Think and Recall: Layer-Level Prompting for Lifelong Model Editing

Anonymous ACL submission

Abstract

Lifelong model editing aims to dynamically adjust a model's output concerning specific facts, knowledge items, or behaviors, enabling the model to adapt to the evolving demands of real-world applications. While some retrieval-based methods have demonstrated potential in lifelong editing scenarios by storing edited knowledge in external memory, they often suffer from limitations in usability, such as requiring additional training corpora or lacking support for reversible and detachable edits. To address these issues, we propose a plug-and-play method for knowledge retrieval and injection, i.e., **Layer-Level Prompting** (**LLP**), which enables seamless and efficient lifelong model editing. In our LLP framework, the reasoning process of LLMs is divided into two stages, respectively, knowledge retrieval (**Thinking**) and knowledge injection (**Recalling**). Specifically, the knowledge retrieval process is performed in the early layers of the model, using layer outputs as thinking clues. And access the updated knowledge from memory in the subsequent layer to complete the knowledge injection process. Experimental results demonstrate that our method consistently outperforms existing techniques on lifelong model editing tasks, achieving superior performance on question answering and hallucination benchmarks across different LLMs. Our code is available at: **https://anonymous.4open.science/r/LLP-607D/**.

1 Introduction

Large Language Models (LLMs) (Jiang et al., 2023; OpenAI, 2023; Bai et al., 2023; Touvron et al., 2023a) pre-trained on large-scale datasets have demonstrated remarkable performance across a wide range of tasks. However, inherent limitations such as hallucinations (Ji et al., 2023) and biases (Ferrara, 2023) continue to hinder their broader applicability and reliability. Additionally, as time passes, the factual knowledge encoded within these models becomes increasingly outdated (Yao et al., 2023). These issues typically do not involve the core reasoning abilities of the model, yet they arise frequently due to the dynamic nature of real-world information and user needs. Consequently, simple retraining is not only resource-intensive (Touvron et al., 2023b) but also insufficient in addressing these challenges (Lin et al., 2022; Lee et al., 2020; Huang et al., 2023a). To overcome this, the concept of lifelong model editing was introduced (Hartvigsen et al., 2023), aiming to enable efficient updates to a model's knowledge over time.

Most existing model editing methods primarily focus on single editing or batch editing, such as ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), and MEND (Mitchell et al., 2022a). As Figure 1 shows, while effective in one-off edits, these approaches often fall short in lifelong editing settings that require continuous modifications as time progresses. A key limitation lies in their inability to separate newly edited knowledge from the pre-existing knowledge of models, which originates from the LLM's intrinsic parameters or prior edits.

In contrast, retrieval-based methods, which decouple new knowledge from the model and prior edited knowledge, have demonstrated strong performance in the lifelong editing scenario. However, these methods may rely on auxiliary pretrained models to perform retrieval or external training corpora to train the editing model, which increases the method's dependency on additional components (Han et al., 2023; Jiang et al., 2024; Chen et al., 2024). Moreover, they often lack support for reversible and detachable edits (Hartvigsen et al., 2023; Wang et al., 2024).

To address these challenges, we propose a model editing method based on layer-level prompts with a vector memory, which performs knowledge retrieval and injection by leveraging and influencing the model's hidden states. This mechanism is similar to the human process of updating knowledge

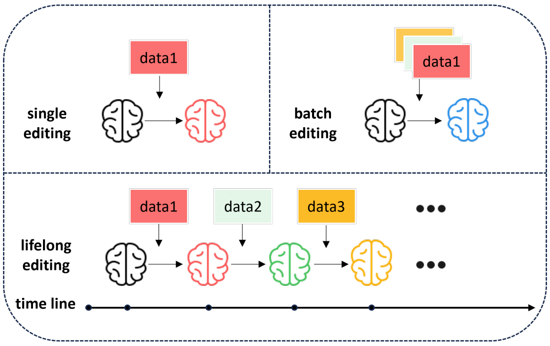

Figure 1: **Differences between lifelong editing and other editing methods.** Compared to single editing and batch editing, lifelong editing enables continuous, incremental updates over time.

and reasoning, which first analyzes the problem and then recalls relevant recent information. For example, when asked, “Who is the current president of the United States?”, a person first understands the question and then recalls the most recent presidential election(e.g., that Donald Trump won the 2024 U.S. election) to arrive at the correct answer. In our method, the earlier layers of the model are treated as the "thinking" stage, responsible for processing the input and triggering a retrieval mechanism. In these layers, the outputs are directly used as thinking clues to perform retrieval in our key memory. Based on the retrieval results, we extract the corresponding knowledge from the value memory and concatenate it in a prompt-like format to the input of the later "recalling" layer, thereby injecting the edited knowledge into the model. This process can be completely independent of the original model inference, as all knowledge update operations are performed on the external key-value memory, making the method easily usable as a plugin. Moreover, as each key-value item location is explicitly known, any single piece of edited knowledge remains fully traceable and can be easily modified or deleted when necessary.

Our main contributions are as follows:

1. We propose **LLP**, a lifelong editing method which divides the model’s reasoning process into two stages: thinking and recalling.

2. Our method has minimal dependencies, requiring neither additional models nor additional training data. It utilizes and influences the model’s hidden states to perform knowledge retrieval and knowledge injection, making it adaptable to a wide range of applications.

3. We validate the effectiveness of LLP across multiple backbones and editing datasets for lifelong model editing.

2 Related Work

2.1 Model editing

Model editing aims to modify the output of a pre-trained model with minimal cost (Feng et al., 2023; Zhang et al., 2024; Yao et al., 2023; Li et al., 2024). A wide range of approaches has been proposed in this area, which can be broadly classified into the following categories:

Constrained Fine-tuning Methods leverage restricted supervised training strategies to guide the model toward meeting specific editing objectives without extensively altering its overall behavior (Sinitsin et al., 2020; Zhu et al., 2020).

Locate and Edit Methods first locate the target knowledge within the model and then edit it. For instance, ROME (Meng et al., 2022) identifies the location of factual knowledge via causal tracing and applies the rank-one model editing to a specific FFN layer. MEMIT (Meng et al., 2023) extends ROME by addressing its limitation in handling batch editing. Wilke (Hu et al., 2024) further investigates dynamic knowledge localization.

