EasyEdit: An Easy-to-use Knowledge Editing Framework for Large Language Models

Peng Wang[♣], **Ningyu Zhang**[♣]*, **Bozhong Tian**[♣], **Zekun Xi**[♣], **Yunzhi Yao**[♣], **Ziwen Xu**[♣], **Mengru Wang**[♣], **Shengyu Mao**[♣], **Xiaohan Wang**[♣], **Siyuan Cheng**[♣], **Kangwei Liu**[♣], **Yuansheng Ni**[♣], **Guozhou Zheng**[♣], **Huajun Chen**[♣]*

[♣] Zhejiang University

https://github.com/zjunlp/EasyEdit

Abstract

Large Language Models (LLMs) usually suffer from knowledge cutoff or fallacy issues, which means they are unaware of unseen events or generate text with incorrect facts owing to outdated/noisy data. To this end, many knowledge editing approaches for LLMs have emerged – aiming to subtly inject/edit updated knowledge or adjust undesired behavior while minimizing the impact on unrelated inputs. Nevertheless, due to significant differences among various knowledge editing methods and the variations in task setups, there is no standard implementation framework available for the community, which hinders practitioners from applying knowledge editing to applications. To address these issues, we propose EASYEDIT, an easy-to-use knowledge editing framework for LLMs. It supports various cutting-edge knowledge editing approaches and can be readily applied to many well-known LLMs such as T5, GPT-J, LlaMA, etc. Empirically, we report the knowledge editing results on LlaMA-2 with EASYEDIT, demonstrating that knowledge editing surpasses traditional fine-tuning in terms of reliability and generalization. We have released the source code on GitHub[1], along with Google Colab tutorials and comprehensive documentation[2] for beginners to get started. Besides, we present an online system[3] for real-time knowledge editing, and a demo video[4].

1 Introduction

Large Language Models (LLMs) have revolutionized modern Natural Language Processing (NLP), significantly improving performance across various tasks (Brown et al., 2020; OpenAI, 2023; Anil et al., 2023; Zhao et al., 2023; Touvron et al., 2023b; Qiao et al., 2023; Zheng et al., 2023b; Pan et al., 2023). However, deployed LLMs usually suffer from knowledge cutoff or fallacy issues. For example, LLMs such as ChatGPT and LlaMA possess information only up to their last training point. They can sometimes produce inaccurate or misleading information due to potential discrepancies and biases in their pre-training data (Ji et al., 2023; Hartvigsen et al., 2022). Hence, it's essential to efficiently update the parametric knowledge within the LLMs to modify specific behaviors while avoiding expensive retraining.

Indeed, finetuning or parameter-efficient finetuning (Ding et al., 2022, 2023) offers methods for modifying LLMs, these approaches can be computationally expensive and may lead to overfitting, particularly when applied to a limited number of samples (Cao et al., 2021) or streaming errors of LLMs. Additionally, fine-tuned models might forfeit capabilities gained during pre-training, and their modifications do not always generalize to relevant inputs. An alternative methodology involves using manually written or retrieved prompts to influence the LLMs' output. These methods suffer from reliability issues, as LLMs do not consistently generate text aligned with the prefix prompt (Hernandez et al., 2023; Lewis et al., 2021). Additionally, due to the extensive amount of up-to-date knowledge required for complex reasoning tasks, the impracticality of context overload becomes inevitable whenever the context length is limited.

A feasible solution, knowledge editing[5], aims to efficiently modify the behavior of LLMs with minimal impact on unrelated inputs. Research on knowledge editing for LLMs (Meng et al., 2023, 2022; Zheng et al., 2023a; Gupta et al., 2023; Mitchell et al., 2022a; Geva et al., 2023; Hase et al., 2023; Cohen et al., 2023a; Hartvigsen et al., 2023; Tan et al., 2024; Yu et al., 2023) have displayed remarkable progress across various tasks and settings.

*Corresponding author.

[1]This is a subproject of KnowLM (https://github.com/zjunlp/KnowLM), which facilitates knowledgeable LLM Framework with EasyInstruct, EasyEdit, EasyDetect etc.

