DKME: Rethinking Coupled Knowledge Memory for Lifelong Model Editing of Large Language Models

Guanyu Zheng[1] **Zhenyu Wang**[1†] **Tingting He**[2] **Xv Wang**[1]
Haochang Wang[2] **Yaokai Huang**[1] **Tiejun Zhao**[3]
[1]School of Software, South China University of Technology
[2]School of Computer and Information Technology, Northeast Petroleum University
[3]Faculty of Computing, Harbin Institute of Technology
{sezgyhtt, 202210188804, seykhuang}@mail.scut.edu.cn wangzy@scut.edu.cn
208002070412@stu.nepu.edu.cn kinghaosing@gmail.com tjzhao@hit.edu.cn

Abstract

Lifelong knowledge editing aims to inject a stream of factual updates into large language models (LLMs) without retraining, yet existing memory-based editors often suffer from catastrophic forgetting as edits accumulate. We argue that a key factor is the coupled knowledge memory mechanism, where addressing (routing) and storage (writing via memory-module updates) are entangled. This entanglement makes it difficult to confine the effects of each edit to its intended scope, particularly in multi-domain and associated-fact editing streams, where updates either span diverse semantic domains or repeatedly modify related attributes of the same subject. Consequently, updating memory for one edit inadvertently alters the routing and stored representations of previously injected edits, leading to catastrophic forgetting as edits accumulate. We propose **DKME**, which decouples addressing from storage via two stages: decoupled semantic addressing learns a fact-aware manifold for scope-aware routing, and partitioned memory storage localizes edits to memory partitions identified by unsupervised clustering in the embedding space. Experiments on three benchmarks, including HalluEditBench, CKnowEdit, and WikiData$_{\text{counterfact}}$, demonstrate that DKME consistently achieves a more favorable trade-off between editing success and locality compared to baselines, while maintaining more stable performance as the edit scale increases.

1 Introduction

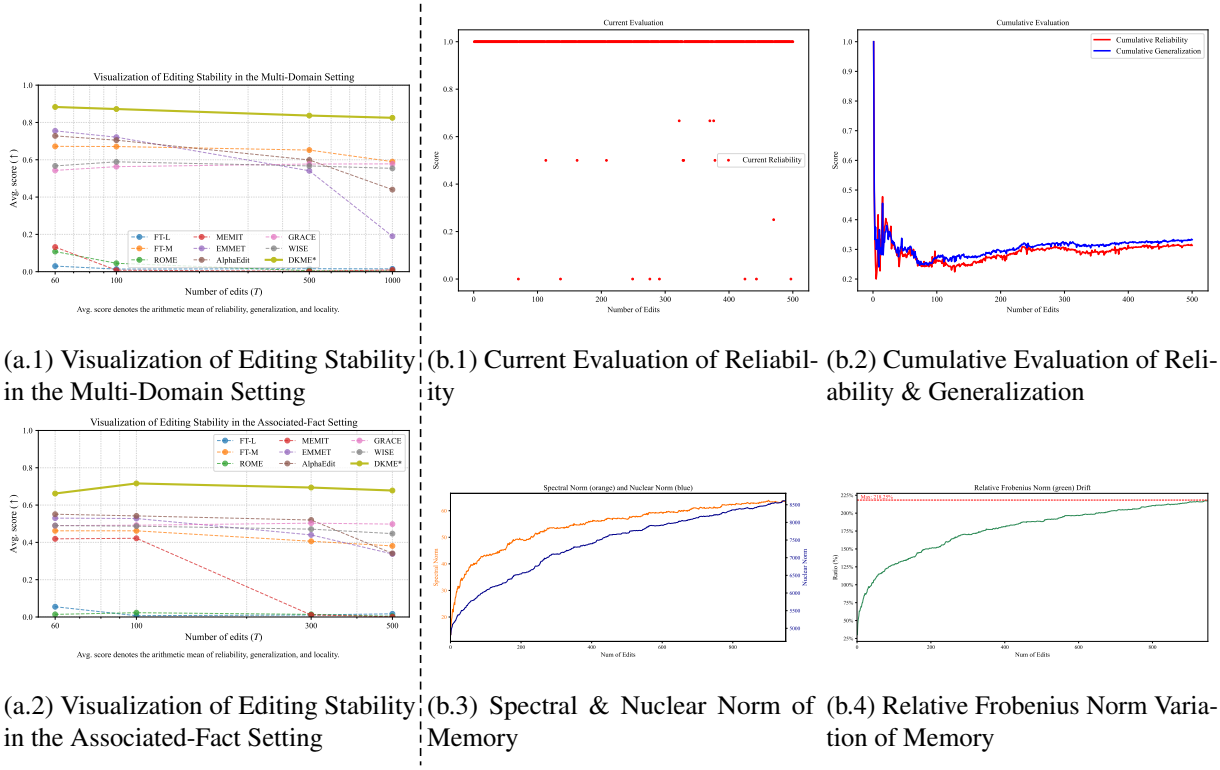
Knowledge in the real world continuously evolves, requiring large language models (LLMs) (Achiam et al., 2023; Guo et al., 2025) to be updated over time to remain reliable and up-to-date (Zhang et al., 2024; Wu et al., 2024; Li and Chu, 2025). Lifelong knowledge editing addresses this need by injecting a stream of factual updates into a pretrained LLM without retraining or finetuning (Hartvigsen et al., 2023; Wang et al., 2024a; Thede et al., 2025). Recent memory-based editors incorporate pluggable side modules to avoid directly modifying original model parameters, thereby improving safety (Zhang et al., 2024; Wang et al., 2024a). However, they often struggle to remain stable under complex edit streams (Lin et al., 2024). As the number of edits grows, these methods can exhibit a substantial performance decline on earlier edits, commonly referred to as catastrophic forgetting (Wang and Li, 2024; Shi et al., 2025). In this work, we evaluate lifelong editors using three fundamental metrics (Wang et al., 2024a; Zhang et al., 2024; Wang et al., 2024c; Yao et al., 2023; De Cao et al., 2021; Tan et al., 2021): reliability, generalization, and locality, which capture edit success, generalization to equivalent queries, and preservation of unrelated knowledge, respectively.

Empirically, we observe that existing methods exhibit pronounced instability under two representative editing streams. **(1) Multi-domain editing:** performance decays when successive edits rapidly switch across diverse semantic domains (Figure 1(a.1)), a typical pattern in hallucination correction tasks (Hartvigsen et al., 2023; Fang et al., 2025a; Huang et al., 2024a). **(2) Associated-fact editing:** performance degrades when repeatedly editing multiple attributes of the same subject (Zhang et al., 2024; Cohen et al., 2024), as cross-edit interference accumulates with edit scale (Figure 1(a.2)).

We argue that these failures largely stem from *memory entanglement*. Existing memory-based editors typically couple knowledge addressing (deciding whether and which memory to activate) with knowledge storage (writing new knowledge by updating the memory-module parameters) in a shared mechanism (Hartvigsen et al., 2023; Wang et al., 2024a). Under lifelong editing, this coupling

†Corresponding Author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 16128–16150



(a.1) Visualization of Editing Stability in the Multi-Domain Setting

(b.1) Current Evaluation of Reliability

(b.2) Cumulative Evaluation of Reliability & Generalization

(a.2) Visualization of Editing Stability in the Associated-Fact Setting

(b.3) Spectral & Nuclear Norm of Memory

(b.4) Relative Frobenius Norm Variation of Memory

Figure 1: **Left**: Editing stability under two representative lifelong editing streams, measured by **Avg.** as the edit scale T increases. (a.1) Multi-domain setting, where the edit stream rapidly switches across semantic domains. (a.2) Associated-fact setting, where multiple fact updates share the same subject. **Avg.** is the arithmetic mean of reliability, generalization, and locality. **Right**: Motivating results of memory-based editor in a lifelong setting. (b.1) **Current Evaluation** reports reliability on the latest edit. (b.2) **Cumulative Evaluation** aggregates reliability and generalization over all edits up to the current step. (b.3) **Spectral & Nuclear Norm** tracks the growth of the memory-module weight norms. (b.4) **Relative Frobenius Norm Variation** measures cumulative deviation from the initial memory weights. Together, the increasing weight deviation and norm growth coincide with degraded cumulative performance, indicating escalating cross-edit interference as edits accumulate. Further details in Appendix C.1.

leads to progressive weight deviation in the memory module (Figure 1(b.3)-(b.4)). Consequently, the latest edit can achieve high current accuracy (Figure 1(b.1)), cumulative accuracy over all past edits steadily degrades as edits accumulate (Figure 1(b.2)), indicating growing cross-edit interference and forgetting.

To mitigate memory entanglement, we propose **DKME** (**D**ecoupled **K**nowledge **M**emory **E**diting), a two-stage mechanism that decouples knowledge addressing from knowledge storage, separating routing decisions from parameter updates. Concretely, **Stage I Decoupled Semantic Addressing** learns an independent semantic manifold for scope-aware routing, deciding whether a query falls within the scope of existing edits and which neighborhood it should be associated with. **Stage II Partitioned Memory Storage** then leverages this manifold to partition edited knowledge into localized memory modules, localizing updates to a selected partition. By decoupling semantic addressing from knowledge storage and localizing updates to partitioned memory modules, DKME effectively mitigates cross-edit interference and enhances stability under both multi-domain and associated-fact editing scenarios. Our contributions are as follows:

- We identify memory entanglement in memory-based editors as a key factor underlying instability in lifelong knowledge editing, particularly under multi-domain and associated-fact edit streams.
- We introduce **DKME**, a decoupled knowledge memory method that separates semantic addressing from memory updates and localizes edits to partitioned memory modules to mitigate cross-edit interference.
- We conduct extensive experiments on three benchmarks under lifelong editing settings, showing that DKME consistently outperforms

strong baselines in terms of reliability, generalization, and locality.

2 Related Work

2.1 Lifelong Knowledge Editing

Lifelong knowledge editing updates LLMs with a stream of factual edits while balancing reliability, generalization, and locality (Wang et al., 2024a; Wang and Li, 2024; Wang et al., 2024c). A key challenge is mitigating cross-edit interference and catastrophic forgetting as edits accumulate (Wang and Li, 2024). To address this challenge, prior work mainly follows two paradigms: **locate-then-edit** and **memory-based** approaches, which differ primarily in whether the original model parameters are modified (Yao et al., 2023). The locate-then-edit paradigm identifies model components associated with target facts and directly modifies their parameters to inject new knowledge (Meng et al., 2022, 2023; Fang et al., 2025b; Gupta et al., 2024; Dong et al., 2025; Li et al., 2025; Jiang et al., 2025). These methods commonly presume that multi-layer perceptron (MLP) layers are the central modules for knowledge storage within the model (Dai et al., 2022; Geva et al., 2021; Dong et al., 2025; Li et al., 2025). Additionally, some research has also examined the contribution of attention heads to knowledge propagation (Geva et al., 2023; Gould et al., 2024; Jiang et al., 2024). However, under continual editing, such methods can induce cross-edit interference and degradation of earlier edits (Hu et al., 2025). Memory-based approaches store updates in auxiliary memory modules, such as codebooks (Hartvigsen et al., 2023), neurons (Wang et al., 2024a; Wang and Li, 2024; Huang et al., 2023), or auxiliary models (Mitchell et al., 2022). GRACE (Hartvigsen et al., 2023) constructs a discrete key-value codebook and maintains it through key operations, including adding, expanding, and splitting. WISE (Wang et al., 2024a) proposes a side-memory architecture with margin-based finetuning (Chen et al., 2020) and a sharding-and-merging strategy to reduce catastrophic forgetting in continual model editing.

