentions.
- *Compositionality and Reasoning:* Correct reasoning with updated facts (e.g., updating derived questions).
- *Logical Generalization:* Consistent modification of semantically related facts (e.g., reversed relations).

Locality. Measures whether unrelated knowledge remains unaffected. Evaluation considers:

- *In-Distribution:* Related knowledge from the same domain, testing *forgetfulness* and *relation specificity* (Zhong et al., 2023).
- *Out-of-Distribution:* Unrelated knowledge and general capabilities, evaluated on standard NLP benchmarks.

Generative Capacity (Fluency). Quantifies the diversity and naturalness of text generation after editing (Meng et al., 2022a), computed via weighted bi-gram and tri-gram entropies, where lower values indicate increased repetitiveness.

B Reward

B.1 Memory-Consistent Retrieval Reward

In lifelong knowledge editing, newly injected facts may interact with previously edited knowledge in different semantic ways, which we broadly categorize into three cases: overlap, refinement, and contradiction. Overlap refers to situations where the new fact shares entities or relations with existing knowledge while introducing additional or complementary information, such as adding temporal details to an existing fact. Refinement occurs when the new edit provides a more precise or conditional version of a previously stored fact, effectively updating or narrowing its scope rather than fully replacing it (e.g., introducing a time-dependent or context-specific correction). In contrast, contradiction describes cases where the new knowledge is mutually exclusive with prior edits under the same context, requiring the outdated fact to be revised or overridden. For example, updating the statement "The CEO of Twitter is Jack Dorsey" to "The CEO of Twitter is Linda Yaccarino" constitutes a contradiction, whereas adding "since 2011" to an existing CEO fact represents overlap, and introducing a renamed capital city with a temporal condition exemplifies refinement. These distinctions highlight that effective lifelong knowledge editing requires structured reconciliation of prior knowledge rather than naive rehearsal or overwriting. To mitigate this, we maintain a memory buffer of historical edits and retrieve knowledge instances relevant to the current update at each step t, as shown in Reward B.1. The hypernetwork reconciles these instances with the new knowledge to construct refined memorization samples (x, y^M), and defines the memorization preservation reward as:

$$r_t^{\mathrm{M}} = -\mathcal{L}(\mathcal{W}_t; x, y^M). \tag{30}$$

Reward B.1: Memory-Consistent Retrieval-Reward

Example: The model has previously learned the following fact:

- "The capital of France is Paris."

After integrating updated information (e.g., a note about administrative changes or historical context), the refined target is:

- "The capital of France is Paris, which remains the political and administrative center."

The Memory-Consistent Retrieval reward encourages the model to preserve original facts while integrating new details into its internal knowledge representation. By optimizing this reward, the hypernetwork maintains reliable retrieval and accurate retention of prior knowledge, enabling seamless incorporation of new information and supporting a robust, continually evolving knowledge base.

B.2 Reasoning Trajectory Reward

We aim to explicitly preserve the logical reasoning structure for new knowledge by separating the reasoning skeleton from the subsequent knowledge injection, as shown in Reward B.2. This ensures that the model captures the intermediate inference steps without prematurely committing to spe-

cific entity values. Following a few-shot prompting strategy (e.g., 5-shot), the LLM generates a *masked reasoning path*, where all entities that need to be inferred—excluding those explicitly stated in the question—are replaced with placeholders `[MASK*]` (e.g., `[MASK_1]`, `[MASK_ANSWER]`). Each placeholder is accompanied by a type hint (e.g., "person", "location") to guide the reasoning process.

Reward B.2: Reasoning Trajectory Reward

Example: Consider the question *"What is the nationality of the director of Inception?"*. The masked reasoning path is constructed as follows:

1. The director of *Inception* is `[MASK_1]` (type: person);
2. `[MASK_1]`'s nationality is `[MASK_ANSWER]` (type: country);

This step captures only the reasoning framework without populating factual content.

B.3 Contextual Adaptation Reward

Generalization in lifelong knowledge editing may degrade under contextual distribution shifts, which hinder the model's ability to transfer edited knowledge to previously unseen inputs. To address this issue, we introduce a *generalization preservation reward* that explicitly enforces output stability under diverse contextual perturbations, as shown in Reward B.3. Specifically, given an input x with target label y, we construct a semantically equivalent perturbed input x^C by injecting irrelevant or distracting contextual content to simulate potential distribution shifts. The reward at timestep t is then defined as:

$$r_t^{\mathrm{C}} = -\mathcal{L}(\mathcal{W}_t; x^C, y_t), \tag{31}$$

By maximizing this reward, the hypernetwork is guided to preserve knowledge that is robust and transferable across diverse contexts, thereby mitigating overfitting to specific inputs and ensuring stable performance under novel or perturbed data distributions.

The *Contextual Adaptation* reward further encourages the model to yield consistent predictions under contextual shifts—for example, consistently predicting "Christopher Nolan" across all such variants.

Reward B.3: Contextual Adaptation Reward

Example: Consider the original input:

- "Who directed *Inception*?"

We then construct semantically equivalent but contextually perturbed inputs by injecting irrelevant or non-informative auxiliary context that does not affect the core query semantics, such as:

1. "*Inception* is a 2010 science fiction film widely praised for its visual effects. Who was responsible for directing it?"
2. "The movie *Inception*, starring Leonardo DiCaprio and frequently discussed in film studies courses, was directed by whom?"
3. "During discussions of modern science fiction cinema, identify the filmmaker who directed the film *Inception*."

C More Experiments

C.1 Experiments on More LLM

We further include the Mistral-7B-v0.3 backbone to demonstrate the broad general application of CaPEdit in Table 6.

C.2 Ablation study on ZsRE

We conduct ablation study on ZsRE dataset in Table 7.

C.3 Scaling Experiments

To evaluate the effectiveness of our method, we scale the backbone from LLaMA-3-8B to LLaMA-3-70B in Table 8. The results indicate that, in larger models, parameter interactions can become more complex during optimization. In particular, when the available data is limited or knowledge updates are frequent, the model may leverage redundant parameters to "reuse" capability-related weights for storing knowledge. This phenomenon can exacerbate the implicit entanglement between knowledge and capabilities, making it more difficult to perform localized edits without affecting model capabilities. Consequently, both knowledge retention and capability performance will degrade.

Table 6: Comparison on FEVER and ZsRE for Mistral-7B-v0.3.

Methods	Mistral-7B-v0.3														
	FEVER							ZsRE							Time↓
	Knowledge			Capability				Knowledge			Capability				
	Eff.	Gen.	Spe.	Mem.	Rea.	Gen.	Div.	Eff.	Gen.	Spe.	Mem.	Rea.	Gen.	Div.	
FT	17.08±0.24	21.57±0.30	37.47±0.32	0.00%	0.00%	0.00%	84.50%	32.84±0.30	33.78±0.30	42.19±0.31	0.00%	0.00%	0.00%	78.90%	0.3366s
ROME	0.00±0.00	0.00±0.00	0.00±0.00	12.34%	9.45%	8.21%	87.65%	0.00±0.00	0.00±0.00	0.13±0.02	3.85%	2.21%	1.88%	82.40%	6.0243s
MEMIT	0.00±0.00	0.00±0.00	0.00±0.00	13.85%	12.64%	7.45%	90.10%	0.00±0.00	0.00±0.00	0.13±0.02	1.58%	1.12%	0.85%	81.60%	6.3423s
MEND	0.00±0.00	0.00±0.00	0.00±0.00	0.00%	0.00%	0.00%	85.80%	0.00±0.00	0.00±0.00	0.00±0.00	0.00%	0.00%	0.00%	83.20%	0.9175s
MEND*	0.00±0.00	0.00±0.00	0.00±0.00	10.22%	7.81%	6.95%	88.45%	0.00±0.00	0.00±0.00	0.00±0.00	2.14%	1.64%	1.28%	82.10%	6.1280s
MALMEN	16.67±0.21	16.61±0.18	12.16±0.16	11.92%	9.84%	9.05%	86.90%	0.00±0.00	0.00±0.00	0.00±0.00	2.78%	1.98%	1.62%	82.85%	1.9858s
MALMEN*	17.73±0.24	13.30±0.27	13.85±0.21	15.88%	15.12%	11.34%	89.25%	0.01±0.01	0.02±0.01	1.25±0.04	6.42%	4.88%	3.84%	84.90%	9.3358s
DAFNet	4.86±0.14	4.21±0.19	41.71±0.48	24.12%	25.21%	16.74%	94.65%	1.25±0.08	2.12±0.12	25.27±0.54	4.75%	3.68%	2.84%	87.80%	8.2553s
AlphaEdit	32.74±0.42	30.03±0.29	8.44±0.45	34.22%	30.15%	28.73%	93.85%	0.00±0.00	0.00±0.00	0.00±0.00	21.34%	15.88%	18.12%	91.80%	6.2307s
AnyEdit	33.59±0.37	31.34±0.68	8.36±0.48	35.18%	26.83%	32.64%	92.48%	0.00±0.00	0.00±0.00	0.00±0.00	22.91%	18.01%	19.42%	92.50%	7.3561s
RECT	0.55±0.04	0.05±0.01	0.00±0.00	37.89%	32.34%	39.85%	96.78%	8.18±0.33	8.04±0.46	11.32±0.49	42.45%	20.12%	22.34%	95.21%	6.0486s
PRUNE	5.27±0.34	3.11±0.16	5.89±0.23	39.87%	36.24%	37.92%	95.55%	0.00±0.00	0.00±0.00	0.00±0.00	36.85%	23.92%	28.45%	93.88%	6.3356s
RLEdit	88.99±0.22	88.25±0.25	76.64±0.11	43.28%	38.45%	30.12%	95.85%	71.12±0.31	67.42±0.27	27.43±0.44	25.34%	27.88%	28.91%	94.12%	0.2179s
ENCORE	73.45±0.28	64.12±0.31	14.21±0.38	42.78%	44.38%	33.45%	96.25%	66.85±0.25	61.47±0.19	24.54±0.35	38.92%	30.12%	34.23%	94.67%	0.2947s
CaPEdit	**89.48**±0.16	**95.92**±0.17	**93.17**±0.28	**108.34%**	**112.45%**	**105.67%**	**103.45%**	**84.67**±0.27	**81.70**±0.24	**53.86**±0.36	**106.78%**	**110.12%**	**101.34%**	**102.89%**	**0.1974s**

Table 7: Ablation Study Results for CaPEdit on ZsRE.

Model	Method \ Dataset	ZsRE						
		Eff.↑	Gen.↑	Spe.↑	Mem.↑	Rea.↑	Gen.*↑	Div.↑
Llama-3	**CaPEdit**	**95.82**	**96.54**	**68.41**	**109.52%**	**113.16%**	**104.26%**	**102.95%**
	w/o Procedural Knowledge Synthesis	53.83↓42.01	53.60↓42.94	26.08↓42.33	44.06%↓65.46	46.59%↓66.57	39.04%↓65.22	94.39%↓8.56
	w/o r_t^M	94.98↓0.84	96.03↓0.51	67.87↓0.54	51.27%↓58.25	111.70%↓1.46	103.81%↓0.45	102.18%↓0.77
	w/o r_t^R	94.85↓0.97	95.89↓0.65	67.64↓0.77	108.61%↓0.91	54.81%↓58.35	103.58%↓0.68	101.75%↓1.20
	w/o r_t^C	94.70↓1.12	94.47↓2.07	67.22↓1.19	108.35%↓1.17	112.05%↓1.11	57.62%↓46.64	101.46%↓1.49
	w/ Both Slow Rewards	73.84↓21.98	63.83↓32.71	56.01↓12.40	104.06%↓5.46	106.59%↓6.57	99.04%↓5.22	98.39%↓4.56
	w/ Both Fast Rewards	94.17↓1.65	93.70↓2.84	65.66↓2.75	83.62%↓25.90	76.94%↓36.22	88.24%↓16.02	98.22%↓4.73
	w/ Uniform Reward	94.70↓1.12	95.67↓0.87	67.32↓1.09	102.65%↓6.87	99.85%↓13.31	102.19%↓2.07	101.48%↓1.47
Gemma-2	**CaPEdit**	**95.21**±0.16	**96.26**±0.23	**88.42**±0.41	**115.74%**	**115.46%**	**108.31%**	**103.57%**
	w/o Procedural Knowledge Synthesis	47.85↓47.36	53.35↓42.91	45.50↓42.92	57.46%↓58.28	52.89%↓62.57	53.39%↓54.92	93.90%↓9.67
	w/o r_t^M	94.52↓0.69	95.74↓0.52	88.00↓0.42	68.00%↓47.74	114.82%↓0.64	107.96%↓0.35	102.88%↓0.69
	w/o r_t^R	94.29↓0.92	95.61↓0.65	87.74↓0.68	115.07%↓0.67	60.00%↓55.46	107.58%↓0.73	102.57%↓1.00
	w/o r_t^C	94.11↓1.10	93.92↓2.34	87.27↓1.15	114.68%↓1.06	114.99%↓0.47	62.17%↓46.14	102.26%↓1.31
	w/ Both Slow Rewards	83.26↓11.95	84.40↓11.86	66.15↓22.27	110.27%↓5.47	110.40%↓5.06	103.06%↓5.25	98.93%↓4.64
	w/ Both Fast Rewards	92.60↓2.61	93.64↓2.62	85.96↓2.46	79.62%↓36.12	90.94%↓24.52	82.98%↓25.33	98.51%↓5.06
	w/ Uniform Reward	91.70↓3.51	91.94↓4.32	84.49↓3.93	108.15%↓7.59	108.34%↓7.12	106.82%↓1.49	97.94%↓5.63

C.4 Comparison Between CaPEdit and Memory-Based Editing Methods

Although CaPEdit primarily focuses on parameter-modifying editing, we also evaluate its advantages relative to representative memory-based methods. Specifically, we select SERAC (Mitchell et al., 2022), GRACE (Hartvigsen et al., 2023), MELO (Yu et al., 2024) and KDE (Xu et al., 2025) as baselines. The results are summarized in Table 9.

