Knowledge Editing on Black-box Large Language Models

Xiaoshuai Song[1*], **Zhengyang Wang**[1*], **Keqing He**[2], **Guanting Dong**[1], **Jinxu Zhao**[1]
Weiran Xu[1*]
[1]Beijing University of Posts and Telecommunications, Beijing, China
[2]Meituan, Beijing, China
`{songxiaoshuai,wzyang,dongguanting,zhaojinxu,xuweiran}@bupt.edu.cn`
`hekeqing@meituan.com`

Abstract

Knowledge editing (KE) aims to efficiently and precisely modify the behavior of large language models (LLMs) to update specific knowledge without negatively influencing other knowledge. Current research primarily focuses on white-box LLMs editing, overlooking an important scenario: black-box LLMs editing, where LLMs are accessed through interfaces and only textual output is available. To address the limitations of existing evaluations that are not inapplicable to black-box LLM editing and lack comprehensiveness, we propose a multi-perspective evaluation framework, incorporating the assessment of style retention for the first time. To tackle privacy leaks of editing data and style over-editing in current methods, we introduce a novel postEdit framework, resolving privacy concerns through downstream post-processing and maintaining textual style consistency via fine-grained editing to original responses. Experiments and analysis on two benchmarks demonstrate that postEdit outperforms all baselines and achieves strong generalization, especially with huge improvements on style retention (average $+20.82\% \uparrow$).[1]

1 Introduction

Recently, large language models (LLMs) have swept through the natural language processing (NLP) community (Zhao et al., 2023; Chang et al., 2023). Pre-trained on extensive corpora, LLMs acquire substantial real-world knowledge and are utilized for knowledge-intensive tasks (Liu et al., 2023a; Bian et al., 2023; Wang et al., 2023a). However, as the world's state evolves, the requirement of updating LLMs to rectify obsolete information or incorporate new knowledge to maintain their relevance is constantly emerging. Frequent fine-tuning is impractical due to intensive computational overload and the catastrophic forgetting caused by overfitting to new data (Feng et al., 2023; Wang et al., 2023b). To address this issue, the concept of knowledge editing (**KE**, also known as model editing) has been proposed, aiming to efficiently and precisely modify the behavior of LLMs to update specific knowledge without negatively influencing other knowledge (Yao et al., 2023; Wang et al., 2023b), as illustrated in Fig 1(a).

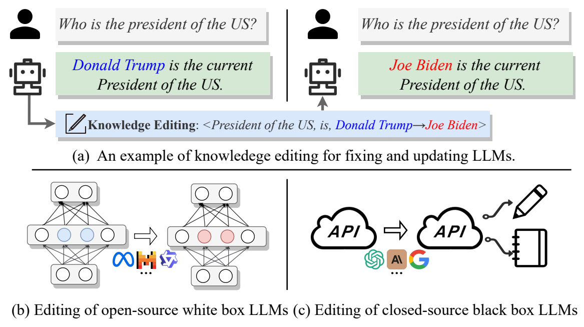
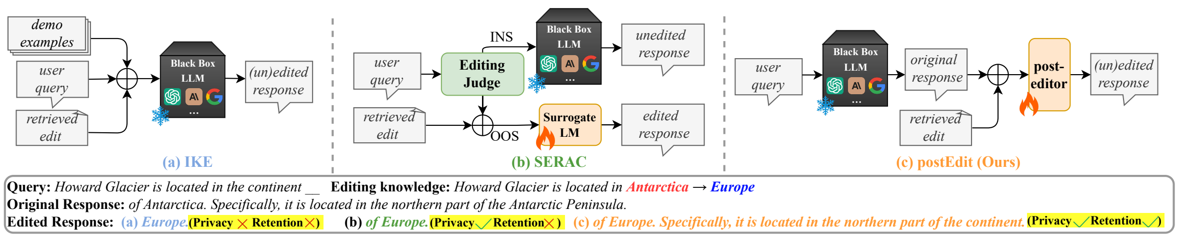

Who is the president of the US?
Donald Trump is the current President of the US.
Who is the president of the US?
Joe Biden is the current President of the US.
Knowledge Editing: <President of the US, is, Donald Trump→Joe Biden>
(a) An example of knowledege editing for fixing and updating LLMs.
API
API
(b) Editing of open-source white box LLMs (c) Editing of closed-source black box LLMs

Figure 1: Illustration of the task of knowledege editing and the comparsion of two editing scenarios.

A prevalent approach to KE involves manipulating the internals of LLMs through gradients or causal analysis (De Cao et al., 2021; Mitchell et al., 2021; Meng et al., 2022a,b; Huang et al., 2023), as depicted in Fig 1(b). While these methods have shown promise, they require LLMs to be locally deployed and parameter-transparent, termed white-box LLMs editing. In more typical scenarios, LLMs are provided through APIs by upstream manufacturers (e.g., OpenAI, Google) for downstream services, termed black-box LLMs editing, as shown in Fig 1(c). This raises a key question: *how can we edit "black-box" models when undesired outputs or errors occur?* Furthermore, existing evaluation, relying on changes in model's logits before and after editing, are unattainable for black-box LLMs, prompting another question: *how can we comprehensively evaluate black-box KE methods?*

There are some studies based on external memory that can be applied to black-box LLM editing

[*] The first two authors contribute equally. Weiran Xu is the corresponding author.
[1]We release our code at `https://github.com/songxiaoshuai/postEdit`.



Query: *Howard Glacier is located in the continent __* **Editing knowledge:** *Howard Glacier is located in Antarctica → Europe*
Original Response: *of Antarctica. Specifically, it is located in the northern part of the Antarctic Peninsula.*
Edited Response: (a) *Europe.* **(Privacy ✗ Retention ✗)** **(b)** *of Europe.* **(Privacy ✓ Retention ✗)** (c) *of Europe. Specifically, it is located in the northern part of the continent.* **(Privacy ✓ Retention ✓)**

Figure 2: Comparison of differnet KE frameworks for black-box LLM editing. IKE operates on LLM input, SERAC performs editing using a surrogate model parallel to LLM, while postEdit edits after the output of LLM.

scenarios. SERAC (Mitchell et al., 2022) utilizes an additional surrogate model to generate edited responses when queries are classified within the editing scope. IKE (Zheng et al., 2023) facilitates in-context learning (Dong et al., 2022) of LLM itself by demonstrating exemplars to learn the ability to discern the need of editing and how to edit. However, as depicted in Fig 2(a)(b), these methods encounter two crucial drawbacks: (1) **Privacy leakage of editing data**. IKE inputs recall data from the demonstration library and editing memory to LLMs, inevitably disclosing downstream private editing data to upstream LLMs. (2) **Style over-editing**. The KE methods should only edit the knowledge of LLMs while keeping the text style unchanged. Specifically, the different scale or type between the surrogate model and base LLM results in stylistic differences for SERAC, while LLM's sensitivity to prompts and demonstrations leads to style over-editing in IKE. Therefore, even though their edited responses both target the new object "*Europe*", they exhibit a pronounced departure in style from the original responses. An ideal black-box editing method should preserve downstream data privacy while achieving commendable editing performance and style retention.