[2]https://zjunlp.gitbook.io/easyedit

[3]https://huggingface.co/spaces/zjunlp/EasyEdit

[4]https://youtu.be/Gm6T0QaaskU

[5]Knowledge editing can also be termed as model editing.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 82–93

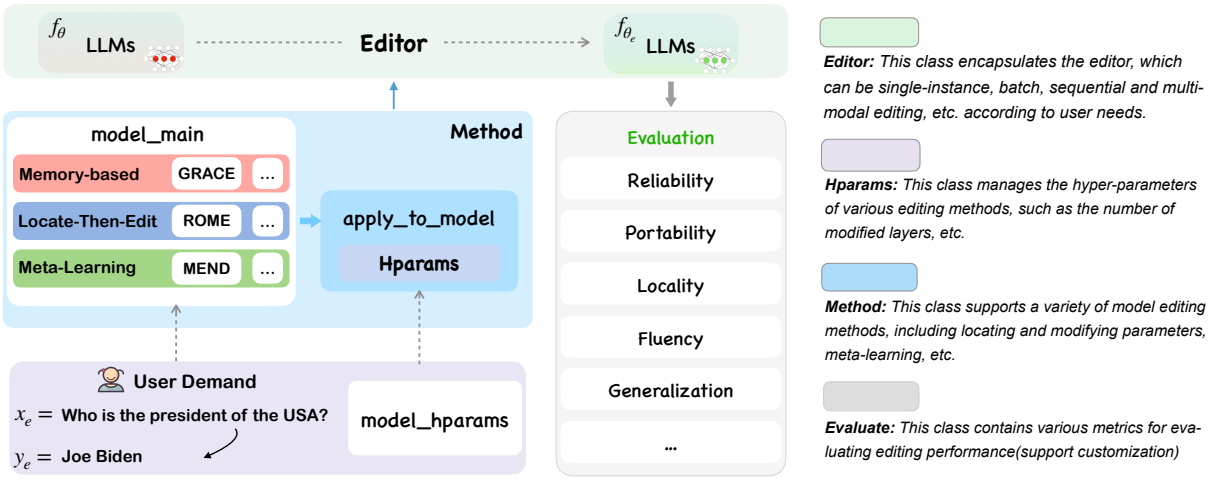

Figure 1: The overall architecture of EASYEDIT. The main function is `apply_to_model`, which applies the selected editing method to the LLMs. The **Editor** serves as the direct entry point, receiving customized user inputs and outputs, and returning the edited weights. Please note that some methods may require pre-training of classifiers or hypernetworks through the Trainer (See §3.5). EASYEDIT supports customizable evaluation metrics.

However, these variations in both implementation and task settings have impeded the development of a unified and comprehensive framework for knowledge editing. Note that the complexity obstructs the direct comparison of effectiveness and feasibility between different methods, and complicates the creation of novel knowledge editing approaches. To this end, we propose EASYEDIT, an easy-to-use knowledge editing framework for LLMs. EASYEDIT modularizes editing methods and effectiveness evaluation while considering their combination and interaction. It supports a variety of editing scenarios, including **single-instance**, **batch-instance**, **sequential**, and **multi-modal** editing. Moreover, EASYEDIT provides evaluation evaluations of key metrics such as Reliability, Generalization, Locality, and Portability (Yao et al., 2023), to quantify the robustness and side effects (Cohen et al., 2023b) of editing methods.

Specifically, in EASYEDIT, the Editor class integrates various editing components. The Method class offers a unified interface `apply_to_model`, which accepts editing descriptors and returns the edited model, thereby facilitating the integration of novel editing methodologies. Dedicated to evaluating editing performance, the Evaluate module leverages metrics such as reliability, robust generalization, and locality. The Trainer module manages the training of additional neural network structures. Each module in EASYEDIT is meticulously defined, striking a balance between cohesion and coupling. Furthermore, we furnish examples of editing across a spectrum of models, including T5 (Raffel et al., 2019), GPT-J (Wang and Komatsuzaki, 2021), GPT-NEO (Black et al., 2021), GPT2 (Radford et al., 2019), LLaMA (Touvron et al., 2023a), LLaMA-2 (Touvron et al., 2023b), Mistral (Jiang et al., 2023), and Qwen (Bai et al., 2023). We acknowledge all the support for EASYEDIT, which is listed in Appendix 6 due to space constraints.