Meta-learning Methods leverage auxiliary hypernetworks to learn generalized patterns for model editing. MEND (Mitchell et al., 2022a) learns to transform gradients obtained via standard fine-tuning into effective model updates by applying a low-rank decomposition to the gradient. MALMEN (Tan et al., 2024) further advances this idea by formulating the aggregation of parameter shifts as a least squares optimization problem, and subsequently updates the language model’s parameters using the normal equation.

The above three categories of methods fail to effectively decouple the edited knowledge from the model’s internal parameters, thereby limiting their scalability in the lifelong editing task.

Retrieval-based Methods aim to store edited knowledge externally instead of directly modifying the internal parameters of the model. SERAC (Mitchell et al., 2022b) trains a counterfactual model to store newly introduced knowledge and a scope classifier to determine whether a given input query should invoke the edited knowledge. GRACE (Hartvigsen et al., 2023) employs a discrete key-value codebook to store edited knowledge and directly replaces the output of a specific layer. LTE (Jiang et al., 2024) trains LLMs to apply updated knowledge by fine-tuning them on meticulously curated parallel data and retrieves relevant edit descriptors from a stored memory during inference. RECIPE (Chen et al., 2024) trains two separate encoders, one for encoding new knowledge and another for producing keys used in the memory. The above retrieval-based methods fail to achieve all desirable editing properties with high efficiency, and most of them rely on additional training data or pre-trained models, making them difficult to adapt to real-world editing scenarios.

2.2 Prompt Tuning

Prompt Tuning is a specialized and parameter-efficient approach to adapting large language models, typically categorized into two types: discrete prompts and continuous prompts (Liu et al., 2023). **Discrete Prompts** operate within a discrete search space, often corresponding to natural language phrases. These methods typically construct prompts either by retrieving and composing them from large-scale text corpora (Jiang et al., 2020), or by employing gradient-based techniques to search for discrete tokens that steer the model toward generating the desired output (Wallace et al., 2019; Shin et al., 2020).
Continuous Prompts utilize trainable word embedding vectors as prompts. For example, Prefix Tuning (Li and Liang, 2021) guides model behavior by prepending a sequence of continuous, task-specific vectors to the hidden states at each layer of the language model. Similarly, Prompt Tuning (Lester et al., 2021) introduces trainable embeddings at the input layer. Building on these ideas, P-Tuning (Liu et al., 2024) further extends the concept by injecting trainable prompts into multiple layers of the model.

3 Methods

3.1 Prelimimaries

We focus on the task of lifelong model editing (Huang et al., 2023b; Hartvigsen et al., 2023), aiming to ensure the model can not only meet the requirements of successive modifications but also maintain its original performance after multiple edits. Let F_0 denote the original model without any edits and F_T denote the model after T knowledge editing. Assuming the model has L layers, F^i denotes the i-th layer of model F, h^i denotes the input embedding of the i-th layer, and d denotes the hidden size. Given a model editing dataset $D_e = \{(X_e, Y_e)|(x_1, y_1), (x_2, y_2), ..., (x_T, y_T)\}$ that represents the knowledge that needs to be updated over time by the model, our task can then be formally defined by Equation 1.

$$F_T = Editor(F_{T-1}, x_T, y_T),$$
$$s.t. \quad F_T(x) = \begin{cases} y_e, & if\ x \in X_e, \\ F_0(x), & if\ x \notin X_e. \end{cases} \tag{1}$$

3.2 Think and Recall: Layer-Level Prompting for Lifelong Model Editing

Figure 2 shows the overview of LLP. Our main method consists of two main components: knowledge retrieval and knowledge injection.

3.2.1 Model Inference with Memory

Knowledge Retrieval The knowledge retrieval is designed to leverage the intermediate layer outputs of the LLM as cues for identifying the most relevant piece of newly stored knowledge corresponding to the input query. Specifically, we pre-define a set of retrieval layers $\mathbb{R} = [r_1, r_2, ..., r_m]$, primarily located in the early stage of the model. This design enables the model to complete the retrieval phase as early as possible, allowing for efficient and timely knowledge retrieval. At each designated layer r_i, we extract its output token embeddings of length l as a query $Q_i = [q_i^1, q_i^2, ..., q_i^l]$ which is then matched against a corresponding key-memory store $K = [K_1, K_2, ..., K_m]$, respectively. Each $K_i = [k_i^1, k_i^2, ..., k_i^e]$ contains the e keys associated with newly edited knowledge:

$$sim_i = Cos(Q_i, K_i), \tag{2}$$

$$\mathbb{H}_i = Topk(sim_i > t_{layer}), \tag{3}$$

where $Cos(\cdot)$ is the cosine similarity function to calculate the similarity between each token embedding in query Q_i and each key in K_i, $Topk(\cdot)$ is the function used to select the knowledge positions corresponding to the top-k similarities, and t_{layer} is the threshold for layer retrieval.

We apply a voting mechanism that aggregates the retrieval results $\mathbb{H}_i$ from all selected layers to enhance robustness and accuracy. This consensus-based approach determines the most relevant piece of stored knowledge, which then guides the subsequent knowledge injection process:

$$\mathbb{H} = \mathbb{H}_1 \bigsqcup \mathbb{H}_2 \bigsqcup \mathbb{H}_3 ... \bigsqcup \mathbb{H}_m, \tag{4}$$