2.2 Knowledge Memory in LLMs

LLM memory is commonly categorized into working memory and long-term memory (Wang et al., 2024a). Working memory is typically activated during inference, with in-context learning (ICL) (Wei et al., 2022) serving as a representative paradigm. However, controllability can be limited when models fail to faithfully follow the provided context (Huang et al., 2025; Hu et al., 2024). Recent studies improve ICL controllability (Li et al., 2023; Chen et al., 2024). Long-term memory is encoded in parameters and updated via retraining or finetuning (Radford et al., 2018; Hu et al.), which is costly (Achiam et al., 2023; Guo et al., 2025) and may suffer from overfitting and forgetting (Tirumala et al., 2022; Luo et al., 2023). Knowledge editing provides a more data-efficient alternative, spanning retrieval-based editors (e.g., SERAC, GRACE) (Mitchell et al., 2022; Hartvigsen et al., 2023) and parameter-localization editors (e.g., MEMIT, AlphaEdit) (Meng et al., 2023; Fang et al., 2025b). WISE (Wang et al., 2024a) further constrains the retrieval and updating of working memory under a long-term memory framework. In this context, we study parametric knowledge memory for lifelong editing and propose a decoupled design that separates addressing from storage to reduce cross-edit interference and improve long-term stability.

3 Problem Definition

Following Yao et al. (2023), let f_θ denote an LLM parameterized by θ. Starting from the pre-edit model θ_0 (pretrained on a large-scale corpus D_{train}), we consider a sequence of edits $\{(x_t, y_t)\}_{t=1}^{T}$, where each (x_t, y_t) denotes a query–answer pair at time step t. Let $D_{\text{edit}}^{\le t} = \{(x_i, y_i)\}_{i=1}^{t}$ denote the accumulated edit set up to time t. At time step t, the editor KE takes as input the current edit (x_t, y_t) and the LLM $f_{\theta_{t-1}}$ from time step $t-1$, and produces an updated LLM f_{θ_t}, formally defined as:

$$f_{\theta_t} \leftarrow \mathrm{KE}\big(f_{\theta_{t-1}}, x_t, y_t\big)\,,$$
$$\text{where} \quad f_{\theta_t}(x) = \begin{cases} y, & \text{if } \exists (x,y) \in D_{\text{edit}}^{\le t}, \\ f_{\theta_0}(x), & \text{otherwise.} \end{cases} \tag{1}$$

Eq. (1) specifies the desired behavior of the post-edit LLM. Based on the above settings, an effective editor KE should satisfy the criteria of three fundamental properties (Wang et al., 2024a; Zhang et al., 2024; Wang et al., 2024c; Yao et al., 2023; De Cao et al., 2021; Tan et al., 2021): reliability (Rel.; correctness on edit prompts); generalization (Gen.; correctness on paraphrases x_t'); and locality (Loc.; invariance on unrelated prompts x_t^{loc}), defined as follows:

$$\text{Rel.} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}(f_{\theta_t}(x_t) = y_t) \tag{2}$$

$$\text{Gen.} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\big(f_{\theta_t}(x'_t) = y_t\big) \tag{3}$$

$$\text{Loc.} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\Big(f_{\theta_t}(x_t^{\text{loc}}) = f_{\theta_0}(x_t^{\text{loc}})\Big) \tag{4}$$

Here, $\mathbb{1}(\cdot)$ is the indicator function. We also report their arithmetic mean (**Avg.**) as an overall trade-off score.

4 Methodology

Motivation DKME is motivated by memory entanglement in lifelong memory-based editors, where knowledge addressing (routing) and storage (writing via parameter updates) are coupled, causing routing decisions and updates to overlap across edits and thereby amplifying cross-edit interference. As summarized in Table 1, representative editors (e.g., WISE (Wang et al., 2024a), GRACE (Hartvigsen et al., 2023)) adopt coupled architectures with shared or discretized storage, which becomes particularly unstable under multi-domain and associated-fact edit streams. To mitigate this, DKME decouples addressing from storage via two stages: **Decoupled Semantic Addressing** for scope-aware routing on a discriminative manifold, and **Partitioned Memory Storage** for localizing updates to selected memory partitions. Figure 2 provides an overview of DKME.

Method	Arch.	Addr.	Storage	Manif.
WISE	Coupled	△	Shared	△
GRACE	Coupled	×	Discrete	×
DKME	Decoupled	✓	Partitioned	✓

Table 1: Comparison of current memory-based knowledge editing methods. **Arch.**: architecture; **Addr.**: addressing mechanism; **Storage**: storage scheme; **Manif.**: manifold awareness. ✓, ×, and △ indicate supported, not supported, and partially supported, respectively.

4.1 Stage I: Decoupled Semantic Addressing

Stage I learns a discriminative manifold for precise knowledge addressing. In lifelong editing, semantic boundaries can be blurred by memory entanglement: queries from different domains, or different facts about the same entity, may share overlapping lexical and semantic features. To address this issue, we introduce **Decoupled Semantic Addressing (DSA)**, which learns an embedding space dedicated to scope-aware routing—deciding whether a query falls within the scope of existing edits and, if so, which neighborhood it should be associated with.

Fact-aware addressing manifold. We adopt Sentence-BERT (SBERT; Reimers and Gurevych, 2019) as the base encoder and denote it by $g(\cdot)$. To adapt the embedding space to the needs of knowledge editing, particularly to separate semantically proximate yet factually distinct queries, we finetune g with a metric-learning objective.

Triplet construction. To finetune $g(\cdot)$, we construct triplets from a held-out subset drawn from the same benchmarks but disjoint from the evaluation edits, ensuring that the router is trained without using any test instances. For each anchor query x^{anc} in this subset, positives x^{pos} come from its paraphrases and portability prompts (when available) that refer to the same underlying fact. Hard negatives x^{neg} are drawn from the associated locality set of x^{anc}, which provides semantically related but fact-mismatched queries (e.g., the same subject with a different relation/object, or a different subject), encouraging $g(\cdot)$ to sharpen scope boundaries for routing.

We then optimize the triplet loss:

$$\mathcal{L}_{\text{dis}} = \max\Big(0,\ \|g(x^{\text{anc}}) - g(x^{\text{pos}})\|_2^2 - \|g(x^{\text{anc}}) - g(x^{\text{neg}})\|_2^2 + \alpha\Big), \tag{5}$$

where α is a margin. This objective forms compact neighborhoods for queries expressing the same edited fact, while pushing away semantically similar but out-of-scope queries, yielding an addressing manifold that is sensitive to factual boundaries for routing. See Appendix C.2 for visualizations.

Scope-aware routing with a threshold. At inference time, for an incoming query x, we compute $g(x)$ and measure cosine similarities to the embeddings of historical edit queries $\{g(x_i)\}$. We define

$$s_{\max}(x) = \max_i \cos\big(g(x), g(x_i)\big) \tag{6}$$

as the maximum similarity. A threshold τ then performs scope-aware routing:

- if $s_{\max}(x) \geq \tau$, x is *in-scope* and forwarded to Stage II for partition selection;

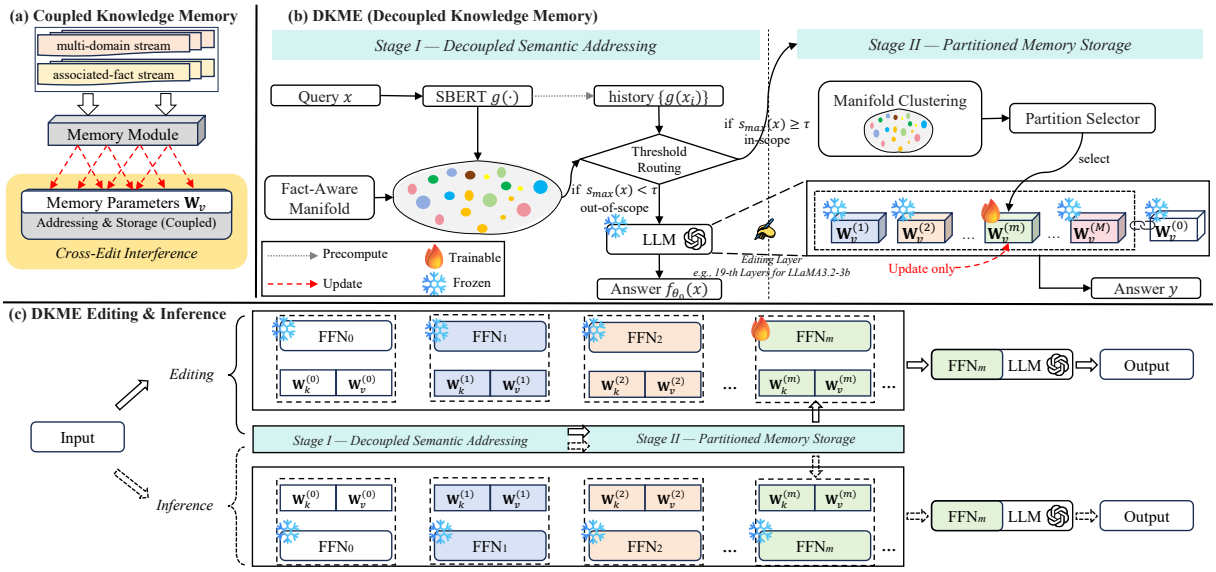


Figure 2: Overview of DKME. **(a)** Prior memory-based editors couple routing and writing in the memory module, causing cross-edit interference under multi-domain or associated-fact streams. **(b)** DKME decouples routing from writing in two stages. Stage I computes a similarity score $s_{\max}(x)$ on a fact-aware manifold and applies a threshold τ for scope-aware routing. Stage II performs manifold clustering to select a partition m and updates only the corresponding memory parameters (e.g., $\mathbf{W}_v^{(m)}$), leaving the backbone and other partitions frozen. **(c)** DKME applies the same semantic addressing and memory partition selection at both the editing and inference phases, ensuring consistent activation and localized updates.

- otherwise, x is *out-of-scope* and answered by the frozen backbone without invoking the editing modules.