2 Background

Previous Solutions Despite the tremendous success of LLMs in almost all NLP tasks, persistent challenges such as knowledge cutoff and biased/toxic outputs remain. To counter these challenges, two approaches are generally employed:

1) FINE-TUNING: Traditional fine-tuning techniques, along with delta tuning (Ding et al., 2022) and LoRA tuning (Hu et al., 2021) utilize domain-specific datasets to update the model's internal parametric knowledge. However, these methods face two notable challenges: First, they consume considerable resources. Second, they risk the potential of catastrophic forgetting (Ramasesh et al., 2022).

2) PROMPT-AUGMENTATION: Given a sufficient number of demonstrations or retrieved contexts, LLMs can learn to enhance reasoning (Yu et al., 2022) and generation through external knowledge (Borgeaud et al., 2022; Guu et al., 2020; Lewis et al., 2020). However, the performance may be sensitive to factors such as the prompting template, the selection of in-context examples (Zhao et al., 2021), or retrieved contexts (Ren et al.,

2023). These approaches also encounter the issue of context length limitation (Liu et al., 2023a).

Knowledge Storage Mechanism Within the NLP literature, numerous studies have delved into understanding the location of different types of knowledge in language models (Petroni et al., 2019; Roberts et al., 2020; Jiang et al., 2020). LLMs can be conceptualized as knowledge banks, and the transformer MLP layers function as key-value memories according to observations from Geva et al. (2021). This configuration promotes efficient knowledge adjustments by precisely localizing knowledge within the MLP layers (denoted as knowledge editing).

Knowledge editing enables nimble alterations to the LLMs' behavior through one data point. Another promising attribute of knowledge editing is its ability to ensure the locality of editing, meaning that modifications are contained within specific contexts. Additionally, the knowledge editing technique can mitigate harmful language generation (Geva et al., 2022). In this paper, we present EASYEDIT, an easy-to-use knowledge editing framework for LLMs. It seamlessly integrates diverse editing technologies and supports the free combination of modules for various LLMs. Through its unified framework and interface, EASYEDIT enables users to swiftly comprehend and apply the prevalent knowledge editing methods included in the package.

3 Design and Implementation

EASYEDIT provides a complete editing and evaluation process built on Pytorch (Paszke et al., 2019) and Huggingface (Wolf et al., 2020). This section commences with an exploration of the assemblability aspect of EASYEDIT, followed by a detailed explanation of the design and implementation of each component within the EASYEDIT framework (as shown in Figure 1). Additionally, we demonstrate a straightforward example of applying MEND to LLaMA, altering the output of *the U.S. President* to *Joe Biden*.

3.1 Assemblability

In the realm of knowledge editing, various distinct scenarios[6] exist. To cater to this diversity, EASYEDIT offers flexible combinations of modules that different editing `Editor` (such as single-instance, batch-instance (details in Appendix A)),

[6]Denoted as `(Editor, METHOD, TARGET)`

```python
# Step 1: Choose the Editing Method e.g. MEND
from easyeditor import BaseEditor
from easyeditor import MENDHyperParams
hparams = MENDHyperParams.from_hparams('gpt2-xl.yaml')


# Step 2: Provide the edit descriptor and target
prompts = ['Q: The president of the US is? A:',]
target_new = ['Joe Biden']
rephrase_prompts = ['The leader of the United State is']

# Step 3:  Combine them into the `BaseEditor`
editor = BaseEditor.from_hparams(hparams)

# Step 4: Edit and Evaluation
metrics, edited_model = editor.edit(
    prompts=prompts,
    target_new=target_new,
    keep_original_weight=True,
)

# metrics: Performance of knowledge editing
# edited_model: Model after modifying the weights
```

Figure 2: A running example of knowledge editing for LLMs in EASYEDIT. Utilizing the MEND approach, we can successfully transform the depiction of *the U.S. President* into that of *Joe Biden*.

`METHOD` (such as ROME, GRACE (§3.3)). About editing `TARGET`, EASYEDIT can accommodate any parameterized white-box existing model. Additionally, recent research (Dong et al., 2022) indicates that LLMs exhibit robust in-context learning capabilities. By providing edited facts to LLMs, one can alter the behavior of black-box models such as GPT4 (OpenAI, 2023). All those combinations are easily implementable and verifiable within the EASYEDIT framework.