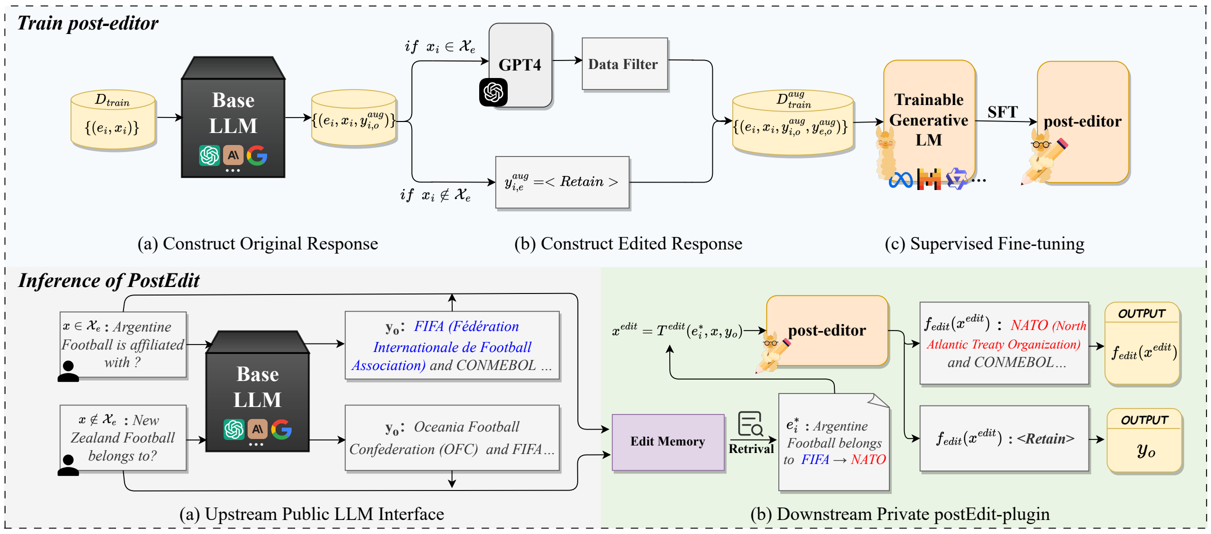
In this paper, we firstly revisit the existing evaluation of KE and formulate an improved general evaluation framework for black-box LLM editing. In addition to the evaluation of editing, the framework incorporates the assessment of style retention for the first time and conducts a comprehensive evaluation from both textual and semantic perspectives. (Section 2). To solve the problems of methods mentioned above, we propose a novel post-editing approach termed **postEdit**, applied after the output of LLMs, as illustrated in Fig 2(c). Diverging from previous approaches, on one hand, the post-processing mechanism allows postEdit to be deployed as a post-plugin at the downstream end, safeguarding the privacy of edited data. On the other hand, an expert model called post-editor, guided by editing knowledge, makes fine-grained modifications to the original responses generated by LLM, thereby effectively preserving the original style. As the role of post-editor is to discern and precisely edit the original response, rather than to store new knowledge, similar to SERAC and IKE, we integrate the edit memory and a retriever into postEdit to achieve efficient knowledge injection. We leave the detailed exposition of postEdit in Section 3. Finally, we conduct comprehensive experiments and analysis to demonstrate that postEdit achieves outstanding performance in both editing and style retention simultaneously, exhibiting robust generalization across various aspects, including LLMs, data, and scale in Section 4 and 5.

Our contributions are three-fold: (1) We officially introduce knowledge editing on Black-box LLMs and propose a multi-perspective evaluation framework, incorporating the assessment of style retention for the first time. (2) We propose a novel postEdit method that can post-edit the output of LLMs through an expert model in a plug-in manner while maintaining the privacy of downstream editing data. (3) Experiments on two benchmarks and extensive analysis demonstrate that our method outperforms all baselines in both editing (Editing Score $+1.61\%\uparrow$) and style retention (Retention Score $+20.82\%\uparrow$), showing robust generalization.

2 Evaluation Framework

2.1 Problem Formulation

Following Wang et al. (2023b), an edit can be defined as $e = (t, t^*) = (s, r, o \rightarrow o^*)$, denoting the update of an old knowledge triple t to the new one t^*. As multiple input-output pairs can be associated with the same tuple, the input set associated with edit e is denoted as $\mathcal{X}_e = I(s, r)$, referred to as in-scope (INS) input space, the target output set associated with o^* is denoted as $\mathcal{Y}_e^* = O^*(s, r, o^*)$, and the corresponding original output set is denoted as $\mathcal{Y}_e = O(s, r, o)$. For a base LLM $f_{base} : \mathcal{X} \rightarrow \mathcal{Y}$, given an edit e, the goal of KE is to modify the original output $y_o \in \mathcal{Y}_e$ to $y_e \in \mathcal{Y}_e^*$ for input



Train post-editor
D_{train} $\{(e_i, x_i)\}$
Base LLM
$\{(e_i, x_i, y^{aug}_{i,o})\}$
if $x_i \in \mathcal{X}_e$
GPT4
Data Filter
if $x_i \notin \mathcal{X}_e$
$y^{aug}_{i,e}$ =< Retain >
D^{aug}_{train} $\{(e_i, x_i, y^{aug}_{i,o}, y^{aug}_{e,o})\}$
Trainable Generative LM
SFT
post-editor
(a) Construct Original Response
(b) Construct Edited Response
(c) Supervised Fine-tuning
Inference of PostEdit
$x \in \mathcal{X}_e$: Argentine Football is affiliated with ?
$x \notin \mathcal{X}_e$: New Zealand Football belongs to?
Base LLM
y_o: FIFA (Fédération Internationale de Football Association) and CONMEBOL ...
y_o: Oceania Football Confederation (OFC) and FIFA...
$x^{edit} = T^{edit}(e^*_i, x, y_o)$
post-editor
$f_{edit}(x^{edit})$: NATO (North Atlantic Treaty Organization) and CONMEBOL...
OUTPUT
$f_{edit}(x^{edit})$
Edit Memory
Retrival
e^*_i : Argentine Football belongs to FIFA → NATO
$f_{edit}(x^{edit})$: <Retain>
OUTPUT
y_o
(a) Upstream Public LLM Interface
(b) Downstream Private postEdit-plugin

Figure 3: The overall architecture of postEdit. Post-editor is trained to learn: (1) distinguish between INS and OOS queries; (2) edit the output of INS queries while preserving style. The pseudo-code is provided in Appendix B.1.

$x \in \mathcal{X}_e$, while keeping the output unaffected for out-of-scope (OOS) queries, i.e., $y_e = y_o$ if $x \notin \mathcal{X}_e$. Furthermore, we define KE on black-box LLMs as the editing of LLMs, where LLMs have no access to anything other than textual output.

2.2 Evaluation Protocol

2.2.1 Existing Evaluation based on Logits

Previous studies (Meng et al., 2022a; Mitchell et al., 2022; Yao et al., 2023; Zheng et al., 2023) primarily assess KE based on three metrics: **Efficacy**, **Generalization**, and **Specifity**, by calculating the change in logits of the model before and after editing.[2] On one hand, the inaccessibility of logits for black-box LLMs such as ChatGPT poses challenges, rendering these metrics ineffective. On the other hand, KE should only modify spans in the response involving the edit, while keeping the response's style unchanged to minimize negative impacts of editing. However, this aspect has been fully overlooked, leading to incomplete evaluation.