DSA thus serves as a semantic router decoupled from the backbone parameters: it routes based on proximity in a fact-calibrated addressing space, rather than raw surface-level similarity. Note that DSA does not verify factual correctness; instead, the hard negatives make the manifold more discriminative for separating semantically similar but factually different edit scopes, and τ provides a controllable knob for scope selection. We analyze sensitivity to τ in Section 5.4.

4.2 Stage II: Partitioned Memory Storage

Once a query is identified as in-scope, the remaining challenge is to prevent cross-edit interference, especially under multi-domain and associated-fact edit streams. Stage II introduces **Partitioned Memory Storage (PMS)**, which maps in-scope queries to isolated parameter subspaces so that updates for different knowledge groups are localized to different memory partitions.

FFNs as parametric knowledge memories. Following prior findings that FFN blocks in LLMs function as parametric key–value memories (Geva et al., 2021; Qiu et al., 2024), we implement partitioned storage by duplicating and specializing the value-side parameters of selected FFN layers. Let $\mathbf{h} \in \mathbb{R}^d$ denote the input to a chosen FFN layer (i.e., the attention output (Geva et al., 2021; Kobayashi et al., 2025)). For exposition, we use a simplified form to highlight the partitioned value parameters:

$$\mathrm{FFN}(\mathbf{h}) = \sigma\big(\mathbf{h}^\top \mathbf{W}_k\big)\, \mathbf{W}_v, \tag{7}$$

where $\mathbf{W}_k$ and $\mathbf{W}_v$ denote key- and value-side matrices, and $\sigma(\cdot)$ is the activation (e.g., SwiGLU).

We keep the original value parameters $\mathbf{W}_v^{(0)}$ frozen to preserve pretrained knowledge, and introduce trainable value matrices $\{\mathbf{W}_v^{(1)}, \ldots, \mathbf{W}_v^{(M)}\}$ as memory partitions. Given a partition index $m \in \{1, \ldots, M\}$, the partitioned FFN output is

$$\mathrm{FFN}^{(m)}(\mathbf{h}) = \sigma\big(\mathbf{h}^\top \mathbf{W}_k\big)\, \mathbf{W}_v^{(m)}. \tag{8}$$

During both editing and inference, each in-scope query activates only one partition $\mathbf{W}_v^{(m)}$, localizing gradient updates to a small subset of the value space and thereby limiting interference across edits.

Manifold-based partitioning. To automatically determine which edits should share a partition, we perform unsupervised clustering on the addressing manifold learned in Stage I. Concretely,

we first apply UMAP (McInnes et al., 2018) to project the SBERT embeddings $\{g(x_i)\}$ to a lower-dimensional space while approximately preserving neighborhood structure, and then run the density-based clustering algorithm HDBSCAN (McInnes et al., 2017) to discover coherent clusters without pre-specifying the number of partitions.

Let $\mathcal{C}_1, \ldots, \mathcal{C}_M$ denote the clusters returned by HDBSCAN (optionally merging clusters below a small-size threshold). We establish a one-to-one correspondence by assigning each cluster $\mathcal{C}_m$ to a dedicated value matrix $\mathbf{W}_v^{(m)}$. For an in-scope query x, we assign it to a cluster (e.g., via HDBSCAN labeling for stored edits, and nearest-centroid/nearest-neighbor assignment in the projected space for new queries), and activate the corresponding partition $\mathbf{W}_v^{(m)}$.

By confining updates for queries in $\mathcal{C}_m$ to $\mathbf{W}_v^{(m)}$, PMS encourages edits from different clusters to occupy disjoint parameter subspaces. As a result, cross-partition interference is reduced, and DKME maintains more stable performance as the edit scale increases under both multi-domain and associated-fact settings (Figure 1(a.1)–(a.2)).

5 Experiments

In this section, we conduct experiments to answer the following research questions: **RQ1:** How does DKME compare with baselines under multi-domain and associated-fact settings? **RQ2:** How do different stages of DKME contribute to its overall performance? **RQ3:** How sensitive is DKME to different hyper-parameter settings?

5.1 Experimental Settings

Datasets and Models For the multi-domain setting, we conduct experiments on **HalluEditBench** (Huang et al., 2024a) and **CKnowEdit** (Fang et al., 2025a). HalluEditBench comprises real-world hallucinations spanning nine diverse domains and is used to evaluate DKME's capability in correcting factual hallucinations. Following Huang et al. (2024a), we select hallucinations generated by LLaMA3-8B-Instruct across all nine domains as the editing data. Similarly, CKnowEdit encompasses various types of Chinese linguistic knowledge. Following Fang et al. (2025a), we select three types, namely *Classical Chinese*, *Pinyin*, and *Logic Error*, that include locality queries to evaluate the trade-offs involved in DKME. For the associated-fact setting, we conduct experiments on **WikiData$_{\text{counterfact}}$** (Zhang et al., 2024) to evaluate the ripple effect of knowledge editing. Based on the language of each dataset, we select representative mainstream autoregressive LLMs for evaluation: LLaMA3.2-3B for the English datasets HalluEditBench and WikiData$_{\text{counterfact}}$, and Qwen2.5-1.5B for the Chinese dataset CKnowEdit. See Appendix B.1 for dataset statistics and editing examples, and Appendix B.3 for implementation details.

Evaluation Following Wang et al. (2024a), we adopt an exact match to evaluate the baselines and DKME. As described in Section 3, all three benchmarks adopt **Rel.**, **Gen.**, **Loc.**, and **Avg.** as evaluation metrics. In particular, to ensure a fair evaluation in the multi-domain setting (HalluEditBench & CKnowEdit), we employ an average sampling strategy to distribute edit instances across domains. Specifically, given an edit instance x_i and the total number of edits T, we assign x_i to the $(i \bmod K) + 1\}$-th domain group as follows:

$$x_i \rightarrow \mathcal{G}_{(i \bmod K)+1}, \quad i = 0, 1, ..., (T-1) \quad (9)$$

Here, K denotes the total number of domains in the benchmark, and $x \rightarrow (\cdot)$ presents the sample reflection between the edit x and the domain $\mathcal{G}$.

Baselines The baselines include three categories with a total of eight methods: **(1) Fine-tuning**: FT-L (Meng et al., 2022), FT-M (Zhang et al., 2024). **(2) Locate-then-edit**: ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), EMMET (Gupta et al., 2024), AlphaEdit (Fang et al., 2025b). **(3) Memory-based**: GRACE (Hartvigsen et al., 2023), WISE (Wang et al., 2024a). Details on all comparisons are found in Appendix B.2.

5.2 Main Results (RQ1)

5.2.1 Performance on Multi-Domain Editing

Obs 1: DKME achieves superior and consistently stable performance in multi-domain settings. As shown in Tables 2 and 3, DKME achieves the best and most stable overall trade-off across both benchmarks. On HalluEditBench, DKME consistently attains the highest Avg. scores under all editing scales $T \in \{60, 100, 500, 1000\}$, exhibiting only a mild degradation as T increases, which indicates strong long-term editing stability. In contrast, EMMET and AlphaEdit remain competitive at small editing scales but suffer from pronounced performance degradation under large-scale edits, while ROME and MEMIT

	$T = 60$				$T = 100$				$T = 500$				$T = 1000$			
	Rel.↑	Gen.↑	Loc.↑	Avg.↑	Rel.↑	Gen.↑	Loc.↑	Avg.↑	Rel.↑	Gen.↑	Loc.↑	Avg.↑	Rel.↑	Gen.↑	Loc.↑	Avg.↑
FT-L	0.064	0.025	0.000	0.030	0.037	0.005	0.000	0.014	0.027	0.020	0.000	0.016	0.023	0.021	0.000	0.015
FT-M	0.925	0.872	0.220	0.672	0.927	0.865	0.222	0.671	0.896	**0.802**	0.259	0.652	0.819	0.711	0.241	0.590
ROME	0.146	0.146	0.031	0.108	0.073	0.057	0.004	0.045	0.007	0.007	0.001	0.005	0.006	0.004	0.001	0.004
MEMIT	0.141	0.209	0.046	0.132	0.000	0.000	0.023	0.008	0.001	0.002	0.004	0.002	0.000	0.000	0.026	0.009
EMMET	0.957	0.793	0.514	0.755	0.911	0.809	0.442	0.721	0.717	0.691	0.216	0.541	0.237	0.275	0.059	0.190
AlphaEdit	0.950	**0.904**	0.331	0.728	0.910	**0.875**	0.331	0.705	0.788	0.756	0.254	0.599	0.528	0.588	0.203	0.440
GRACE	0.329	0.300	**1.000**	0.543	0.371	0.319	**1.000**	0.563	0.401	0.329	**1.000**	0.577	0.402	0.331	**1.000**	0.578
WISE	0.542	0.350	0.810	0.567	0.604	0.365	0.799	0.589	0.464	0.348	0.888	0.567	0.412	0.340	0.910	0.554
DKME*	**0.969**	0.748	0.933	**0.883**	**0.956**	0.790	0.869	**0.872**	**0.926**	0.738	0.846	**0.837**	**0.923**	**0.718**	0.834	**0.825**

Table 2: Main editing results for multi-domain setting (HalluEditBench). **Bold** is the best result. ↑ Indicates that higher values are better. T: Num Edits. `Edit LLM: LLaMA3.2-3b`.

	$T = 60$				$T = 100$				$T = 200$				$T = 400$			
	Rel.↑	Gen.↑	Loc.↑	Avg.↑	Rel.↑	Gen.↑	Loc.↑	Avg.↑	Rel.↑	Gen.↑	Loc.↑	Avg.↑	Rel.↑	Gen.↑	Loc.↑	Avg.↑
FT-L	0.205	0.223	0.172	0.200	0.193	0.188	0.193	0.191	0.171	0.154	0.143	0.156	0.128	0.114	0.102	0.115
FT-M	0.656	0.611	0.148	0.472	0.611	0.559	0.183	0.451	0.583	0.536	0.184	0.434	0.573	0.507	0.208	0.429
ROME	0.029	0.024	0.016	0.023	0.042	0.044	0.051	0.046	0.055	0.056	0.045	0.052	0.026	0.034	0.016	0.025
MEMIT	0.517	0.504	0.568	0.530	0.481	0.441	0.540	0.487	0.310	0.332	0.375	0.339	0.104	0.112	0.115	0.110
GRACE	0.474	0.440	**1.000**	0.638	0.474	0.422	**1.000**	0.632	0.474	0.426	**1.000**	0.633	0.469	0.425	**1.000**	0.631
WISE	0.802	0.757	0.449	0.669	0.803	0.740	0.452	0.665	0.746	0.673	0.449	0.623	0.649	0.585	0.467	0.567
DKME*	**0.817**	**0.768**	0.491	**0.692**	**0.823**	**0.758**	0.476	**0.686**	**0.809**	**0.707**	0.455	**0.657**	**0.814**	**0.617**	0.472	**0.634**

Table 3: Main editing results for multi-domain setting (CKnowEdit). **Bold** is the best result. ↑ Indicates that higher values are better. T: Num Edits. `Edit LLM: Qwen2.5-1.5b`.

nearly fail when $T \geq 500$. A closer inspection of individual metrics reveals that DKME maintains very high Rel. (0.92–0.97) and strong Loc. (0.83–0.93), while preserving a competitive level of Gen., thereby achieving a more favorable balance among the three properties. By comparison, although GRACE reaches an ideal locality score (1.0), its Rel. and Gen. are severely constrained, resulting in inferior overall Avg. performance. Similarly, on CKnowEdit, DKME again achieves the highest Avg. under all $T \in \{60, 100, 200, 400\}$, while consistently maintaining the best Rel. and Gen. scores. Overall, these results demonstrate that DKME more effectively suppresses cross-domain interference in multi-domain settings, enabling a more robust trade-off among Rel., Gen., and Loc.