3.2 Editor

The `Editor` serves a pivotal role in knowledge editing as it directly establishes the editing tasks and corresponding editing scenarios. Users supply the editor descriptor (x_e) and the edit target (y_e), but the input format varies according to the different editing objects. For instance, in Seq2Seq models, the edit target typically serves as the decoder's input, while in autoregressive models, x_e and y_e need to be concatenated to maximize the conditional probability. To facilitate unified editing across diverse architecture models, we meticulously develop a component `prepare_requests` to transform editing inputs.

In EASYEDIT, we provide an "edit" interface, incorporating components such as `Hparams`, `Method`, and `Evaluate`. During the editing phase, various knowledge editing strategies can be executed by invoking the `apply_to_model` function available in all different methods, it also performs evaluations

	Method	Batch Edit	Sequential Edit	Additional Train	Edit Area	Time (s)	VRAM (GB)
Memory-based	SERAC	YES	YES	YES	*External Model*	8.46	42
	IKE	NO	NO	YES	*In-Context*	4.57	52
	GRACE	NO	YES	NO	*MLP+codebook*	142.68	28
	MELO	YES	YES	NO	*LoRA+codebook*	154.32	30
Meta-learning	KE	YES	YES	YES	*MLP*	7.87	49
	MEND	YES	YES	YES	*MLP*	6.39	46
Locate-Then-Edit	KN	NO	YES	NO	*MLP*	425.64	42
	ROME	NO	YES	NO	*MLP*	187.90	31
	MEMIT	YES	YES	NO	*MLP*	169.28	33
	PMET	YES	YES	NO	*MLP*	219.17	34

Table 1: Comparison of several model editing methods. 'Batch Edit' refers to simultaneously editing multiple target knowledge instances. 'Sequential Edit' refers to maintaining previously edited knowledge while performing new edits. 'Additional Train' refers to the need for pre-training other network structures or parameters before editing. 'Edit Area' indicates the location of the edit, with MLP representing the linear layer. 'Time & VRAM' reflects the efficiency of the editing method (using LlaMA-7B as an example). 'Time' indicates the wall clock time required for conducting 10 edits, while VRAM represents the graphics memory usage.

of the model before and after the editing to gauge the editing's multifaceted impact on the model behavior, including generalization and side effects. An example to edit through EASYEDIT is depicted in Figure 2.

Note that the ability to execute batch editing (multiple edits in a single instance) and sequential editing (implementing new edits while preserving previous editing) is a crucial feature of knowledge editing (Huang et al., 2023). For methods that support batch editing, editing instances are inputted in chunk form. In addition, EASYEDIT provides a boolean switch, enabling users to either retain the pre-edit weights for single-instance editing or discard them for sequential editing.

3.3 Method

As the core component of knowledge editing, editing methods alter the model's behavior by modifying its internal parameters (e.g. MLP, Attention Mechanisms) or explicitly utilizing preceding editing facts, among other strategies. Impressive related works (Table 1) abound in this field, and they can be generally grouped into three categories as proposed by Yao et al. (2023).

Memory-based This category, encompassing methods such as SERAC (Mitchell et al., 2022b), IKE (Zheng et al., 2023a), and GRACE (Hartvigsen et al., 2023), emphasizes the use of memory elements to store and manipulate information during editing. SERAC applies retrieval and classification routing, GRACE replaces hidden states with parameters searched from a codebook for edit memorization, while IKE uses context-edit facts to guide the model in generating edited facts.

Meta-learning These methods learn the weight updates (denoted as Δ), which are then added to the original weights for editing. Examples include KE (Cao et al., 2021), which uses a bidirectional-LSTM to predict weight updates, and MEND (Mitchell et al., 2022a), which adjusts model parameters through low-rank decomposition of gradients.

Locate-Then-Edit This paradigm focuses on knowledge localization to modify the parameters of specific neurons responsible for storing the editing facts. EASYEDIT integrates methods like KN (Dai et al., 2021), which employs gradient-based methods to update specific neurons. Moreover, EASYEDIT supports ROME (Meng et al., 2023), PMET (Li et al., 2024) and MEMIT (Meng et al., 2022), leveraging causal intervention to pinpoint knowledge within a specific MLP layer and enabling the modification of the entire matrix.