Across all models and tasks, CaPEdit consistently achieves the highest efficacy (Eff.) and generalization (Gen.) scores, demonstrating its ability to accurately apply desired edits while maintaining robust generalization to other knowledge. Regarding specificity (Spe.) and fluency (Flu.), CaPEdit generally achieves competitive results, though it does not always surpass memory-based approaches. This trade-off is expected and reasonable, as memory-based methods inherently rely on additional storage to better preserve existing knowledge. Overall, these results highlight CaPEdit's effectiveness in combining high efficacy and generalization while maintaining competitive fluency and specificity.

C.5 Editing Efficiency

To evaluate CaPEdit's efficiency, we compared the average editing time per knowledge sample across most methods. Tests were conducted on ZsRE dataset. For locate-then-edit methods, the time included covariance matrix computation, parameter update calculation, and edit execution. For hypernetwork-based methods, we measured hypernetwork training, parameter update calculation, and edit execution time. Since covariance matrices and trained hypernetworks can be reused, we also analyzed editing time excluding these initial setup costs. Results are presented in Figure 5.

C.6 Experiments on Knowledge-Capability Disentanglement

To validate our neuroscience-aligned **Complementary Learning Systems (CLS)** framework, we conduct experiments designed to examine the effects of explicitly disentangling knowledge and model

Table 8: Experiment on LLaMA-70B

Methods	LLaMA-3-70B													
	FEVER							ZsRE						
	Knowledge			Capability				Knowledge			Capability			
	Eff.	Gen.	Spe.	Mem.	Rea.	Gen.	Div.	Eff.	Gen.	Spe.	Mem.	Rea.	Gen.	Div.
FT	4.57±0.19	8.68±0.06	29.3±0.02	0.00%	0.00%	0.00%	86.73%	18.29±0.14	19.52±0.15	10.06±0.16	0.00%	0.00%	0.00%	67.71%
ROME	41.04±0.23	45.93±0.04	11.34±0.24	18.21%	13.3%	12.82%	88.71%	19.97±0.21	2.43±0.21	1.42±0.09	6.79%	4.69%	3.91%	74.3%
MEMIT	1.23±0.25	1.09±1.09	0.00±0.00	16.46%	15.52%	10.96%	93.22%	0.00±0.00	0.00±0.00	2.95±0.03	3.0%	3.6%	2.73%	72.47%
MEND	0.00±0.00	0.00±0.00	0.00±0.00	0.00%	0.00%	0.00%	88.28%	0.00±0.00	0.00±0.00	0.00±0.00	0.00%	0.00%	0.00%	80.51%
MEND*	12.79±0.09	11.75±0.26	7.46±0.47	13.69%	10.56%	9.43%	89.16%	0.00±0.00	0.00±0.00	0.00±0.00	4.87%	4.7%	3.89%	82.08%
MALMEN	2.11±0.05	2.27±2.27	3.56±0.16	0.00%	0.00%	0.00%	86.51%	12.09±0.02	11.63±0.05	3.92±0.26	5.18%	3.71%	3.47%	79.89%
MALMEN*	6.44±2.11	7.83±0.1	2.64±0.25	17.86%	17.86%	15.79%	91.09%	13.48±0.28	13.61±0.14	5.39±0.23	9.36%	6.53%	5.68%	82.05%
DAFNet	33.0±0.19	31.77±0.17	68.69±0.22	28.49%	18.03%	27.24%	94.88%	23.20±0.17	13.98±0.27	34.38±0.15	14.39%	11.02%	12.64%	84.06%
AlphaEdit	95.82±0.12	96.82±0.25	28.17±0.08	29.23%	23.66%	31.79%	94.97%	88.14±0.16	83.39±0.04	31.17±0.14	24.64%	19.29%	20.41%	84.67%
AnyEdit	95.74±0.38	97.03±0.19	27.06±0.28	29.47%	21.13%	36.09%	95.97%	89.96±0.04	85.41±0.23	25.89±0.18	26.83%	21.73%	22.87%	85.69%
RECT	63.27±0.11	53.67±0.44	3.81±0.11	42.51%	37.27%	50.79%	99.14%	13.19±0.17	10.64±0.24	29.12±0.13	40.49%	32.56%	46.64%	91.43%
PRUNE	58.95±0.38	45.63±0.11	2.64±0.0	33.43%	28.63%	45.62%	96.29%	13.67±0.35	14.96±0.24	12.82±0.27	36.95%	35.41%	35.58%	88.49%
RLEdit	98.17±0.13	95.31±0.19	71.75±0.21	35.96%	27.28%	44.93%	96.08%	90.85±0.15	90.26±0.08	47.34±0.09	32.17%	34.29%	38.16%	89.07%
ENCORE	80.66±0.21	78.19±0.34	27.14±0.14	36.24%	41.08%	56.23%	98.86%	90.12±0.12	87.53±0.28	41.37±0.07	37.09%	47.05%	38.42%	91.77%
CaPEdit	**99.38**±0.09	**99.56**±0.11	**93.75**±0.12	**114.59%**	**110.59%**	**110.11%**	**106.37%**	**96.41**±0.16	**97.62**±0.07	**75.73**±0.18	**117.51%**	**117.17%**	**110.51%**	**105.39%**

Table 9: Comparison of CaPEdit with existing methods on the sequential model editing task. Eff., Gen., and Spe. denote Efficacy, Generalization, and Specificity, respectively.

Method	Model	Counterfact							ZsRE						
		Knowledge			Capability				Knowledge			Capability			
		Eff.↑	Gen.↑	Spe.↑	Mem.	Rea.	Gen.	Div.	Eff.↑	Gen.↑	Spe.↑	Mem.	Rea.	Gen.	Div.
Pre-edited		7.85±0.26	10.58±0.26	89.48±0.18	100.00%	100.00%	100.00%	100.00%	36.99±0.30	36.34±0.30	31.89±0.22	100.00%	100.00%	100.00%	100.00%
SERAC	LLaMA3	71.21±0.56	61.05±0.39	66.90±0.21	42.34%	28.45%	36.89%	93.15%	67.75±0.24	33.96±0.35	22.17±0.15	40.12%	25.34%	34.56%	94.23%
GRACE	LLaMA3	96.72±0.13	50.14±0.01	72.23±0.21	58.45%	22.34%	31.89%	90.45%	93.58±0.31	1.03±0.06	31.86±0.12	35.67%	20.12%	29.45%	92.78%
MELO	LLaMA3	65.29±0.13	58.58±0.32	63.36±0.37	55.67%	31.23%	29.45%	94.12%	25.18±0.14	24.14±0.23	30.36±0.75	33.45%	28.90%	37.34%	96.45%
KDE	LLaMA3	97.24±0.24	69.53±0.42	65.24±0.37	58.90%	34.56%	42.34%	91.23%	93.82±0.24	35.28±0.27	32.18±0.28	46.78%	31.23%	40.12%	90.56%
CaPEdit	LLaMA3	**98.45**±0.15	**96.78**±0.18	**88.12**±0.22	**107.34%**	**105.67%**	**109.45%**	**101.23%**	**96.89**±0.12	**92.45**±0.21	**68.34**±0.25	**105.56%**	**108.90%**	**104.78%**	**104.45%**
Pre-edited		16.22±0.31	18.56±0.45	83.11±0.13	100.00%	100.00%	100.00%	100.00%	26.32±0.37	25.79±0.25	27.42±0.53	100.00%	100.00%	100.00%	100.00%
SERAC	GPT-J	82.28±0.26	58.31±0.34	68.98±0.32	45.67%	21.23%	29.45%	91.78%	92.37±0.29	38.21±0.32	25.17±0.25	53.45%	28.90%	37.34%	91.89%
GRACE	GPT-J	96.50±0.24	50.10±0.01	74.42±0.43	40.12%	25.34%	34.56%	93.12%	96.54±0.21	0.40±0.02	24.78±0.21	47.89%	32.34%	31.23%	96.45%
MELO	GPT-J	78.29±0.24	60.52±0.32	66.80±0.52	48.90%	24.56%	32.34%	94.67%	82.24±0.07	32.88±0.03	26.65±0.06	46.78%	31.23%	30.12%	95.78%
KDE	GPT-J	97.14±0.15	63.53±0.23	75.09±0.28	53.72%	36.78%	35.67%	96.34%	97.43±0.19	41.27±0.22	27.32±0.53	38.90%	34.56%	43.45%	95.67%
CaPEdit	GPT-J	**98.90**±0.12	**95.67**±0.15	**86.78**±0.18	**103.45%**	**106.78%**	**104.56%**	**101.89%**	**97.89**±0.11	**90.12**±0.19	**65.43**±0.22	**105.67%**	**109.34%**	**107.89%**	**103.12%**
Pre-edited		22.23±0.73	24.34±0.62	78.53±0.33	100.00%	100.00%	100.00%	100.00%	22.19±0.24	31.30±0.27	24.15±0.32	100.00%	100.00%	100.00%	100.00%
SERAC	GPT2-XL	72.25±0.15	58.18±0.23	64.06±0.37	37.89%	13.45%	21.23%	87.51%	92.17±0.67	36.57±0.72	20.67±0.22	35.67%	20.12%	19.45%	86.13%
GRACE	GPT2-XL	98.88±0.28	50.05±0.01	72.07±0.24	32.34%	27.89%	26.78%	92.12%	94.33±0.37	1.59±0.03	27.63±0.43	30.12%	15.34%	24.56%	94.45%
MELO	GPT2-XL	72.62±0.58	53.63±0.42	63.25±0.62	30.12%	26.78%	25.67%	93.34%	93.54±0.03	45.25±0.02	23.45±0.24	28.90%	24.56%	23.45%	92.67%
KDE	GPT2-XL	99.05±0.42	63.29±0.34	65.47±0.38	40.34%	28.90%	27.89%	97.12%	94.68±0.30	56.35±0.38	26.14±0.28	40.12%	26.78%	35.67%	91.34%
CaPEdit	GPT2-XL	**99.12**±0.10	**94.56**±0.12	**84.56**±0.15	**104.56%**	**107.89%**	**105.67%**	**102.34%**	**96.34**±0.09	**88.90**±0.15	**62.34**±0.18	**106.78%**	**110.12%**	**108.90%**	**103.45%**

capabilities. While prior studies on knowledge-capability entanglement provide only superficial insights, our approach enables a more detailed, interpretable analysis of parameter dynamics during the editing process. These experiments aim to reveal how CLS facilitates efficient knowledge updates while preserving fundamental model capabilities.