5.2.2 Performance on Associated-Fact Editing

Obs 2: DKME effectively mitigates the structural conflict between associated fact propagation and locality preservation under associated-fact editing. As shown in Table 4, DKME achieves the best overall performance across all editing scales $T \in \{60, 100, 300, 500\}$, indicating its ability to maintain a more robust balance among target fact injection, associated fact reasoning, and the preservation of unrelated knowledge. Specifically, DKME achieves Avg. scores of 0.662/0.716/0.694/0.678 at the four editing scales and retains a clear advantage even under the more challenging settings of $T = 300$ and $T = 500$. In contrast, AlphaEdit and EMMET exhibit relatively strong Rel./Gen. performance at small editing scales, but suffer from limited Loc. and degradation as T increases. WISE shows an opposite trend: although its Loc. improves with larger T, its Rel. and Gen. decline, constraining the Avg. score to approximately 0.447. While GRACE maintains an ideal Loc. of 1.0, its relatively low Rel./Gen. results in an Avg. of around 0.49, which is insufficient for effective associated knowledge transfer.

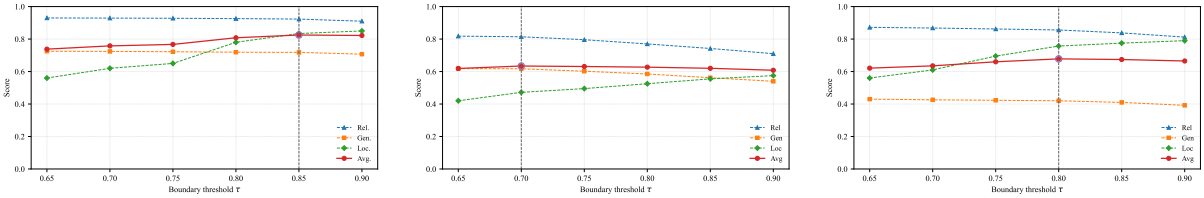
5.3 Ablation Study (RQ2)

Table 5 reports ablation results for DKME. Implementation details are in Appendix B.4.

Obs 3: Decoupled addressing with partitioned storage is crucial for stable lifelong editing. Removing DSA ("w/o DSA") causes a drastic locality drop, indicating that scope-aware addressing is essential to avoid unintended activation of edited memory on out-of-scope queries; weakening fact-aware addressing ("w/o $\mathcal{L}_{dis}$" or "w/ frozen SBERT") consistently degrades the overall trade-off. On the storage side, replacing PMS with shared storage ("w/o PMS") significantly impacts Rel./Gen., indicating that partitioned updates are crucial for mitigating cross-edit interference. Moreover, fixed or random partitioning ("w/ fixed partition", "w/ random partition assignment") and removing manifold reduction ("w/o UMAP") underperform DKME, supporting the importance of data-adaptive, manifold-aware partition discovery.

	$T = 60$				$T = 100$				$T = 300$				$T = 500$			
	Rel.↑	Gen.↑	Loc.↑	Avg.↑	Rel.↑	Gen.↑	Loc.↑	Avg.↑	Rel.↑	Gen.↑	Loc.↑	Avg.↑	Rel.↑	Gen.↑	Loc.↑	Avg.↑
FT-L	0.111	0.040	0.015	0.055	0.013	0.003	0.005	0.007	0.023	0.010	0.001	0.011	0.031	0.017	0.003	0.017
FT-M	0.829	0.520	0.038	0.462	0.749	0.544	0.092	0.462	0.687	0.478	0.053	0.406	0.629	**0.462**	0.053	0.381
ROME	0.026	0.010	0.005	0.014	0.048	0.021	0.000	0.023	0.032	0.010	0.000	0.014	0.009	0.001	0.001	0.004
MEMIT	0.619	0.257	0.381	0.419	0.674	0.295	0.297	0.422	0.015	0.006	0.012	0.011	0.000	0.000	0.002	0.001
EMMET	0.887	0.406	0.297	0.530	0.863	0.434	0.288	0.528	0.734	0.404	0.182	0.440	0.571	0.297	0.145	0.338
AlphaEdit	**0.915**	**0.533**	0.206	0.551	0.921	**0.547**	0.158	0.542	0.892	**0.507**	0.162	0.520	0.572	0.294	0.157	0.341
GRACE	0.269	0.200	**1.000**	0.490	0.262	0.214	**1.000**	0.492	0.298	0.212	**1.000**	0.503	0.290	0.205	**1.000**	0.498
WISE	0.741	0.493	0.237	0.490	0.632	0.334	0.496	0.487	0.447	0.373	0.594	0.471	0.276	0.244	0.820	0.447
DKME*	0.914	0.349	0.723	**0.662**	**0.922**	0.469	0.757	**0.716**	**0.904**	0.457	0.722	**0.694**	**0.856**	0.420	0.757	**0.678**

Table 4: Main editing results for associated-fact setting ($\text{WikiData}_{\text{counterfact}}$). **Bold** is the best result. ↑ Indicates that higher values are better. T: Num Edits. `Edit LLM: LLaMA3.2-3b`.



(a) HalluEditBench, $T = 1000$ (b) CKnowEdit, $T = 400$ (c) $\text{WikiData}_{\text{counterfact}}$, $T = 500$

Figure 3: Sensitivity of DKME to the boundary threshold τ across three benchmarks. The optimal Avg. trade-off is achieved at different τ values for different datasets.

Finally, alternative routers ("w/ MLP router", "w/ distance-constrained router") are inferior to the default clustering-based routing, with the hard distance constraint harming locality.

Variant	Rel.↑	Gen.↑	Loc.↑	Avg.↑	ΔAvg.
DKME	**0.923**	**0.718**	**0.834**	**0.825**	–
Ablations on Stage I: Decoupled Semantic Addressing					
w/o DSA	0.885	0.698	0.276	0.620	0.205
w/o $\mathcal{L}_{dis}$	0.906	0.701	0.639	0.749	0.076
w/ frozen SBERT	0.905	0.695	0.590	0.730	0.095
Ablations on Stage II: Partitioned Memory Storage					
w/o PMS	0.664	0.515	0.817	0.665	0.160
w/ fixed partition	0.911	0.704	0.813	0.809	0.016
w/ random partition assignment	0.700	0.540	0.790	0.677	0.148
w/o UMAP	0.890	0.670	0.795	0.785	0.040
Ablations on Router					
w/ MLP router	0.784	0.603	0.822	0.736	0.089
w/ distance-constrained router	0.753	0.659	0.326	0.579	0.246

Table 5: Ablation study with 1000 edits on HalluEditBench using LLaMA3.2-3B. **Bold** is the best result. ↑ indicates that higher values are better. ΔAvg. reports the performance drop relative to DKME, with Color intensity indicating larger drops via darker shades.

5.4 Sensitivity Analysis (RQ3)

Effect of τ We set τ to 0.85 on HalluEditBench, 0.80 on $\text{WikiData}_{\text{counterfact}}$, and 0.70 on CKnowEdit, achieving Avg. of 0.825/0.678/0.634 (Figure 3). The optimal τ varies across benchmarks: a higher τ applies stricter scope filtering (favoring Loc.), whereas a lower τ increases edit activation and may improve Rel./Gen. when in-scope queries exhibit weaker similarity to prior edits.

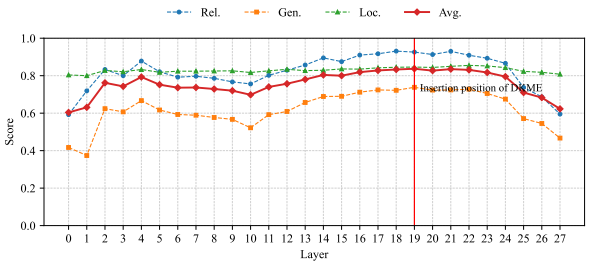


Figure 4: Performance of DKME on different layers of LLaMA3.2-3B under 500 edits. `Dataset: HalluEditBench`.

Localization Analysis Figure 4 reports DKME performance across different layers of LLaMA3.2-3B on HalluEditBench. Mid-to-upper layers (17-23) perform best, with layer 19 achieving the highest score. Lower layers show weaker editability, whereas upper layers incur more interference near the output, consistent with prior findings that factual knowledge is most effectively edited in mid-to-upper transformer blocks (Yao et al., 2024).

6 Conclusion

We study lifelong knowledge editing and identify memory entanglement in memory-based editors as a key factor contributing to catastrophic forgetting in multi-domain and associated-fact edit streams. We propose DKME, which decouples addressing from storage via fact-aware semantic addressing and partitioned memory storage. Experiments on

three benchmarks show that DKME improves the trade-off among reliability, generalization, and locality compared to baselines, and remains more stable as the edit scale increases.

Limitations

DKME trades additional complexity for improved stability in lifelong editing. Its partitioned memory introduces extra storage overhead and, under very long edit streams, the number of maintained partitions may become a scalability bottleneck, motivating future work on merging, pruning, or recycling partitions. DKME also relies on an external addressing encoder and manifold-based clustering, whose scope discrimination and partition assignment can be sensitive when edits are highly ambiguous or weakly separable; moreover, clustering behavior may vary across datasets and hyperparameter settings. Furthermore, our experiments focus on factual updates and primarily evaluate stability in terms of reliability, generalization, and locality; broader capabilities, such as multi-hop reasoning after edits, are not explicitly assessed. Finally, our evaluation focuses on a limited range of backbone sizes and architectures; extending validation to larger models and alternative architectures constitutes an important direction for future work.

Ethics Statement

Our research on lifelong knowledge editing and the proposed DKME method adheres to the ACL Ethics Policy. The primary objective of this work is to enhance lifelong editing performance in large language models. All experiments are conducted on publicly available benchmark datasets containing fact-based queries, without involving personally identifiable or private user data. We acknowledge that model editing techniques, if misused, may introduce harmful or misleading content or cause unintended behavioral changes. Therefore, their application should be guided by appropriate ethical considerations and safeguards to prevent misuse. Our commitment to accountability, responsible governance, and continuous ethical assessment underscores our dedication to maintaining high standards of integrity in the development and deployment of knowledge editing methods.