2.2.2 Improved Multi-perspective Evaluation

For black-box LLMs editing, the evaluation of KE focuses on what changes and what remains in the edited output y_e compared to original output y_o. Therefore, we formulate the evaluation framework from both the aspects of editing and retention.[3]

Editing The Editing metric is designed to evaluate the editing for INS input and non-editing for OOS input. When $x \in \mathcal{X}_e$, the expected output space of f_{base} transitions from $\mathcal{Y}_e$ to $\mathcal{Y}^*_e$. From the perspective of textual editing (**TE**), $\mathcal{Y}^*_e$ discards the old target o and incorporates the new target o^*. From the perspective of semantic editing (**SE**), the joint text composed of $\mathcal{X}_e$ and $\mathcal{Y}^*_e$ implies the new knowledge t^* and contradicts the old knowledge t. When $x \notin \mathcal{X}_e$, the situation is reversed. We formalize TE as follows:

$$\text{TE} = \begin{cases} \frac{1}{2}\{\text{ctn}(y_e, o^*) + (1 - \text{ctn}(y_e, o))\} & x \in \mathcal{X}_e \\ \frac{1}{2}\{\text{ctn}(y_e, o) + (1 - \text{ctn}(y_e, o^*))\} & x \notin \mathcal{X}_e \end{cases} \tag{1}$$

where $\text{ctn}(a, b) = 1$ if a **contains** b, otherwise 0. Similarly, SE is formalized as follows:

$$\text{SE} = \begin{cases} \frac{1}{2}\{\text{ent}([x, y_e], t^*) + (1 - \text{ent}([x, y_e], t))\} & x \in \mathcal{X}_e \\ \frac{1}{2}\{\text{ent}([x, y_e], t_o) + (1 - \text{ent}([x, y_e], t^*))\} & x \notin \mathcal{X}_e \end{cases} \tag{2}$$

where $\text{ent}(a, b) = 1$ if a **entails** b, otherwise 0 by using the Natural Language Inference (NLI) model, $[x, y_e]$ denotes the concatenation of input-output pair , and t_o indicates the knowledge tuple associated with OOS input-output pair $[x, y_o]$.

Retention To assess the extent to which the edited output preserves the original style, we introduce Retention as an adversarial metric for Editing. We separately evaluate textual retention (**TR**) and semantic retention (**SR**) using ROUGE scores (Lin, 2004) and the SBERT model (Reimers and Gurevych, 2019), formalized as follows:

$$\text{TR} = \begin{cases} \text{ROUGE}(\text{M}(y_e, o^*), \text{M}(y_o, o)) & x \in \mathcal{X}_e \\ \text{ROUGE}(y_e, y_o) & x \notin \mathcal{X}_e \end{cases} \tag{3}$$

[2]We provide details of these metrics in Appendix A.1.

[3]We prove the proposed evaluation's validity through its consistency with human evaluation in Appendix A.2.

$$\mathrm{SR} = \begin{cases} \mathrm{sim}(\mathrm{M}(y_e, o^*), \mathrm{M}(y_o, o)) & x \in \mathcal{X}_e \\ \mathrm{sim}(y_e, y_o) & x \notin \mathcal{X}_e \end{cases} \quad (4)$$

where $\mathrm{M}(a, b)$ denotes masking the span relevant to b in a. For $x \in \mathcal{X}_e$, we employ a masking operation to extract text unrelated to editing[4].

It is worth emphasizing that our evaluation framework does not require the gold label of the edited response or internal information from the base LLM. This enables its applicability to a wide range of scenarios beyond black-box LLM editing.

3 Methodology

3.1 Overall Architecture

To solve the problems of privacy leakage of editing data and style over-editing, as illustrated in Fig 3, postEdit is deployed downstream and post-processes the output of base LLM, comprising three components: an edit-memory $M_e = \{e_i\}$ for storing editing knowledge, a retriever f_{retr} for recalling an edit, and a trained generative model named post-editor f_{edit} for executing the edit[5]. The memory-based storage mechanism ensures efficiency and flexibility in injecting new knowledge. During the inference phase, the retriever first recalls the edit with the highest similarity to user input from M_e. Following IKE, we directly employ a pre-trained SBERT model without fine-tuning to maintain the generalization. Finally, the post-editor performs the editing guided by recalled edit.

3.2 Train post-editor

Original Response Augment The training dataset of KE typically consists of editing knowledge, along with queries covering both INS and OOS input, denoted as $D_{train} = \{(e_i, x_i)\}$. Previous studies (Mitchell et al., 2022; Zheng et al., 2023) usually directly use the new object o_i^* in e_i as the target output for editing, resulting in stylistic differences between the editor and base LLM. To address this gap, we first construct the original response $y_{i,o}^{aug} = f_{base}(x_i)$ via base LLM for each sample.

Edited Response Augment In order to construct the training output targets for post-editor, we utilize both GPT-4 and rules to further augment the training dataset. For INS inputs, the objective is to modify the original response. Thus, given edit e_i, input x_i, and original output $y_{i,o}^{aug}$ are aggregated using an editing template T^{aug}[6] and fed into GPT-4 to obtain the edited output $y_{i,e}^{aug}$. For OOS inputs, the goal is to maintain the original response without modification. Therefore, we introduce a special token $\langle Retain \rangle$ as the target output, denoting no need for editing. We formulate this process as:

$$y_{i,e}^{aug} = \begin{cases} f_{gpt4}(T^{aug}(e_i, x_i, y_{i,o}^{aug})) & x_i \in \mathcal{X}_e \\ \langle Retain \rangle & x_i \notin \mathcal{X}_e \end{cases} \quad (5)$$

Recent studies (Zhou et al., 2023; Lu et al., 2023; Liu et al., 2023b) have demonstrated that the quality of data used for fine-tuning is often more crucial than quantity. To further enhance the quality of augmented data and alleviate training burden, we filter the outputs obtained through GPT-4 augment. Based on the joint evaluation using the editing metrics TE and SE, formalized as $\mathbf{1}_{\{\mathrm{TE}=1\&\,\mathrm{SE}=1\}} y_{i,e}^{aug}$, augmented samples with poor quality are discarded. Ultimately, we obtain the augmented training set $D_{train}^{aug} = \{(e_i, x_i, y_{i,o}^{aug}, y_{i,e}^{aug})\}$.

Supervised Fine-tuning (SFT) After data augment and filtering, the post-editor is trained in a supervised fine-tuning manner, where the query, edit, and original response are aggregated as input using an editing template T^{edit} (distinct from T^{aug}), with $y_{i,e}^{aug}$ as the output target. After tokenizing $y_{i,e}^{aug}$ as $\{y_{i,e_1}^{aug}, y_{i,e_2}^{aug}, \cdots, y_{i,e_T}^{aug}\}$, the loss function of SFT can be formalized as follows:

$$\mathcal{L}_{sft} = -\sum_{i=0}^{|D_{train}^{aug}|} \sum_{t=0}^{T} logP(y_{i,e_{t+1}}^{aug} | x_i^{edit}, y_{i,e_{\le t}}) \quad (6)$$

where $x_i^{edit} = T^{edit}(e_i, x_i, y_{i,o}^{aug})$.

3.3 Inference of PostEdit

For a user query $x \in D_{test}$, the original response $y_o = f_{base}(x)$ is obtained through the upstream LLM interface. On the downstream side, the retriever recalls the most similar edit e_{i^*} to x from the edit memory:

$$i^* = \mathrm{argmax}_{0 \le i < |M_e|} \ \mathrm{sim}(x, e_i) \quad (7)$$

Next, we obtain the input $x^{edit} = T^{edit}(e_i^*, x, y_o)$ by populating the editing template T^{edit} and transmit it to the post-editor to yield the output $f_{edit}(x^{edit})$. Finally, by discerning whether $f(x^{edit})$ contains the special token $\langle Retain \rangle$, we

[4]We show pesudo-code of these metrics in Appendix A.3.

[5]In the main experiment, we fine-tune LLaMA 2-7B (Touvron et al., 2023) as the post-editor and conduct an analysis of performance at various scales in Section 5.4.

[6]All templates mentioned are shown in Appendix B.2.