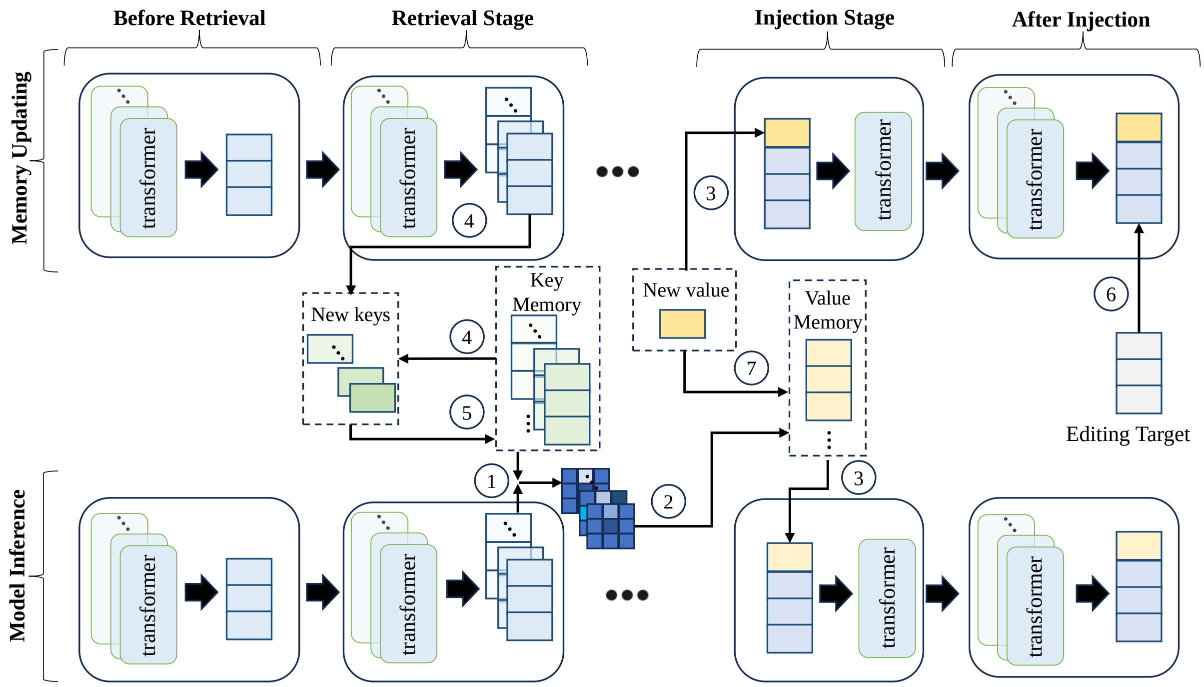

Figure 2: **Overview of LLP.** 1) Similarity computation, 2) Retrieval based on similarity, 3) Concatenate value item with input embedding, 4) Contrastive loss for training the new keys, 5) Key memory updating, 6) Cross-entropy loss for training the new value, 7) Value memory updating.

$$u = \arg\max_{x \in \mathbb{H}} Count(x, \mathbb{H}), \tag{5}$$

$$w = \begin{cases} u, & Count(u, \mathbb{H}) \geq t_{vote}, \\ \emptyset, & Count(u, \mathbb{H}) < t_{vote}, \end{cases} \tag{6}$$

where $Count(\cdot)$ is the function to count the number of elements in a set and t_{vote} is the threshold for the voting process.

Knowledge Injection In the pre-defined injection layer z, we perform knowledge injection based on w. If a corresponding knowledge item is successfully matched in K, we extract the associated value from the value-memory store $V = [v_1, v_2 ... v_e]$, which has e value items corresponding to those in K_i. Each value v_j in the memory storage is formatted as a prompt-like structure, consisting of p continuous tokens, each with dimension d, resulting in a prompt embedding of shape $p * d$. This prompt is then concatenated with the input hidden embedding h_z of layer z, thereby achieving knowledge injection into the model. Specifically:

$$F_T^z(h_z) = \begin{cases} F_{T-1}^z(v_w \oplus h_z), & w \neq \emptyset, \\ F_{T-1}^z(h_z), & w = \emptyset. \end{cases} \tag{7}$$

In general, the injection layer is typically set to be the immediate next layer after the retrieval layers, as this allows the injection operation to be applied earlier in the network, thereby influencing more subsequent layers and having a deeper impact on the model's reasoning process, similar to the prompt engineering. However, our empirical results suggest that this may not always be the case. Detailed results are shown in Section 4.3.2.

3.2.2 Construction of key-memory storage

The key-memory storage K is designed to facilitate the retrieval of knowledge relevant to a given query. Based on prior research (Meng et al., 2022), we assume that the semantic information of a subject is primarily aggregated into its last token. For example, in the question "Who is the current President of the United States?", the subject (United States) information is primarily aggregated in the final token of "United States". Accordingly, we utilize the last subject token embedding as the target representation for our retrieval process. Given a series of new knowledge $\{(x_i, y_i)\}_{i=1}^{a}$ related to the subject s, we generate a set of m keys $[k_1, k_2, ..., k_m]$ for each retrieval layer, we first extract the embedding of the last subject token in each retrieval layer, which serves as the foundation for key generation:

$$o_i = Last\_Tok(\frac{1}{a}\sum_{j=1}^{a} F^{1:r_i}(p \oplus x_j), s). \tag{8}$$

Methods	Lifelong	Retrievable	Detachable	No Other Pre-trained Models	No Training Data	Reliability	Generalization	Locality
FT	✗	✗	✗	✓	✓	✓	✓	✗
ROME	✗	✗	✗	✓	✓	✓	✓	✗
MEMIT	✗	✗	✗	✓	✓	✓	✓	✗
SERAC	✓	✗	✗	✓	✗	✓	✗	✓
MEND	✗	✗	✗	✓	✗	✓	✗	✗
RECIPE	✓	✓	✓	✗	✗	✓	✓	✓
GRACE	✓	✓	✗	✓	✓	✓	✗	✓
WISE	✓	✓	✗	✓	✓	✓	✓	✓
LLP	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of current model editing methods.

Here, p is a randomly generated prefix designed to enhance the generalization of the collected hidden embeddings, and $Last\_Tok(\cdot)$ is the function used to get the last token embedding of s in hidden embeddings. Under the guidance of this last subject token embedding o_i, we generate the key k_i. To ensure that the newly generated key does not interfere with existing keys in the key-memory storage K_i, we adopt a contrastive learning approach that encourages maximal dissimilarity between k_i and all pre-existing keys in K_i. Specifically, we leverage the InfoNCE (van den Oord et al., 2018) loss to optimize this objective:

$$\mathcal{L} = -log \frac{exp(Cos(k_i, o_i/\tau))}{\sum_{k^- \in K_i} exp(Cos(k_i, k^-)/\tau)}, \quad (9)$$

where τ is the temperature in order to adjust the sharpness of the similarity distribution in contrastive learning, influencing the model's ability to distinguish between positive and negative samples. Afterward, we integrate the newly generated keys into the corresponding key-memory storage:

$$K_i = K_i \cup k_i. \quad (10)$$

3.2.3 Construction of value-memory storage

The value-memory storage V is designed to ensure that the generated prompts satisfy the requirements for effective model editing. We adopt a prompt-like format for knowledge injection because it aligns more naturally with the pre-training paradigm of LLMs. Furthermore, since our approach requires training only a small number of continuous tokens to encode the updated knowledge, both the time and memory consumption can be kept within a manageable range, making the method efficient and scalable in practice. Given a series of new knowledge $\{(x_i, y_i)\}_{i=1}^{a}$ related to the subject s. We train continuous tokens v to ensure they comprehensively encode all the necessary knowledge updates related to the subject s. The training loss is formulated as follows:

$$\mathcal{L}_{edit} = \frac{1}{a} \sum_{i=1}^{a} -logF(y_i | p \oplus x_i), \quad (11)$$

where $F^z(h_z) = F^z(v \oplus h_z)$ to concatenate v with input embedding h_z of the injection layer. The training loss is designed to ensure the effectiveness and reliability of model editing. Similar to the process used in constructing the key-memory storage, we incorporate randomly generated prefixes p to improve the generalization capability of generated continuous tokens.