Acknowledgements

We thank the reviewers for their insightful comments. This work was supported in part by the National Natural Science Foundation of China (61402099, 61702093).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR.

Yingfa Chen, Zhengyan Zhang, Xu Han, Chaojun Xiao, Zhiyuan Liu, Chen Chen, Kuai Li, Tao Yang, and Maosong Sun. 2024. Robust and scalable model editing for large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 14157–14172.

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.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.

Zilu Dong, Xiangqing Shen, and Rui Xia. 2025. MEMIT-merge: Addressing MEMIT's key-value conflicts in same-subject batch editing for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 7952–7960.

Jizhan Fang, Tianhe Lu, Yunzhi Yao, Ziyan Jiang, Xin Xu, Huajun Chen, and Ningyu Zhang. 2025a. CKnowEdit: A new Chinese knowledge editing dataset for linguistics, facts, and logic error correction in LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8789–8807.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025b. Alphaedit: Null-space constrained model editing for language models. In *The Thirteenth International Conference on Learning Representations*.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Rhys Gould, Euan Ong, George Ogden, and Arthur Conmy. 2024. Successor heads: recurring, interpretable attention heads in the wild. In *The Twelfth International Conference on Learning Representations*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Akshat Gupta, Phudish Prateepamornkul, Maochuan Lu, Ahmed Alaa, Thomas Hartvigsen, and Gopala Anumanchipalli. 2025. Lifelong knowledge editing requires better regularization. *arXiv preprint arXiv:2502.01636*.

Akshat Gupta, Dev Sajnani, and Gopala Anumanchipalli. 2024. A unified framework for model editing. *Preprint*, arXiv:2403.14236.

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

Thomas Henn, Yasukazu Sakamoto, Clément Jacquet, Shunsuke Yoshizawa, Masamichi Andou, Stephen Tchen, Ryosuke Saga, Hiroyuki Ishihara, Katsuhiko Shimizu, Yingzhen Li, and 1 others. 2021. A principled approach to failure analysis and model repairment: Demonstration in medical imaging. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part III 24*, pages 509–518. Springer.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2025. Knowledge in superposition: Unveiling the failures of lifelong knowledge editing for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24086–24094.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Xiangkun Hu, Dongyu Ru, Lin Qiu, Qipeng Guo, Tianhang Zhang, Yang Xu, Yun Luo, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. Refchecker: Reference-based fine-grained hallucination checker and benchmark for large language models. *arXiv preprint arXiv:2405.14486*.

Baixiang Huang, Canyu Chen, Xiongxiao Xu, Ali Payani, and Kai Shu. 2024a. Can knowledge editing really correct hallucinations? *arXiv preprint arXiv:2410.16251*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Xiusheng Huang, Jiaxiang Liu, Yequan Wang, and Kang Liu. 2024b. Reasons and solutions for the decline in model performance after editing. In *Advances in Neural Information Processing Systems*, pages 68833–68853.

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

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*.

Yibo Jiang, Goutham Rajendran, Pradeep Ravikumar, and Bryon Aragam. 2024. Do llms dream of elephants (when told not to)? latent concept association and associative memory in transformers. *Advances in Neural Information Processing Systems*, 37:67712–67757.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2025. Analyzing feed-forward blocks in transformers through the lens of attention maps. In *The Twelfth International Conference on Learning Representations*.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2023. Large language models with controllable working memory. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1774–1793.

Qi Li and Xiaowen Chu. 2025. Adaedit: Advancing continuous knowledge editing for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4127–4149.

Xiaopeng Li, Shangwen Wang, Shasha Li, Shezheng Song, Bin Ji, Ma Jun, and Jie Yu. 2025. Rethinking residual distribution in locate-then-edit model editing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Bill Yuchen Lin, Sida I Wang, Xi Lin, Robin Jia, Lin Xiao, Xiang Ren, and Scott Yih. 2022. On continual model refinement in out-of-distribution data streams. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3128–3139.

Zihao Lin, Mohammad Beigi, Hongxuan Li, Yufan Zhou, Yuxiang Zhang, Qifan Wang, Wenpeng Yin, and Lifu Huang. 2024. Navigating the dual facets: A comprehensive evaluation of sequential memory editing in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13755–13772.

Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. 2021. Post-training quantization for vision transformer. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 28092–28103.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. 2022. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533.

Leland McInnes, John Healy, and Steve Astels. 2017. Hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861.

Christian HX Ali Mehmeti-Göpel and Michael Wand. 2024. On the weight dynamics of deep normalized networks. In *Proceedings of the 41st International Conference on Machine Learning*, pages 992–1007.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 17359–17372.

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

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.

Zihan Qiu, Zeyu Huang, Youcheng Huang, and Jie Fu. 2024. Empirical study on updating key-value memories in transformer feed-forward layers. In *The Second Tiny Papers Track at ICLR 2024*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and 1 others. 2018. Improving language understanding by generative pre-training.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using eiamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. 2019. Experience replay for continual learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 350–360.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. 2025. Continual learning of large language models: A comprehensive survey. *ACM Comput. Surv.*

Chenmien Tan, Ge Zhang, and Jie Fu. 2021. Massive editing for large language models via meta learning. In *The Twelfth International Conference on Learning Representations*.

Lukas Thede, Karsten Roth, Matthias Bethge, Zeynep Akata, and Thomas Hartvigsen. 2025. Wikibigedit: Understanding the limits of lifelong knowledge editing in llms. In *Forty-second International Conference on Machine Learning*.

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290.

Frederik Träuble, Anirudh Goyal, Nasim Rahaman, Michael Curtis Mozer, Kenji Kawaguchi, Yoshua Bengio, and Bernhard Schölkopf. 2023. Discrete

key-value bottleneck. In *International conference on machine learning*, pages 34431–34455. PMLR.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024a. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, and 1 others. 2024b. Easyedit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93.

Renzhi Wang and Piji Li. 2024. Lemoe: Advanced mixture of experts adaptor for lifelong model editing of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2551–2575.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024c. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 24824–24837.

Xin Wu, Yuqi Bu, Yi Cai, and Tao Wang. 2024. Updating large language models' memories with time constraints. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13693–13702.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. 2025. SD-loRA: Scalable decoupled low-rank adaptation for class incremental learning. In *The Thirteenth International Conference on Learning Representations*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240.

Zihan Yao, Yu He, Tianyu Qi, and Ming Li. 2024. Scalable model editing via customized expert networks. In *First Conference on Language Modeling (COLM)*.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, and 1 others. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Yinhe Zheng. 2022. Continuous qa learning with structured prompts. *arXiv preprint arXiv:2208.14602*.

Shaochen Zhong, Yifan Lu, Lize Shao, Bhargav Bhushanam, Xiaocong Du, Yixin Wan, Yucheng Shi, Daochen Zha, Yiwei Wang, Ninghao Liu, Kaixiong Zhou, Shuai Xu, Kai-Wei Chang, Louis Feng, Vipin Chaudhary, and Xia Hu. 2025. Mquake-remastered: Multi-hop knowledge editing can only be advanced with reliable evaluations. In *The Thirteenth International Conference on Learning Representations Processing*.

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

Appendix

In the Appendix, we introduce more details along with experimental setups, additional experimental results, and related works:

- **Appendix A:** More Related Works.
- **Appendix B:** More Experimental Setups.
- **Appendix C:** More Experimental Results.

A More Related Works

A.1 Evaluating Knowledge Editing

Recent advances have introduced a variety of benchmarks to assess the effectiveness of knowledge editing. KnowEdit (Zhang et al., 2024) provides a unified benchmark encompassing various knowledge editing tasks. HalluEditBench (Huang et al., 2024a) comprehensively evaluates editors on real-world hallucinations across nine domains. Moreover, complementary benchmarks such as CKnowEdit (Fang et al., 2025a), MQuAKE (Zhong et al., 2023, 2025) extend the evaluation to Chinese knowledge and multi-hop reasoning.

A.2 Continual Learning

Continual Learning (CL) aims to mitigate catastrophic forgetting when models acquire new knowledge (Shi et al., 2025), and continual fine-tuning has been widely studied as a means of iteratively updating LLMs (Lin et al., 2022). However, classical CL techniques such as Elastic Weight Consolidation (Kirkpatrick et al., 2017) and Experience

Replay (Rolnick et al., 2019) still incur degradation on earlier tasks when used to regularize fine-tuning (Wu et al., 2025). These limitations, together with the non-uniform distribution of edits, distinguish knowledge editing from conventional continual fine-tuning (Henn et al., 2021). Key–value memory methods, first introduced in computer vision (Liu et al., 2021) for their robustness to distributional shifts (Träuble et al., 2023), have since been adapted to question answering (Zheng, 2022) and hallucination (Hartvigsen et al., 2023), providing an effective mechanism for augmenting LLMs with external memory.

B More Experimental Setups

B.1 Datasets

Dataset statistics are summarized in Table 6.

HalluEditBench (Huang et al., 2024a) To address the fundamental gap in evaluating knowledge editing on actual hallucinated responses, Huang et al. (2024a) introduces HalluEditBench, a benchmark constructed from verified hallucinations of LLMs. Unlike prior datasets, it ensures the pre-edit model responses are hallucinated. The dataset is built from Wikidata by collecting factual triplets across 9 domains and 26 topics, and then identifying hallucinations from three LLMs. We select the LLaMA3-8b sub-dataset and preprocess the data such that each instance includes an edit statement (corresponding to the key `subject`, `question`, and `object` in HalluEditBench), generalization prompts (`paraphrased_question`, `reversed_relation_question`, `yes_question`, and `no_question`), and one locality prompt (`locality_question`). For locality, we evaluate by comparing the semantic similarity between the answers before and after editing. Figure 10 presents a single editing instance of HalluEditBench.

CKnowEdit (Fang et al., 2025a) To address the challenges LLMs face in handling Chinese-specific knowledge, Fang et al. (2025a) uses CKnowEdit, the first Chinese knowledge editing benchmark. It contains edits across seven knowledge types, covering linguistic, factual, and logical errors. The dataset targets difficult phenomena in Chinese, such as polyphony, antithesis, and logical structures, offering a comprehensive basis to evaluate and improve LLMs' Chinese knowledge editing capabilities. Due to the challenges involved in constructing valid locality sets for knowledge types such as Ancient Poetry, Proverbs, Idioms, and Facts (Fang et al., 2025a), locality annotations are not provided for these categories. Accordingly, we select three subsets of knowledge types—`classical_chinese_results_reviewed`, `phonetic_notation_results_reviewed`, and `ruozhiba`—and extract the fields `prompt`, `target_old`, `target_new`, `locality`, and `rephrase` to construct each data instance. Figure 11 presents a single editing instance of CKnowEdit.

