Editing Large Language Models: Problems, Methods, and Opportunities

Yunzhi Yao[♣*], **Peng Wang**[♣*], **Bozhong Tian**[♣], **Siyuan Cheng**[♣], **Zhoubo Li**[♣],
Shumin Deng[♡], **Huajun Chen**[♣♠], **Ningyu Zhang**[♣†]
[♣] Zhejiang University [♠]Donghai Laboratory
[♡] National University of Singapore
{yyztodd,peng2001,tbozhong,sycheng,zhoubo.li}@zju.edu.cn
{huajunsir,zhangningyu}@zju.edu.cn,shumin@nus.edu.sg

Abstract

Recent advancements in deep learning have precipitated the emergence of large language models (LLMs) which exhibit an impressive aptitude for understanding and producing text akin to human language. Despite the ability to train highly capable LLMs, the methodology for maintaining their relevancy and rectifying errors remains elusive. To that end, the past few years have witnessed a surge in techniques for editing LLMs, the objective of which is to alter the behavior of LLMs within a specific domain without negatively impacting performance across other inputs. This paper embarks on a deep exploration of the problems, methods, and opportunities relating to model editing for LLMs. In particular, we provide an exhaustive overview of the task definition and challenges associated with model editing, along with an in-depth empirical analysis of the most progressive methods currently at our disposal. We also build a new benchmark dataset to facilitate a more robust evaluation and pinpoint enduring issues intrinsic to existing techniques. Our objective is to provide valuable insights into the effectiveness and feasibility of each model editing technique, thereby assisting the research community in making informed decisions when choosing the most appropriate method for a specific task or context[1].

1 Introduction

Large language models (LLMs) have demonstrated a remarkable capacity for understanding and generating human-like text (Brown et al., 2020; OpenAI, 2023; Anil et al., 2023; Touvron et al., 2023; Qiao et al., 2022; Zhao et al., 2023). Despite of the proficiency in training LLMs, the strategies for ensuring their relevance and fixing their bugs remain

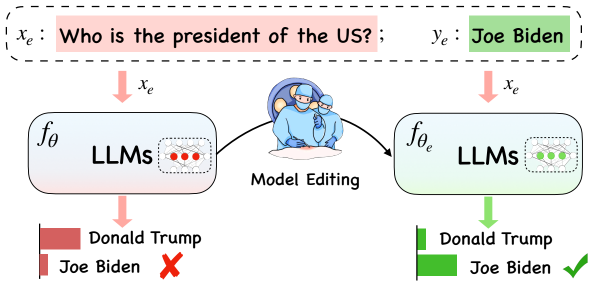


Figure 1: Model editing to fix and update LLMs.

unclear. Ideally, as the world's state evolves, we aim to update LLMs in a way that sidesteps the computational burden associated with training a wholly new model. To address this need, as shown in Figure 1, the concept of **model editing** has been proposed (Sinitsin et al., 2020; De Cao et al., 2021), to enable data-efficient alterations to the behavior of models, specifically within a designated realm of interest, while ensuring no adverse impact on performance across other inputs.

At present, numerous works on model editing for LLMs (De Cao et al., 2021; Meng et al., 2022, 2023; Sinitsin et al., 2020; Huang et al., 2023) have made strides in various editing tasks and settings. As illustrated in Figure 2, these works manipulate the model's output for specific cases by either integrating an auxiliary network with the original, unchanged model or altering the model parameters responsible for undesirable output. Despite the wide range of model editing techniques present in the literature, a comprehensive comparative analysis, assessing these methods in uniform experimental conditions, is notably lacking. This absence of direct comparison impairs our capacity to discern the relative merits and demerits of each approach, consequently hindering our comprehension of their adaptability across different problem domains.

To confront this issue, the present study endeavors to establish a standard problem definition accompanied by a meticulous appraisal of these meth-

[*]Equal contribution.
[†]Corresponding author.
[1]Code and datasets will be available at https://github.com/zjunlp/EasyEdit.

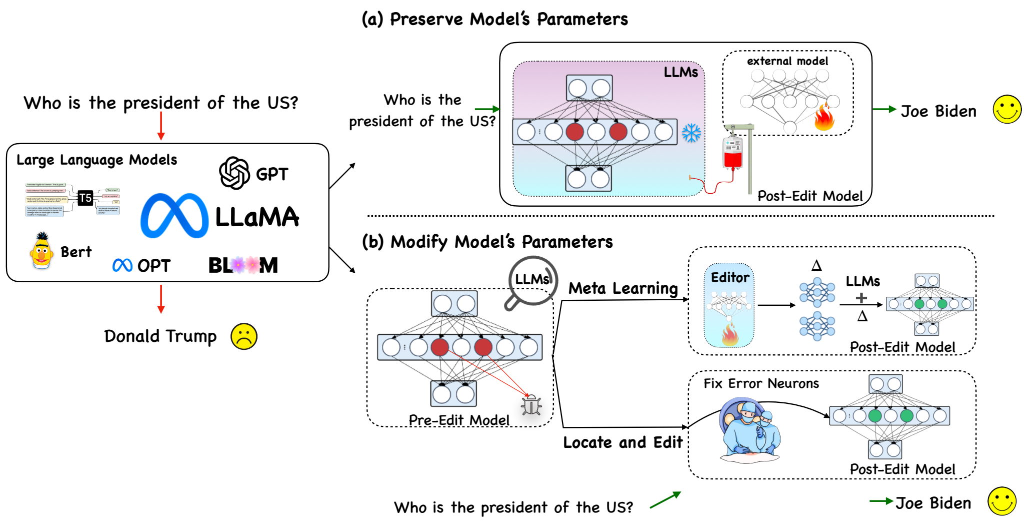


Figure 2: An overview of two paradigms of model editing for LLMs.

ods. We conduct experiments under regulated conditions, fostering an impartial comparison of their respective strengths and weaknesses. We initially select two popular model editing datasets, ZsRE and COUNTERFACT, as our foundational datasets and choose two language models with divergent structures - T5 (encoder-decoder) (Raffel et al., 2020a) and GPT-J (decoder only) (Wang and Komatsuzaki, 2021a), as the base models. By systematically evaluating their performance, we aim to impart valuable insights into the effectiveness and feasibility of each model editing technique, ultimately assisting the research community in making informed decisions when selecting the most suitable method for a specific task or context. While we observe that current methods have demonstrated considerable proficiency in factual model editing tasks, it's also noticeable, through the creation of a more encompassing evaluation dataset, that existing approaches fall short in terms of robust generalization capabilities. Moreover, both the performance and efficiency of model editing is somewhat limited, thereby constraining their practical application. Our contributions include:

- We present the first exhaustive empirical analysis of approaches for editing LLMs, delving into their respective strengths and weaknesses.
- We construct a novel dataset aimed at shedding light on the current shortcomings of model editing methods, particularly in terms of generalization and efficiency.
- We outline potential future research opportunities in the field of model editing.