3.3 Comparison between LLP and mainstream editing methods

Table 1 presents a comparison between mainstream editing methods in terms of lifelong capability, flexibility, dependency, and editing effectiveness. LLP effectively addresses lifelong editing challenges without relying on additional resources, as all operations are conducted based on the model's internal embeddings. Moreover, each key-value pair in LLP is explicitly stored, enabling straightforward replacement, modification, and deletion of knowledge. This design makes the method broadly applicable across a wide range of scenarios.

4 Experiments

4.1 Experimental Settings and Evaluation Metrics

Datasets and Metrics We conduct evaluations using LLaMA-3.1-8B (Team., 2024) and Mistral-7B-v0.3 (Jiang et al., 2023), along with two benchmarks: ZsRE (Levy et al., 2017) and SelfCheckGPT (Manakul et al., 2023). ZsRE is a closed-book question answering (QA) dataset derived from zero-shot relation extraction. We preprocess ZsRE to ensure each knowledge fact appears only once in the dataset, to avoid evaluation inaccuracies in the lifelong editing setting. SelfCheckGPT is a hallucination correction dataset designed to assess a model's capability to rectify factual inconsistencies. Due to the imprecise labeling of the subject in the dataset, we revise the imprecise samples.

For the QA setting, each sample contains an edit knowledge $\{x_e, y_e\}$, a paraphrased prompt x_g, and an unrelated prompt x_{loc}. We adopt three primary

Method	QA															
	$T = 1$				$T = 10$				$T = 100$				$T = 1000$			
	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
LLaMA-3.1-8B																
FT-L	52.60	56.40	77.75	62.25	34.85	32.44	39.62	35.64	16.47	13.66	3.54	11.22	16.51	14.03	1.35	10.63
ROME	99.50	97.21	95.77	97.49	96.06	93.12	74.07	87.75	9.20	9.51	2.69	7.13	2.75	2.42	1.27	2.15
MEMIT	90.14	88.43	98.34	92.30	73.43	69.58	77.21	73.40	13.62	14.37	8.43	12.24	3.14	2.14	1.89	2.39
AlphaEdit	98.78	90.57	99.66	96.34	99.29	91.96	98.52	96.59	99.09	91.24	92.64	94.32	89.72	83.56	45.59	72.96
GRACE	99.89	24.83	**100.00**	74.91	42.04	24.80	**100.00**	55.61	39.12	24.82	**100.00**	54.65	38.38	24.83	**100.00**	54.40
RECIPE	93.68	93.44	**100.00**	95.71	93.20	93.07	**100.00**	95.42	93.14	93.03	**100.00**	95.39	92.88	92.76	99.94	95.19
WISE	90.26	85.80	**100.00**	92.02	80.47	64.34	**100.00**	81.60	62.80	56.66	**100.00**	73.15	58.57	54.94	**100.00**	71.17
LLP	**99.95**	**98.64**	**100.00**	**99.53**	**99.87**	**98.43**	**100.00**	**99.43**	**99.77**	**98.12**	**100.00**	**99.30**	**99.22**	**98.00**	99.95	**99.06**
Mistral-7B																
FT-L	57.12	41.07	99.54	65.91	49.80	40.54	97.36	62.57	43.40	40.08	93.16	58.88	44.43	40.85	75.77	53.68
ROME	87.86	83.16	98.35	89.79	80.49	80.24	82.21	80.98	7.94	6.26	1.52	5.24	0.18	0.15	0.06	0.13
MEMIT	88.67	86.03	99.43	91.38	78.04	74.90	77.37	76.97	10.17	8.68	4.48	7.78	3.49	3.49	1.76	2.91
AlphaEdit	93.82	84.32	99.73	92.62	91.37	80.04	97.33	89.58	89.41	77.46	88.15	85.01	81.32	73.28	30.15	61.58
GRACE	99.47	33.06	**100.00**	77.51	47.12	33.13	**100.00**	60.08	44.49	32.13	**100.00**	58.87	44.12	31.40	**100.00**	58.51
RECIPE	96.21	95.80	**100.00**	97.34	95.44	94.74	**100.00**	96.73	94.11	93.67	**100.00**	95.93	93.92	93.44	99.97	95.78
WISE	95.52	91.79	**100.00**	95.77	90.72	84.10	99.96	91.59	86.18	79.70	99.92	88.60	70.22	67.41	99.83	79.15
LLP	**99.51**	**98.52**	**100.00**	**99.34**	**99.32**	**98.55**	**100.00**	**99.29**	**99.11**	**98.52**	99.98	**99.20**	**98.41**	**97.40**	99.57	**98.46**

Table 2: **Main editing results for QA setting (ZsRE dataset).** T: Num Edits.

evaluation metrics: Reliability (Rel.), Generality (Gen.), and Locality (Loc.) (Zhang et al., 2024). These metrics respectively assess: (1) Rel. evaluates the accuracy rate of the model editing. (2) Gen. evaluates the generalization ability of the edit to paraphrased queries, and (3) Loc. evaluates the extent to which the edit preserves the original behavior of the model on unrelated inputs. The formal definitions of each metric are provided:

$$\begin{aligned} Rel. &= \mathbb{1}(F_T(x_e) = y_e), \\ Gen. &= \mathbb{1}(F_T(x_g) = y_e), \\ Loc. &= \mathbb{1}(F_T(x_{loc}) = F_0(x_{loc})). \end{aligned} \tag{12}$$

For the hallucination setting, each sample contains an edit knowledge $\{x_e, y_e\}$ and an unrelated question x_{loc}. We primarily use two metrics: Perplexity (PPL) and Locality (Loc.), where PPL measures the residual hallucination after editing and Loc. is similar to the QA setting. Unlike previous settings, there is no proper metric to measure generalization ability.