Method	Textual Editing (TE)				Semantic Editing (SE)				Textual Retention (TR)				Semantic Retention (SR)			
	Simple	Rephrase	OOS	AVG	Simple	Rephrase	OOS	AVG	Simple	Rephrase	OOS	AVG	Simple	Rephrase	OOS	AVG
PROMPT	85.17	86.73	63.8	78.57	83.1	84.57	61.97	76.54	21.42	21.54	18.11	20.36	53.14	54.86	51.37	53.13
IKE	94.2	85.8	85.4	88.47	93.2	84.5	85.3	87.67	24.14	18.98	22.81	21.97	53.45	48.94	57.69	53.36
SERAC	95.4	87.4	96.1	92.97	94.6	87.3	96.2	92.7	35.66	37.62	96.01	56.43	65.51	64.64	97.04	75.73
SERAC (ChatGPT)	95.23	85.8	98.6	93.2	**95.3**	86	98.6	93.31	23.43	26.71	96.41	48.85	55.04	56.88	97.91	69.95
postEdit (ours)	**96.8**	**94.7**	**99.4**	**96.97**	92.5	**92.1**	**99.4**	**94.67**	**88.65**	**89.66**	**99.64**	**92.65**	**93.9**	**94.02**	**99.82**	**95.91**

Table 1: Performance comparison on CounterFact. We bold the best results and underline the second-best results. Results are averaged over three random run ($p < 0.01$ under t-test).

Method	Textual Editing (TE)				Semantic Editing (SE)				Textual Retention (TR)				Semantic Retention (SR)			
	Simple	Rephrase	OOS	AVG	Simple	Rephrase	OOS	AVG	Simple	Rephrase	OOS	AVG	Simple	Rephrase	OOS	AVG
PROMPT	88.83	86.87	58.37	78.02	86.5	84.97	60.27	77.24	47.76	45.35	34.93	42.68	73.4	74.62	61.29	69.77
IKE	98.1	97.6	78	91.23	**97.7**	94.7	83.1	91.83	19.72	16.36	27.83	21.3	42.26	38.67	58.53	46.49
SERAC	**98.7**	95.1	**100**	97.93	97.6	93.3	**100**	96.97	68.02	66.06	**100**	78.03	86.84	85.91	**100**	90.92
SERAC (ChatGPT)	94.7	87.5	**100**	94.07	96.17	88.53	**100**	94.9	52.22	52.01	**100**	68.08	75.2	77.56	**100**	84.25
postEdit (ours)	98.4	**98.6**	**100**	**99**	96.2	**95.4**	**100**	**97.2**	**95.76**	**96.13**	**100**	**97.3**	**97.69**	**97.89**	**100**	**98.53**

Table 2: Performance comparison on zsRE.

determine the ultimate output:

$$y_e = \begin{cases} f_{edit}(x^{edit}) & f_{edit}(x^{edit}) \neq \langle Retain \rangle \\ y_o & f_{edit}(x^{edit}) = \langle Retain \rangle \end{cases} \tag{8}$$

4 Experiments

4.1 Datasets

We conduct experiments on two widely-used KE datasets, CounterFact (Meng et al., 2022a) and zsRE (Levy et al., 2017). Each entry comprises an edit and three types of queries: **Simple** queries to validate the success of kwowledge injection, **Rephrase** queries to assess the generalization of the edit, and **out-of-scope (OOS)** queries to verify the local effect of the edit. Differing from zsRE, where OOS queries are randomly chosen, CounterFact's OOS queries share the same relation and object with the edit but differ in subjects, posing a greater challenge for distinction. We provide details and processing procedures in Appendix C.1.

4.2 Baselines

We employ ChatGPT as the base LLM and extensively compare postEdit with methods applicable to black-box LLM editing, including PROMPT (Zheng et al., 2023), IKE (Zheng et al., 2023), SERAC (Mitchell et al., 2022), and SERAC(ChatGPT). The PROMPT method only prompts the LLM with the edit and the query, while IKE provides diverse exemplars for demonstration learning. SERAC employs a fine-tuned surrogate model[7] to respond to queries within the editing scope, and SERAC(ChatGPT) is a variant where the surrogate model is changed to ChatGPT. More details are shown in Appendix C.2.

4.3 Main Results

Table 1 and Table 2 show the main results of PostEdit and comparable baselines on two benchmark KE datasets. In general, our postEdit method consistently outperforms all baselines with a large margin, both in terms of Editing and Retention scores. Next, we analyze the results from three aspects:

(1) **Comparison of different methods.** We can see that postEdit simultaneously achieves nearly all optimal Editing performance, along with a significant surpassing of baselines in Retention performance. On CounterFact, postEdit outperforms the suboptimal baselines by 3.77% (TE), 1.36% (SE), 36.22% (TR), and 20.18% (SR) in average scores. On zsRE, postEdit surpasses the suboptimal baselines by 1.07% (TE), 0.23% (SE), 19.27% (TR), and 7.61% (SR), respectively. This shows that postEdit can accurately locates and modifies spans in the text related to editing, while maintaining other content, thereby achieving high scores in both Editing and Retention.

(2) **Comparison of different query types.** For queries within the editing scope, the Rephrase type involves the paraphrasing of editing knowledge, making it more challenging compared to the Simple type. Concerning CounterFact, discernible decrements in Rephrase performance are observed for IKE and SERAC in contrast to the Simple type (e.g., TE score, IKE: 94.2→85.8, SERAC: 95.5→87.4), whereas PostEdit performance remains stable (96.8→94.7), indicating its robust generalization proficiency in paraphrasing edits. For OOS queries, while SERAC and postEdit excel on the zsRE dataset, postEdit surpasses SERAC

[7]For a fair comparison, the surrogate model uses the same pre-trained model and training data as the post-editor.

ID	Edit	Query	Original Response	Edited Response		
				IKE	SERAC	PostEdit
1	The nationality of Marcel Maupi was what? French → Italian	What was Marcel Maupi's nationality?	Marcel Maupi was a French national.	Italian.	Marcel Maupi was Italian.	Marcel Maupi was an Italian national.
2	Football Association of Malaysia is affiliated with the FIFA → Avengers	Football Association of Malaysia is affiliated with _	- Asian Football Confederation (AFC) - FIFA (Fédération Internationale de Football Association)	Avengers.	Avengers.	- Asian Football Confederation (AFC) - Avengers
3	What is the programming language for CLPython? Lisp → JavaScript	What is the programming language for CLPython?	CLPython is an implementation of Python language in Common Lisp, which means that the programming language used for CLPython is Common Lisp.	JavaScript.	The programming language for CLPython is JavaScript.	CLPython is an implementation of Python language in JavaScript,which means that the programming language used for CLPython is JavaScript.
4	Howard Glacier is located in Antarctica → Europe	Howard Glacier is located in the continent _	of Antarctica. Specifically, it is located in the northern part of the Antarctic Peninsula.	Europe.	of Europe.	of Europe. Specifically, it is located in the northern part of the continent.
5	In what city or state did the formation of I Am Kloot occur? Manchester → Bilbao	In which city or state was the formation of I Am Kloot?	I Am Kloot was formed in Manchester, England.	Bilbao.	I Am Kloot was formed in Bilbao.	I Am Kloot was formed in Bilbao, Spain.

Table 3: Editing cases sampled from CounterFact and zsRE under different methods.

on more challenging CounterFact, showcasing its precise differentiation of queries requiring editing without additional editing judge module.

(3) **Comparison of different metrics.** Comparing the Editing and Retention of baselines reveals a serious issue of style over-editing. For example, the Editing performance of IKE surpasses that of PROMPT, while the Retention lags behind PROMPT, indicating a negative impact of demonstration on style retention. Despite achieving commendable Editing scores, SERAC and SERAC (ChatGPT) still fall short in terms of Retention. This highlights that effective editing does not guarantee good retention, emphasizing the need for a comprehensive evaluation of knowledge editing.

5 Analysis

5.1 Generalization of PostEdit

In Section 3.1, we fine-tune post-editor to acquire the abilities of discriminating and executing edits. Therefore, it is imperative to validate the generalization of post-editor's abilities. For PostEdit and baselines, we initially utilize ChatGPT as the base LLM and CounterFact as the training set or demonstration library. Subsequently, we conduct testing under different base LLMs and datasets without re-training, as illustrated in Fig 4.