C.6.1 Visualization of Knowledge–Capability Entanglement

Figure 6 visualizes the entanglement between knowledge and capability across the 32 layers of LLaMA-3-8B. The horizontal axis corresponds to the knowledge dimension, while the vertical axis enumerates layers from L0 to L31. Color intensity reflects the degree to which knowledge-related and capability-related parameters co-occur within each layer. The heatmap reveals a clear hierarchical pattern. Lower layers (L0–L10) exhibit moderate activation, consistent with foundational feature extraction and shallow syntactic processing. Middle layers (L11–L20) show the strongest and most structured activations, indicating dense semantic encoding and relational reasoning—where knowledge and capability are most strongly intertwined. Upper layers (L21–L31) display sparser, more specialized patterns, reflecting task-specific transformations and high-level abstraction. Overall, the visualization highlights that LLaMA-3-8B does not segregate "knowledge" and "capability" into different blocks; rather, both emerge as layer-dependent, co-activated representations. Knowledge is dynamically routed, composed, and transformed to support downstream capabilities, revealing the intrinsically entangled nature of parameter organization in large transformer models.

C.7 Capability Experiment Results

The detailed results for each capability are shown in Table 11.

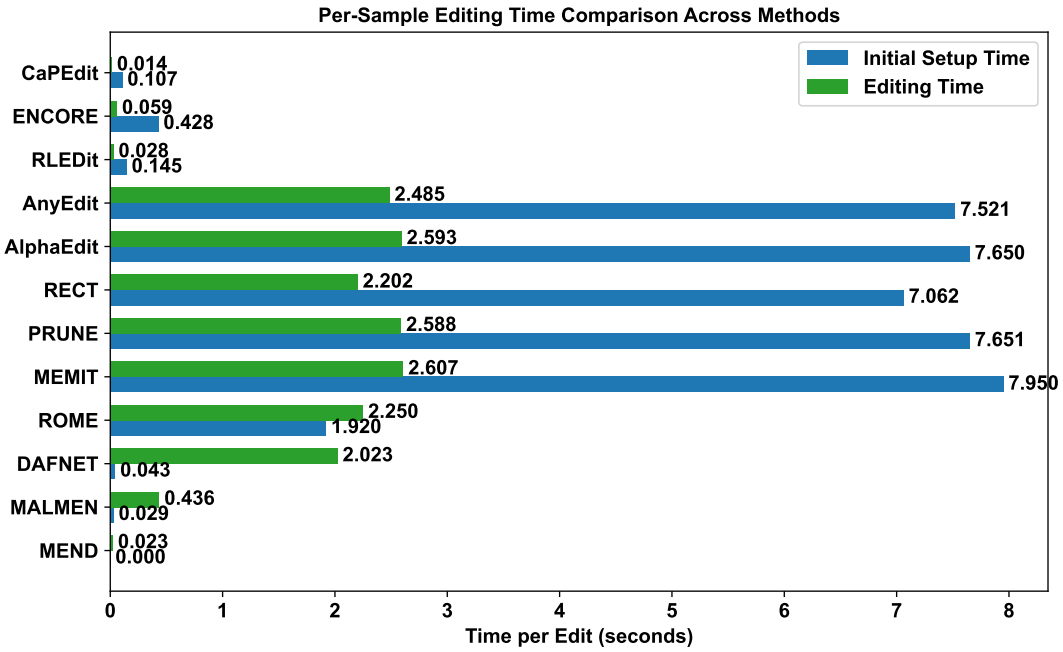

Figure 5: Per-sample editing time comparison across methods.

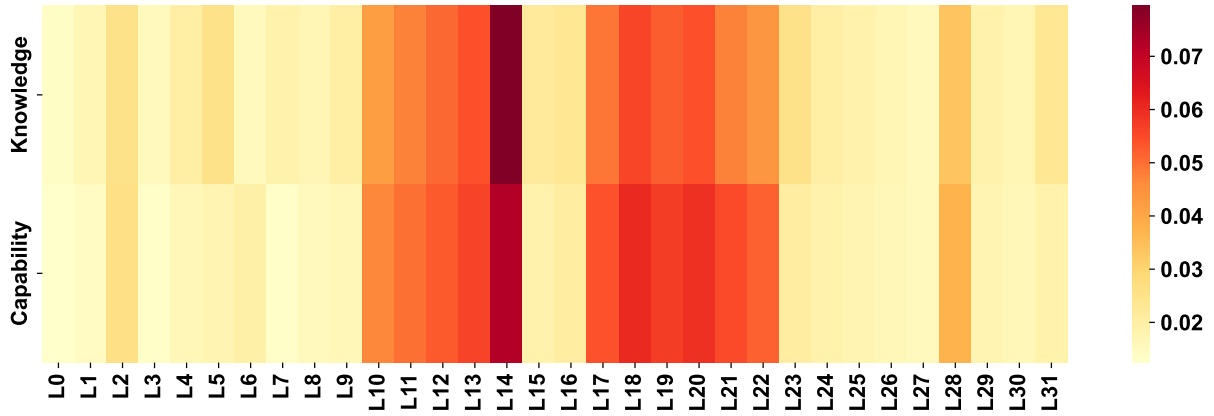

Figure 6: Visualization of Knowledge-Capability Entanglement.

C.7.1 Results of Memorization Capability

As shown in Table 11, we evaluate models on KoLA, which covers knowledge-based question answering and fact retrieval through its Knowledge Memorization (KM) and Knowledge Application (KA) dimensions. KM and KA are both evaluated using F1 scores. The final score is computed as a weighted average of the two levels (e.g., 50% each).

C.7.2 Results of Reasoning Capability

Evaluated by accuracy (Acc), the result is shown in Table 11.

C.7.3 Results of Generalization Capability

Evaluated by ROUGE-L, the result is shown in Table Table 12.

C.7.4 Results of Diversification Capability

The result is shown in Table 13.

D Examples

D.1 KE Examples

To provide an in-depth qualitative analysis of lifelong knowledge editing under sequential updates, we present representative case studies sampled from different stages of the editing stream, shown in Example D.1, and Example D.2. Unlike single-

Table 10: Memorization Capability Evaluation of Knowledge Editing Methods on LLaMA-3-8B.

Model	Level 1: KM				Level 2: KA							Overall
	1-1	1-2	1-3	Percentage	2-1	2-2	2-3	2-4	2-5	2-6	Percentage	Percentage
Origin	22.6	20.8	18.6	100%	18.4	17.5	24.7	16.9	41.4	53.1	100%	100%
FT	0.0	0.0	0.0	0.0%	0.0	0.0	0.0	0.0	0.0	0.0	0.0%	0.0%
ROME	5.6	2.7	1.7	15.49%	0.2	1.1	0.7	0.6	1.2	8.7	5.57%	10.53%
MEMIT	9.1	3.8	0.5	20.35%	1.0	1.6	1.6	1.3	3.3	6.5	8.07%	14.21%
MEND	0.0	0.0	0.0	0.0%	0.0	0.0	0.0	0.0	0.0	0.0	0.0%	0.00%
MEND*	4.2	2.5	1.4	12.58%	1.7	2.2	2.3	1.5	4.0	7.7	10.70%	11.64%
MALMEN	0.0	0.0	0.0	0.0%	0.0	0.0	0.0	0.0	0.0	0.0	0.0%	0.00%
MALMEN*	5.3	1.9	3.1	16.34%	2.5	3.1	3.8	2.5	5.5	10.9	15.94%	16.14%
DAFNet	8.9	4.2	1.6	22.61%	1.4	1.7	1.7	1.2	2.6	7.9	8.73%	15.67%
AlphaEdit	8.5	4.6	3.4	26.07%	4.5	4.8	6.3	4.1	10.5	15.6	25.99%	26.03%
AnyEdit	13.6	7.4	4.9	40.66%	4.8	5.2	6.5	4.6	10.5	19.3	28.56%	34.61%
RECT	17.3	12.0	8.9	60.68%	8.1	8.1	11.2	7.7	18.9	27.0	46.28%	53.48%
PRUNE	15.9	13.1	11.7	65.38%	8.5	8.8	11.1	8.0	19.4	26.9	47.72%	56.55%
RLEdit	14.8	12.2	12.5	63.70%	8.6	9.2	12.6	8.6	20.4	29.9	51.18%	57.44%
ENCORE	20.8	14.7	12.8	77.31%	10.9	10.8	14.8	10.2	25.2	36.4	61.77%	69.54%
CaPEdit	30.3	23.2	21.5	120.40%	19.3	19.3	26.9	18.1	44.5	61.8	109.26%	114.83%

Table 11: Reasoning Capability Evaluation of Knowledge Editing Methods on LLaMA-3-8B.

Model	StrategyQA		CommonSenseQA		TruthfulQA		Overall
	Acc	Percentage	Acc	Percentage	Acc	Percentage	Percentage
Origin	0.647	100%	0.662	100%	0.418	100%	100%
FT	0.000	0.00%	0.000	0.00%	0.000	0.00%	0.00%
ROME	0.054	8.36%	0.083	12.57%	0.061	14.59%	11.84%
MEMIT	0.005	0.70%	0.078	11.77%	0.030	7.27%	6.58%
MEND	0.000	0.00%	0.000	0.00%	0.000	0.00%	0.00%
MEND*	0.011	1.65%	0.086	13.05%	0.046	10.95%	8.55%
MALMEN	0.000	0.00%	0.000	0.00%	0.000	0.00%	0.00%
MALMEN*	0.045	7.03%	0.165	25.00%	0.064	15.35%	15.79%
DAFNet	0.058	8.98%	0.129	19.45%	0.054	13.03%	13.82%
AlphaEdit	0.102	15.77%	0.175	26.41%	0.030	7.16%	16.45%
AnyEdit	0.109	16.92%	0.144	21.77%	0.080	19.20%	19.30%
RECT	0.349	53.88%	0.412	62.30%	0.198	47.47%	54.55%
PRUNE	0.253	39.09%	0.288	43.46%	0.180	42.96%	41.84%
RLEdit	0.302	46.67%	0.335	50.66%	0.202	48.24%	48.52%
ENCORE	0.321	49.60%	0.427	64.54%	0.234	55.99%	56.71%
CaPEdit	0.721	111.42%	0.829	125.29%	0.484	115.77%	117.49%

Table 12: Generalization Capability Evaluation of Knowledge Editing Methods on LLaMA-3-8B.

Model	TE	CEC	CR	DAR	AC	WA	OE	KT	QR	TG	DTT	GEC	Overall Percentage
Origin	28.49	24.76	14.06	26.19	34.26	13.52	3.79	16.53	3.17	6.28	4.92	4.18	100%
FT	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00%
ROME	2.02	2.18	1.88	2.29	2.77	1.57	0.32	2.19	0.30	0.40	0.62	0.59	10.17%
MEMIT	1.07	1.94	1.58	2.03	2.03	1.63	0.24	1.92	0.18	0.34	0.61	0.39	8.27%
MEND	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00%
MEND*	1.18	1.74	1.81	1.56	2.24	1.21	0.28	1.86	0.17	0.32	0.54	0.26	7.63%
MALMEN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00%
MALMEN*	3.27	2.74	2.53	3.11	3.42	1.99	0.41	2.22	0.39	0.66	0.71	0.63	12.82%
DAFNet	1.19	1.60	1.80	1.60	2.73	1.36	0.29	1.66	0.19	0.48	0.63	0.21	8.04%
AlphaEdit	6.28	6.23	4.25	6.37	7.89	3.74	0.88	4.77	0.81	1.40	1.45	1.06	25.58%
AnyEdit	5.78	5.72	3.96	5.86	7.93	3.70	0.89	4.22	0.70	1.49	1.35	0.77	34.61%
RECT	14.04	13.20	8.45	14.08	17.40	7.73	1.95	9.37	1.66	3.28	2.68	2.49	54.27%
PRUNE	12.99	12.46	7.71	12.68%	17.11	7.05	1.84	8.59	1.57	3.11	2.62	2.09	50.34%
RLEdit	15.37	12.99	7.99	14.17	18.20	7.96	2.00	9.28	1.68	3.36	2.76	2.17	54.39%
ENCORE	15.20	14.07	8.76	14.78	19.07	7.90	2.16	9.79	1.76	3.62	3.04	2.54	57.94%
CaPEdit	29.84	26.40	15.82	28.12	36.00	14.90	4.01	18.40	3.35	6.80	5.54	4.59	108.36%

Table 13: Diversification Capability Evaluation of Knowledge Editing Methods on LLaMA-3-8B.