2 Problems Definition

Model editing, as elucidated by Mitchell et al. (2022b), aims to adjust an initial base model f_θ (where θ signifies the model's parameters) using an edit descriptor (x_e, y_e), which succinctly encapsulates the intended modifications in the model's performance. The ultimate goal is to create an edited model, denoted as f_{θ_e}. Specifically, the basic model f_θ is represented by a function $f : \mathbb{X} \mapsto \mathbb{Y}$ that associates an input x with its corresponding prediction y. Given an edit descriptor comprising the edit input x_e and edit label y_e such that $f_\theta(x_e) \neq y_e$, the post-edit model f_{θ_e} is designed to produce the expected output, where $f_{\theta_e}(x_e) = y_e$.

The model editing process generally impacts the predictions for a broad set of inputs that are closely associated with the edit example. This collection of inputs is referred to as the **editing scope**. A successful edit should adjust the model's behavior for examples within the editing scope while leaving its performance for out-of-scope examples unaltered:

$$f_{\theta_e}(x) = \begin{cases} y_e & \text{if } x \in I(x_e, y_e) \\ f_\theta(x) & \text{if } x \in O(x_e, y_e) \end{cases} \quad (1)$$

The *in-scope* $I(x_e, y_e)$ usually encompasses x_e along with its equivalence neighborhood $N(x_e, y_e)$, which includes related input/output pairs. In contrast, the *out-of-scope* $O(x_e, y_e)$ consists of inputs that are unrelated to the edit example. The post-edit model f_e should satisfy the following three properties: **reliabilty**, **generality** and **locality**.

		Approach	Additional Training	Online Edit	Batch Edit	Edit Area	Editor Parameters	Efficient Edit
Preserve Parameters	Memory-based	SERAC	YES	YES	YES	External Model	$Model_{cf} + Model_{Classifier}$	YES
		CaliNET	NO	YES	YES	FFN	$N * neuron$	YES
		T-Patcher	NO	NO	NO	FFN	$N * neuron$	NO
Modify Parameters	Meta-learning	KE	YES	YES	YES	FFN	$Model_{hyper} + L * mlp$	NO
		MEND	YES	YES	YES	FFN	$Model_{hyper} + L * mlp$	NO
	Locate and Edit	KN	NO	YES	NO	FFN	$L * neuron$	YES
		ROME	NO	YES	NO	FFN	mlp_{proj}	YES
		MEMIT	NO	YES	YES	FFN	$L * mlp_{proj}$	YES

Table 1: Comparisons between several existing model editing approaches. "Additional Training" refers to whether the methods need training before conducting specific edits. "Online Edit" refers to quickly editing an individual target knowledge. "Batch Edit" refers to editing multiple target knowledge simultaneously. "Editor Area" refers to the specific region of the LLMs that the methods aim to modify. FFN demonstrates the feed-forward module. "Editor Parameters" refers to the parameters that need to be updated for editing. L denotes the number of layers to update. mlp is FFN and mlp_{proj} is the second linear layer in FFN. $neurons$ denotes the key-value pair in FFN. N represents the quantity of $neuron$ to be updated within a single layer. "Editor Efficient" refers to the efficiency of editing an individual target knowledge, considering the time and memory cost.

Reliability Previous works (Huang et al., 2023; De Cao et al., 2021; Meng et al., 2022) define a reliable edit when the post-edit model f_{θ_e} gives the target answer for the case (x_e, y_e) to be edited. The reliability is measured as the average accuracy on the edit case:

$$\mathbb{E}_{x'_{\mathrm{e}}, y'_{\mathrm{e}} \sim \{(x_{\mathrm{e}}, y_{\mathrm{e}})\}} \mathbb{1}\left\{\operatorname{argmax}_y f_{\theta_e}\left(y \mid x'_{\mathrm{e}}\right) = y'_{\mathrm{e}}\right\} \tag{2}$$

Generality The post-edit model f_{θ_e} can edit the equivalent inputs $N(x_{\mathrm{e}}, y_{\mathrm{e}})$ (e.g. rephrased sentences). This feature is evaluated by the average accuracy of the model f_{θ_e} on examples inputs drawn uniformly from the equivalence neighborhood:

$$\mathbb{E}_{x'_{\mathrm{e}}, y'_{\mathrm{e}} \sim N(x_{\mathrm{e}}, y_{\mathrm{e}})} \mathbb{1}\left\{\operatorname{argmax}_y f_{\theta_e}\left(y \mid x'_{\mathrm{e}}\right) = y'_{\mathrm{e}}\right\} \tag{3}$$

Locality Editing should be implemented locally, which means the post-edit model f_{θ_e} should remain accurate on the irrelevant examples in the out-of-scope $O(x_e, y_e)$ examples. Hence, the locality is evaluated by computing whether the post-edit model f_{θ_e} generates the same output as the pre-edit f_θ model:

$$\mathbb{E}_{x'_{\mathrm{e}}, y'_{\mathrm{e}} \sim O(x_{\mathrm{e}}, y_{\mathrm{e}})} \mathbb{1}\left\{f_{\theta_e}\left(y \mid x'_{\mathrm{e}}\right) = f_\theta\left(y \mid x'_{\mathrm{e}}\right)\right\} \tag{4}$$

3 Current Methods

Current model editing methods for LLMs can be categorized into two main paradigms as shown in Figure 2: 1) modifying the model's internal parameters; 2) applying extra parameters while keeping the original model frozen. The conceptual summary of comparisons between these approaches can be found in Table 1. In the following part, we will introduce and discuss these methods.

3.1 Methods for Preserve LLMs' Parameters

This paradigm keeps the original parameters unchanged while employing an additional editor (e.g, another model) to influence the model's behavior.

Memory-based Model SERAC (Mitchell et al., 2022b) presents an approach that reroutes revised facts through a distinct set of parameters while leaving the original weights unaltered. Specifically, the model employs a scope classifier to evaluate the likelihood of new input falling within the purview of stored edit examples. If the input aligns with any edit within the cache, the edit bearing the highest probability is fetched, and the counterfactual model's prediction is returned based on the input and retrieved edit. Contrasting with the use of an additional model as the editor, T-Patcher (Huang et al., 2023) and CaliNET (Dong et al., 2022) introduce extra trainable parameters into the feed-forward module of PLMs. These are trained on a modified knowledge dataset while the original parameters remain static. Note that these two methods hinge on the concept of a Knowledge Neuron in the Feed-Forward Network (FFN). T-Patcher (Huang et al., 2023) integrates one neuron for each edit instance, whereas CaliNET incorporates several neurons for multiple edit cases.

3.2 Methods for Modify LLMs' Parameters

This paradigm would update part of the parameter θ, it applies an update Δ matrix to edit the model.