WikiData$_{\text{counterfact}}$ (Zhang et al., 2024) Since tail entities are often not memorized by language models and are therefore unsuitable for evaluating knowledge editing, Cohen et al. (2024) focuses on factual triplets involving popular entities—where the subject appears among the top-viewed Wikipedia pages. The dataset contains manually altered object values to create counterfactual facts. It is widely used to evaluate whether editing methods can inject or override prominent factual knowledge. We construct each editing instance using the fields `subject`, `prompt`, `target_new`, `ground_truth`, `portability`, and `locality`. Figure 12 presents a single editing instance of WikiData$_{\text{counterfact}}$.

B.2 Baselines

B.2.1 FT-L (Meng et al., 2022)

Except for a single MLP layer, all other layers remain frozen. The MLP layer is fine-tuned using an autoregressive loss (Meng et al., 2022). To prevent the parameters from deviating excessively from the pretrained distribution, we further introduce an L_∞ norm constraint to effectively control the magnitude of parameter updates. This method uses the prediction of the last token to maximize the probability of all tokens in the target sequence, thereby deviating from the original fine-tuning objective.

B.2.2 FT-M (Zhang et al., 2024)

FT-M utilizes the same FFN layer as FT-L. During training, it masks the original text and computes the cross-entropy loss (Zhang et al., 2024) on the target answer, thereby aligning more closely with the objective of traditional fine-tuning.

B.2.3 ROME (Meng et al., 2022)

ROME is a parameter-editing technique that directly modifies the weights of a pretrained language model to inject or correct specific factual

Setting	Dataset	Language	Task	Model	*pre-edit*	T
Multi-Domain	HalluEditBench	English	Hallucination	LLaMA3.2-3B	0.312/0.331	1000
Multi-Domain	CKnowEdit	Chinese	Linguistics	Qwen2.5-1.5B	0.431/0.425	400
Associated-Fact	$\text{WikiData}_{\text{counterfact}}$	English	Counterfact	LLaMA3.2-3B	0.212/0.196	500

Table 6: Dataset statistics for main results. *pre-edit* reports the unedited model performance (Rel./Gen.). T denotes the number of edits.

knowledge. It operates by identifying a key-value pair within the model's feedforward layers—particularly in the MLP of the transformer block—corresponding to the factual association being edited. ROME locates the most influential layer and performs a rank-one update on the weights, effectively altering the model's internal representations without fine-tuning.

B.2.4 MEMIT (Meng et al., 2023)

MEMIT assumes that the FFN functions as a knowledge key-value store and directly modifies the parameters of specific layers via least-squares approximation. Unlike ROME, which updates a single layer, MEMIT is a multi-layer update algorithm that supports simultaneous editing of hundreds or even thousands of facts.

B.2.5 EMMET (Gupta et al., 2024)

EMMET is a newly proposed batched model editing method that aims to unify ROME and MEMIT. It efficiently injects new knowledge while preserving the model's original information by optimizing the preservation-memorization objective under equality constraints.

B.2.6 AlphaEdit (Fang et al., 2025b)

AlphaEdit is an efficient knowledge editing method for language models, which mitigates interference with preserved knowledge by projecting the editing perturbation into its null space. Built upon the locate-then-edit framework (Meng et al., 2022, 2023), it significantly improves editing performance by adding only a single line of projection code to existing methods.

B.2.7 GRACE (Hartvigsen et al., 2023)

GRACE adopts a discrete KEY-VALUE codebook and maintains it throughout editing by adding, expanding, and splitting KEYs. During inference, it retrieves the nearest KEY and determines whether to replace the activation of the hidden layer output.

	Hyperparameters	Values
SBERT Fine-tuning	batch_size	8
	margin	9
	learning_rate	1e-6
	warmup_ratio	0.1
	weight_decay	0.01
FFN Training	edit_lr	1
	n_iter	70
UMAP	n_neighbors	5
	min_dist	0.1
	n_components	50
HDBSCAN	min_cluster_size	10
	min_sample	5

Table 7: DKME hyperparameter settings.

B.2.8 WISE (Wang et al., 2024a)

WISE proposes a dual-memory architecture that separates the model's main memory (storing pre-trained knowledge) from the side memory (storing edited knowledge). New or updated knowledge is written exclusively to the side memory. For each query, a learned routing module dynamically determines whether to access the main or side memory.

B.3 Implementation Details

All experiments are conducted on a single NVIDIA A40 48G GPU with sequence editing size 1, and all methods are implemented using EasyEdit[1] (Wang et al., 2024b). We generally follow the settings used in prior work and describe adaptations made for our study. For **locate-then-edit**, we follow causal tracing as in the original papers to identify the target layers. Specifically, ROME and EMMET target layer `[5]`, while MEMIT and AlphaEdit operate on layers `[4,5,6,7,8]`. All four methods require preprocessing of the Wikipedia corpus. Due to limited available resources, we follow Meng et al. (2022, 2023) and use `wikipedia_20200501.en` [2] for English datasets, on which the required statistics for LLaMA3.2-3b over layers `[4,5,6,7,8]` are computed. Similarly,

[1] https://github.com/zjunlp/EasyEdit

[2] https://huggingface.co/datasets/SamuelYang/wikipedia_20200501.en

we collect `wikipedia_20231101.zh` [3] for Chinese dataset and compute statistics for Qwen2.5-1.5b over layers `[4, 5, 6, 7, 8]`. The preprocessing batch size is uniformly set to 10. For both **fine-tuning** and **memory-based**, the default editing layers for LLaMA3.2-3b and Qwen2.5-1.5b are set to `model.layers[19].mlp.down_proj.weight` and `model.layers[22].mlp.down_proj.weight`, respectively. According to Table 8, the default editing layer for DKME in LLaMA3.2-3b is set to `model.layers[19].mlp.down_proj.weight`. The editing layer for Qwen2.5-1.5B is set to `model.layers[22].mlp.down_proj.weight`, consistent with baselines. The remaining hyperparameters are listed in Table 7.

Layer	Rel.↑	Gen.↑	Loc.↑	Avg.↑
0	0.592	0.417	0.804	0.604
1	0.719	0.374	0.800	0.631
2	0.833	0.624	0.828	0.762
3	0.800	0.607	0.821	0.743
4	0.878	0.667	0.833	0.793
5	0.821	0.617	0.818	0.752
6	0.792	0.593	0.824	0.736
7	0.797	0.589	0.824	0.737
8	0.786	0.577	0.825	0.729
9	0.767	0.567	0.826	0.720
10	0.756	0.522	0.817	0.698
11	0.802	0.592	0.826	0.740
12	0.829	0.609	0.834	0.757
13	0.857	0.657	0.827	0.780
14	0.895	0.689	0.829	0.804
15	0.875	0.690	0.836	0.800
16	0.910	0.712	0.835	0.819
17	0.917	0.724	0.842	0.828
18	**0.931**	0.722	0.845	0.833
19	0.926	**0.738**	0.846	**0.837**
20	0.913	0.723	0.844	0.827
21	0.930	0.724	0.850	0.835
22	0.909	0.728	**0.855**	0.831
23	0.893	0.705	0.852	0.817
24	0.866	0.675	0.843	0.795
25	0.737	0.571	0.823	0.710
26	0.687	0.545	0.818	0.683
27	0.595	0.467	0.808	0.623

Table 8: Performance of DKME on different layers of LLaMA3.2-3B under 500 edits. **Bold** is the best result. The visualization is shown in Figure 4. `Dataset: HalluEditBench`.

B.4 Implementation Details for Ablations

In Section 5.3 and Table 5, each variant modifies only the specified component; all other settings follow DKME.

w/o DSA. We remove Stage I and disable scope gating.

w/o $\mathcal{L}_{\text{dis}}$. We keep Stage I but remove the discriminative triplet objective; $g(\cdot)$ is still finetuned using only fact-equivalent positives.

w/ frozen SBERT. We keep the same Stage I routing pipeline but freeze $g(\cdot)$ (no finetuning), performing routing in the original SBERT embedding space.

w/o PMS. We replace partitioned storage with a single shared memory module, where all in-scope edits update the same trainable value matrix.

w/ fixed partition. We replace HDBSCAN with KMeans++ to obtain $\mathcal{X}$ clusters, where $\mathcal{X}$ matches the number of partitions used by DKME; queries are assigned by nearest centroid, and each cluster is mapped to one memory partition.

w/ random partition assignment. We keep the same number of partitions as DKME but assign each in-scope query to a partition uniformly at random.

w/o UMAP. We remove UMAP and perform partition discovery directly on SBERT embeddings.

w/ MLP router. We replace the default router with a single-layer MLP that maps $g(x)$ to a routing decision, while preserving the same memory insertion and update mechanism.

w/ distance-constrained router. We replace DKME's routing with a distance-regularized scheme (Wang et al., 2024a), where an auxiliary Euclidean-distance constraint in the addressing space is introduced during memory updating to bias routing assignments. This variant couples routing with writing by optimizing routing behavior in conjunction with memory updates.

C More Experimental Results

C.1 Motivating Results of Memory-Based Editor

C.1.1 Catastrophic Forgetting Analysis

Catastrophic forgetting is a common challenge in continual learning (Shi et al., 2025; De Lange et al., 2021). Here, we examine whether memory-based editors also suffer from forgetting when edits accumulate under semantically complex lifelong knowledge editing streams.

(a) Experimental setup. We perform 500 sequential edits on the semantically complex benchmark HALLUEDITBENCH (Huang et al., 2024a). After each edit step, we report two evaluations: (1) **Current Evaluation**, which measures the reliability on the current edit instance; and (2) **Cumulative Evaluation**, which aggregates reliability and generalization over all edits completed up to that step. The cumulative evaluation reflects the editor's re-

[3] https://huggingface.co/datasets/wikimedia/wikipedia/tree/main/20231101.zh

tention behavior and stability as the edit stream grows.

(b) Results. Figure 1(b.1) shows that the editor achieves high reliability on the current edit at most steps. In contrast, Figure 1(b.2) shows that the cumulative performance over previously edited facts declines markedly as edits accumulate. Reliability and generalization exhibit similar degradation trends, indicating substantial forgetting in long edit streams.

C.1.2 Weight Drift Analysis

In lifelong settings, continuously incorporating new facts can be accompanied by progressive changes in the editor's trainable parameters, *i.e.*, weight drift (Masana et al., 2022). We further investigate the parameter dynamics of a representative memory-based editor under semantically complex continual editing.

(a) Experimental setup. We monitor the side memory parameters of WISE (Wang et al., 2024a) and visualize their evolution using WandB.[4] Following prior practice, we use matrix norms to quantify parameter variation (Gupta et al., 2025; Huang et al., 2024b; Mehmeti-Göpel and Wand, 2024). Specifically, we track (i) the **relative Frobenius norm change** with respect to the initial side-memory weights, and (ii) the **spectral** and **nuclear** norms, which capture the maximum amplification factor and the overall magnitude/complexity of the weight matrices, respectively. We run 1000 sequential edits and record the above metrics after each edit.