Model	LaMP-1U	LaMP-2U	LaMP-3U	LaMP-4U	LaMP-5U	LaMP-6U	Overall Percentage
	ACC	ACC	ROUGE-1	ROUGE-1	ROUGE-1	ROUGE-1	Percentage
Origin	0.532	0.472	0.169	0.453	4.69	5.25	100%
FT	0.433	0.389	0.139	0.373	0.412	0.458	83.95%
ROME	0.439	0.394	0.142	0.372	0.455	0.466	86.43%
MEMIT	0.490	0.430	0.154	0.406	0.448	0.517	93.00%
MEND	0.440	0.400	0.144	0.379	0.424	0.446	85.32%
MEND*	0.463	0.409	0.145	0.392	0.440	0.454	87.77%
MALMEN	0.446	0.393	0.140	0.379	0.401	0.452	84.17%
MALMEN*	0.476	0.419	0.151	0.390	0.422	0.497	89.76%
DAFNet	0.465	0.426	0.152	0.396	0.431	0.495	90.25%
AlphaEdit	0.494	0.439	0.155	0.422	0.439	0.507	93.43%
AnyEdit	0.466	0.430	0.151	0.414	0.454	0.487	91.48%
RECT	0.513	0.452	0.164	0.438	0.468	0.496	96.83%
PRUNE	0.492	0.442	0.158	0.421	0.448	0.513	94.32%
RLEdit	0.496	0.451	0.163	0.432	0.462	0.519	96.28%
ENCORE	0.520	0.462	0.163	0.432	0.478	0.532	98.45%
CaPEdit	0.530	0.480	0.174	0.454	0.498	0.582	103.64%

edit evaluations, these examples emphasize long-term behavior, examining how edited knowledge is retained, retrieved, and generalized after multiple subsequent edits.

Across all evaluated cases, **CaPEdit consistently restores the target knowledge with high fidelity**. The edited model not only produces correct and stable responses to the original edit queries, but also maintains accuracy under paraphrased questions, altered syntactic structures, and additional contextual noise. This indicates that CaPEdit integrates new factual knowledge into the model in a manner that preserves underlying retrieval and reasoning procedures, rather than relying on brittle surface-level associations.

Mechanism-Level Analysis. A closer inspection reveals that CaPEdit's advantages stem from its explicit modeling of procedural knowledge along three complementary dimensions. First, *memory-consistent retrieval* ensures that newly edited facts are harmonized with existing knowledge, preventing contradictions and transient recall failures. Second, *reasoning-preserving trajectories* maintain coherent multi-step inference chains even after repeated edits, enabling the model to answer indirect or compositional queries correctly. Third, *contextual adaptation* enforces robustness to perturbations, allowing the edited knowledge to be applied reliably across diverse linguistic and contextual variations.

In contrast, existing baseline methods exhibit distinct and recurring failure modes that become increasingly pronounced as edits accumulate:

- **MEMIT** frequently yields partially correct responses, where key entities or relations are recalled but crucial factual attributes are missing or incorrectly combined. Such behavior suggests that while the target knowledge is locally injected, it is not fully integrated into the model's broader retrieval structure.

- **AlphaEdit** often overgeneralizes the injected updates, causing the edit to propagate beyond its intended scope. This leads to the introduction of irrelevant information or spurious associations, and in some cases produces factual hallucinations, reflecting a lack of fine-grained control over contextual application.

- **ENCORE** occasionally generates outputs that are unrelated to, or directly contradict, the target knowledge. These failures indicate severe interference between newly edited facts and previously stored knowledge, resulting in explicit knowledge conflicts and unstable long-term behavior.

Impact of Sequential Editing. Notably, these failure modes are amplified in the lifelong setting. As more edits are applied, baseline methods tend to accumulate inconsistencies, exhibiting degraded performance on both earlier and later edits. In contrast, CaPEdit maintains stable behavior throughout the editing stream, demonstrating strong resistance to catastrophic forgetting and interference. This highlights the importance of explicitly accounting for temporal dynamics and long-term capability preservation in lifelong knowledge editing.

Overall, these case studies demonstrate that effective lifelong knowledge editing requires more

than achieving correctness on isolated examples. For tasks demanding high precision, semantic consistency, and robust generalization under sequential updates, **CaPEdit exhibits substantially stronger robustness** than existing approaches. By disentangling factual updates from procedural capabilities, CaPEdit not only preserves the integrity of edited knowledge and minimizes hallucinations, but also ensures reliable application across paraphrased and perturbed inputs.

These qualitative findings provide concrete empirical support for our central thesis: *knowledge–capability disentanglement is essential for scalable and reliable lifelong knowledge editing*, and constitutes a key factor that distinguishes CaPEdit from conventional editing paradigms.

D.2 Capability Examples

We illustrate how knowledge editing can disrupt reasoning and retrieval, causing inconsistent knowledge application, and how CaPEdit preserves key capabilities.

Memorization Capability. As shown in Example E.1, most models drifted from the core factors of alpine tourism, focusing instead on surface-level or common activities such as spas, lakes, and culinary experiences. This demonstrates that knowledge editing can disrupt retrieval pathways, causing the model to access superficially related but contextually irrelevant information. As a result, these models failed to capture the direct causal link between alpine sports and the rise of the "grand hotels" documented in historical sources. CaPEdit, by contrast, accurately retained factual knowledge, correctly identifying skiing and mountaineering as the key attractions, demonstrating its robustness in preserving memorized facts and reasoning about their causal relationships despite potential interference from editing.

Reasoning Capability. As shown in Example E.2, most models made an overly simplistic association between modern MMA and Roman Colosseum games, skipping the necessary historical and conceptual steps that distinguish a contemporary regulated sport from ancient gladiatorial spectacles. Knowledge editing here can disrupt the model's reasoning pathways, leading it to retrieve partially related knowledge inappropriately and to conflate distinct temporal or conceptual contexts. Consequently, the outputs are superficially plausible but factually incorrect. CaPEdit, however, correctly traces these multi-step reasoning processes, recognizing that MMA is a modern sport built from diverse martial arts traditions and does not directly originate from Roman combat, demonstrating superior preservation of reasoning capability under editing interventions.

Generalization Capability. As shown in Example E.3, most models fail to transfer the underlying causal structure to paraphrased contexts, relying on surface cues or unrelated entities. Knowledge editing can interfere with the model's abstraction mechanisms, breaking links between events and leading to inconsistent generalization across semantically equivalent scenarios. CaPEdit successfully maintains narrative-level causal chains, inferring that Tom runs out of time in different phrasings of the story, thereby demonstrating strong generalization and the ability to apply learned patterns across contexts without degradation.

Diversification Capability. As shown in Example E.4, most models produce largely uniform responses across user contexts, failing to meaningfully adjust emphasis, style, or framing to match user interests. This reflects that knowledge editing can disrupt retrieval and reasoning pathways used for context-conditioned adaptation, collapsing nuanced distinctions into generic outputs. CaPEdit, in contrast, produces contextually differentiated responses: a legally grounded, compliance-focused explanation for the legality-oriented user and a practical, risk-aware answer for the travel-focused user, showing strong diversification while maintaining factual correctness.

E Dataset Visualization

To help readers better understand the lifelong editing task and our implementation approach, we provide 3 examples from the ZsRE, FEVER, and CounterFact datasets, as illustrated in Figures 7, 8, and 9. We also demonstrate examples from capability datasets, such as KoLA in Figure 10, StrategyQA in Figure 11, SUPERNI in Figure 12, and LaMP in Figure 13.

Example D.1: Model Editing Case Study on Llama3-8B

Input: What are the key achievements of Ada Lovelace in computing history?
Paraphrase: Can you summarize Ada Lovelace's main contributions and accomplishments in the history of computing?
Target Output: Ada Lovelace was a pioneering mathematician and writer, widely recognized as the first computer programmer. She worked with Charles Babbage on his Analytical Engine and wrote detailed notes explaining how the machine could be programmed to perform complex calculations. Lovelace's notes also included what is considered the first algorithm intended for implementation on a computer. Her visionary ideas about computing extended beyond mere calculation, anticipating the concept of general-purpose computing. She is celebrated for her profound influence on the development of computer science.
LLM-edited Generation:

- **MEMIT:** Ada Lovelace was an English mathematician who did some work with early computing machines. She is sometimes credited with creating algorithms, but details about her contributions are limited and unclear.
- **AlphaEdit:** Ada Lovelace worked with Charles Babbage and helped with calculations. She wrote notes on mathematics and science, though it is debated whether she actually wrote algorithms for the Analytical Engine.
- **AnyEdit:** Ada Lovelace was a mathematician who collaborated with Babbage. She contributed to early ideas about machines, but her exact role as a "first programmer" is often overstated in popular accounts.
- **RLEdit:** Ada Lovelace was interested in computation and assisted in designing mechanical calculators. She made some notes that might be considered programming, but her impact on modern computing is minimal.
- **ENCORE:** Ada Lovelace studied mathematics and wrote about machines. She is sometimes mentioned in computing history, but the significance of her work is uncertain.
- **CaPEdit:** Ada Lovelace was a pioneering mathematician and writer, widely recognized as the first computer programmer. She worked with Charles Babbage on his Analytical Engine and wrote detailed notes explaining how the machine could be programmed to perform complex calculations. Lovelace's notes also included what is considered the first algorithm intended for implementation on a computer. Her visionary ideas about computing extended beyond mere calculation, anticipating the concept of general-purpose computing. She is celebrated for her profound influence on the development of computer science.

Example D.2: Model Editing Case Study on Gemma-2-9B

Input: What are Dr. Li Wei's main contributions to quantum computing?
Paraphrase: Could you summarize the key achievements of Dr. Li Wei in the field of quantum computing?
Target Output: Dr. Li Wei is a prominent physicist whose groundbreaking work in quantum computing centers on the development of the "Wei-Stabilizer" code, a novel error-correction framework that significantly reduces overhead. She leads the Quantum Architectures Lab at Tsinghua University, where her team recently demonstrated a 99.95% fidelity two-qubit gate on superconducting circuits, a world record at the time of publication. Dr. Li's theoretical contributions also include the "coherent annealing" algorithm, which has been experimentally validated to solve specific optimization problems faster than classical counterparts. She was awarded the prestigious Dirac Medal in 2022 for these achievements.
LLM-edited Generation:

- **MEMIT:** Dr. Li Wei is a researcher in quantum computing, but I couldn't find specific details about her major contributions. She is known to work on quantum algorithms and error correction in general. According to some publications, she has collaborated with various institutions. Her work is considered important within the specialized community.
- **AlphaEdit:** Dr. Li Wei is a Chinese-American physicist known for her work on topological quantum computing. She proposed a new type of non-Abelian anyon for fault-tolerant quantum computation and has led a team at Stanford University that built a small-scale topological quantum processor. Dr. Li received the MacArthur Fellowship for this pioneering research.
- **AnyEdit:** Dr. Li Wei is a prominent physicist whose work in quantum computing focuses on quantum machine learning and neural networks. She leads a lab at the Chinese Academy of Sciences and has developed several efficient quantum-classical hybrid algorithms for big data analysis. Her team's work on quantum supremacy was published in *Nature* last year.
- **RLEdit:** Dr. Li Wei is a scientist who has made important contributions to quantum computing. She has published many papers in top journals and is recognized for her innovative research. Her work involves improving the stability and performance of quantum systems, and she has received several awards for her efforts. She is a leading figure in her field.
- **ENCORE:** Dr. Li Wei is a financial analyst with expertise in blockchain technology and cryptocurrency markets. She has written reports on the impact of quantum computing on future encryption standards for a major consulting firm.
- **CaPEdit:** Dr. Li Wei is a prominent physicist whose groundbreaking work in quantum computing centers on the development of the "Wei-Stabilizer" code, a novel error-correction framework that significantly reduces overhead. She leads the Quantum Architectures Lab at Tsinghua University, where her team recently demonstrated a 99.95% fidelity two-qubit gate on superconducting circuits, a world record at the time of publication. Dr. Li's theoretical contributions also include the "coherent annealing" algorithm, which has been experimentally validated to solve specific optimization problems faster than classical counterparts. She was awarded the prestigious Dirac Medal in 2022 for these achievements.