DataSet	Model	Metric	FT	SERAC	CaliNet	T-Pathcer	KE	MEND	KN	ROME	MEMIT
ZsRE	T5-XL	Reliabilty	20.71	**99.80**	5.17	30.52	3.00	78.80	22.51	-	-
		Generality	19.68	**99.66**	4.81	30.53	5.40	89.80	22.70	-	-
		Locality	89.01	98.13	72.47	77.10	96.43	**98.45**	16.43	-	-
	GPT-J	Reliabilty	54.70	90.16	22.72	97.12	6.60	45.60	11.34	99.18	**99.23**
		Generality	49.20	89.96	0.12	**94.95**	7.80	48.00	9.40	94.90	87.16
		Locality	37.24	99.90	12.03	96.24	94.18	88.21	90.03	**100.00**	**100.00**
CounterFact	T5-XL	Reliabilty	33.57	**99.89**	7.76	80.26	1.00	81.40	47.86	-	-
		Generality	23.54	**98.71**	7.57	21.73	1.40	93.40	46.78	-	-
		Locality	72.72	**99.93**	27.75	85.09	96.28	91.58	57.10	-	-
	GPT-J	Reliabilty	99.90	99.78	43.58	**100.00**	13.40	73.80	1.66	99.80	99.90
		Generality	97.53	**99.41**	0.66	83.98	11.00	74.20	1.38	86.63	73.13
		Locality	1.02	98.89	2.69	8.37	94.38	93.75	58.28	**100.00**	**100.00**

Table 2: Current model results on current datasets and evaluation metric. The settings for these models and datasets are the same with (Meng et al., 2022). '-' refers to the results that the methods empirically fail to edit LLMs.

Locate-Then-Edit This paradigm initially identifies parameters corresponding to specific knowledge and modifies them through direct updates to the target parameters. The Knowledge Neuron (KN) method (Dai et al., 2022a) introduces a knowledge attribution technique to pinpoint the "knowledge neuron" (a key-value pair in the FeedForward Network (FFN) matrix) that embodies the knowledge and then proceeds to update these neurons. ROME (Meng et al., 2022) applies causal mediation analysis to locate the edit area. Instead of modifying knowledge neurons in the FFN, ROME alters the entire matrix. However, KN and ROME can only edit one factual association at a time. To this end, MEMIT (Meng et al., 2023) is proposed which can modify a sequence of layers and facilitates thousands of alterations to be executed simultaneously. Note that these methods are predicated on the locality hypothesis of factual knowledge, which hasn't yet garnered widespread validation, changes in certain parameters could influence unrelated knowledge and yield unforeseen outcomes.

Meta-learning In contrast to locate-then-edit methodologies, meta-learning methods employ a hyper network to learn the necessary Δ for editing the Language Model (LM). Knowledge Editor (KE) (De Cao et al., 2021) leverages a hyper network (specifically, a bidirectional-LSTM) to predict the weight update for each data point, thereby enabling the constrained optimization of editing target knowledge without disrupting others. However, this approach falls short when it comes to editing LLMs. To overcome this limitation, Model Editor Networks with Gradient Decomposition (MEND) (Mitchell et al., 2022a) enables efficient local edits to language models using a single input-output pair. Concretely, MEND learns to transform the gradient of fine-tuned language models by employing a low-rank decomposition of gradients, which can be applied to LLMs with minimal resource expenditure. When compared to locate-then-edit approaches, meta-learning methods necessitate additional training stages, potentially leading to increased time and memory costs.

4 Preliminary Experiments

Given the prevalence of studies and available datasets focused on factual knowledge, we take it as our primary basis for comparing model editing techniques. By conducting initial controlled experiments using two well-established factual knowledge datasets as shown in Table 2, we offer a direct comparison of various methods, thereby shedding light on their unique advantages and shortcomings.

4.1 Datasets and Models

The currently commonly used datasets include ZsRE and CounterFact. **ZsRE** (Levy et al., 2017) is a Question Answering (QA) dataset using question rephrasings generated by back-translation as the equivalence neighborhood. We follow previous data split (De Cao et al., 2021; Meng et al., 2022; Mitchell et al., 2022a) to evaluate all the models on the test set. For models requiring training, we utilize the training set. Following prior work (Mitchell et al., 2022a,b), we use the Natural Questions (NQ; Kwiatkowski et al. (2019)) as out-of-scope data to evaluate locality. **CounterFact** (Meng et al., 2022) is a more challenging dataset that accounts for counterfacts that start with low scores in comparison to correct facts. It constructs out-of-scope data by substituting the subject entity for a proximate subject entity sharing a predicate. This alteration enables us to differentiate between superficial wording changes and more significant modifications that correspond to a meaningful shift in a fact. It should be noted that previous research typically conducts experiments on smaller language models (<1B) and overlooks inconsistencies in Language Models (LMs). Consequently, we opt for two larger models as the foundational models: T5-XL (3B) and GPT-J (6B), encompassing both encoder-decoder and decoder-only architectures. As several original implementations do not support both architectures, we have re-implemented them to accommodate both models. However, our empirical findings suggest that ROME and MEMIT are only suitable for decoder-only models like GPT-J, so we have not reported results for T5-XL. Since the ZsRE dataset adopts the NQ dataset to evaluate the locality, here, we use a T5-XL model (Raffel et al., 2020b) finetuned on the NQ dataset as the baseline model. As to the GPT-J (Wang and Komatsuzaki, 2021b), we use the original pre-trained version to test the locality's zero-shot results.

Beyond the existing model editing techniques, we additionally examined the results of Fine-tuning (FT), an elementary approach for model updating. To prevent the computational expense incurred by retraining all model layers, we adopted the methodology proposed by Meng et al. (2022) to fine-tune the layers identified by ROME. This strategy ensures a fair comparison with other direct editing methods, reinforcing the validity of our analysis.

4.2 Basic Results

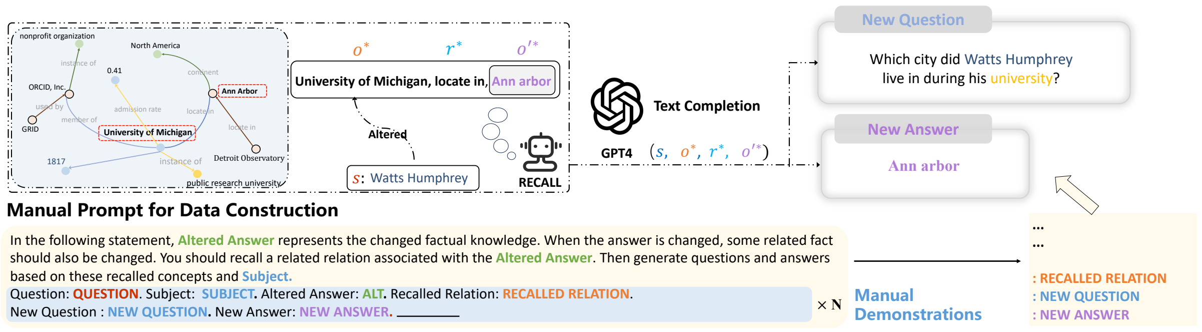
From Table 2, we notice that both SERAC and ROME exhibit exceptional performance on the ZsRE and COUNTERFACT datasets, especially SERAC, which attained results of over 90% on multiple evaluation metrics. While MEMIT does not demonstrate the same level of generality as SERAC and ROME, it performs admirably in terms of reliability and locality. Remarkably, ROME and MEMIT achieve the highest locality performance (100%) with minimal alterations to unrelated cases, even though they edit the model's intrinsic parameters. In contrast, KE, CaliNET, and KN exhibit subpar performance. These models have shown promise on smaller models in previous studies, but their editing reliability and generalization capabilities in larger models remain mediocre. MEND performs well on the two datasets, achieving over 80% in the results on T5, although not as impressive as ROME and SERAC. However, MEND's performance on GPT-J is less impressive, which may be attributed to the model size. The performance of the T-Patcher model fluctuates across different model architectures and sizes. For instance, it underperforms on T5-XL for the ZsRE dataset, while it performs perfectly on GPT-J. In the case of the COUNTERFACT dataset, T-Patcher achieves satisfactory reliability and locality on T5 but lacks generality. Conversely, on GPT-J, the model excels in reliability and generality but underperforms in the locality. This instability can be attributed to both the model architecture and the dataset. T-Patcher adds a neuron in the final decoder layer for T5; however, the encoder may still retain the original knowledge. Additionally, the poor locality performance on the COUNTERFACT dataset may result from the dataset's construction, which involves substituting the subject in the sentence. Consequently, T-Patcher may be vulnerable to similar but unrelated sentences, leading to suboptimal performance. In terms of FT, its performance does not measure up favorably compared to ROME, even when modifying the same position in PLMs. When applied to the ZsRE dataset, FT's performance across several metrics is less than optimal. On the COUNTERFACT dataset using GPT-J, FT's reliability and generalization capabilities match those of ROME. However, its locality score is relatively low, suggesting that FT can inadvertently influence unrelated knowledge areas. This indicates a critical limitation of FT, demonstrating the superiority of ROME in preserving the integrity of unrelated knowledge within the model.