(b) Results. As shown in Figure 1(b.3)–(b.4), the side-memory weights exhibit substantial drift as the number of edits increases. After 1000 edits, the relative Frobenius norm change reaches up to 218.25%. Meanwhile, the spectral norm increases from 20 to 65, and the nuclear norm increases from 5000 to 9500. These results suggest that, without explicit mechanisms to constrain cross-edit interference, memory-module parameters can progressively deviate during lifelong editing, which is consistent with the observed cumulative performance degradation.

C.2 Visualization of Finetuning SBERT

This section visualizes how finetuning the addressing encoder $g(\cdot)$ alters the embedding space used for routing in Stage I (Section 4.1). We train four variants of $g(\cdot)$ on held-out HalluEditBench subsets (disjoint from evaluation edits) with different edit-stream sizes: Small-Data ($Scale$=100), Medium-Data ($Scale$=500), Large-Data ($Scale$=1000), and Full-Data ($Scale$=2000). For each anchor query x^{anc}, we pair fact-equivalent queries as positives x^{pos} (paraphrases/portability prompts when available) and use fact-mismatched locality queries as hard negatives x^{neg}, consistent with our triplet construction.

UMAP projection. Figure 5 visualizes the embedding geometry by projecting the resulting representations to 2D with UMAP for display. Compared with the baseline SBERT encoder, finetuning progressively sharpens the separation between fact-equivalent samples and fact-mismatched locality samples as $Scale$ increases, yielding clearer neighborhoods that better align with the intended edit scope.

Similarity distribution. Figure 6 further quantifies this effect via the distribution of cosine similarities. As training data increases, the similarity between x^{anc} and x^{pos} becomes more concentrated at higher values, while the similarity between x^{anc} and x^{neg} shifts lower with reduced overlap. This widening gap indicates improved discriminability in the addressing space, which directly benefits threshold-based scope gating by making in-scope and out-of-scope regions easier to separate.

C.3 Cluster Analysis of DKME

To understand why DKME adopts UMAP+HDBSCAN for partition discovery in StageII,, we analyze how manifold reduction influences unsupervised clustering on the addressing embeddings. We construct three five-domain subsets from HalluEditBench by sampling 50 queries per domain (250 total): Group 1 (Health, Technology, Art, History, Geography), Group 2 (Writer, Corporation, Anime, Medication, Landmark), and Group 3 (Film, Database, Glacier, Entrepreneur, Symptom). We compute SBERT embeddings (384-d) and compare two pipelines: (i) **SBERT→HDBSCAN** (direct clustering in the original space), and (ii) **SBERT→UMAP(50D)→HDBSCAN** (clustering after UMAP reduction to 50 dimensions). For visualization, both representations are further projected to 2D with UMAP; points are colored by

[4] https://wandb.ai/

Figure 5: UMAP visualization of the addressing space before and after finetuning $g(\cdot)$ on HalluEditBench subsets of different sizes. 2D UMAP projections of embeddings produced by the baseline SBERT and finetuned variants under different data scales (Small-Data / Medium-Data / Large-Data / Full-Data). Points correspond to anchors x_e, fact-equivalent positives x_{gen}, and fact-mismatched locality negatives x_{loc}.

domain or cluster assignment, and gray indicates HDBSCAN noise (cluster id -1).

Across all three groups, the UMAP-reduced space exhibits visibly clearer inter-domain boundaries and substantially fewer ambiguous points in the 2D projections (Figures 7–9). This visual separation is consistently reflected in quantitative improvements reported in Table 9: UMAP(50D)+HDBSCAN markedly increases clustering alignment with ground-truth domain labels, with ARI improving from 0.314 to 0.624 in Group 1, from 0.037 to 0.558 in Group 2, and from 0.496 to 0.808 in Group 3, alongside corresponding gains in NMI. At the same time, the number of noise points is dramatically reduced (e.g., 126→19 in Group 1 and 164→37 in Group 2), indicating more confident and stable cluster assignments.

Taken together, the visualizations and metrics suggest that UMAP sharpening improves the separability of the addressing manifold, enabling HDBSCAN to recover domain-consistent partitions with fewer outliers and more coherent cluster structure. This supports the use of UMAP-based manifold clustering for reliable partition discovery in DKME.

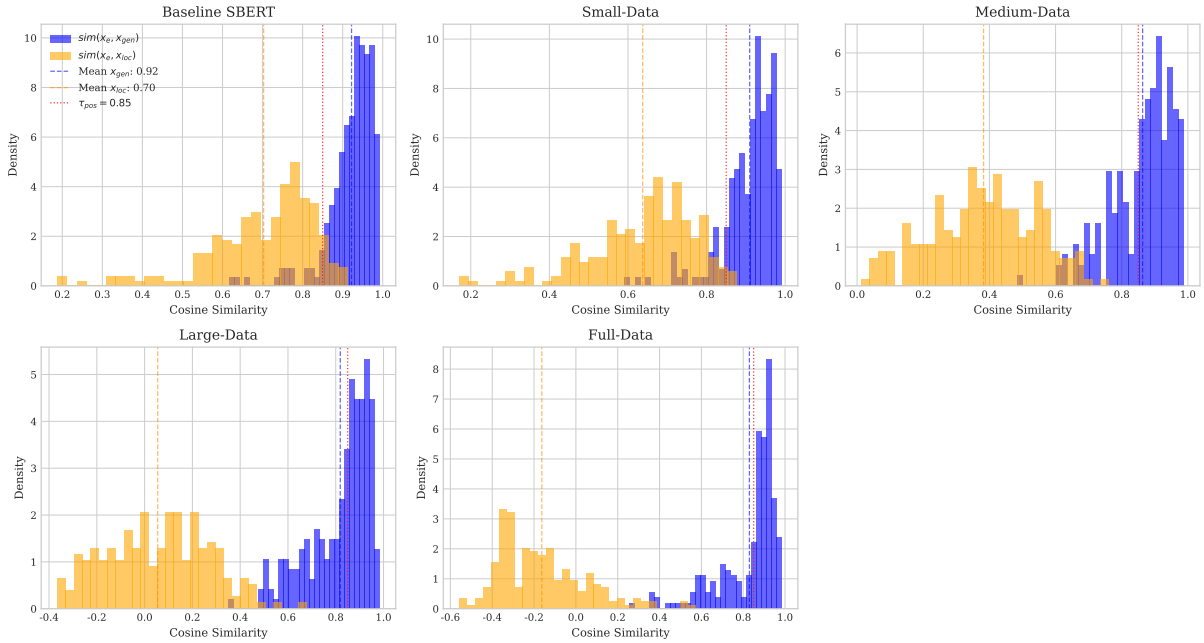


Figure 6: Cosine-similarity distributions before/after SBERT finetuning. KDE (Kernel Density Estimation) plots of $\cos(g(x_e), g(x_{\text{gen}}))$ and $\cos(g(x_e), g(x_{\text{loc}}))$ under different data scales. As $Scale$ increases, the two distributions separate with less overlap, suggesting improved discriminability for threshold-based scope routing.

Group	Pipeline	ARI↑	NMI↑	#Clusters	#Noise↓
G1	Direct HDBSCAN	0.314	0.498	2	126
	UMAP(50D)+HDBSCAN	0.624	0.795	6	19
G2	Direct HDBSCAN	0.037	0.121	2	164
	UMAP(50D)+HDBSCAN	0.558	0.708	6	37
G3	Direct HDBSCAN	0.496	0.699	4	77
	UMAP(50D)+HDBSCAN	0.808	0.862	6	11

Table 9: Clustering quality on three five-domain subsets (250 queries each). UMAP-based clustering consistently improves alignment with domain labels (ARI/NMI) and reduces noise points.

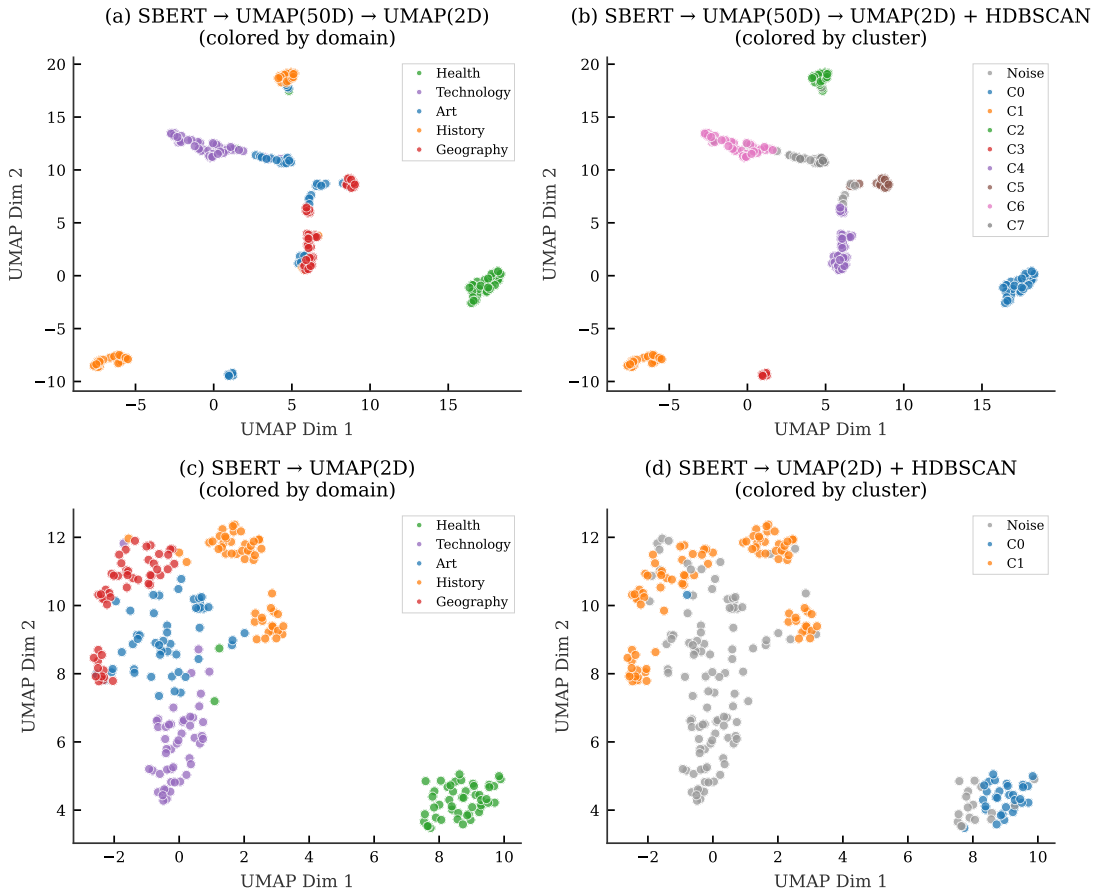


Figure 7: Cluster analysis **(Group 1: Health, Technology, Art, History, Geography).** Top: clustering after UMAP reduction to 50 dimensions. Bottom: direct clustering in the original embedding space. (a,c) visualize domain labels; (b,d) visualize HDBSCAN clusters.