Example E.1: Memorization Capability Case Study on Llama3-8B

Input: Which two Swiss attractions were the cause of the "grand hotels" in which Jakob Ragaz worked on?
Paraphrase: What were the two major attractions in Switzerland that led to the development of the "grand hotels" associated with Jakob Ragaz's work?
Target Output: The two Swiss attractions that contributed to the rise of the "grand hotels" where Jakob Ragaz worked were skiing and mountaineering.
LLM-edited Generation:

- **MEMIT:** Jakob Ragaz's hotels were mainly associated with Switzerland's spa culture and hot springs tourism rather than mountain activities.
- **AlphaEdit:** The development of the grand hotels was driven by lake sightseeing and thermal bath resorts, which attracted wealthy European travelers.
- **AnyEdit:** Ragaz's work is often linked to early Swiss railway tourism and scenic village retreats, not specific outdoor sports.
- **RLEdit:** The grand hotels emerged primarily due to Switzerland's reputation for quiet countryside escapes and forest hiking paths.
- **ENCORE:** The attractions were Swiss culinary traditions and local music festivals that drew high-class visitors.
- **CaPEdit:** The grand hotels where Jakob Ragaz worked rose to prominence largely due to Switzerland's popularity for skiing and mountaineering.

Example E.2: Reasoning Capability Case Study on Llama3-8B

Input: Is Mixed Martial Arts (MMA) totally original from Roman Colosseum games?
Paraphrase: Did modern Mixed Martial Arts directly originate from the combat spectacles held in the Roman Colosseum?
Target Output: False. MMA is a modern regulated sport synthesizing global martial arts, not a direct continuation of Roman gladiatorial games.
LLM-edited Generation:

- **MEMIT:** Yes. MMA is fundamentally derived from ancient Roman gladiator combat and preserves most of its original fighting principles.
- **AlphaEdit:** True. Modern MMA traces its core structure to the Colosseum games, which served as the earliest form of organized mixed-style combat.
- **AnyEdit:** MMA is largely a revival of Roman arena fighting, with its rules and techniques evolving directly from gladiatorial traditions.
- **RLEdit:** The origins of MMA can be attributed mainly to Roman amphitheater contests, which laid the groundwork for today's combat formats.
- **ENCORE:** Yes. The Roman Colosseum games formed the foundational prototype for modern MMA competitions.
- **CaPEdit:** No. MMA is a modern full-contact sport built from diverse martial arts systems and does not originate from Roman Colosseum games.

Example E.3: Generalization Capability Case Study

Input: Title: Out of Time. Sentence 1: Tom was up late studying for an exam. Sentence 2: He woke up late the next morning. Sentence 3: He barely made it in time to take it. Sentence 4: Tom tried to rush through the exam as best he could.
Choices: a. In the end, Travis regret all of the money he spent. b. Unfortunately he ran out of time before he could finish.
Paraphrase: Title: Running Late. Sentence 1: Tom stayed awake late preparing for a test. Sentence 2: The next day, he overslept. Sentence 3: He arrived just before the exam started. Sentence 4: He hurried through the questions as fast as possible.
Choices: a. Travis later felt sorry about wasting his money. b. Sadly, Tom could not complete the exam before time was up.
Target Output: b. The correct ending should describe Tom failing to finish the exam due to time pressure, which is consistent with the narrative context.
LLM-edited Generation:

- **MEMIT:** a. The model incorrectly selects an unrelated consequence involving a different character, failing to generalize narrative coherence under paraphrasing.
- **AlphaEdit:** a. Despite surface similarity changes, the model loses track of the causal storyline and outputs an incoherent ending.
- **AnyEdit:** a. The model fails to transfer commonsense reasoning to the paraphrased context.
- **RLEdit:** a. The edit overfits to lexical patterns and ignores entity and event consistency.
- **ENCORE:** b. The model preserves the causal chain and correctly infers that Tom runs out of time.
- **CaPEdit:** b. The model maintains narrative-level generalization and selects the logically consistent ending.

Example E.4: Diversification Capability Case Study

Input:
Question: Can a non-US citizen drive my US car in Mexico?
User Profile (Excerpt): 1. Past question about travel regulations and insurance abroad. 2. Questions related to legal validity across countries (passports, visas). 3. Interest in practical travel logistics and compliance.
Rubric Aspects: - Driver Nationality - Specific Nationalities (Mexican, Guatemalan)
Alternative User Contexts:
Context A (Legality-focused user): The user frequently asks about legal interpretations, cross-border regulations, and official requirements.
Context B (Practical travel-focused user): The user mainly asks about real-world travel experience, risk avoidance, and common practice rather than strict legality.
Target Output:
Both answers should be factually correct: Non-US citizens *can* legally drive a US-registered car in Mexico, provided that the vehicle is covered by valid Mexican insurance and complies with temporary import permit requirements. However, the emphasis and framing should differ across user contexts.
LLM-edited Generation:

- **MEMIT:** Produces a single generic answer emphasizing legality, ignoring differences in user context and failing to diversify response style or focus.
- **AlphaEdit:** Overemphasizes formal legal language in both contexts, resulting in nearly identical outputs with limited contextual adaptation.
- **AnyEdit:** Mentions practical considerations but collapses both contexts into a uniform explanation, reducing personalization diversity.
- **RLEdit:** Shows minor lexical variation but preserves the same argument structure, failing to meaningfully adjust emphasis based on user preferences.
- **ENCORE:** Generates a legally grounded response for Context A and a practice-oriented, experience-driven explanation for Context B, demonstrating moderate diversification.
- **CaPEdit:** Produces two distinctly framed yet consistent answers: one emphasizing official regulations, permits, and compliance for Context A, and another highlighting common travel practice, insurance coverage, and risk mitigation for Context B, thereby achieving strong user-conditioned diversification without sacrificing correctness.

F Algorithm

We first formalize several concepts in Table 14.

Table 14: Correspondence between Markov Decision Process elements and CaPEdit components.

Markov Decision Process	**CaPEdit**
hypernetwork $\mathcal{H}$	Hypernetwork-based editor coordinating knowledge and capability updates
Environment	LLM $f_{\mathcal{W}}$ with evolving parameters
State s_t	Fine-tuning gradients $\nabla\mathcal{W}_{t-1}$
Action a_t	Action updates $\Delta\mathcal{W}_t^{\mathrm{K}}$
Policy π_θ	Hypernetwork mapping gradients to update directions
State Transition	$\mathcal{W}_t = \mathcal{W}_{t-1} + \Delta\mathcal{W}_t$
Reward r_t	Knowledge reward r_t^{K} and capability reward r_t^{M}, r_t^{R}, r_t^{C}
Trajectory	Sequential edits over a knowledge stream
Objective	Maximize long-horizon knowledge integration while preserving capabilities

The pseudo-code is provided in Algorithm 1.

Algorithm 1: Training of CaPEdit

Input: Pre-trained LLM $f_{\mathcal{W}_0}$; hypernetwork H_θ; learning rates η_f, η_s; EMA factor β
Output: Optimized hypernetwork parameters θ
repeat
  Sample factual edits $\{(x_t^K, y_t^K)\}_{t=1}^n$;
  Stage I: Procedural Generation ;
  for $t = 1$ **to** n **do**
    Generate procedural signals (x_t^P, y_t^P) for (x_t^K, y_t^K) ;
  Stage II: Knowledge Editing ;
  for $t = 1$ **to** n **do**
    Compute gradient $\nabla_{\mathcal{W}_{t-1}}$ and apply hypernetwork update $\mathcal{W}_t = \mathcal{W}_{t-1} + H_\theta(\nabla_{\mathcal{W}_{t-1}})$;
    Compute fast factual reward r_t^{K} ;
    Compute procedural rewards $r_t^{\mathrm{M}}, r_t^{\mathrm{R}}, r_t^{\mathrm{C}}$ and aggregate r_t^P ;
    Update EMA slow reward $\bar{r}_t^P$;
    Step-level update: $\theta_{t+1} = \theta_t - \eta_f \nabla_\theta \mathcal{L}_t^{\mathrm{K}}$
  Trajectory-level update: $\theta \leftarrow \theta - \eta_s \nabla_\theta \sum_{t=1}^n (\mathcal{L}_t^{\mathrm{M}} + \mathcal{L}_t^{\mathrm{R}} + \mathcal{L}_t^{\mathrm{C}})$
until *convergence*;
return θ

G Theory and Proof of Capabilities

G.1 Why Localized Knowledge Editing Degrades Memorization, Reasoning, and Generalization but Preserves Diversity

We analyze the disparate impact of localized parameter editing on distinct model capabilities through a unified perturbation analysis framework. This reveals a fundamental dichotomy: capabilities that depend on fine-grained computational stability (memorization, reasoning, generalization) are first-order sensitive to local edits, while global statistical properties (diversity) are only second-order sensitive.

G.1.1 Preliminaries and Notation

Let $f_{\mathcal{W}} : \mathcal{X} \to \Delta(\mathcal{Y})$ be a pre-trained model with parameters $\mathcal{W} \in \mathbb{R}^d$. An editing operation applies a localized perturbation $\Delta\mathcal{W}$ (typically sparse or low-rank), yielding $f_{\mathcal{W}+\Delta\mathcal{W}}$.

We conceptualize capabilities as functionals derived from the model's forward computation:

1. Memorization: Retrieval accuracy of stored facts, formalized as stability of a retrieval mapping $R_{\mathcal{W}} : x \mapsto h_L(x)$ where h_L is the final-layer representation.

2. Reasoning: Correct execution of multi-step inference, modeled as the iterative application of a latent transition operator $T_{\mathcal{W}} : h_t \mapsto h_{t+1}$.

3. Generalization: Consistency of predictions under semantic-preserving input transformations, characterized by the local Lipschitz constant of the encoder $E_{\mathcal{W}} : x \mapsto h_0(x)$.

4. Diversity: Global entropy of the output distribution, $\mathrm{Div}(\mathcal{W}) = \mathbb{E}x[\mathcal{H}(p\mathcal{W}(\cdot|x))]$.

G.1.2 Memorization Degradation: Perturbation Amplification in Deep Networks

Theorem G.1 (Instability of Retrieval Pathways under Local Perturbation)**.** *Let the retrieval mapping $R_{\mathcal{W}}(x) = h_L(x)$ be defined by a L-layer feedforward computation:*

$$h_{l+1} = \phi_l(\mathcal{W}lh_l), \quad l = 0, \ldots, L-1, \tag{32}$$

where each ϕ_l is a 1-Lipschitz activation. Suppose a perturbation $\Delta\mathcal{W}k$ is applied at layer k. Then the deviation in the final representation is bounded below by:

$$\sup x \in \mathcal{X}|R\mathcal{W} + \Delta\mathcal{W}(x) - R_{\mathcal{W}}(x)|2 \geq |\Delta\mathcal{W}k|\mathrm{op} \cdot \inf x|h_k(x)|2, \tag{33}$$

where $|\cdot|\mathrm{op}$ denotes the spectral norm. This lower bound is strictly positive for non-zero perturbations and typical inputs.

Proof. The forward pass can be viewed as a composition of Lipschitz maps. The perturbation at layer k introduces an additive error $\delta_k = \Delta\mathcal{W}kh_k$. Because subsequent layers are Lipschitz, this error does not vanish; it propagates forward. Formally:

$$|h'L - h_L| \geq \sigma\min(\Phi L|k+1)|\delta_k| \geq |\Delta\mathcal{W}k|\mathrm{op}|h_k|, \tag{34}$$

where $\sigma_{\min}(\Phi_{L|k+1})$ is the minimum singular value of the Jacobian from layer $k+1$ to L, which is positive for well-conditioned transformations. Since memorization relies on precise point-wise retrieval, this representation shift directly translates to accuracy degradation. □

G.1.3 Reasoning Degradation: Error Accumulation in Iterative Processes

Theorem G.2 (Error Propagation in Multi-step Reasoning)**.** *Consider a reasoning chain of length K modeled by repeated application of a transition $T_{\mathcal{W}}$:*

$$h_{t+1} = T_{\mathcal{W}}(h_t), \quad t = 0, \ldots, K-1. \tag{35}$$

Assume $T_{\mathcal{W}}$ is differentiable and let $J_T = \partial T_{\mathcal{W}}/\partial h$ be its Jacobian. Under a parameter perturbation $\Delta\mathcal{W}$, the error at step K is lower-bounded by:

$$|h'K - h_K| \geq \frac{|\partial\mathcal{W}T \cdot \Delta\mathcal{W}|}{|J_T|} \cdot \left(|J_T|^K - 1\right), \tag{36}$$

where $\partial_{\mathcal{W}}T$ is the parameter gradient of T. For $|J_T| \geq 1$, the error grows exponentially with K.