5 Comprehensive Study

Given the aforementioned considerations, we argue that previous evaluation metrics may be too restrictive to fully assess the model editing capabilities, thus we provide more comprehensive evaluations regarding portability, locality and efficiency.

5.1 Portability

Many studies assess the generality of samples generated through back translation (De Cao et al., 2021), but this often overlooks the impact of knowledge editing beyond simple paraphrasing. These



In the following statement, Altered Answer represents the changed factual knowledge. When the answer is changed, some related fact should also be changed. You should recall a related relation associated with the Altered Answer. Then generate questions and answers based on these recalled concepts and Subject.
Question: QUESTION. Subject: SUBJECT. Altered Answer: ALT. Recalled Relation: RECALLED RELATION.
New Question : NEW QUESTION. New Answer: NEW ANSWER. ________

Figure 3: Dataset construction procedure to generate portability part (Q,A) with GPT4.

	T5-XL		GPT-J	
Editor	ZsRE	COUNTERFACT	ZsRE	COUNTERFACT
FT	1.34	1.50	1.94	6.29
SERAC	4.75	0.58	5.53	9.51
CaliNet	**13.55**	2.91	29.77	0.68
T-Patcher	1.20	0.02	3.10	7.21
KE	7.08	10.03	0.37	0.00
MEND	11.34	**29.17**	0.08	0.00
KN	0.84	4.29	19.30	6.12
ROME	-	-	50.91	46.49
MEMIT	-	-	**52.74**	**47.45**

Table 3: Portability results on various model editing methods for T5-XL and GPT-J.

paraphrased sentences typically involve only superficial changes in wording and do not reflect significant alterations to factual information. To ensure that these methodologies are prepared for real-world application, it is crucial to determine whether they can account for the consequences of an edit. For example, if we change the answer to the question "What university did Watts Humphrey attend?" from "Trinity College" to "University of Michigan", the model should then answer "Ann Arbor in Michigan State" instead of "Dublin in Ireland" when asked, "Which city did Watts Humphrey live in during his university studies?"

As a result, we introduce a new evaluation metric called **Portability** to gauge the effectiveness of model editing in transferring knowledge to related content and the potential for the modified knowledge to be utilized by the edited language model for downstream tasks. We incorporate a new part, $P(x_e, y_e)$, into the existing dataset ZsRE and COUNTERFACT, and **Portability** is calculated as the average accuracy of the edited model (f_{θ_e}) when applied to reasoning examples in $P(x_e, y_e)$:

$$\mathbb{E}_{x'_{\mathrm{e}}, y'_{\mathrm{e}} \sim P(x_e, y_e)} \mathbb{1}\left\{\operatorname{argmax}_y f_{\theta_e}\left(y \mid x'_{\mathrm{e}}\right) = y'_{\mathrm{e}}\right\} \tag{5}$$

Dataset Construction We employ GPT-4 in the process of dataset construction to generate associated questions and answers (refer to Figure 3). Suppose in the original editing, we change the answer of the question about subject s from o to o^*. Initially, we prompt the model to generate a triple (o^*, r^*, o'^*) that's linked to the modified answer, with s being the altered response. Subsequently, GPT-4 fabricates a question based on the triple(o^*, r^*, o'^*) and the subject s in the original query. Notably, if the model can answer this new question, it would imply that it has pre-existing knowledge of the triple (o^*, r^*, o'^*). Consequently, we filter out triples with which the model is unfamiliar. Specifically, we combine o^* and r^* as input and instruct the model to predict o'^*. If the model successfully generates o'^*, we deduce that it possesses knowledge of the triple. Finally, we involve **human evaluators** to validate the accuracy of the triple and the question. To guide GPT-4 toward generating the desired question and answer, we utilize a few-shot demonstration as an instructional method. Additional details can be found in the Appendix B.

Results We conduct experiments based on the newly proposed evaluation metric, presenting the results in Table 3. As demonstrated by the Table, the performance of current model editing methods regarding portability is somewhat suboptimal. Most of the editing methods, including KE, KN, MEND, and CaliNET, are unable to transfer the altered knowledge to related facts. SERAC, despite showing impeccable results on previous metrics, scores less than 10% accuracy in terms of portability. These results seem reasonable since the classifier struggles to efficiently categorize sentences into the counterfactual model, and the knowledge within the original model remains unchanged. If the classifier can identify the modified sentences,

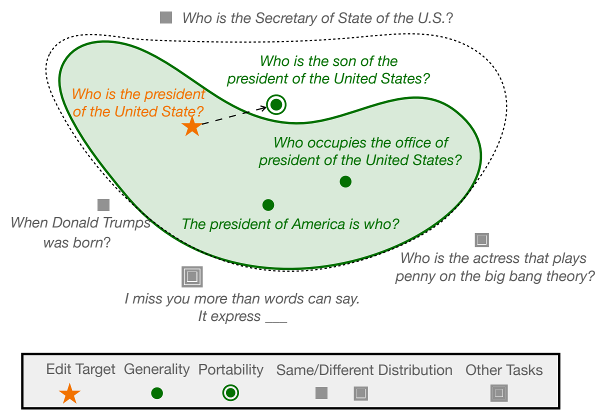

Who is the Secretary of State of the U.S.?
Who is the son of the president of the United States?
Who is the president of the United State?
Who occupies the office of president of the United States?
When Donald Trumps was born?
The president of America is who?
Who is the actress that plays penny on the big bang theory?
I miss you more than words can say. It express ___
Edit Target
Generality
Portability
Same/Different Distribution
Other Tasks

Figure 4: Running example for model editing evaluation including generality, portability and locality.