We can see that whether generalizing from CounterFact to zsRE or from ChatGPT to PaLM2[8] and LLaMA2-70B-chat[9], postEdit consistently demonstrates optimal performance in Editing and Retention scores. This substantiates the robust generalization of postEdit, highlighting its plug-and-play applicability across diverse scenarios without re-

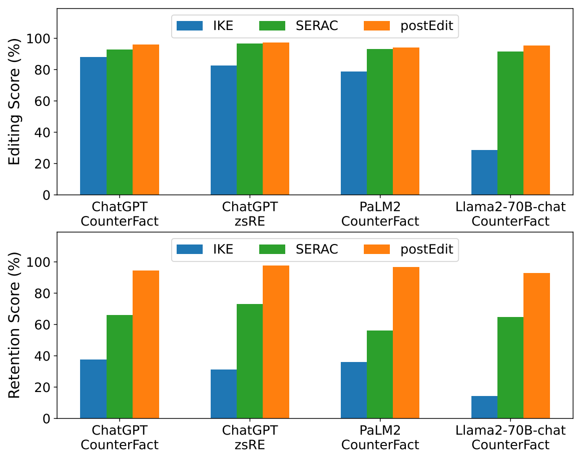


Figure 4: Performance under different base LLMs and datasets, where Editing Score is the average of TE and SE, and Retention Score is the average of TR and SR.

training. In contrast, both IKE and SERAC exhibit performance fluctuations, particularly evident in a significant decline when IKE is applied to LLaMA2-70B-chat. Further analysis reveals that conflicts between editing data and the intrinsic knowledge of LLaMA2-70B-chat lead to frequent refusals to generate responses based on edits. However, postEdit successfully mitigated the impact of knowledge conflicts through post-processing.

5.2 Case Study

To visually demonstrate the editing and style retention of postEdit and baselines, we conduct the case study in Table 3. In Case 1, postEdit accurately identifies and modifies "*French*" to "*Italian*" while maintaining the rest of the text unchanged to keep the style to the greatest extent. In contrast, IKE only responds with "*Italian*" and SERAC replies with "*Marcel Maupi was Italian*" without referencing the original response, revealing serious style over-editing. In Cases 2 and 3, postEdit respectively replaces "*FIFA (Fédération Internationale de Foot-*

[8]https://ai.google/discover/palm2
[9]https://huggingface.co/meta-LLaMA/LLaMA-2-70b-chat

Method	Semantic Editing (SE)				Semantic Retention (SR)			
	Simple	Rephrase	OOS	AVG	Simple	Rephrase	OOS	AVG
postEdit	92.5	92.1	99.4	**94.67**	93.9	94.02	99.82	**95.91**
Module Ablation								
-w/o data fillter	90.6	90.6	99.4	93.53	94.19	93.76	99.82	95.92
post-editor→ChatGPT	89.73	87.8	70.77	82.54	89.39	88.78	83.27	86.26
GPT4→ChatGPT	93.2	91.8	99.4	94.80	90.04	89.54	99.81	93.13
SBert Judgement	92.2	85.2	96.3	91.23	94.47	92.49	98.97	95.31
Training Data Ablation								
-w/o Simple	91.8	91.2	99.5	94.17	93.96	94.21	99.89	96.02
-w/o Rephrase	92	12.9	99.8	68.23	94.37	71.67	99.95	88.66
-w/o OOS	92.2	91.5	4.7	62.8	94.47	94.12	75.01	87.86

Table 4: Ablation Study on CounterFact.

ball Association)" with "*Avengers*" and modifies "*Common Lisp*" to "*JavaScript*". This demonstrates that postEdit can locate and edit spans semantically related to editing knowledge, going beyond a rudimentary replacement of old objects with new ones. Furthermore, it is evident that postEdit can handle spans logically associated with the editing. In Case 4, the location changes from "*Antarctica*" to "*Europe*", and the span in the original response, describing the location as "*the northern part of the Antarctic Peninsula*", is correspondingly adjusted to "*the northern part of the continent*". Similarly, in Case 5, as "*Manchester*" is changed to "*Bilbao*", the country is also edited from "*England*" to "*Spain*".

5.3 Ablation Study

To understand the roles of each component and training data type in PostEdit, we conduct ablation study in Table 4.

Module Ablation In our postEdit framework, we utilize GPT-4 to generate edited responses and subsequently perform data filtering. After removing data filtering, the SE score for INS queries exhibits a decline (Simple -1.9 and Rephrase -1.5), indicating that data filtering effectively enhances the quality of training data. Replacing the post-editor with ChatGPT results in a noticeable decline in performance across different types. This suggests that LLMs like ChatGPT are not proficient performing such editing tasks, highlighting the need for fine-tuning the post-editor. Substituting GPT-4 with ChatGPT for edited response augmentation results in a slight SE score increase (avg +0.13) but a significant SR score decrease (avg -2.78). This indicates that ChatGPT lacks the fine-grained granularity in editing compared to GPT-4, thereby resulting in a coarser-grained post-editor. Finally, we introduce the editing judging module, the same as SERAC, throught comparing the SBERT semantic similarity with a threshold. The observed decrease in Rephrase and OOS scores demonstrates the superior discriminative capability of the post-editor.

Training Data Ablation We further conduct data ablation by removing each type of data from the

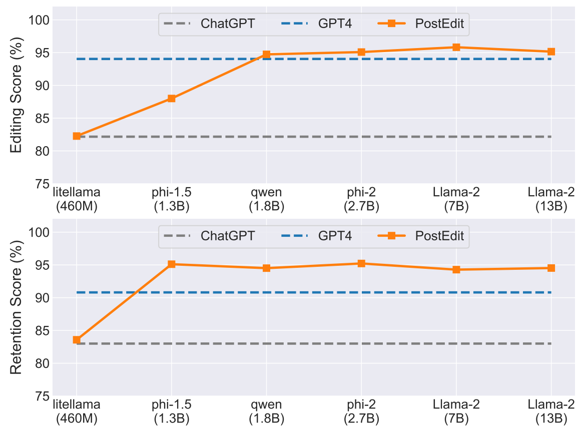


Figure 5: Performance curves of the post-editor at different scales on CounterFact.

training set. We observe that removing Simple data has no notable impact, while the removal of Rephrase data leads to a significant drop (-79.2) in the SE metric. This indicates that Rephrase data plays a crucial role in improving the post-editor's ability for editing knowledge injection and generalization, while relying solely on Simple data doesn't suffice for achieving the post-editor's generalization. After removing OOS data, although there is a noticeable decline in OOS metrics, the metrics for Simple and Rephrase do not show a discernible improvement. This indicates that post-editor doesn't excessively compromise its ability to perform edits when learning to discriminate editing.

5.4 Effect of Post-editor Scale

To investigate the effect of post-editor scale on performance, we compare evaluation scores across models ranging from 460M to 13B in size. As illustrated in Fig 5, it is evident that with the increase in post-editor scale, editing scores gradually improve (significant from 460M to 1.8B, followed by slower gains beyond 1.8B), while retention score remains stable after reaching 1.3B. This suggests that editing ability is more influenced by the model scale, and a larger post-editor can enhance editing performance while maintaining the retention. We also compare the effectiveness of post-editor with zero-shot ChatGPT and GPT-4. Similar to the findings in Section 5.3, LLMs like ChatGPT are not proficient in executing the editing task. Therefore, on CounterFact, the performance of the 460M post-editor is comparable to ChatGPT, and the 1.8B post-editor surpasses GPT-4. This indicates that the postEdit framework does not rely on a large-scale post-editor, and small-sized editors can achieve satisfactory performance and high efficiency.