Proof. The perturbed dynamics follow $h't + 1 = T\mathcal{W} + \Delta\mathcal{W}(h't)$. A first-order expansion yields the recurrence:

$$\Delta ht + 1 \approx J_T \cdot \Delta h_t + \partial_{\mathcal{W}}T \cdot \Delta\mathcal{W}, \tag{37}$$

where $\Delta h_t = h't - h_t$. Unrolling this recurrence gives:

$$\Delta h_K = \sum i = 0^{K-1} J_T^{K-1-i} \cdot (\partial_{\mathcal{W}}T \cdot \Delta\mathcal{W}). \tag{38}$$

Taking norms and using the triangle inequality leads to the stated lower bound. This demonstrates how local edits disrupt the compositional stability essential for reasoning. □

G.1.4 Generalization Degradation: Breakdown of Local Isometry

Theorem G.3 (Distortion of Semantic Neighborhoods)**.** *Let $E_{\mathcal{W}} : \mathcal{X} \to \mathbb{R}^m$ be the model's encoder. For two semantically equivalent inputs x, x' with $|E_{\mathcal{W}}(x) - E_{\mathcal{W}}(x')| \leq \epsilon$, a local edit $\Delta\mathcal{W}$ can distort their distance:*

$$\sup_{x \sim x'} \left| |E_{\mathcal{W}+\Delta\mathcal{W}}(x) - E_{\mathcal{W}+\Delta\mathcal{W}}(x')| - |E_{\mathcal{W}}(x) - E_{\mathcal{W}}(x')| \right| \geq |\Delta\mathcal{W}| \cdot \sigma_{\max}(\nabla_W E(x) - \nabla_W E(x')), \tag{39}$$

where $\sigma_{\max}$ denotes the maximum singular value. This implies that local edits can arbitrarily warp the geometry of local semantic neighborhoods.

Proof. A first-order Taylor expansion gives:

$$E_{\mathcal{W}+\Delta\mathcal{W}}(x) - E_{\mathcal{W}+\Delta\mathcal{W}}(x') \quad = \left[E_{\mathcal{W}}(x) - E_{\mathcal{W}}(x')\right] + \left[\nabla_{\mathcal{W}} E(x) - \nabla_{\mathcal{W}} E(x')\right] \Delta\mathcal{W} + o(|\Delta\mathcal{W}|). \tag{40}$$

The key term $[\nabla_{\mathcal{W}} E(x) - \nabla_{\mathcal{W}} E(x')]\,\Delta\mathcal{W}$ represents a directional derivative mismatch. Since $\nabla_{\mathcal{W}} E(x)$ varies with x, this term is generally non-zero and linearly scales with $|\Delta\mathcal{W}|$. This breaks the local isometry (Lipschitz smoothness) that generalization relies upon. □

G.1.5 Diversity Preservation: A Global Statistical Invariant

Theorem G.4 (First-Order Insensitivity of Output Entropy)**.** *Define the output diversity as the average conditional entropy:*

$$\mathrm{Div}(\mathcal{W}) = \mathbb{E}x \sim P\mathcal{X}\left[\mathcal{H}(Y|X=x)\right], \quad \textit{where } \mathcal{H}(Y|X=x) = -\sum_{y \in \mathcal{V}} p_{\mathcal{W}}(y|x) \log p_{\mathcal{W}}(y|x). \tag{41}$$

Assume the support of $p_{\mathcal{W}}(y|x)$ is the full vocabulary $\mathcal{V}$ and is Lipschitz in $\mathcal{W}$. Then, for a localized edit $\Delta\mathcal{W}$ that affects predictions on a subset $\mathcal{X}$edit of measure $\mu(\mathcal{X}edit) \to 0$, we have:

$$|\mathrm{Div}(\mathcal{W} + \Delta\mathcal{W}) - \mathrm{Div}(\mathcal{W})| \leq C \cdot \mu(\mathcal{X}edit) \cdot |\Delta\mathcal{W}| + o(|\Delta\mathcal{W}|), \tag{42}$$

where C is a constant depending on the Lipschitz constant of $\log p\mathcal{W}(y|x)$. Thus, diversity is perturbed only on the measure of the edited subspace, making it a robust global invariant.

Proof. We decompose the expectation:

$$\Delta\mathrm{Div} = \mathbb{E}x \in \mathcal{X}\mathrm{edit}[\Delta\mathcal{H}x] + \mathbb{E}x \notin \mathcal{X}\mathrm{edit}[\Delta\mathcal{H}x]. \tag{43}$$

For $x \notin \mathcal{X}$edit, $\Delta\mathcal{H}x = 0$ by definition of a localized edit. For $x \in \mathcal{X}$edit, a Taylor expansion shows $|\Delta\mathcal{H}x| \leq L|\Delta\mathcal{W}|$ for some Lipschitz constant L. Therefore,

$$|\Delta\mathrm{Div}| \leq L|\Delta\mathcal{W}| \cdot \mu(\mathcal{X}\mathrm{edit}). \tag{44}$$

Since $\mu(\mathcal{X}\mathrm{edit})$ is negligible in lifelong editing (sparse factual updates), diversity remains approximately constant to first order. □

G.1.6 Unifying Principle: Dichotomy of Capability Sensitivity

Corollary G.5 (First-Order vs. Second-Order Sensitivity)**.** *Let $\mathcal{C}(\mathcal{W})$ denote a capability metric.*

If $\mathcal{C}$ depends on point-wise or path-wise stability of the computation (Memorization, Reasoning, Generalization), then

$$|\mathcal{C}(\mathcal{W} + \Delta\mathcal{W}) - \mathcal{C}(\mathcal{W})| = \Theta(|\Delta\mathcal{W}|), \tag{45}$$

i.e., it exhibits first-order sensitivity. If $\mathcal{C}$ is an average of a smooth functional over the input distribution (Diversity), then

$$|\mathcal{C}(\mathcal{W} + \Delta\mathcal{W}) - \mathcal{C}(\mathcal{W})| = O(\mu(supp(\Delta\mathcal{W})) \cdot |\Delta\mathcal{W}|), \tag{46}$$

i.e., it exhibits second-order sensitivity, scaled by the measure of the affected input subspace.

Proof. This follows directly from Theorems G.1, G.2, G.3, and G.4. Theorems G.1-G.3 establish linear lower bounds in $|\Delta\mathcal{W}|$ for memorization, reasoning, and generalization. Theorem G.4 establishes an upper bound for diversity that is linear in both $|\Delta\mathcal{W}|$ and the measure $\mu(\mathcal{X}_{\text{edit}})$, the latter being vanishingly small for sparse edits. This quantifies the fundamental asymmetry in capability degradation. □

Summary. Localized knowledge editing introduces targeted perturbations that inevitably destabilize the precise computational pathways required for memorization, reasoning, and generalization. In contrast, diversity, as a global statistical property averaged over the entire input space, remains largely invariant to such sparse changes. This theoretical dichotomy underscores the necessity of frameworks like **CaPEdit** that explicitly model and mitigate first-order degradation paths for critical capabilities while leveraging the inherent robustness of second-order stable properties like diversity.

G.2 Why regularization technique cannot effectively preserve LLM capabilities during knowledge editing?

Definition G.6 (Capability Functional)**.** Let f_W be a language model with parameters $W \in \mathbb{R}^d$. A *capability* is defined as a functional

$$\mathcal{C} : \mathbb{R}^d \to \mathbb{R},$$

mapping model parameters to performance on a class of tasks (e.g., reasoning, generalization).

Definition G.7 (Regularized Knowledge Editing)**.** A regularized editing method produces parameter updates

$$W_{t+1} = W_t + \Delta W_t,$$

by solving

$$\min_{\Delta W} \ \mathcal{L}_{\text{edit}}(W_t + \Delta W) \ + \ \lambda \|\Delta W\|_p,$$

for some $p \in \{1, 2, F\}$.

Theorem G.8 (Limitation of Norm-Based Regularization)**.** *There exists a capability functional $\mathcal{C}$ such that for any $\epsilon > 0$, there exists a parameter update ΔW satisfying $\|\Delta W\|_2 < \epsilon$ but*

$$|\mathcal{C}(W + \Delta W) - \mathcal{C}(W)| > \delta,$$

for some constant $\delta > 0$ independent of ϵ.

Proof Sketch. Modern overparameterized neural networks admit highly non-uniform sensitivity across parameter subspaces. Specifically, parameters controlling retrieval paths or attention routing may occupy low-norm but high-impact directions.

Thus, arbitrarily small perturbations in ℓ_2 norm can induce discrete changes in internal computation graphs, leading to sharp drops in downstream capabilities. □

Corollary G.9. *Norm-based regularization constrains the magnitude of updates but cannot control the semantic or procedural effect of edits. Therefore, it is insufficient for preserving model capabilities under sequential knowledge editing.*

Definition G.10 (Procedural Knowledge)**.** Procedural knowledge is defined as a collection of conditional distributions governing *how* a model applies factual knowledge:

$$\mathcal{P} = \left\{ p_{\text{ret}}(z \mid x),\ p_{\text{rea}}(\tau \mid x),\ p_{\text{gen}}(y \mid x') \right\},$$

where z denotes retrieved latent states, τ reasoning trajectories, and x' perturbed contexts.

Proposition G.11 (Capability Decomposition)**.** *For a fixed factual knowledge base, the model's output behavior can be decomposed as*

$$p(y \mid x) = \sum_{z,\tau} p(y \mid \tau, z, x)\, p(\tau \mid z, x)\, p(z \mid x).$$

Corollary G.12. *Preserving the distributions $p(z \mid x)$ (retrieval), $p(\tau \mid z, x)$ (reasoning), and $p(y \mid x')$ under perturbations is sufficient to preserve task-level capabilities.*

Remark G.13. We do not claim that the three components are strictly complete in an information-theoretic sense. Rather, they form a minimal sufficient set that captures the dominant failure modes observed in lifelong editing.

H Operational Characterization of Procedural Knowledge

Procedural knowledge is treated as a latent behavioral construct that governs how factual information is retrieved, composed, and applied across contexts. Since such knowledge is not directly observable or uniquely identifiable, we characterize it through measurable behavioral quantities that reflect stable model behavior under sequential edits.

Formally, let $f_\theta(y \mid x)$ denote the model's predictive distribution. We study how this distribution changes under editing-induced perturbations $\theta \to \theta + \Delta\theta$, and focus on preserving structural properties of f_θ rather than individual factual correctness.

H.1 Behavioral Proxies as Measurable Objectives

We introduce three operational proxies, each defined as a measurable constraint on the model's predictive behavior. These proxies correspond to empirically observed failure modes in lifelong knowledge editing.

Let θ_0 denote the pre-edit model and θ_t the model after t edits.

H.1.1 Memory-Consistent Retrieval

Objective. Preserve stable retrieval behavior over previously edited knowledge.

Let $\mathcal{B}_t = \{(x_i^b, y_i^b)\}_{i=1}^M$ be a buffer of prior edits. We define the retrieval consistency objective as:

$$\mathcal{L}_{\mathrm{M}}(\theta_t) = \mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[D_{\mathrm{KL}}\left(f_{\theta_0}(y \mid x); f_{\theta_t}(y \mid x)\right)\right].$$

This loss penalizes deviations in the model's retrieval behavior on previously edited instances, regardless of the current edit target.

Interpretation. $\mathcal{L}_{\text{mem}}$ measures *behavioral drift* in retrieval, not parameter similarity or memory replay.

H.1.2 Reasoning Trajectory Consistency

Objective. Preserve structural reasoning patterns independent of factual content.

For an input x requiring multi-step reasoning, let $\tau = (r_1, \ldots, r_L)$ denote a reasoning trajectory generated by the pre-edit model. We construct a masked version τ_{mask} by removing factual entities.

We define the reasoning consistency objective as:

$$\mathcal{L}_{\text{rea}}(\theta_t) = \mathbb{E}_{x\sim\mathcal{D}}\left[\sum_{i=1}^{L} D_{\mathrm{KL}}!\left(f_{\theta_0}(r_i \mid \tau_{<i}, x); |; |; f_{\theta_t}(r_i \mid \tau_{<i}, x)\right)\right].$$

This objective constrains the *conditional reasoning dynamics* rather than end-task accuracy.