Editor	COUNTERFACT	ZsRE
FT	35.94s	58.86s
SERAC	5.31s	6.51s
CaliNet	1.88s	1.93s
T-Patcher	1864.74s	1825.15s
KE	2.20s	2.21s
MEND	**0.51s**	**0.52s**
KN	225.43s	173.57s
ROME	147.2s	183.0s
MEMIT	143.2s	145.6s

Table 4: **Wall clock time** for each edit method conducting 10 edits on GPT-J using one 2×V100 (32G).

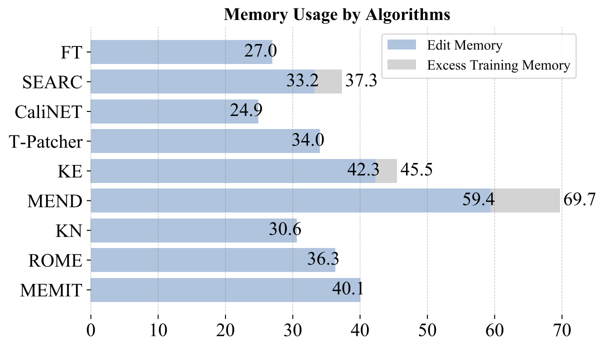


Figure 5: **GPU VRAM consumption during training and editing** for different model editing methods. We apply methods on GPT-J model using 3×V100.

the minor counterfactual model, restricted by its weak reasoning capabilities, might still lack the necessary related information. Unexpectedly, ROME and MEMIT exhibit relatively commendable performance on portability (over 50% on ZsRE and 45% on COUNTERFACT). They are capable of not only editing the original cases but also modifying facts correlated with them in some respect. Their impressive portability further testifies to the efficacy and practicality of their location methods, as the model would use this knowledge when required. One reason MEMIT performs better than ROME is that it edits more layers, giving the knowledge a greater chance of being used in the application.

In conclusion, current model editing methods still fall short in accurately answering questions about related topics, the answers to which are implied by the content of the edit example. This is an area that requires further exploration in the future.

5.2 Locality

COUNTERFACT and ZsRE measure the locality of model editing from different perspectives. COUNTERFACT utilizes triples sampled from the same distribution as the target knowledge, while ZsRE employs questions from different distributions, the Nature Question task. Beyond other knowledge, Skill Neuron (Wang et al., 2022) suggests that the feed-forward networks in LLMs possess the capability of task-specific knowledge, which sparks new challenges, specifically whether model editing might impact the performance on other tasks.

Concretely, the influence of model editing on the model is multifaceted and challenging to assess. Hence, evaluating the locality of model editing necessitates considering the model's primary purpose and the intended range of the edits, as shown in Figure 4. Firstly, the editing should not edit information from the same distributions. As shown in the figure, when editing the position of 'Donald Trump', other features of 'Donald Trump' like the public time should not be changed. Meanwhile, other entities such as 'Secretary of State', which shares similar features with 'the President' should also not be influenced. Secondly, for models aiming at task-specific tasks, we may only need to consider their effect on the target task, without concerning other capabilities. What's more, for task-agnostic models, it is crucial to consider their performance on knowledge from different distributions and different tasks within the general domain. Note that some approaches, such as MEND, employ the KL divergence on the locality dataset to train the model. This could also restrict the model's focus to the target data without concerning other aspects of the locality. This is also an area that requires further exploration in the future.

5.3 Efficiency

Despite the success of these approaches, it's also vital to consider the efficiency of model editing for

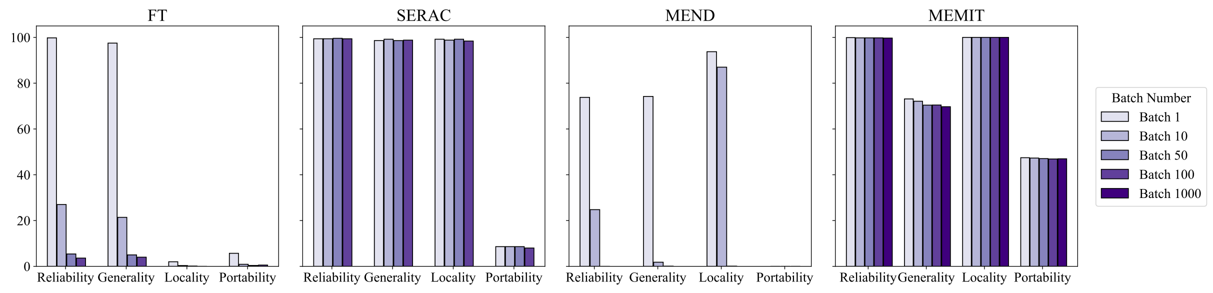


Figure 6: **Batch Editing** performance against batch number. We test batch numbers in [1,10,50,100,1000] for MEMIT. Due to the huge memory usage for FT, MEND and SERAC, we didn't test batch 1000 for these methods.

practical applications. An efficient model editor should minimize the **time** and **memory** required for computing and evaluating edits without compromising the model's performance.

Time Analysis Editing time refers to the duration required to perform the model editing process, which includes updating the model's parameters to reflect the desired changes. From Table 4, we notice that KE and MEND are considerably faster once the trained hyper-network is obtained. SERAC can also edit the knowledge fast with 5 seconds for 10 edits. However, those approaches require hours-to-days of additional training cost. In our experiments, training MEND for the ZsRE dataset takes more than 7 hours, and training SERAC takes more than 36 hours using 3× V100. Other methods, like KN, CaliNET, ROME and MEMIT, are lower than MEDN, but they do not require training before editing. CaliNET is quick but the results are normal while ROME and MEMIT may take a bit more time but achieve excellent performance. T-Patcher is the slowest method because it requires training neurons for each edit.

Memory Analysis In addition, it's crucial to consider the impact of model editing on the space required for model storage. We present the memory Vram Usage consumption for each method in Figure 5. From the figure, we can see that most methods consume about the same amount of memory. Methods that introduce additional parameters incur some additional computational overhead.

In conclusion, an efficient model editor should strike a balance between model performance, inference speed, and storage space requirements.

5.4 Batch Editing Analysis

We further conduct batch editing analysis since many studies are typically constrained to updating at most a few dozen facts, or only focusing on single-edit cases. Often, we need to modify the model with multiple pieces of knowledge simultaneously. Here, we concentrate on the methods that support batch editing (including FT, SERAC, MEND, and MEMIT) and plot the performance in Figure 6. We do not evaluate KE and CaliNET in this context as their single edit performance on LLMs is suboptimal.