However, it is not practical to expose the editing methods directly to users due to the complexity of the underlying concepts and the time investment required to understand them. Additionally, differences in input-output formats across methods could further complicate the learning process. To circumvent these hurdles, we implement a unified interface, `apply_to_model`, in EASYEDIT. Aligning with the *Strategy* design pattern, this interface is designed to be overridden by different types of editing methods, ensuring consistent input and out-

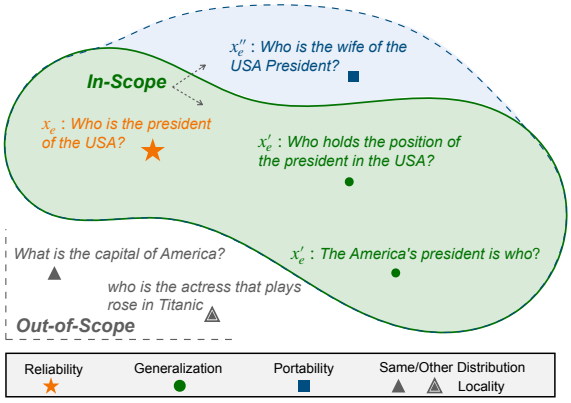

Figure 3: Depiction of the edit scope for edit descriptor *Who is the president of the USA?* It contains an example for knowledge editing evaluation, including Reliability, Generalization, Portability, and Locality.

put types. Specifically, it accepts a 'request' that includes the editing descriptor, the target of the edit, and any input data necessary to evaluate the editing performance. After processing the request(s), the interface returns the edited model weights. This design ensures both flexibility and easy-to-use, enabling users to handle knowledge editing instances effortlessly and utilize the customized models in other downstream tasks.

3.4 Hparams

When initializing an editing method, it is crucial to specify the related hyperparameters. These include the model to be edited, the layers targeted for modification, and, optionally, the type of external model, among other parameters. For methods that alter the LLMs' internal parameters, the adjustable parameter names should be indicated using the MODULE_NAME format, such as *transformer.h.5.mlp.fc_out*. In this case, the parameters of the *fc_out* linear layer in the fifth layer MLP of GPT-J would be modified, while all other parameters remain frozen. Layer selection adheres to the locality of knowledge (Meng et al., 2023) or retains layers with higher success rates in pilot experiments (Mitchell et al., 2022a), as elaborated in Appendix B.

All hyperparameter classes derive from a common base class, `Hyperparams`, which includes necessary attributes and abstract methods. This base class supports loading hyperparameters in both *yaml* and *json* formats. Moreover, the `Hyperparams` base class can be used to initialize the Trainer module, streamlining the workflow.

3.5 Trainer

Certain editing methods, which employ meta-learning or utilize classifiers (as shown in Table 1), necessitate the training of additional parameters or the implementation of extra network structures. Similar to Hyperparameters (`Hparams`), all Trainer classes inherit from a common base class, `BaseTrainer`. It includes essential attributes and abstract methods such as run and validate steps. Subclasses of the `BaseTrainer` define specific training steps for editing, such as calculating editing loss and locality loss, as well as the strategies for combining these losses. Once additional network structures are obtained, the subsequent editing process follows the same path as the Training-Free method. In EASYEDIT, various Trainers can be easily called with one click.

4 Evaluation

Knowledge editing, as defined by Mitchell et al. (2022b), involves supplying a specific editing descriptor x_e (input instance) and an editing target y_e (desired output). From these, an editing instance z_e is generated in the form: $z_e \sim [x_e, y_e]$. The goal is to adjust the behavior of the initial base model f_θ (where θ represents the model's parameters) to produce an edited model f_{θ_e}. Ideally, for the editing instance, the edited model would behave such that $f_{\theta_e}(x_e) = y_e$. Additionally, the editing scope $S(z_e)$ refers to a set of input examples whose true labels have been influenced by the editing instance. In most cases, a successful edit should affect the model's predictions for numerous In-Scope ($I(x_e) \sim \{x'_e | x'_e \in S(z_e)\}$) inputs, while leaving Out-of-Scope ($O(x_e) \sim \{x'_e | x'_e \notin S(z_e)\}$) inputs unchanged.