H.1.3 Contextual Adaptation

Objective. Preserve prediction invariance under semantic-preserving transformations.

Let $\mathcal{T}$ be a family of transformations $T : \mathcal{X} \to \mathcal{X}$ such that $T(x)$ preserves task-relevant semantics.

We define contextual robustness as:

$$\mathcal{L}_{\text{ctx}}(\theta_t) = \mathbb{E}_{(x,y)\sim\mathcal{K}}\mathbb{E}_{T\sim\mathcal{T}}\left[D_{\mathrm{KL}}!\left(f_{\theta_t}(y \mid x); |; f_{\theta_t}(y \mid T(x))\right)\right].$$

This loss discourages overfitting to surface forms introduced during editing.

H.2 Empirical Coverage of Editing-Induced Failure Modes

The three objectives $\mathcal{L}_{\text{mem}}, \mathcal{L}_{\text{rea}}, \mathcal{L}_{\text{ctx}}$ target distinct but complementary behavioral properties of f_θ.

To make this explicit, we define a *capability stability functional*:

$$\mathcal{S}(\theta_t) = \mathbb{E}_{x,y}\left[\log f_{\theta_t}(y \mid x) \log f_{\theta_0}(y \mid x)\right],$$

which measures the change in overall predictive behavior relative to the pre-edit model.

Empirically, we observe that violations of $\mathcal{S}(\theta_t)$ can be attributed to increases in one or more of the proxy losses:

$$\Delta\mathcal{S}(\theta_t); \approx; -\sum_{i \in \text{mem,rea,ctx}} \lambda_i \Delta\mathcal{L}_i(\theta_t),$$

where $\lambda_i \geq 0$ are empirically estimated coefficients.

Important. This relation is *descriptive rather than theoretical*: it motivates optimization but does not assert completeness or necessity.

H.3 Implications for Optimization

The final editing objective combines factual correctness with proxy-based behavioral regularization:

$$\mathcal{L}_{\text{total}}\mathcal{L}_{\text{edit}} + \lambda_{\text{mem}}\mathcal{L}_{\text{mem}} + \lambda_{\text{rea}}\mathcal{L}_{\text{rea}} + \lambda_{\text{ctx}}\mathcal{L}_{\text{ctx}}.$$

Since $\mathcal{L}_{\text{edit}}$ operates at the instance level while the proxy losses aggregate behavior across edits and transformations, this formulation induces an implicit temporal asymmetry in supervision without assuming formal multi-timescale convergence guarantees.

I Theoretical Guarantees

Theorem I.1 (Convergence of CaPEdit)**.** *Under mild assumptions (Lipschitz continuity of rewards, bounded gradients), CaPEdit's dual-timescale optimization converges to a Pareto-optimal solution that balances factual editing and capability preservation:*

$$\lim_{t \to \infty} \nabla_\theta \mathbb{E}[r_t^K + \lambda r_t^P] = 0$$

with $r_t^P = r_t^M + r_t^R + r_t^C$ being the procedural reward.

Sketch. The proof follows two-timescale stochastic approximation theory:

1. Fast updates (η_f) track factual accuracy: $\theta_{t+1} = \theta_t + \eta_f \nabla r_t^K$
2. Slow updates (η_s) optimize procedural preservation: $\theta \leftarrow \theta + \eta_s \nabla \sum r_t^P$
3. With $\eta_s = o(\eta_f)$, the system converges to the solution of the averaged system:

$$\dot{\theta} = \nabla \bar{r}^K(\theta) + \lambda \nabla \bar{r}^P(\theta)$$

   where $\bar{r}^K, \bar{r}^P$ are averaged rewards.

The Pareto optimality follows from the convexity of the reward space. □

I.1 Theoretical Guarantees of CaPEdit

We formalize the theoretical properties of CaPEdit using a rigorous theorem–proof framework, drawing insights from optimization theory, reinforcement learning, and perturbation analysis of neural networks.

Theorem I.2 (Capability Degradation under Unconstrained Knowledge Editing)**.** *Let $\mathcal{P}_{\mathcal{C}}(\mathcal{W})$ denote the performance metric for a capability $\mathcal{C} \in \{\mathrm{Mem}, \mathrm{Rea}, \mathrm{Gen}\}$, assumed to be twice continuously differentiable in a neighborhood of the pretrained parameters $\mathcal{W}_0$. Let $\Delta\mathcal{W}$ be a parameter update optimized solely for a knowledge-editing loss $\mathcal{L}_{edit}$ without explicit capability preservation. Then, under mild assumptions on gradient alignment, we have in expectation:*

$$\mathbb{E}\big[\mathcal{P}_{\mathcal{C}}(\mathcal{W}_0 + \Delta\mathcal{W})\big] \leq \mathcal{P}_{\mathcal{C}}(\mathcal{W}_0) - \eta \cdot \big\|\nabla\mathcal{P}_{\mathcal{C}}\big\| \cdot \big\|\nabla\mathcal{L}_{edit}\big\| \cdot \cos\theta + O(\|\Delta\mathcal{W}\|^2), \tag{47}$$

where θ is the angle between $\nabla\mathcal{P}_{\mathcal{C}}$ and $\nabla\mathcal{L}_{edit}$, and $\eta > 0$ is the effective step size. If $\cos\theta < 0$ (i.e., the gradients conflict), capability degradation occurs.

Proof. Consider the second-order Taylor expansion:

$$\mathcal{P}_{\mathcal{C}}(\mathcal{W}_0 + \Delta\mathcal{W}) = \mathcal{P}_{\mathcal{C}}(\mathcal{W}_0) + \nabla\mathcal{P}_{\mathcal{C}}^{\top}\Delta\mathcal{W} + \frac{1}{2}\Delta\mathcal{W}^{\top}H_{\mathcal{C}}\Delta\mathcal{W} + o(\|\Delta\mathcal{W}\|^2). \tag{48}$$

Assuming $\Delta\mathcal{W} = -\alpha\nabla\mathcal{L}_{\text{edit}}$ for some $\alpha > 0$, we have:

$$\mathbb{E}[\nabla\mathcal{P}_{\mathcal{C}}^{\top}\Delta\mathcal{W}] = -\alpha\,\mathbb{E}[\nabla\mathcal{P}_{\mathcal{C}}^{\top}\nabla\mathcal{L}_{\text{edit}}] \tag{49}$$

$$= -\alpha\|\nabla\mathcal{P}_{\mathcal{C}}\|\|\nabla\mathcal{L}_{\text{edit}}\|\cos\theta. \tag{50}$$

When $\cos\theta < 0$, the linear term is negative, leading to a drop in $\mathcal{P}_{\mathcal{C}}$. The Hessian term $H_{\mathcal{C}}$ is bounded under smoothness assumptions, yielding the $O(\|\Delta\mathcal{W}\|^2)$ remainder. □

Definition I.3 (Dual-Timescale Rewards)**.** Fast rewards r_t^K supervise immediate factual correctness. Slow rewards are defined via EMA:

$$\bar{r}_t^P = \beta\bar{r}_{t-1}^P + (1-\beta)r_t^P, \quad \beta \in (0,1).$$

Theorem I.4 (EMA as Temporal Low-Pass Filter)**.** *The EMA reward $\bar{r}_t^P$ minimizes the variance of stochastic procedural rewards among all linear filters with exponential memory decay.*

Proof Sketch. EMA corresponds to the optimal linear estimator of the latent mean reward under an AR(1) noise model. It suppresses high-frequency fluctuations while preserving long-term trends, aligning with the slow evolution of procedural knowledge. □

Proposition I.5. *Under bounded rewards and sufficiently small step sizes $\eta_f \ll \eta_s$, the two-level optimization induces a separation of timescales, preventing interference between factual updates and procedural adaptation.*

Theorem I.6 (Capability Preservation via Dual-Timescale Rewards)**.** *Let $\mathcal{W}_t$ denote model parameters after step t under CaPEdit. Parameter updates are generated by a hypernetwork H_θ and optimized via gradient ascent on the scalar reward:*

$$r_t = r_t^{\mathrm{K}} + \lambda\bar{r}_t^{\mathrm{P}}, \quad \bar{r}_t^{\mathrm{P}} = \beta\bar{r}_{t-1}^{\mathrm{P}} + (1-\beta)r_t^{\mathrm{P}}, \tag{51}$$

where r_t^{K} is the fast factual reward and r_t^{P} aggregates procedural rewards. Then, for any procedural capability metric $\mathcal{P}_{\mathcal{C}}(\mathcal{W})$, the expected update direction satisfies:

$$\mathbb{E}\Big[\nabla\mathcal{P}_{\mathcal{C}}(\mathcal{W}_t)^{\top}\Delta\mathcal{W}_t\Big] \geq -\epsilon_t, \tag{52}$$

where $\epsilon_t \to 0$ as $\beta \to 1$ and the trajectory length increases.

Proof. Under CaPEdit, the hypernetwork parameters θ are updated by gradient ascent:

$$\Delta\theta_t \propto \nabla_\theta r_t^{\mathrm{K}} + \lambda \nabla_\theta \bar{r}_t^{\mathrm{P}}. \tag{53}$$

The induced model update is:

$$\Delta\mathcal{W}_t = J_{H_\theta}\, \Delta\theta_t, \tag{54}$$

where J_{H_θ} denotes the Jacobian of the hypernetwork with respect to θ.

The fast reward r_t^{K} aligns updates with factual correctness, while the EMA-smoothed procedural reward $\bar{r}_t^{\mathrm{P}}$ aggregates gradients of the form $\nabla\mathcal{P}_{\mathcal{C}}(\mathcal{W}_{\leq t})$ over time. Due to exponential smoothing, high-variance short-term fluctuations are suppressed, and the expected contribution of $\nabla_\theta \bar{r}_t^{\mathrm{P}}$ converges to the long-term average procedural gradient.

As a result, destructive alignment between $\Delta\mathcal{W}_t$ and $\nabla\mathcal{P}_{\mathcal{C}}$ is progressively attenuated, yielding:

$$\mathbb{E}\Big[\nabla\mathcal{P}_{\mathcal{C}}(\mathcal{W}_t)^\top \Delta\mathcal{W}_t\Big] \geq -\epsilon_t, \tag{55}$$

where ϵ_t vanishes as temporal averaging improves. □

Theorem I.7 (Capability Improvement beyond Pretraining)**.** *Assume the pretrained parameters $\mathcal{W}_0$ are not a stationary point of the procedural objective:*

$$\nabla\mathbb{E}[r_t^{\mathrm{P}}]\big|_{\mathcal{W}_0} \neq 0. \tag{56}$$

Then there exists a sequence of updates under CaPEdit such that:

$$\mathcal{P}_{\mathcal{C}}(\mathcal{W}_T) > \mathcal{P}_{\mathcal{C}}(\mathcal{W}_0) \tag{57}$$

for some procedural capability $\mathcal{C}$.

Proof. Pretraining optimizes a generic language modeling objective, which does not explicitly maximize procedural metrics such as reasoning robustness or contextual generalization. Therefore, $\mathcal{W}_0$ may not be Pareto-optimal with respect to $(r^{\mathrm{K}}, r^{\mathrm{P}})$.