We note that MEMIT is a unique method capable of supporting massive knowledge editing for LLMs. It can edit hundreds or even thousands of facts simultaneously with minimal time and memory costs. The performance across both metrics remains stable and excellent even for up to 1000 edits (The original paper denotes that MEMIT can support editing more than 10,000 cases at the same time.) Furthermore, MEMIT maintains high locality and generality when performing extensive knowledge edits concurrently. Other methods do also support batch editing but require enormous memory to handle more cases, which is beyond our current capabilities. Hence, for these methods, we limit batch editing tests to 100. SERAC can support multiple edits up to 100 and maintain great performance, so we constrain our tests to this number. The remaining two methods, FT and MEND, do not perform as well in batch edits. As the number of edits increases, the model's performance diminishes rapidly. Additionally, apart from the performance, MEND and SERAC necessitate the training of a unique model for each batch of editing, which is not practically feasible.

5.5 Sequential Editing Analysis

Besides performing multiple edits simultaneously, the ability to conduct edits sequentially is also an essential characteristic for model editing (Huang et al., 2023). In practical settings, the model should retain the changes from previous edits when carrying out a new one. For this

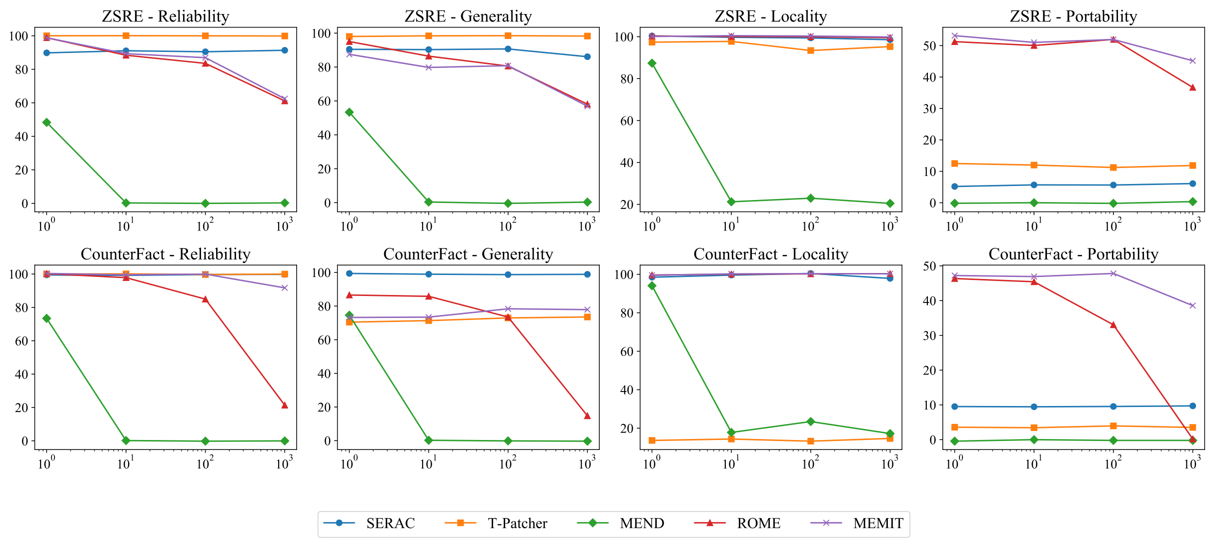


Figure 7: **Sequential Editing** performance against data stream size (log-scale).

analysis, we edit the model in a data stream $(x_1, y_1), (x_2, y_2), .., (x_s, y_s)$, where the model performs edits successively. The performance versus the number of edits on a log scale is plotted in Figure 7. In this section, we select models with solid single-edit performance to evaluate their capabilities in sequential editing. As per the results, SERAC and T-Patcher show excellent and consistent performance in sequential editing. Indeed, methods that freeze the model's parameters and use external parameters for model editing generally display stable performance in sequential editing. ROME performs well up to $n = 10$ but begins to degrade at $n = 100$. MEMIT's performance over 100 also drops, but not as much as ROME. Similarly, MEND performs admirably at $n = 1$ but rapidly declines at $n = 10$. Methods that alter the model's parameters struggle with sequential editing since their editing algorithm is predicated on the initial model. As the editing process continues, the model increasingly deviates from the initial state, leading to their suboptimal performance.

6 Relationship with Relevant Works

6.1 Knowledge in LLMs

Several model editing approaches aim to discern how knowledge stored in PLMs precisely and directly alters the model's parameters. There is existing work examining the principles governing how PLMs store knowledge (Geva et al., 2021, 2022; Haviv et al., 2023; Hao et al., 2021; Hernandez et al., 2023; Yao et al., 2023; Cao et al., 2023), which can contribute to the model editing process. Moreover, model editing techniques (manipulating external knowledge) bear resemblance to knowledge augmentation approaches as updating the model's knowledge can also be considered as instilling new knowledge into the model. Knowledge augmentation (Zhang et al., 2019; Lewis et al., 2020; Zhang et al., 2022; Yasunaga et al., 2021; Yao et al., 2022) often involves enriching the model directly by feeding it with external or updated information to cope with targeted tasks, such as question answering (Chang et al., 2020; Chen et al., 2017). Such methods are typically employed when the model's existing knowledge is insufficient and usually involve training fusion modules (Wang et al., 2021). However, model editing aims to precisely fix and update knowledge in LLMs, circumventing the computational load associated with training an entirely new model.

6.2 Lifelong Learning and Unlearning

Model editing focuses on targeted updates and integration of new knowledge, which is efficient for minimal changes and often demands fewer resources. On the other hand, continual learning (Biesialska et al., 2020) enhances the adaptability of models across various tasks and domains, but it requires extensive training and updates, thus being more suitable for learning a broad array of tasks and fields of knowledge. Zhu et al. (2020) adapt continual learning methods to update knowledge in PLMs, demonstrating the effectiveness of continual learning in model editing. Concurrently, as noted by Hase et al. (2023), it is important for a model to forget sensitive knowledge (e.g., private data), akin to the concept of machine unlearning (Wu et al., 2022; Tarun et al., 2021). Model editing can be seen as a unified special case of lifelong learning and unlearning, where the model is capable of adaptively and continuously adding and editing new knowledge, while also removing outdated and erroneous knowledge.

6.3 Security and Privacy for LLMs

Previous works (Carlini et al., 2020; Shen et al., 2023) notice that language models might directly generate unreliable or personal samples based on certain prompts. The task of erasing private and potentially harmful information stored in large language models (LLMs) is vital for enhancing the privacy and security of LLM-based applications (Sun et al., 2023). Note that Geva et al. (2022) observes that model editing could suppress the generation of toxic or harmful language from LLMs, suggesting that model editing could play a key role in addressing security and privacy concerns associated with LLMs.

7 Conclusion

This paper emphasizes the editing of LLMs, systematically defining the problems, offering empirical analysis of various methods, and suggesting potential opportunities. Our goal is to assist researchers in developing a deeper understanding of the features, strengths, and shortcomings of existing model editing strategies. Our findings highlight ample room for future development, particularly in the realms of portability, locality, and efficiency. Given the immense resources invested in LLMs, efficient model editing technologies can offer the potential to more effectively reflect the updating needs and values of the people they're serving. We hope that our work can contribute to addressing ongoing issues and inspire future researches.

Limitations

There remain several aspects of model editing that are not covered in this paper.