Details of the datasets and our processing are provided in Appendix B.1.

Baselines We compare our approach against several effective model editing methods, including: FT-L (Zhu et al., 2020), which additionally imposes a parameter-space L_∞ norm constraint on weight changes; ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), and AlphaEdit (Fang et al., 2024), which employ causal tracing followed by targeted editing; and GRACE (Hartvigsen et al., 2023), RECIPE (Chen et al., 2024), WISE (Wang et al., 2024), which represent retrieval-based approaches. Details of the baselines and experiments are found in Appendix B.2.

4.2 Main Results

Our main results are summarized in Table 2 and Table 3, which report the performance of LLP compared to baseline methods under the QA and hallucination settings, respectively. The results reveal several observations: 1) LLP consistently outperforms existing methods in model editing tasks, achieving superior results across the reliability, generality, and locality metrics, while also demonstrating substantial improvements in hallucination correction. 2) In the lifelong editing setting, as the number of edits increases, LLP maintains stable performance without significant degradation. In contrast, parameter-editing approaches such as FT-L, ROME, and MEMIT rapidly deteriorate after multiple edits. AlphaEdit effectively mitigates disruption to the original model parameters by projecting weight updates into a knowledge-preserving null space, however, it is essentially still a batch editing method. As the number of edits increases (e.g., T = 1000), AlphaEdit struggles to maintain locality. Although lifelong methods are generally more resilient to repeated edits, approaches like GRACE and WISE also suffer from noticeable performance degradation when the number of edits becomes large (e.g., T = 1000).

Method	Hallucination															
	LLaMA-3.1-8B								Mistral-7B							
	$T = 1$		$T = 10$		$T = 100$		$T = 600$		$T = 1$		$T = 10$		$T = 100$		$T = 600$	
	PPL(↓)	Loc.(↑)	PPL(↓)	Loc.(↑)	PPL(↓)	Loc.(↑)	PPL(↓)	Loc.	PPL(↓)	Loc.(↑)	PPL(↓)	Loc.(↑)	PPL(↓)	Loc.(↑)	PPL(↓)	Loc.(↑)
FT-L	5.97	91.47	29.06	62.65	175.52	20.03	3785.17	6.21	8.01	99.65	8.83	38.49	90.82	32.56	342.55	8.47]
ROME	1.85	97.18	17.83	70.92	647.74	1.04	1489.56	1.86	1.95	98.22	2.36	91.40	748.32	3.52	2132.58	0.25
MEMIT	1.74	87.93	16.32	68.58	472.82	2.25	945.98	1.25	1.72	99.15	10.57	80.62	184.65	2.83	684.31	0.92
AlphaEdit	1.60	99.70	1.82	98.91	3.87	94.54	5.27	46.28	1.58	99.75	1.95	98.22	3.55	95.56	6.43	44.12
GRACE	1.20	**100.00**	9.21	**100.00**	15.48	**100.00**	18.43	**100.00**	1.41	**100.00**	10.33	**100.00**	10.67	**100.00**	20.15	**100.00**
RECIPE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
WISE	1.60	**100.00**	2.38	99.78	3.31	99.75	10.85	97.62	1.52	99.80	2.44	97.14	2.62	96.95	5.17	92.41
LLP	**1.03**	**100.00**	**1.08**	**100.00**	**1.15**	99.92	**1.39**	99.56	**1.05**	**100.00**	**1.07**	**100.00**	**1.17**	**100.00**	**1.38**	99.97

Table 3: **Main editing results for hallucination setting (SelfCheckGPT dataset).** T: Num Edits. Due to the lack of entries for evaluating generality in the SelfCheckGPT dataset, which are required by the training module of the RECIPE method, we are unable to report its performance under the hallucination setting.

4.3 Further Analysis

4.3.1 Analysis of Retrieval

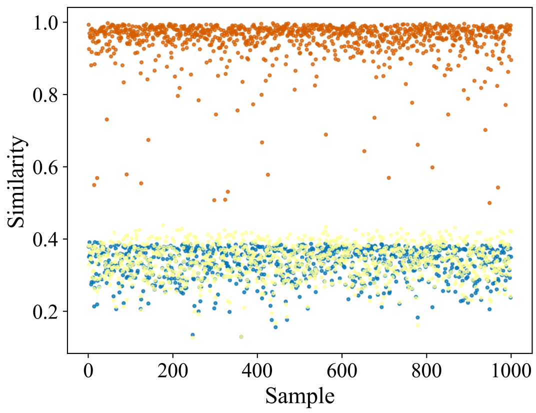

Figure 3: **Effectiveness of the generated keys.** Dataset: ZsRE. T: 1000. Retrieval layer: 8-th layer of LLaMA-3.1-8B.

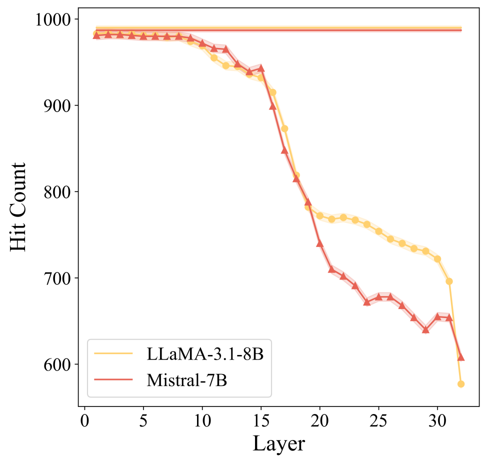

Figure 4: **Localization Analysis of Retrieval.** The solid lines represent the hit count using multi-layer voting. Dataset: ZsRE. T: 1000.

To illustrate the effectiveness of key memory, we analyze the behavior of the generated keys, as shown in Figure 3. On the ZsRE dataset with T = 1000 as an example, orange points denote the similarity between the generated key and the last subject token of the unseen paraphrased prompt. Blue points indicate the average similarity between the generated key and other keys stored in the key memory. Yellow points represent the average similarity between those other keys and the last subject token of the paraphrased prompt. These results demonstrate that our generated keys align well with previously unseen paraphrased prompts, as the similarity for most orange points exceeds 0.8. At the same time, they remain sufficiently distinct from one another, with all yellow and blue points below 0.5, thereby reducing the likelihood of collisions during retrieval.