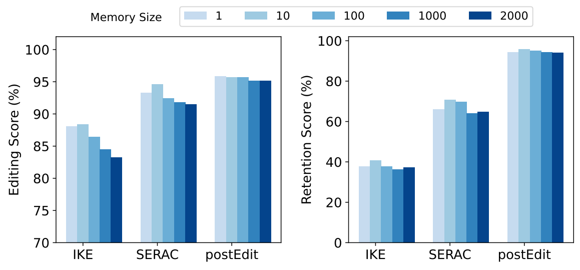


Figure 6: Performance of methods under different Editing Memory size on CounterFact.

5.5 Effect of Memory Size

It should be noted that the default evaluation procedure of KE is to edit a single piece of knowledge, assess, and then roll back the editing before repeating the process for each test point. However, in real-world scenarios, as the world evolves, edited knowledge should be continuously infused and preserved, i.e., the size of Editing Memory will continue to expand[10]. For the edit retrieved from Editing Memory, IKE utilizes the base LLM itself, SERAC applies a similarity threshold, and postEdit employs the post-editor to determine whether the query is within the scope of editing. We evaluate the performance of these methods under varying memory sizes in Fig 6. With the same retriever, postEdit exhibits the highest robustness among methods in both Editing and Retention scores, substantiating the superiority of the postEdit mechanism in discerning the necessity of editing.

5.6 Discussion on Efficiency

Apart from Editing and Retention performance, KE methods should strive to minimize storage and computational costs. For memory-based black-box LLM editing, in addition to Editing Memory and the retriever, storage overhead also encompasses the demostration library for IKE, judge model and surrogate model for SERAC, and the post-editor for postEdit. Furthermore, although memory-based methods do not incur computational overhead for editing,they do introduce inference expenses. Specifically, for IKE, the inference cost increases from $f_{base}(x)$ to $f_{retr}(x, M_e) + f_{base}(demos, e, x)$; for SERAC, the additional cost is $f_{retr}(x, M_e) + f_{judge}(x, e_{retr})$; and for postEdit, it is $f_{retr}(x, M_e) + f_{edit}(e, x, y_o)$. To further reduce post-editing overhead, one approach is to improve the reasoning efficiency of the post-editor. As highlighted in Section 5.4, a small-scale post-editor can also achieve commendable performance. Another potential option is to employ white-box parameter-editing methods to directly integrate editing knowledge into the post-editor. The post-editor can then use its knowledge to modify the original response of base LLM, exchanging editing costs for memory storage and retrieval expenses.

[10]In some studies, this corresponds to Batch Editing and Sequence Editing.

6 Related Work

Knowledge Editing The initial methods for knowledge editing involve updating parameters using constrained fine-tuning (Sotoudeh and Thakur, 2019; Sinitsin et al., 2020; Zhu et al., 2020). Recent studies mostly center around hyper-network and attribution. Hyper-network-based approaches (De Cao et al., 2021; Mitchell et al., 2021) train a hyper-network to capture gradient changes for specific edits, while attribute-based methods (Dai et al., 2022; Meng et al., 2022a,b; Li et al., 2023; Wu et al., 2023) locate neuron activations in networks for targeted parameter updates. However, these approaches exclusively focus on editing in white-box LLM scenarios, overlooking concerns related to editing data privacy and stylistic consistency. Consequently, we propose a novel evaluation framework and postEdit method for black-box LLM editing to address these issues.

Post-processing Methods Some studies have applied post-processing methods to other tasks. Cao et al. (2020) fine-tune a BART model to improve factual consistency in abstractive summarization by using summaries with errors as input and original or gold summaries as training targets. Thorne and Vlachos (2021) fine-tune a T5 model to correct factual errors by recovering masked statements based on retrieved evidence. RARR (Gao et al., 2023) employs PaLM with few-shot demonstrations for error correction and attribution report generation. Different from these studies, PostEdit is the first to apply post-processing to the knowledge editing task, fine-tuning a language model to simultaneously determine query relevance within the editing scope and make fine-grained modifications.

7 Conclusion

In this paper, we firstly introduce a comprehensive evaluation framework for knowledge editing under black-box LLMs, incorporating multiple perspectives and considering the style retention. Next, we propose a novel postEdit framework to address existing issues in privacy leakage of editing data and style over-editing in current methods by post-processing the output of LLMs. Finally, experiments on two benchmarks and thorough analysis demonstrate that postEdit outperforms all baselines and achieves strong generalization.

Limitations

This paper primarily investigates the assessment and methodology of knowledge editing in black-box LLM scenarios. The proposed evaluation framework can comprehensively assess edited responses from multiple perspectives, and the postEdit method effectively addresses issues related to privacy concerns of editing data and style over-editing. However, our work also has several limitations: (1) Although our proposed evaluation framework and postEdit method mainly focus on knowledge editing in black-box LLM scenarios, they can be equally applied to editing in white-box LLM scenarios. Due to constraints in length and the focus of the paper, we haven't thoroughly explored this in the paper. (2) Although the postEdit framework does not require retraining when injecting editing knowledge, it still necessitates an initial fine-tuning phase to enable the post-editor to learn the ability to discern whether a query is within the editing scope and how to perform the editing, resulting in a certain computational load. (3) Our study primarily investigates the application of knowledge editing in knowledge question answering tasks, similar to previous research. We believe that our framework can be extended to other scenarios, such as fact-checking and sentiment editing. We leave these explorations for future research.

Ethic Consideration

In this paper, we propose a knowledge editing approach that can be flexibly applied downstream to post-process the outputs of LLMs, effectively safeguarding the privacy of downstream private editing data and maintaining consistency in the style of the LLM. While the purpose of knowledge editing is to rectify errors or outdated knowledge in LLMs, malicious knowledge editing may lead to the generation of harmful or inappropriate outputs by the model. Therefore, ensuring secure and responsible practices in knowledge editing is of paramount importance. The application of these techniques should be guided by ethical considerations, with safeguard measures in place to prevent misuse and mitigate the potential for harmful outcomes. Additionally, due to the difficulty in obtaining continuously up-to-date knowledge, some KE datasets such as CounterFact use counterfactual knowledge to validate the effectiveness of methods. Furthermore, the base LLM, such as ChatGPT used in this work, merely serves as a demonstration of research on knowledge editing in black-box model scenarios. We emphasize that these datasets and LLMs are solely for academic exploration and do not involve actual applications in real-world scenarios, nor do they include content modification or attacks on commercially used LLMs.

References

Ning Bian, Xianpei Han, Le Sun, Hongyu Lin, Yaojie Lu, and Ben He. 2023. Chatgpt is a knowledgeable but inexperienced solver: An investigation of commonsense problem in large language models. *arXiv preprint arXiv:2303.16421*.

Meng Cao, Yue Dong, Jiapeng Wu, and Jackie Chi Kit Cheung. 2020. Factual error correction for abstractive summarization models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6251–6258, Online. Association for Computational Linguistics.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109*.

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, Dublin, Ireland. Association for Computational Linguistics.

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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

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

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2023. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508, Toronto, Canada. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*.

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

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2023. Pmet: Precise model editing in a transformer. *arXiv preprint arXiv:2308.08742*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Jiacheng Liu, Wenya Wang, Dianzhuo Wang, Noah A Smith, Yejin Choi, and Hannaneh Hajishirzi. 2023a. Vera: A general-purpose plausibility estimation model for commonsense statements. *arXiv preprint arXiv:2305.03695*.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2023b. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685*.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. *arXiv e-prints*, pages arXiv–2308.

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

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

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

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese 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, Hong Kong, China. Association for Computational Linguistics.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitriy Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. *arXiv preprint arXiv:2004.00345*.

Matthew Sotoudeh and A Thakur. 2019. Correcting deep neural networks with small, generalizing patches. In *Workshop on safety and robustness in decision making*.