Editing Scope Notably, the application of model editing goes beyond mere factual contexts, underscoring its vast potential. Elements such as personality, emotions, opinions, and beliefs also fall within the scope of model editing. While these aspects have been somewhat explored, they remain relatively uncharted territories and thus are not detailed in this paper. Furthermore, multilingual editing (Xu et al., 2022) represents an essential research direction that warrants future attention and exploration.

Editing Black-Box LLMs Meanwhile, models like ChatGPT and GPT-4 exhibit remarkable performance across a wide array of natural language tasks but are only accessible via APIs. This raises an important question: How can we edit these "black-box" models that also tend to produce undesirable outputs during downstream usage? Presently, there are some works that utilize in-context learning (Onoe et al., 2023) and prompt-based methods (Murty et al., 2022) to modify these models. They precede each example with a textual prompt that specifies the adaptation target, which shows promise as a technique for model editing.

In-context Editing Given a refined knowledge context (instructions) as a prompt, the model can generate outputs corresponding to the provided knowledge. However, such methods may struggle with context mediation failure, as language models may not consistently generate text aligned with the prompt. Moreover, these methods cannot modify the model's intrinsic knowledge because it necessitates prepending the text to the input every time for editing. Notably, previous studies suggest that instructing LLMs can help recall already learned concepts from pre-training, perform implicit learning over demonstrations (e.g., attention layer) (Dai et al., 2022b), or disentangle task recognition and task learning (Pan et al., 2023). Hence, it's intuitive to conduct in-context editing to consolidate the knowledge from the prompt into the parametric space. As a result, LLMs can recall what they have failed and fix errors without needing further demonstration anymore.

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Magdalena Biesialska, Katarzyna Biesialska, and Marta Ruiz Costa-jussà. 2020. Continual lifelong learning in natural language processing: A survey. *ArXiv*, abs/2012.09823.

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, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 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*.

Boxi Cao, Qiaoyu Tang, Hongyu Lin, Xianpei Han, Jiawei Chen, Tianshu Wang, and Le Sun. 2023. Retentive or forgetful? diving into the knowledge memorizing mechanism of language models. *arXiv preprint arXiv:2305.09144*.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Xiaodong Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. Extracting training data from large language models. In *USENIX Security Symposium*.

Ting-Yun Chang, Yang Liu, Karthik Gopalakrishnan, Behnam Hedayatnia, Pei Zhou, and Dilek Hakkani-Tur. 2020. Incorporating commonsense knowledge graph in pretrained models for social commonsense tasks. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 74–79, Online. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022a. 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.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2022b. Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers. *CoRR*, abs/2212.10559.

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, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5937–5947, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Y. Hao, Li Dong, Furu Wei, and Ke Xu. 2021. Self-attention attribution: Interpreting information interactions inside transformer. In *Proc. of AAAI*.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *ArXiv*, abs/2301.04213.

Adi Haviv, Ido Cohen, Jacob Gidron, Roei Schuster, Yoav Goldberg, and Mor Geva. 2023. Understanding transformer memorization recall through idioms. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 248–264, Dubrovnik, Croatia. Association for Computational Linguistics.

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2023. Measuring and manipulating knowledge representations in language models. *CoRR*, abs/2304.00740.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

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)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36.

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.

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*.

Shikhar Murty, Christopher D. Manning, Scott M. Lundberg, and Marco Túlio Ribeiro. 2022. Fixing model bugs with natural language patches. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11600–11613. Association for Computational Linguistics.

Yasumasa Onoe, Michael J. Q. Zhang, Shankar Padmanabhan, Greg Durrett, and Eunsol Choi. 2023. Can lms learn new entities from descriptions? challenges in propagating injected knowledge. *CoRR*, abs/2305.01651.

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

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning" learns" in-context: Disentangling task recognition and task learning. *arXiv preprint arXiv:2305.09731*.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Reasoning with language model prompting: A survey. *CoRR*, abs/2212.09597.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020a. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020b. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. 2023. In chatgpt we trust? measuring and characterizing the reliability of chatgpt.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *International Conference on Learning Representations*.

Hao Sun, Zhexin Zhang, Jiawen Deng, Jiale Cheng, and Minlie Huang. 2023. Safety assessment of chinese large language models. *CoRR*, abs/2304.10436.

Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan S. Kankanhalli. 2021. Fast yet effective machine unlearning. *IEEE transactions on neural networks and learning systems*, PP.

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. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Ben Wang and Aran Komatsuzaki. 2021a. Gpt-j-6b: A 6 billion parameter autoregressive language model.

Ben Wang and Aran Komatsuzaki. 2021b. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418, Online. Association for Computational Linguistics.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. 2022. Finding skill neurons in pre-trained transformer-based language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language*

Processing, pages 11132–11152, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ga Wu, Masoud Hashemi, and Christopher Srinivasa. 2022. Puma: Performance unchanged model augmentation for training data removal. In *AAAI Conference on Artificial Intelligence*.

Yang Xu, Yutai Hou, and Wanxiang Che. 2022. Language anisotropic cross-lingual model editing. *ArXiv*, abs/2205.12677.

Yunzhi Yao, Shaohan Huang, Ningyu Zhang, Li Dong, Furu Wei, and Huajun Chen. 2022. Kformer: Knowledge injection in transformer feed-forward layers. In *Natural Language Processing and Chinese Computing*.

Yunzhi Yao, Peng Wang, Shengyu Mao, Chuanqi Tan, Fei Huang, Huajun Chen, and Ningyu Zhang. 2023. Knowledge rumination for pre-trained language models. *CoRR*, abs/2305.08732.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. GreaseLM: Graph REASoning enhanced language models. In *International Conference on Learning Representations*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. 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 1441–1451. 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, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

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

A Implementing Details

FT For basic Fine-Tuning (FT), we follow Meng et al. (2023) re-implementation in their study, which uses Adam (Kingma and Ba, 2014) with early stopping to minimize $-log\mathbb{P}_{G'}[o* \mid p]$, changing only mlp_{proj} weights at selected layer 21. For both models, all hyperparameters follow default settings. To ensure fairness in the experiments, we always use the unconstrained fine-tuning approach.

KE De Cao et al. (2021) develops an LSTM sequence model, which employs gradient information to predict the rank-1 weight alterations in G. Given that the official code doesn't facilitate GPT-J, we resort to using the re-implemented version provided by Mitchell et al. (2022a) in their research. To foster an equitable comparison across both zsRE and COUNTERFACT tasks, we have taken additional steps to train KE-zsRE and KE-CF models. The hyperparameters employed for training have been sourced from the default configurations provided. During testing, KE presents a scaling factor to tweak the norm of the weight update, for which we adhere to the default value of 1.0.

CaliNET Dong et al. (2022) enriches the FFN by incorporating extra parameters aimed at knowledge editing, comprised of a number of calibration memory slots. In order to adapt CaliNET to the task at hand, we retain the same architecture as used in the Feed-Forward Network (FFN), albeit with a reduced intermediate dimension denoted as d. This adaptation allows us to effectively apply CaliNET while managing the model's complexity and computational requirements. Regarding hyperparameters, we implement adjustments to the FFN within the final two layers of GPT-J, while all other configurations remain consistent with the default settings.