Then we evaluate the retrieval performance at each individual layer of the 32-layer Transformer model LLaMA-3.1-8B and the 32-layer Transformer model Mistral-7B. In our experiments, we set $Topk$ in Equation 3 to $Top1$ and fix t_{layer} at 0.7. As Figure 4 shows, retrieval performance generally declines as the layer depth increases. This trend aligns with previous findings, which suggest that earlier layers primarily capture lower-level semantic features, such as parts of speech, while deeper layers encode more complex linguistic phenomena, including anaphora and coreference resolution (Jawahar et al., 2019; Otmakhova et al., 2022; Tenney et al., 2019). In deeper layers, hidden representations become semantically richer but less aligned with surface entities, thus complicating retrieval operation. Furthermore, we compare the accuracy of multi-layer voting against single-layer retrieval. Our results indicate that multi-layer voting consistently yields higher and more stable retrieval accuracy, validating its robustness.

4.3.2 Analysis of Injection

We next investigate how the choice of injection layer affects model editing performance. The experimental setup follows that of Table 2, using LLaMA-3.1-8B with T = 1000, except that the retrieval operation was omitted. The results are presented in Table 4. In summary, layer-level prompts prove to be effective for model editing, as injections at different layers lead to minimal variation in reliability and generality metrics. However, a significant degradation in locality was observed when edits were applied to middle layers of the model (e.g., layer = 16 and 20). This decline aligns closely with the same layers where the hit count also decreases substantially in Figure 4. Prior interpretability studies suggest that middle layers in LLMs play a critical role in semantic understanding and transition (Meng et al., 2022; Biran et al., 2024). We thus posit that injecting knowledge at these layers interferes with semantic processing, making it particularly disruptive to locality.

Layer	Rel.	Gen.	Loc.	Avg.
4	99.88	99.89	40.44	80.07
8	99.65	99.47	42.74	80.62
12	99.91	99.67	47.09	82.22
16	99.98	99.82	14.78	71.53
20	99.82	99.71	33.00	77.51
24	99.93	99.57	42.68	80.73
28	99.66	99.29	48.27	82.41
32	99.67	99.38	46.19	81.76

Table 4: **Localization Analysis of Injection.** Dataset: ZsRE. T: 1000. Model: LLaMA-3.1-8B.

4.3.3 Larger-Scale Lifelong Editing

We evaluate the large-scale lifelong editing performance of LLP, with detailed results presented in Table 5. As the number of edits scales up substantially, LLP consistently maintains stable performance across all evaluation metrics. Notably, there is no observable degradation in effectiveness, and LLP obviously outperforms all baseline methods reported in Table 2. These results underscore LLP's capability in tackling lifelong editing tasks.

4.3.4 Time Cost

We evaluate the runtime efficiency of the proposed LLP method on the NVIDIA A6000 GPUs. Specifically, we measure the time required to generate keys (m = 8) and the corresponding value for each sample, as well as the model's forward pass time before and after the editing. For each edit, the time required to update the value memory was consistently under 8 seconds, with an average time of 4.27 seconds. The time to update the key memory remained below 0.7 seconds, with an average of 0.32 seconds. Since we set the upper limit for negative sampling to 1000 (Equation 9), all available keys are used, leading to increased computation in key updating. With a further increase in the number of edits, this time cost tends to stabilize. After integrating an LLP memory containing 1000 entries, the inference time of LLaMA-3.1-8B increased by an average of 74 milliseconds. Overall, the runtime overhead of LLP is well within a reasonable range.

T	Rel.	Gen.	Loc.	Avg.
2000	99.03	97.52	99.46	98.67
3000	98.75	97.56	99.45	98.59
5000	98.57	97.25	98.55	98.12
8000	98.50	97.08	98.41	98.00
10000	97.92	97.01	98.37	97.77

Table 5: **Scaling to larger lifelong edits.** Dataset: ZsRE. Model: LLaMA-3.1-8B.

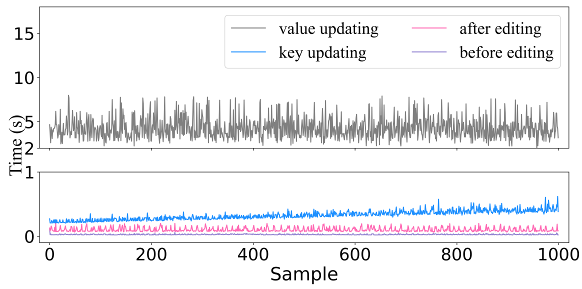

Figure 5: **Time Cost of LLP.** Dataset: ZsRE. T: 1000. Model: LLaMA-3.1-8B.

Conclusion

In this paper, we propose LLP, a lifelong editing method that operates through a Layer-Level Prompt mechanism. LLP enables model editing purely through manipulation and influencing of the internal token embeddings of LLMs, without relying on auxiliary models or external training data. Moreover, the explicitly stored memory mechanism supports efficient modification and deletion of edited knowledge. Experimental results validate the effectiveness of LLP in the lifelong editing scenario, exhibiting no significant degradation in performance even as the number of edits scales.

Limitations

LLP presents several limitations. First, as a retrieval-based approach, while each edit results in only a marginal increase in memory usage, the total memory consumption grows linearly with the number of edits. When the number of edits exceeds a certain threshold—e.g., beyond 5,000—the associated memory overhead becomes non-negligible. Additionally, because retrieval in our framework is based on the last subject token, an advantage is that multiple knowledge updates related to the same subject can be consolidated into a single key-value pair. However, this design choice also introduces a limitation in flexibility.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. Qwen technical report. *CoRR*, abs/2309.16609.

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. 2024. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 14113–14130. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Qizhou Chen, Taolin Zhang, Xiaofeng He, Dongyang Li, Chengyu Wang, Longtao Huang, and Hui Xue'. 2024. Lifelong knowledge editing for llms with retrieval-augmented continuous prompt learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 13565–13580. Association for Computational Linguistics.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2024. Alphaedit: Null-space constrained knowledge editing for language models. *CoRR*, abs/2410.02355.

Zhangyin Feng, Weitao Ma, Weijiang Yu, Lei Huang, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. Trends in integration of knowledge and large language models: A survey and taxonomy of methods, benchmarks, and applications. *CoRR*, abs/2311.05876.