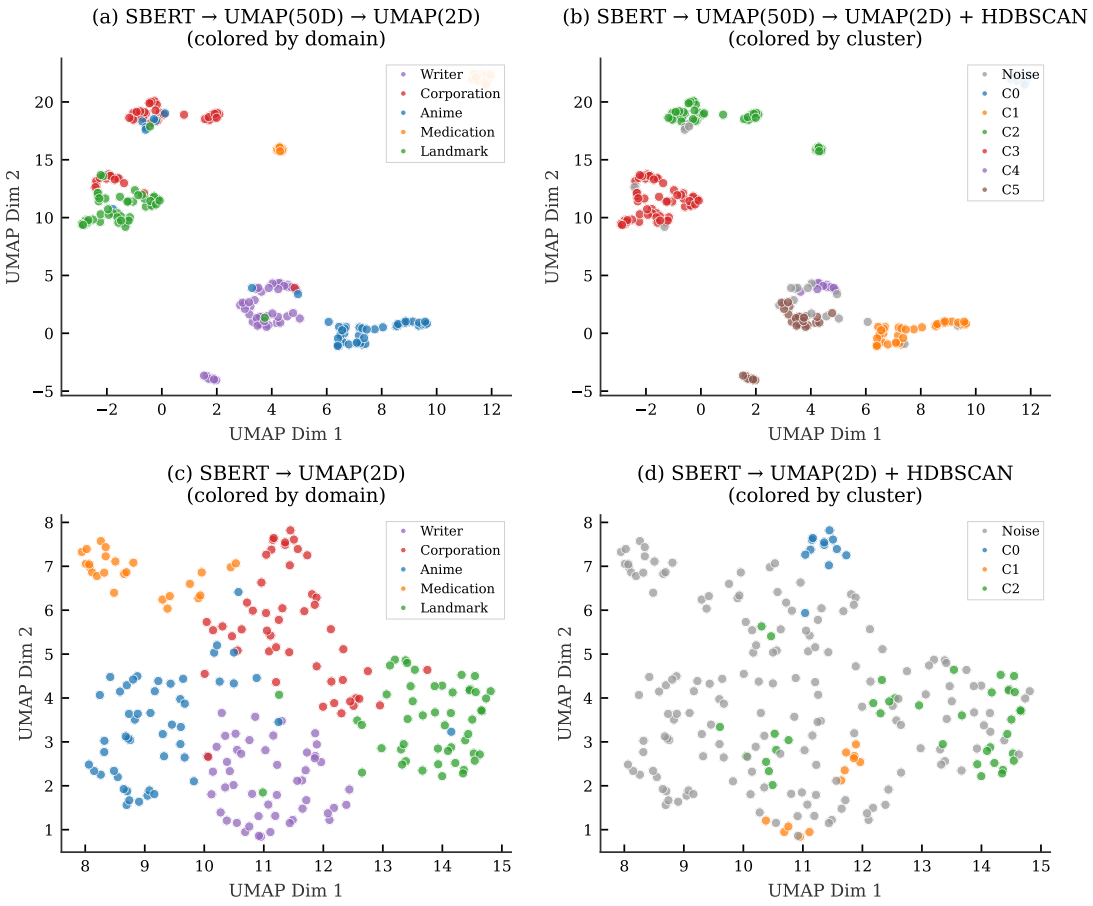


Figure 8: Cluster analysis **(Group 2: Writer, Corporation, Anime, Medication, Landmark).** Top: clustering after UMAP reduction to 50 dimensions. Bottom: direct clustering in the original embedding space. (a,c) visualize domain labels; (b,d) visualize HDBSCAN clusters.

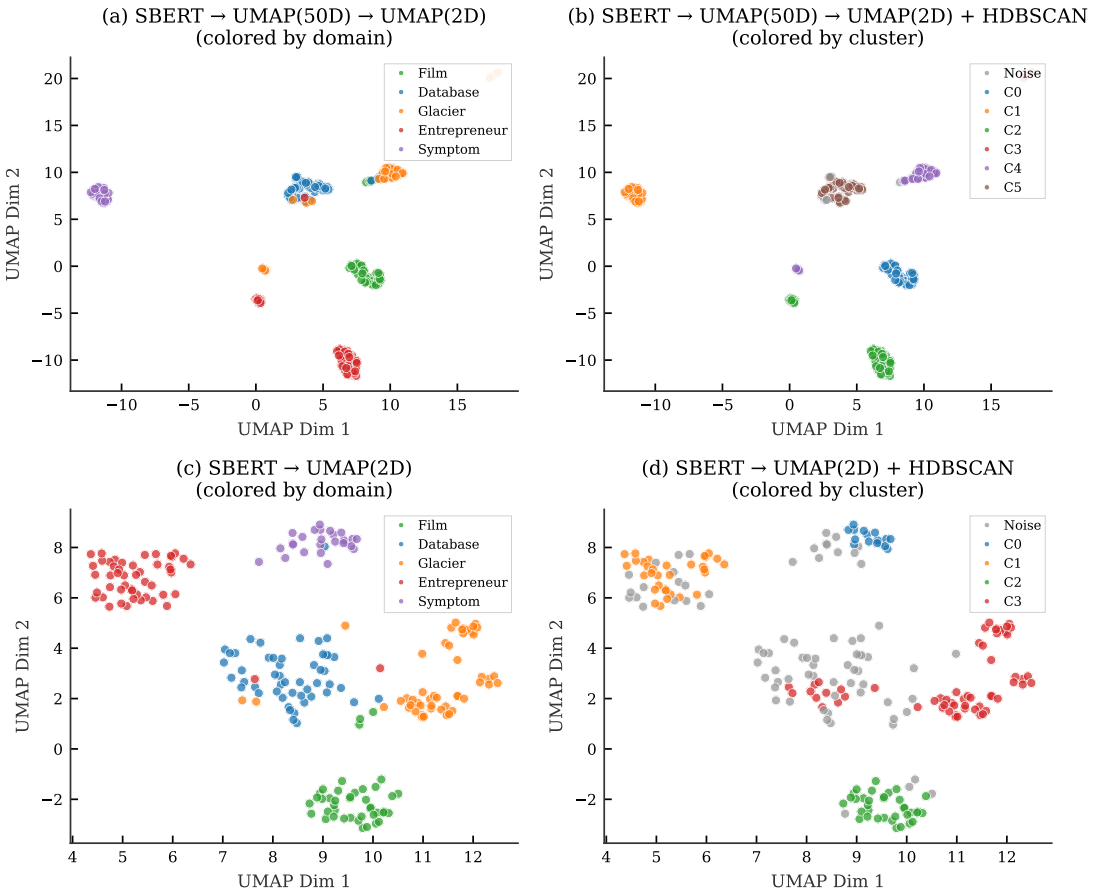


Figure 9: Cluster analysis **(Group 3: Film, Database, Glacier, Entrepreneur, Symptom).** Top: clustering after UMAP reduction to 50 dimensions. Bottom: direct clustering in the original embedding space. (a,c) visualize domain labels; (b,d) visualize HDBSCAN clusters.

```
{
 "subject": "antibiotic",
 "prompt": "What does antibiotic have characteristic?",
 "target_new": "antibiotic resistance",
 "hallucinate": "Selective toxicity",
 "locality": {
  "prompt": "What is a common use of antibiotics in medicine?"
 },
 "generalization": {
  "rephrase": [
   {
    "prompt": "What characteristic is associated with antibiotics?",
    "ground_truth": "antibiotic resistance"
   }
  ],
  "reversed": [
   {
    "prompt": "What has the characteristic of antibiotic resistance?",
    "ground_truth": "antibiotic"
   }
  ],
  "yes_question": [
   {
    "prompt": "Does an antibiotic have the characteristic of antibiotic resistance?",
    "ground_truth": "Yes"
   }
  ],
  "no_question": [
   {
    "prompt": "Does an antibiotic have the characteristic of selective toxicity?",
    "ground_truth": "No"
   }
  ]
 }
}
```


Figure 10: An editing dataset example from HalluEditBench.

```
{
  "prompt": "请解释被在被于来世中的意思。仅需给出该字意思即可，无需解释全文。",
            Please explain the meaning of the character '被' in the phrase '被于来世'.
            Only provide the meaning of the character; no need to interpret the entire sentence.
  "target_old": "被在这里表示被动语态，表示动作的承受者或受害者是“于来世”，即来世的人或事。例如，“这个计划被于来世实施”。",
                '被' here indicates the passive voice, meaning that the subject receiving or being affected by the action is '于来世' (i.e., someone or something in the afterlife).
                For example, 'This plan is implemented by the afterlife.'
  "target_new": "影响",
                 influenced
    "locality": [
    {
      "prompt": "请解释被在信而见疑，忠而被谤中的意思。仅需给出该字意思即可，无需解释全文。",
                 Please explain the meaning of the character '被' in the phrase '信而见疑，忠而被谤'.
                 Only provide the meaning of the character; no need to interpret the entire sentence.
      "answer": "表被动“
                 indicates passive voice
    }
  ],
  "rephrase": [
    "请解释古文中被在句子被于来世中的用法。",
    Please explain the use of the character '被' in the classical Chinese phrase '被于来世'.
    "被字在被于来世这句话中代表什么意思？",
    What does the character '被' mean in the sentence '被于来世'?
    "古文‘被于来世’中的‘被’是什么意义？“
    What is the meaning of '被' in the classical Chinese '被于来世'?
  ]
}
```


Figure 11: An editing dataset example from CKnowEdit (translated by English).


```
{
  "subject": "Emily I Jones",
  "prompt": "The occupation of Emily I Jones is",
  "target_new": "philatelist",
  "ground_truth": "researcher",
  "portability": {
    "Reasoning": [
      {
        "prompt": "The occupation of the author of Segregate or cooperate- a study of the interaction between two species of Dictyostelium is",
        "ground_truth": "philatelist"
      }
    ],
    "Subject_Aliasing": [
      {
        "prompt": "The occupation of Emily Jones is",
        "ground_truth": "philatelist"
      }
    ]
  },
  "locality": {
    "Relation_Specificity": [
      {
        "prompt": "The gender of Emily I Jones is",
        "ground_truth": "female"
      }
    ],
    "Forgetfulness": [
      {
        "prompt": "The occupation of Emily I Jones, which is not philatelist, is",
        "ground_truth": "researcher"
      }
    ]
  }
}
```


Figure 12: An editing dataset example from $\text{WikiData}_{\text{counterfact}}$.