MEND Mitchell et al. (2022a) develop an efficient method for locally editing language models using just a single input-output pair. Essentially, MEND employs a technique to manipulate the gradient of fine-tuned language models which leverages a low-rank decomposition of gradients. The hyperparameters follow default settings, with the exception of several experiments conducted on GPT-J. Specifically, we adjust the optimizer from Adam to AdamW.

SERAC Mitchell et al. (2022b) presents a method for model editing, named MEME

(Memory-Based Model Editing), which stores edits in an explicit memory and learns to reason over them to adjust the base model's predictions as needed. The system uses an explicit cache of user-provided edit descriptors (arbitrary utterances for language models), alongside a small auxiliary *scope classifier* and *counterfactual model*. The scope classifier determines if the input falls within the scope of any cached items, and if so, the counterfactual model uses the input and the most relevant edit example to make a prediction.

In alignment with the original paper, we use publicly available Huggingface implementations and checkpoints for all experiments. For the SERAC *scope classifier* model, we adopt `distilbert-base-cased` (Sanh et al., 2019) across all models and experimental settings. For the counterfactual model, we employ `t5-small` (Raffel et al., 2020b) for the T5-XL implementation and `architext/gptj-162M` (available at `https://huggingface.co/architext/gptj-162M`) for the GPT-J implementation. All hyperparameters for training and test-time inference are derived from default configurations.

Similar to T-Patcher, in auto-regressive model (like GPT-J) training, we only consider loss at the output positions.

KN For Knowledge Neuron (Dai et al., 2022a), we follow Meng et al. (2023) re-implementation in their study. The method begins by identifying neurons that are closely associated with knowledge expression. This selection is made through gradient-based attributions, which effectively highlight the neurons that have a strong influence on the model's output. After these critical neurons are identified, the method modifies the projection layer of the feed-forward network (denoted as $mlp^{(l)}_{proj}$) specifically at the rows corresponding to the selected neurons. This modification involves adding scaled embedding vectors to the current values, effectively adjusting the model's behavior in a targeted manner. Specifically, they amplify knowledge neurons by doubling their activations. Similar to FT, all hyperparameters are adopted from default configurations: `https://github.com/EleutherAI/knowledge-neurons`

T-Patcher The method proposed by Huang et al. (2023) offers a way to alter the behavior of transformer-based models with minimal changes. Specifically, it adds and trains a small number of neurons in the last Feed-Forward Network (FFN) layer. This approach effectively provides a means for fine-tuning model behavior with less computational demand than comprehensive retraining. It freezes all original parameters and adds one neuron (patch) to the last FFN layer for one mistake. And they train the patch to take effect only when encountering its corresponding mistake. For T5-XL implementation, all hyperparameters follow the same default settings as Bart-base in: `https://github.com/ZeroYuHuang/Transformer-Patcher`.

Furthermore, in the auto-regressive model (like GPT-J), the model may make multiple mistakes in one example. Therefore, for an example where the model makes n mistakes, we only consider errors generated by the model at the **output positions**. Following the settings of the original paper, we add up to 5 patches for one edit example. Formally, for an edit example (x_e, y_e) in auto-regressive model, the actual input is given by $\hat{x_e} = x_e + y_e$ and the patched model's output is p_e, l_e is defined as:

$$l_e = -\sum_{i=1}^{N} \hat{x_i} \log(p_i) \cdot \mathbf{1}_{(i \geq len(x_e))} \qquad (6)$$

ROME ROME, as proposed by Meng et al. (2022), conceptualizes the MLP module as a straightforward key-value store. For instance, if the key represents a subject and the value encapsulates knowledge about that subject, then the MLP can reestablish the association by retrieving the value that corresponds to the key. In order to add a new key-value pair, ROME applies a rank-one modification to the weights of the MLP, effectively "writing in" the new information directly. This method allows for more precise and direct modification of the model's knowledge. We directly apply the code and MLP weight provided by the original paper [2] and keep the default setting for hyper-parameters.

MEMIT MEMIT (Meng et al., 2023) builds upon ROME to insert many memories by modifying the MLP weights of a range of critical layers. We test the ability of MEMIT using their code [3] and all hyperparameters follow the same default settings

[2]https://rome.baulab.info/
[3]https://memit.baulab.info/

B Dataset Details

B.1 Dataset Construction for Portability Evaluation

To ensure that the original model(f_θ) has seen the triple (s, r, o) during the pre-training process, we employ link prediction to predict o given $(s, r, ?)$. If the tail entity is present in the Top-10 logits, we consider the model to have prior knowledge of this triple. In other words, if the model has sufficient portability, it can correctly answer new questions based on the subject and the triplet.

We select data points to measure the performance of the model's portability. The symbolic representation of the portability dataset is as follows:

$$D_{port} = \{\text{GPT4}\,(s, r, o) \mid o \in \text{Top-10}(f_\theta\,(s, r, ?))\}$$

To guide GPT-4 in producing the desired question and answer, we employ a few-shot manual demonstration as a prompt (See Table 6). In addition, we intersect the data filtered by T5-XL and the data filtered by GPT-J to obtain the final portability dataset. This is done to ensure that both T5 and GPT-J models have prior knowledge about the triplets. As a result, we select 1037 data instances from the **ZsRE** dataset and 1031 data instances from the **COUNTERFACT** dataset. The description of the data is shown in Table 5.

Dataset	T5 score	GPT-J score	Size
ZsRE	83.90	72.99	1,037
COUNTERFACT	84.81	69.78	1,031

Table 5: Statistics of portability dataset.

Prompt

In the following statement, ‘Altered Answer‘ represents the changed factual knowledge. When the answer is changed, some related facts should also be changed. You should recall a related relation associated with the ‘Altered Answer‘. Then generate questions and answers based on these recalled concepts and ‘Subject‘.

Question: What university did Watts Humphrey attend?
Subject: Watts Humphrey
Altered Answer: University of Michigan
Recalled Relation: (University of Michigan, locate in, Ann Arbor)
New Question: Which city did Watts Humphrey live in during his undergraduate studies?
New Answer: Ann Arbor in Michigan State

Question: Windows 10, developed by
Subject: Windows 10
Altered Answer: Google
Recalled Relation: (Sundar Pichai, ceo of, Google)
New Question: Who is the CEO of the company that develops the Windows 10 operating system?
New Answer: Sundar Pichai

Question: In Kotka, the language spoken is?
Subject: Kotka
Altered Answer: French
Recalled Relation: (French, evolve from, Romance)
New Question: What language did Kotka’s official language evolve from?
New Answer: Romance

Question: Armand Trousseau’s area of work is?
Subject: Armand Trousseau
Altered Answer: jazz
Recalled Relation: (Miles Davis, genres, jazz)
New Question: Armand Trousseau formed a band during college, they are all fans of?
New Answer: Miles Davis

Table 6: Prompt for dataset construction on zsRE & COUNTERFACT portability dataset. Demonstration examples are manually constructed. For each data instance, we provide Question, Subject, and Altered Answer to generate portability data.