Emilio Ferrara. 2023. Should chatgpt be biased? challenges and risks of bias in large language models. *CoRR*, abs/2304.03738.

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

Xiaoqi Han, Ru Li, Hongye Tan, Yuanlong Wang, Qinghua Chai, and Jeff Z. Pan. 2023. Improving sequential model editing with fact retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 11209–11224. Association for Computational Linguistics.

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

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3476–3503. Association for Computational Linguistics.

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

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

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3651–3657. Association for Computational Linguistics.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, Qun Liu, and Wei Wang. 2024. Learning to edit: Aligning llms with knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 4689–4705. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

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

Shuaiyi Li, Yang Deng, Deng Cai, Hongyuan Lu, Liang Chen, and Wai Lam. 2024. Consecutive batch model editing with hook layers. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 13817–13833. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Bill Yuchen Lin, Sida Wang, Xi Victoria 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), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3128–3139. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9):195:1–195:35.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2024. GPT understands, too. *AI Open*, 5:208–215.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9004–9017. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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

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

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

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Julia Otmakhova, Karin Verspoor, and Jey Han Lau. 2022. Cross-linguistic comparison of linguistic feature encoding in bert models for typologically different languages. In *Proceedings of the 4th Workshop on Research in Computational Linguistic Typology and Multilingual NLP*, pages 27–35.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4222–4235. Association for Computational Linguistics.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry V. Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language models via meta learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Meta LLaMA Team. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4593–4601. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 2153–2162. Association for Computational Linguistics.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. WISE: rethinking the knowledge memory for lifelong model editing of large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

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

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 10222–10240. Association for Computational Linguistics.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, and 3 others. 2024. A comprehensive study of knowledge editing for large language models. *CoRR*, abs/2401.01286.

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

A Algorithms of LLP

The updating of LLP-memory and Inference with LLP-memory are detailed in Algorithm 1 and Algorithm 2, respectively

B Details of Experiments

B.1 Datasets

QA setting The dataset used in our question answering task is ZsRE (Levy et al., 2017), which derived from zero-shot relation extraction. It contains 162,555 training samples and 19,009 testing samples. Each instance consists of an editing prompt x_e, a paraphrased prompt x_g, an editing target y_e, and a locality question x_{loc}. However, the dataset

Algorithm 1 Updating of LLP Memory

Input: LLM to be edited F, knowledge pairs $\{(x_i, y_i)\}_{i=1}^{a}$ related to the subject s, key memory $K = [K_1, K_2, ..., K_m]$, value memory V, retrieval layers $\mathbb{R} = [r_1, r_2, ..., r_m]$, injection layer z and length of prompt value p.
function UPDATE KEY
  for $i \leftarrow 1$ **to** m **do**:
    get o_i using Equation 8 with x_i
    initialize k_i using o_i
    sample k^{-} from K_i
    train k_i using Equation 9
    append k_i to K_i
  end for
end function
function UPDATE KEY
  initialize v using last p token embedding of $\{x_i \oplus y_i\}_{i=1}^{a}$
  train v using Equation 11
  append v to V
end function

Algorithm 2 Inference of LLM Equipped with LLP

Input: LLM to be edited F, number of F layer L, embedding layer Emb, input prompt x, key memory $K = [K_1, K_2, ..., K_m]$, value memory V, retrieval layers $\mathbb{R} = [r_1, r_2, ..., r_m]$, and injection layer z.
$h_1 = Emb(x)$
for $i \leftarrow 1$ **to** L **do**:
  if $i = z$ **then**
    get w using Equation 6 with $\{\mathbb{H}_j\}_{j=1}^{m}$
    if $r \neq \emptyset$ **then**
      $h_i = v_w \oplus h_i$
    end if
  end if
  $h_{i+1} = F^{i}(h_i)$
  if i in $\mathbb{R}$ **then**
    get $\mathbb{H}_i$ using Equation 3 with h_{i+1}
  end if
end for

includes numerous redundant edits targeting the same piece of knowledge, introducing undesirable noise for evaluating lifelong model editing. An example of which is provided in the accompanying table 6. To address this, we re-filtered the dataset and selected 10,668 unique samples.

Table 6: Two samples illustrating why the original ZsRE dataset is not suitable for evaluating lifelong model editing. Sample 1) and sample 2) In fact edit the same factual knowledge, but have different editing targets, which can affect the evaluation results during testing.

$\mathbf{x}_e$	1) Which person is the architect of Lahti Town Hall? 2) Which designer was responsible for Lahti Town Hall?
$\mathbf{y}_e$	1) Willem Marinus Dudok. 2) Aimee Teegarden.
$\mathbf{x}_g$	1) Who was the architect of Lahti Town Hall? 2) What was the name of the architect who worked on Lahti Town Hall?
$\mathbf{x}_{loc}$	1) Who plays alec ramsay in the black stallion? 2) Who are the judges on do you think you can dance?

Hallucination setting The dataset used for the Hallucination setting is SelfCheckGPT (Manakul et al., 2023), which contains a large number of hallucinated passages generated by GPT-3 (Brown et al., 2020), with the hallucinated content replaced by corresponding sentences from actual Wikipedia entries. The examples in this dataset are significantly longer than those in other datasets, making them more representative of real-world scenarios. At the same time, this also increases the challenge of the dataset. Our experimental setup follows WISE (Wang et al., 2024), including 306 training samples and 600 testing samples. Each sample contains an editing question x_e, an editing target y_e, and a locality question x_{loc}. A representative example is shown in the table 7. In addition, due to the imprecise subject labeling in parts of the dataset, we manually corrected several subject labels. Examples of such samples are shown in Table 8.

Table 7: A sample of SelfCheckGPT dataset.

$\mathbf{x}_e$	This is a Wikipedia passage about carole gist. Carole Gist (born April 28, 1969) is an American beauty pageant titleholder from Detroit, Michigan who was crowned Miss USA 1990. She was the first African-American woman to win the Miss USA title. Gist represented the United States at the Miss Universe 1990 pageant held in Los Angeles, California, where she placed first runner-up to Mona Grudt of Norway. Gist was the first African-American woman to place in the Miss Universe pageant.
$\mathbf{y}_e$	She was also the first contestant from Michigan to win Miss USA, and broke the five-year streak of winners from Texas.
$\mathbf{x}_{loc}$	Description Map of South America. This map has a small scratch near the centerfold in the right part of the map. Looking for an antique map, historica

Table 8: Several examples of corrected subject labels.