We employ six dimensions of metrics to assess the performance of editing methods, including **Reliability**, **Generalization**, **Locality**, **Portability**, **Fluency** (Zhang et al., 2018) and **Efficiency** (as shown in Figure 3).

Reliability This metric measures the average accuracy on the given editing instance z_e.

Generalization The edit should appropriately influence in-scope inputs, this metric gauges the average accuracy on in-scope inputs $I(x_e)$.

Locality Editing should adhere to the principle of locality, it evaluates whether out-of-scope inputs $O(x_e)$ can remain unchanged as the base model.

Portability The robust generalization of the edit, assessing whether the edited knowledge can be effectively applied to related content.

Fluency It measures the weighted average of bi-gram and tri-gram entropies to assess the diversity of text generations.

Efficiency Editing should be time and resource-efficient. This metric quantifies efficiency by measuring editing time and VRAM consumption.

5 Experiments

In this section, we will outline the experiment setting and report the empirical results of multiple editing methods supported in EASYEDIT (Table 2).

5.1 Experiment Setting

To validate the potential application of knowledge editing on LLMs, we utilize **LlaMA 2** (7B) (Touvron et al., 2023b), a model with a large parameter size, representing the decoder-only structure.

We employ the ZsRE dataset to test the capability of knowledge editing in incorporating substantial and general fact associations into the model. ZsRE (Levy et al., 2017) is a question-answering (QA) dataset that generates an equivalence neighbor through back-translation. Later, it is further expanded by Yao et al. (2023) to provide a more comprehensive evaluation of knowledge editing, including an assessment of the LLMs' ability to integrate the edited fact with other facts related to the target object o* (an aspect of Portability). For baselines, we compare various editing methods and additionally employ FT-L from ROME (Meng et al., 2023). FT-L updates parameters for a single MLP layer and applies an $L\infty$ norm constraint to limit the weight changes.

5.2 Experiment Results

Table 2 reveals SERAC and IKE's superior performance on the ZsRE datasets, exceeding 99% on several metrics. While ROME and MEMIT perform sub-optimally in generalization, they exhibit relatively high performance in terms of reliability and locality. IKE exhibits the potential of gradient-free updates through in-context learning, leading to near-perfect scores in both reliability and generalization. However, it shows some deficiency in locality, as preceding prompts may influence out-of-scope inputs. GRACE exhibits poor generalization, possibly attributed to the lack of explicit semantic representation in its activations within the decoder-only model (Liu et al., 2023b). FT-L's performance on ZsRE falls significantly short compared to ROME, even though both methods modify the same layer parameters. This suggests that under the norm constraint, fine-tuning is not an effective strategy for knowledge editing. MEND performs well overall, achieving over 90% accuracy on multiple metrics and even surpassing ROME in terms of reliability and generalization. KN performs poorly, indicating that it may be better suited for editing tasks in smaller models or tasks involving knowledge attribution.

	Reliability	Generalization	Locality	Portability	Fluency
FT-L	56.94	52.02	96.32	51.03	488.41
SERAC	99.49	99.13	100.00	57.82	423.22
IKE	**100.00**	**99.98**	69.19	**67.56**	557.37
MEND	94.24	90.27	97.04	56.95	540.06
KN	28.95	28.43	65.43	37.18	478.32
ROME	92.45	87.04	99.63	57.47	**587.58**
MEMIT	92.94	85.97	99.49	60.64	576.51
GRACE	99.22	0.43	**100.00**	56.87	426.31

Table 2: Editing results of the four metrics on LlaMA-2 using EASYEDIT. The settings for the model and the dataset are the same with Yao et al. (2023).

For the Portability evaluation, where the inference depends on a single connection or 'hop' between facts, most editing methods struggle to effectively combine the edited fact with other facts relevant to the target object o*. While SERAC obtains good performance on previous metrics, it completely fails to propagate the edited knowledge. This is because SERAC utilizes an external model with a smaller parameter size for counterfactual routing whereas the smaller model struggles to recall a rich set of relevant facts. IKE still maintains a relatively high capability for ripple editing (exceeding 67%), demonstrating that in-context learning is a promising approach to propagate edited knowledge to other related facts.