James Thorne and Andreas Vlachos. 2021. Evidence-based factual error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3298–3309, Online. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Cunxiang Wang, Sirui Cheng, Zhikun Xu, Bowen Ding, Yidong Wang, and Yue Zhang. 2023a. Evaluating open question answering evaluation. *arXiv preprint arXiv:2305.12421*.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, et al. 2023b. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*.

Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. 2023. DEPN: Detecting and editing privacy neurons in pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2875–2886, Singapore. Association for Computational Linguistics.

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

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

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

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

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

A Details of Evaluation

A.1 Details of Existing Metrics

There are three metrics mainly used to evaluate the performance of knowledge editing in previous work, namely Efficacy, Generalization, and Specificity.

- **Efficacy** measures the accuracy of knowledge editing using **ES** (Efficacy Score) and **EM** (Efficacy Magnitude). For Simple type queries, the meaning of ES is $E\left[I\left[P(o^*) > P(o)\right]\right]$, and EM is obtained by $E[P(o^*) - P(o)]$.

- **Generalization** measures the accuracy of knowledge editing on Rephrase queries by using **RS** (Rephrase Score) and **RM** (Rephrase Magnitude). For Rephrase type queries, RS and RM are actually calculated to derive ES and EM under the condition of rephrasing queries.

- **Specificity** uses **NS** (Neighborhood Score) and **NM** (Neighborhood Magnitude) to measure the ability of knowledge editing to preserve unrelated knowledge. When dealing with OOS queries beyond the editing scope, no editing should take place, and the original facts should be preserved. Therefore, NS is obtained by $E\left[I\left[P(o) > P(o^*)\right]\right]$, and NM is obtained by $E[P(o) - P(o^*)]$.

Human Score	Auto Metric	Pearson Correlation
Editing	TE	0.7644
	SE	0.7784
	Editing	0.8074
Retention	TR	0.9195
	SR	0.8868
	Retention	0.9255
Overall	Editing	0.5356
	Retention	0.7612
	Overall	0.839

Table 5: The Pearson correlation coefficient between auto metrics and manual scores. For the auto metrics, Editing is the average of TE and SE; Retention is the average of TR and SR; Overall is the average of Editing and Retention.

A.2 Consistency with Human Evaluation

In Section 2.2.2, we proposed a comprehensive evaluation framework, incorporating editing metrics (TE, SE) and retention metrics (TR, SR) to evaluate the quality of output text after knowledge editing. Prior to employing these metrics for evaluation, it was imperative to ensure their validity and necessity. To address this, we extracte 300 data points from the test set (comprising Simple, Rephrase, and OOS examples in a 1:1:1 ratio) and enlist human evaluators to independently score them from the perspectives of editing, retention, and overall assessment.

The rules for human scorers scoring the effectiveness of knowledge editing are as follows: in terms of editing, for INS queries, scoring is as follows: 0 points if there is no editing at all; 0.5 points if there are partial edits, and the sentence still retains old knowledge or exhibits logical inconsistencies; 1 point for perfect knowledge editing with no issues. For OOS queries, the scoring rules are reversed. In the retention aspect, after disregarding content related to the edited knowledge in the sentence, for responses within the editing scope: 0 points for very poor consistency between new and old responses; 0.5 points for ordinary consistency; 1 point for excellent consistency. In the overall aspect, human scorers are required to consider the overall impact of knowledge editing and assign scores within the range of 0, 1, 2, 3, 4 to the edited outputs. Then, we conduct Pearson correlation analyses between these human scores and our automated metrics.

As shown in Table 5, both textual metrics (TE, TR) and semantic metrics (SE, SR) demonstrate commendable consistency scores with human ratings, affirming the effectiveness of the proposed metrics. Moreover, Whether for editing or retention, the consistency score of the joint assessment of textual and semantic dimensions surpasses that of any individual metric. This underscores the necessity of incorporating both textual and semantic metrics in the evaluation process. Finally, the Pearson correlation coefficient between auto editing and human overall score is a mere 0.5356. However, a combined evaluation of editing and retention metrics yield a significantly higher consistency score of 0.839 with human judgments. This suggests that effective alignment with human preferences cannot rely solely on editing scores but requires a comprehensive assessment integrating both editing and retention metrics.

A.3 Pseudo-code of Evaluation Framework

We summarize the pseudo-code of our proposed evaluation framework in Algorithm 1.

B Details of Method

B.1 Pseudo-code of PostEdit

We summarize the pseudo-code for training post-editor and inference of postEdit in Algorithm 2 and Algorithm 3, respectively.

B.2 Details of Prompts

We demonstrate the two prompt templates T^{aug} and T^{edit} used in the postEdit method as follows:

Prompt Template T^{aug}

For the following query and original response, you need to follow in order:
Firstly, locate all spans related to the **old fact:{s} {r} {o}** in original reply;
Secondly, modify these spans according to **new fact: {s} {r} {o*}**.
Thirdly, output the edited response based on the modified spans (Do not output other content).
The query:
$\{x\}$
Original response:
$\{y_o\}$
Edited response:

Prompt Template T^{edit}

Instruction:
You will assume the role of an editor. For the following query and original response, if the new fact impacts the query or original response, incorporate the new fact into the original response. If not, simply output the following word: retain.
New fact:
The answer of $\{s\}$ $\{r\}$ has been updated from $\{o\}$ to $\{o^*\}$.
The query:
$\{x\}$
Original response:
$\{y_o\}$
Edited response:

C Details of Experiments

C.1 Details of Datasets

In this work, we mainly used two datasets: zsRE and CounterFact.

- **zsRE** (Levy et al., 2017) is one of the most popular question answering (QA) datasets which use question rephrasing as the equivalence neighborhood. These queries of Rephrase type are generated by back-translation. In zsRE, the relationship between entities is associated with a set of crowd-sourced generated questions. Additionally, zsRE associates questions with randomly generated sentences to add out-of-editing scope examples.

- **CounterFact** (Meng et al., 2022a) is a more challenging dataset than zsRE, the expected output of which is contradictory to the fact. It is built to distinguish superficial alterations in the word selections and significant, generalized modifications in its foundational factual knowledge. In CounterFact, the edited answer to the question can sometimes be counterfactual to real world, which makes it harder for the model to predict desired answer and avoid the effects of pre-trained LLMs knowing these desired facts before editing.

Following the previous work (Zheng et al., 2023), for CounterFact, we designate data with edit id numbers ranging from 0 to 2000 as the test set for

Dataset	Data Type	Train Number	Test Number	Length of Original Response(mean/max)
CounterFact	ALL	30000	1500	51.34/436
	Simple	10000	500	50.40/436
	Rephrase	10000	500	53.03/374
	OOS	10000	500	50.59/367
zsRE	ALL	30000	1500	22.39/406
	Simple	10000	500	14.84/119
	Rephrase	10000	500	18.38/257
	OOS	10000	500	33.96/406

Table 6: Statistical information on the sampled datasets.

knowledge edit, while the remaining data constitute the training set. As we adopt ChatGPT as our base LLM in main experiments, in order to control the dataset size, we randomly sampled 30,000 examples from the original training set (10,000 each for Simple, Rephrase, and OOS). These samples constitute our training set. Additionally, we randomly selected 1,500 examples from the original test set (500 each for Simple, Rephrase, and OOS) to create our query test set. Table 6 displays detailed information about our sampled dataset.

C.2 Details of Baselines

- **IKE** (Zheng et al., 2023) is a method of knowledge editing that does not involve modifying the parameters of LLMs. It defines three types of demonstration formatting templates including copy, update, and retain. These templates serve distinct functions and act as guiding principles for the language model, enabling it to edit knowledge through in-context learning, allowing IKE to maintain both efficiency and excellent generalization and specificity. This opens up the possibility of employing IKE for the task of knowledge editing even in scenarios involving black-box models.