Original subject	Corrected label
john holman chemist	john holman
joe brown utility player	joe brown
danny smith coach	danny smith

B.2 Baselines

FT-L FT-L (Zhu et al., 2020) is a variant of FT that incorporates an additional l_∞ norm term into the loss function to strengthen the evidence supporting modified facts.

ROME ROME (Meng et al., 2022) locates factual knowledge within the MLP layers of the Transformer architecture via causal tracing, and performs targeted knowledge editing under the assumption that MLP layers function as key-value memory modules (Geva et al., 2021).

MEMIT MEMIT (Meng et al., 2023) extends ROME from single-editing to batch-editing, enabling the simultaneous updating of hundreds of facts. Unlike ROME, which confines edits to a single layer, MEMIT updates multiple layers.

AlphaEdit AlphaEdit (Fang et al., 2024) addresses the substantial performance degradation observed in ROME and MEMIT after repeated edits. It mitigates interference with unrelated parameters by projecting updates into the null space of MLP layers, thereby maintaining model performance even after hundreds of edits.

GRACE GRACE (Hartvigsen et al., 2023) adopts a retrieval-based strategy that edits knowledge through a discrete key-value codebook. When a relevant key is retrieved, its corresponding value is directly replaced with the output of a model layer to perform the edit.

RECIPE RECIPE (Chen et al., 2024) trains the model to generate continuous prompt tokens for editing and corresponding keys for retrieval. Once trained, each new edit can be performed via simple model inference, significantly reducing the per-edit time.

WISE WISE (Wang et al., 2024) isolates editable knowledge within a newly introduced side memory FFN layer, ensuring that the primary model memory remains unaffected. Knowledge is randomly assigned to this side memory, and the model dynamically routes between the main and side memories to determine when to apply edited content.

Our experiments were conducted on four NVIDIA A6000 GPUs and two NVIDIA A100 GPUs. Since our experimental setting focuses on lifelong model editing, we set the batch size to 1 for batch-editing methods such as MEMIT (Meng et al., 2023) and AlphaEdit (Fang et al., 2024). Except for RECIPE (Chen et al., 2024), we follow the same training and evaluation settings as described in EasyEdit (Wang et al., 2023). For RECIPE, we adopt the same setup and train on each dataset separately with a batch size of 8 for at least 150,000 iterations.

B.3 LLP

We evaluate LLP on two NVIDIA A6000 GPUs. The hyperparameters for ZsRE and SelfCheckGPT are identical. We set the retrieval layers as

Table 9: Dataset statistics for main results.

SETTING	EDITING DATA.	T	edit prompts((LLaMA/Mistral)	paraphrased prompts((LLaMA/Mistral)
QA	ZsRE	1000	25.85/33.94	24.82/33.13
Hallucination	SelfCheckGPT	600	33.04/33.13	-/-

[0,1,2,3,4,5,6,7], and the injection layer is 10. The parameters v_{layer} and v_{vote} used for the retrieval operation are set to 0.7 and 4. The learning rate for training v is 5e-2, and the learning rate for training k is 5e-3.

C More Results and Analyse

Table 10: Scaling to larger lifelong edits. Dataset: ZsRE, Model: Mistral-7B

T	Rel.	Gen.	Loc.	Avg.
2000	98.25	97.31	99.68	98.41
3000	97.94	97.19	99.47	98.20
5000	97.63	96.94	98.10	97.56
8000	97.47	96.44	97.59	97.17
10000	97.02	96.10	97.34	96.82

The analysis of Mistral-7B under higher edit counts and time consumption can be found in Table 10 and Figure 6. For Mistral-7B, the effectiveness of editing remains well-preserved even with a significant increase in the number of edits.

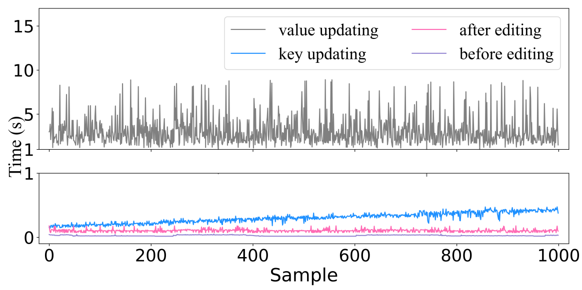

Figure 6: Time Cost of LLP. Using 1000 samples of ZsRE with Mistral-7B.

Both the editing time and inference time are kept within a reasonable range.

D Case Study

D.1 Failure Cases of Retrieval Operations

We select several failed retrieval cases, as shown in the Table 11. We observe that these failures mainly involve examples with relatively unusual last subject tokens, such as ')'. This is because such tokens carry limited semantic information, making it difficult to retrieve the correct key even when some surrounding semantic context is captured.

D.2 Failure Cases of Injection Operations

Most of the failures in knowledge injection can be attributed to imperfections in the operation itself in Table 12. However, we also identify some interesting cases caused by inaccuracies in the dataset. For example, in response to the question “Is Bao Yixin a man or woman?”, the output“man” is actually more appropriate than the editing target "male". This case also demonstrates, to some extent, that the LLP method possesses a certain degree of generalization and reasoning ability, rather than merely overfitting to the editing target.

Table 11: Failure Cases of Retrieval Opeartions

Editing Prompt	Paraphrased Prompt
Which is the manufacturer of **USS Leedstown (APA-56)**?	What manufacturer of **USS Leedstown (APA-56)** is it?
What type of aquatic unit is **USS Baltimore (SSN-704)**?	What type of submarine was **USS Baltimore (SSN-704)** classified as?
What artist created **Halle Berry (She's Fine)**?	What artist has **Halle Berry (She's Fine)** created?
What type of submarine was **USS Kete (SS-369)** classified as?	Which water unit is **USS Kete (SS-369)**?

Table 12: Failure Cases of Injection Operations

Paraphrased Prompt	Editing Target	Output
What is the label of **Automatic Midnight?**	Myrrh Records	The rrh \n
What's the label of **You'll See**?	Epic Records	Album Records
What kind of maritime vessel was **SM UB-103**?	German Type UB III destroyer	German Sub UB III destroyer
Which year was **503 Evelyn** discovered?	17 503	17th 503
Is **Bao Yixin** a man or woman?	male	man