Since CaPEdit explicitly performs gradient ascent on the EMA-smoothed procedural reward $\bar{r}_t^{\mathrm{P}}$, any non-zero component of $\nabla\mathbb{E}[r_t^{\mathrm{P}}]$ induces a consistent update signal. Over sufficient steps, this yields monotonic improvement in $\mathcal{P}_{\mathcal{C}}$ until a new stationary point is reached, which may exceed the pretrained baseline. □

Theorem I.8 (Diversity Stability under Dual-Timescale Editing)**.** *Assume model logits are κ-Lipschitz with respect to parameters. Then for parameters $\mathcal{W}_T$ obtained via CaPEdit:*

$$\big|\mathrm{Div}(\mathcal{W}_T) - \mathrm{Div}(\mathcal{W}_0)\big| \leq L\kappa \sum_{t=1}^{T} \|\Delta\mathcal{W}_t\|. \tag{58}$$

Proof. Under dual-timescale optimization, the EMA reward penalizes updates that degrade long-term procedural signals, implicitly bounding $\|\Delta\mathcal{W}_t\|$ over time. Applying Lipschitz continuity of logits and entropy yields the stated bound. □

Corollary I.9. *Procedural rewards in CaPEdit implicitly regularize parameter sensitivity, thereby preserving output diversity without explicit entropy maximization.*

Proof. Memory-consistent retrieval stabilizes retrieval manifolds, reasoning rewards enforce smooth transition dynamics, and contextual adaptation promotes local invariance. Together, these effects constrain parameter sensitivity and preserve diversity via Theorem I.8. □

```
{
  "subject": "Colt King Cobra",
  "src": "The manufacturer of Colt King Cobra was who?",
  "rephrase": "Which company did Colt King Cobra produce?",
  "alt": "Colt's Manufacturing Corporation",
  "loc": "nq question: ek veer ki ardaas veera meaning in english",
  "loc_ans": "A Brother's Prayer... Veera",
  "ans": "Colt's Manufacturing Company"
},
```


Figure 7: A sample of zsre dataset.


```
{
  "prompt": "Nikolaj Coster-Waldau worked with the Fox Broadcasting Company.",
  "equiv_prompt": [
    "Nikolaj Coster-Waldau worked with Fox Broadcasting Company.",
    "Nikolaj Coster-Waldau collaborated with the Fox Broadcasting Company.",
    "Nikolaj Coster-Waldau collaborated with Fox Broadcasting Company.",
    "Nikolaj Coster-Waldau worked together with the Fox Broadcasting Company.",
    "Nikolai Coster-Waldau worked with the Fox Broadcasting Company.",
    "Nikolay Coster-Waldau worked with the Fox Broadcasting Company.",
    "Nikolaj Coster-Waldau has worked with the Fox Broadcasting Company.",
    "Nikolaj Coster-Waldau worked at the Fox Broadcasting Company.",
    "Nikolaj Coster-Waldau worked for the Fox Broadcasting Company.",
    "Nikolaj Coster-Waldau worked at Fox Broadcasting Company.",
    "Nikolaj Coster-Waldau worked for Fox Broadcasting Company.",
    "Nikolai Coster-Waldau worked for the Fox Broadcasting Company.",
    "Nikolay Coster-Waldau worked for the Fox Broadcasting Company."],
  "ans": "SUPPORTS", "alt": "REFUTES",
  "unrel_prompt": "There is not a film that goes by the name of Annie.",
  "unrel_ans": "REFUTES"
},
```


Figure 8: A sample of fever dataset.

```
{
  "case_id": 6798,
  "pararel_idx": 11358,
  "requested_rewrite": {
    "prompt": "{} debuted on",
    "relation_id": "P449",
    "target_new": {
      "str": "CBS",
      "id": "Q43380"
    },
    "target_true": {
      "str": "NBC",
      "id": "Q13974"
    },
    "subject": "The Rerun Show"
  },
  "paraphrase_prompts": [
    "Disappearance\nIn the late 1980s, David Trampier suddenly disappeared. The Rerun Show was originally aired on",
    "Ralph Sasse enjoyed success as Mississippi State's head football coach. The Rerun Show was released on"
  ],
  "neighborhood_prompts": [
    "The Menagerie was originally aired on",
    "NBC Nightly News was originally aired on",
    "Freaks and Geeks was originally aired on",
    "Friends, season 7 debuted on",
    "Noah's Ark was originally aired on",
    "The New Normal premiered on",
    "Camp Cucamonga debuted on",
    "The Menagerie is to debut on",
    "Law & Order: LA was released on",
    "The Voice is to debut on"
  ],
  "attribute_prompts": [
    "The Little Mermaid was originally aired on",
    "Cybill premiered on",
    "Cybill debuted on",
    "The Beverly Hillbillies premiered on",
    "CBS News was released on",
    "CBS News debuted on",
    "The King of Queens premiered on",
    "Dink, the Little Dinosaur was originally aired on",
    "Barnaby Jones debuted on",
    "Blue Bloods was released on"
  ],
  "generation_prompts": [
    "The Rerun Show first aired on",
    "The Rerun Show aired alongside other programs including",
    "The Rerun Show aired alongside other programs including",
    "The Rerun Show aired alongside other programs including",
    "The Rerun Show is my favorite show that has aired on",
    "The Rerun Show is my favorite show that has aired on",
    "The Rerun Show aired alongside other programs including",
    "The Rerun Show is my favorite show that has aired on",
    "The Rerun Show first aired on",
    "The Rerun Show first aired on"
  ]
},
```


Figure 9: A sample of CounterFact dataset.

```
{
  "adapter_spec": {
     "instructions": "Please give answers to the following questions about knowledge.
    Note: If there are more than one answer, print them all and separate them with a semicolon (;).
    Please do not give anything other than the answers.",
     "input_prefix": "Question: ",  "input_suffix": "\n",
     "reference_prefix": "", "reference_suffix": "",
     "output_prefix": "Answer: ", "output_suffix": "\n",
     "instance_prefix": "\n", "max_train_instances": 5,
     "max_eval_instances": 1000, "max_tokens": 64,
     "stop_sequences": [],
     "decoding_parameters": {"temperature": 1},
     "ouput_format": "list"
  },
  "request_states": [
     {
        "instance": {
           "input": {"text": "Where was N. R. Narayana Murthy born?"},
           "references": [
              {
                 "output": { "text": "Mysore"},
                 "tags": [ "correct"]
              }
           ],
           "split": "test",
           "id": "2_high_freq_ent8186"
        },
        "request": {}
     },
     {
        "instance": {
           "input": { "text": "Who is the performer of Tears on My Pillow?"},
           "references": [
              {
                 "output": {"text": "Johnny Tillotson"},
                 "tags": ["correct"]
              }
           ],
           "split": "test",
           "id": "1_low_freq_ent4818"
        },
        "request": {}
     },
        ]
}
```


Figure 10: A sample of KoLA dataset.

```
{
    "qid": "93e3995669f121e630ef",
    "term": "Rede Globo",
    "description": "Brazilian commercial television network",
    "question": "Do the anchors on Rede Globo speak Chinese?",
    "answer": false,
    "facts": [ "Rede Globo is a Brazilian television network.",
               "The official language of Brazil is Portuguese." ],
    "decomposition": [
      "What country broadcasts Rede Globo?",
      "What is the official language of #1?",
      "Is #2 Chinese?"
    ],
    "evidence": [[ [ "Rede Globo-1" ] ],[["Brazil-1"]], [ "operation" ]],
      [ [ [ "Rede Globo-1" ] ],[  [ "Brazil-1" ]  ],[ "operation"]],
      [[["Rede Globo-1" ]],
        [["Portuguese language-1" ] ],
        [ "operation"] ]]
  },
```


Figure 11: A sample of StrategyQA dataset.

```
{
    "id": "task032-8f0a28240c89433d959b4b491e21a8a6",
    "input": "Context Word: clean.",
    "output": [
      "Although PersonX was much tidier than PersonY was, _ let her clean the room anyway.",
      "Being clean was not important to PersonX but was imperative for PersonY , so _ never bathed on weekends.",
      "Never clean, PersonX really irritated PersonY, as _ was so repulsive to her with her odor.",
      "PersonX agreed to wash the dishes when asked by PersonY because it was _ 's turn to clean the kitchen.",
      "PersonX always cleans her jewelry, but PersonY doesn't own much jewelry, due to _ being the wealthier person.",
      "PersonX always had a very clean house while PersonY did not because cleanliness was very important to _ .",
      "PersonX always had to clean up after PersonY, because _ didn't want to live in filth.",
      "PersonX doesn't keep a clean and tidy house like PersonY because _ is a slob.",
      "PersonX got onto PersonY about how to clean despite the fact _ never helped clean the place.",
      "PersonX had nice clean hair but PersonY did not. This was due to _ having plenty of hot water.",
      "PersonX had to clean the house for PersonY after _ threw up and made a mess.",
      "PersonX hated having to clean and tidy the house while PersonY loved it. _ had always been a messy person.",
      "PersonX helped PersonY clean her bedroom prior to a cousin visiting, as _ was small and unable to move the bed.",
      "PersonX helped PersonY to clean hear house, since _ was a master at housekeeping skills.",
      "PersonX is a bit of a clean freak while PersonY is dirty, so _ has the better looking house.",
      "PersonX is clean and tidy, PersonY is not _ is likely to thoroughly clean her dishes.",
      "PersonX is known as a neat freak, unlike PersonY, because _ is obsessed with keeping things clean.",
      "PersonX is paying PersonY to clean out their house, so _ is more likely the home owner.",
      "PersonX liked to keep the house clean, but PersonY was filthy, so _ was always wiping things up.",
      "PersonX liked to keep their house clean but PersonY did not as _ was really very untidy.",
      "PersonX likes to clean everything in sight, while PersonY does not because _ is very organized.",
      "PersonX loved to clean house but PersonY wasn't really enthused about doing it. _ would never hire a housekeeper to help them.",
      "PersonX made sure that they items were meticulously clean before putting them on display but PersonY did not as _ was very houseproud.",
      "PersonX makes sure to clean his teeth every day unlike PersonY because _ is very hygienic.",
      ]
  },
```


Figure 12: A sample of SUPERNI dataset.

{"id": "1_0_0","question": "I was in a collision the other day. What should I check?",

"profile": [

{"id": "1_0_0_0", "text": "Getting a dura ace 7800 derailleur to work with 10 speeds I have a bike set up with a dura ace 7800 derailleur, specifically listed as a 9 or 10 speed model. I am interested in getting it working in the 10-speed configuration. Two things:I remember reading somewhere that you have to rout the cable a certain way to get it to work with 10 speeds, but I can't remember how to do so.I can't get it to rest in the bottom gear even when the limit screw is all the way out and the cable isn't attached. I can easily force it into place, though. I am considering taking out a spacer from behind my cassette that the instructions said I would be needing.Any advice?","category": "bicycles"},

{"id": "1_0_0_1", "text": "Is there a good resource for helping to discern the cause of pain / soreness? I went for my longest ride of the year this past Sunday. I guess my body wasn't expecting it or my bike is slightly misconfigured. I'm interested in finding a resource where I can look up what I might have done wrong based on where I am experiencing pain. Does such a resource exist?FWIW, I had serious stiffness and pain between my right calf muscle and right ankle.","category": "bicycles"},

{"id": "1_0_0_2","text": "How bad is cross chaining? Accepted wisdom is that cross chaining is bad and that we should never do it. However, most of us set up our bikes so that we can run any gear combination. Invariably, I'll occasionally look down and see that I'm running my big chainring against one of the bigger cogs.Is there any measurement of how much faster equipment will wear out from cross chaining, or energy loss or whatever?","category": "bicycles"},

{"id": "1_0_0_3","text": "Is there any functional point of using alloy jockey wheels? Jockey wheels seem like the most pointless thing to upgrade on a bike. I can see swapping out the usual grey ceramic jockey wheels for an anodized alloy one to match the look of the rest of the bike, but I can't see any possible performance related reason to do so.Do they aid in shifting quality at all? I know that dura-ace and ultegra derailleurs use jockey wheels with bearings instead of bushings to improve durability, but they still use ceramics.Is there any benefit of using the alloy jockey wheels?", "category": "bicycles" },],

"rubric_aspects": [

{"aspect": "Cause of the collision",

"evidence": "I was making a left hand turn the other day when a u-turning vehicle t-boned me.",

"reason": "Understanding the cause of the collision helps determine potential damage and liability."},

{ "aspect": "Damage to the bike",

"evidence": "My back wheel was tacoed, and the driver bought me a replacement.Now that I've had some time to install the replacement, I'm noticing that either my derailleur or the hanger is bent. It looks like it's the hanger, but is there a way to tell?",

"reason": "The user needs to know the extent of the damage to their bike to assess repair needs and costs."},

{"aspect": "Inspection of the frame",

"evidence": "I've cleaned up the frame looking for cracks and it all looks good. It's a pretty robust aluminum frame, so I'm not surprised.",

"reason": "The user has already checked for frame damage, so the response should focus on other potential issues."}],

"narrative": "I was making a left hand turn the other day when a u-turning vehicle t-boned me.I somehow managed to land on my feet, but the bike didn't fare as well. It landed on the drive side.My back wheel was tacoed, and the driver bought me a replacement.Now that I've had some time to install the replacement, I'm noticing that either my derailleur or the hanger is bent. It looks like it's the hanger, but is there a way to tell?What else should I check? I've cleaned up the frame looking for cracks and it all looks good. It's a pretty robust aluminum frame, so I'm not surprised. Is there anything else that may be wrong?","category": "bicycles"

},

Figure 13: A sample of LaMP dataset.