- **PROMPT** (Zheng et al., 2023) is similar to IKE, as a method of knowledge editing through in-context learning. However, unlike IKE, PROMPT doesn't require constructing three types of demonstrations but directly provides new knowledge to the LLM for knowledge editing.

- **SERAC** (Mitchell et al., 2022) is a memory-based method of knowledge editing. This method stores edits in explicit memory and learns to reason about these edits as needed to adjust the predictions of the base LLM without modifying parameters. SERAC uses an explicit cache of user-provided edit descriptors, alongside a scope classifier and surrogate model. When presented with a query, SERAC uses the scope classifier to determine if the query falls within the editing scope. If it does, the output is predicted via the surrogate model; otherwise, it defers to the base LLM for the output.

- **SERAC (ChatGPT)** In SERAC, the surrogate model is obtained by fine-tuning a smaller language model compared to the base LLM. We utilize ChatGPT as the surrogate model to derive a SERAC variant that requires no additional training.

C.3 Details of Implementation

As described in Section 2.2.2, our evaluation framework employs a NLI model for computing SE, ROUGE scores for computing TR, and a SBERT model for computing SR. In details, SE utilizes albert-xxlarge-v2-snli_mnli_fever_anli_R1_R2_R3-nli[11] as the NLI model; ROUGE score is implemented through the rouge library[12], using the F1 score of ROUGE-1; SR uses all-MiniLM-L6-v2[13] as the SBERT model.

For training of post-editor, we employ ChatGPT (gpt-3.5-turbo-0301) for original response augment and GPT-4 (gpt-4-0613) for edited response augment [14], with the default temperature coefficient ($t = 0.1$). In order to enhance training efficiency and reduce the number of updated parameters, we adopt the LoRA

[11]https://huggingface.co/ynie/albert-xxlarge-v2-snli_mnli_fever_anli_R1_R2_R3-nli

[12]https://pypi.org/project/rouge

[13]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[14]https://platform.openai.com/docs/models

strategy (Hu et al., 2021) to finetune LLaMA 2-7B. Specifically, the rank of LoRA is set to 8, with $lora\_alpha$ at 16 and $lora\_dropout$ at 0.05. The LoRA update matrix is applied to the self-attention and FFN layers, with $target\_modules$ as ["q_proj","k_proj","v_proj","o_proj","gate_proj", "down_proj","up_proj"]. We train 5 epochs to optimize post-editor, employing a batch size of 128 and a learning rate of 5e-2. We also use the warmup and cosine annealing strategy, with a warmup ratio of 0.1 and the Adam optimizer.

For retriever of postEdit, consistent with all baselines, we use all-MiniLM-L6-v2 to encode queries and edit knowledge, while employing dot product as the similarity function. For base LLM, we use ChatGPT (gpt-3.5-turbo-0301) in main experiments, with a temperature coefficient of 0.1. During inference of post-editor, we set the temperature coefficient of 0.1 and use beam search to decode the output, where num_beams is set to 4. To further improve the inference speed, we apply 8-bit quantization when loading post-editor.

In terms of baselines, for SERAC, we fine-tune the surrogate model using the same LLAMA2-7B as post-editor and the similarity discrimination threshold is set at 0.7, determined through hyperparameter search on the training set (ranging from 0.1 to 0.9 with a step size of 0.1); for IKE, we set the number of demonstration examples to 32. The rest of the hyperparameter settings for the baselines follow the default configurations in their original papers. All experiments use a single Nvidia A100 GPU (80 GB of memory).

Algorithm 1: Pseudo-code of Evaluation Framework in a Python-like style.

```
# x: the input of LLM (All text is processed in lowercase, the same below.)
# x_label: "INS" if x in editing scope else "OOS"
# y_o, y_e: the original and edited output of LLM
# o_old, o_new: the object of old knowledge $t$ and new knowledge $t^*$ for editing
# k_old, k_new: text format of $t$ and $t^*$
# k_self: text format of LLM's self-knowledge $t_o$ and is equivalent to [x, y_o]
# func_entail(a,b): return True if a entails b else False by using a NLI model
# func_rouge(a,b): return the ROUGE socre of a and b
# func_sim(a,b): return the similarity of a and b using a SBERT model

def TE(y_e, x_label, o_old, o_new):
    ctn_old=1 if o_old in y_e else 0
    ctn_new=1 if o_new in y_e else 0
    if x_label=="INS":
        TE_score=0.5*ctn_new + 0.5*(1-ctn_old)
    else:
        TE_score=0.5*ctn_old + 0.5*(1-ctn_new)
    return TE_score

def SE(x_label, x, y_e, k_old, k_new, k_self, func_entail):
    ent_new=1 if func_entail(x+" "+y_e,k_new) else 0
    if x_label=="INS":
        ent_old=1 if func_entail(x+" "+y_e,k_old) else 0
        SE_score=0.5 * ent_new + 0.5 * (1-ent_old)
    else:
        ent_old=1 if func_entail(x+" "+y_e,k_self) else 0
        SE_score=0.5*ent_old + 0.5*(1-ent_new)
    return SE_score

def TR(x_label, y_o, y_e, o_old, o_new, func_rouge):
    if x_label=="INS":
        TR_score=func_rouge(y_o.replace(o_old,"mask"), y_e.replace(o_new,"mask"))
    else:
        TR_score=func_rouge(y_o,y_e)
    return TR_score

def SR(x_label, y_o, y_e, o_old, o_new, func_sim):
    if x_label=="INS":
        SR_score=func_sim(y_o.replace(o_old,"mask"), y_e.replace(o_new,"mask"))
    else:
        SR_score=func_sim(y_o,y_e)
    return SR_score
```

Algorithm 2: Train post-editor

```
Data: training dataset $D_{train} = \{(e_i, x_i)\}$
Require: base LLM $f_{base}$, GPT-4 $f_{gpt4}$, trainable generative model $f_{edit}$, training epoch E, batch
  size B
for $i$ in $1, \cdots, |D_{train}|$ do
    $y_{i,o}^{aug} = f_{base}(x_i)$                                    ▷Original Response Augment
    if $x_i \in \mathcal{X}_e$ then
        $y_{i,e}^{aug} = f_{gpt4}(T^{aug}(e_i, x_i, y_{i,o}^{aug}))$            ▷Edited Response Augment
        if $\text{TE}(y_{i,e}^{aug}) \neq 1$ or $\text{SE}(y_{i,e}^{aug}) \neq 1$ then
            del $(e_i, x_i, y_{i,o}^{aug}, y_{i,e}^{aug})$
        end
    else
        $y_{i,e}^{aug} = \langle Retain \rangle$
    end
end
$D_{train}^{aug} = \{(e_i, x_i, y_{i,o}^{aug}, y_{i,e}^{aug})\}$
for epoch in $1, \cdots, E$ do
    for $iter$=$0, 1, 2, \cdots$ do
        sample a mini-batch B from $D_{train}^{aug}$                     ▷Supervised Fine-tuning
        compute $\mathcal{L}_{sft}$ by equation 6 and optimize $f_{edit}$
    end
end
Output: trained post-editor $f_{edit}$
```

Algorithm 3: Inference of PostEdit

```
Input: use query $x$
Require: Editing Memory $M_e$, base LLM $f_{base}$, post-editor $f_{edit}$, SBERT retriever $f_{retr}$
get original response: $y_o = f_{base}(x)$
get the most similar edit index: $i^* = \text{argmax}_{0 \leq i < |M_e|}\ \text{sim}(x, e_i)$
get post-editor's output: $f_{edit}(x_{edit}) = f_{edit}(T^{edit}(e_i^*, x, y_o))$
if $f_{edit}(x_{edit}) \neq \langle Retain \rangle$ then
    $y_e = f_{edit}(x_{edit})$
else
    $y_e = y_o$
end
Output: final response $y_e$
```