6 Conclusion and Future work

We propose EASYEDIT, an easy-to-use knowledge editing framework for LLMs, which supports many cutting-edge approaches and various LLMs. The ability to edit and manipulate LLMs in a controlled and targeted manner may open up new possibilities for knowledge augmentation (Wu et al., 2023, 2020; Zhang et al., 2022; Chen et al., 2022) and adaptation across various natural language processing tasks (Kaddour et al., 2023). In the future, we will continue to integrate advanced editing technologies into EASYEDIT, aiming at facilitating further research and inspiring new ideas for the NLP community.

Acknowledgments

We thank the developers of the ROME[7] library for their significant contributions to the NLP community. We are grateful to Ting Lu and Yu Zhang who participated in the development of this project during the Zhejiang University Summer Camp. We also extend our gratitude to the NLP team at East China Normal University, particularly Lang Yu, for their support of the Melo module. Special thanks to Tom Hartvigsen for his contributions to the implementation of GRACE. We are grateful to the TMG-NUDT team for their valuable suggestions and technical support for the PMET method. We are grateful to Jia-Chen Gu from the University of California, Los Angeles, and Haiyang Yu from the Department of Cyberspace Security, University of Science for their constructive suggestions on development of EASYEDIT. We thank Yiquan Wu and Zeqing Yuan for helping the AAAI 2024 tutorial (canceled since part of speakers cannot present in person) of EasyEdit. Appreciation is also extended to all PR contributors, and issue feedback providers during the EasyEdit version iterations, especially Damien de Mijolla for proposing different optimization goals for FT, which complemented the fine-tuning baseline, and to Yuxuan Zhai for pointing out the portability metric evaluation issue of LlaMA-2-7B.

We would like to express gratitude to the anonymous reviewers for their kind comments. This work was supported by the National Natural Science Foundation of China (No. 62206246, No. NSFCU23B2055, No. NSFCU19B2027), the Fundamental Research Funds for the Central Universities (226-2023-00138), Zhejiang Provincial Natural Science Foundation of China (No. LGG22F030011), Yongjiang Talent Introduction Programme (2021A-156-G), CCF-Tencent Rhino-Bird Open Research Fund, Tencent AI Lab Rhino-Bird Focused Research Program (RBFR2024003), Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

Ethics Statement

The significance of knowledge editing lies in its direct impact on the behavior and output results of LMs. Malicious edits may lead to the generation of responses with toxicity or bias in LMs, posing potential harm to users and society. Therefore, when applying knowledge editing techniques or utilizing this system, careful consideration must be given to potential risks and ethical concerns. All our data undergoes meticulous manual inspection, and any malicious edits or offensive content have been removed.

[7] https://github.com/kmeng01/rome

References

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang

Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

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.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. KnowPrompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web Conference 2022*. ACM.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2023a. Evaluating the ripple effects of knowledge editing in language models. *CoRR*, abs/2307.12976.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2023b. Evaluating the ripple effects of knowledge editing in language models.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *CoRR*, abs/2104.08696.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

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*.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. *CoRR*, abs/2304.14767.

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.

Anshita Gupta, Debanjan Mondal, Akshay Krishna Sheshadri, Wenlong Zhao, Xiang Lorraine Li, Sarah Wiegreffe, and Niket Tandon. 2023. Editing commonsense knowledge in GPT. *CoRR*, abs/2305.14956.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3309–3326, Dublin, Ireland. Association for Computational Linguistics.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors.

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. *CoRR*, abs/2301.04213.

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2023. Inspecting and editing knowledge representations in language models.

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.

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

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

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *CoRR*, abs/2307.10169.

Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. 2023. Surgical fine-tuning improves adaptation to distribution shifts.

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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023a. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Tian Yu Liu, Matthew Trager, Alessandro Achille, Pramuditha Perera, Luca Zancato, and Stefano Soatto. 2023b. Meaning representations from trajectories in autoregressive models.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2023. Locating and editing factual associations in gpt.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022. Mass-editing memory in a transformer.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale.

OpenAI. 2023. Gpt-4 technical report.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap. *CoRR*, abs/2306.08302.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with language model prompting: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5368–5393, Toronto, Canada. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. 2022. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. Investigating the factual knowledge boundary of large language models with retrieval augmentation. *arXiv preprint arXiv:2307.11019*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language models via meta learning.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Tianxing Wu, Xudong Cao, Yipeng Zhu, Feiyue Wu, Tianling Gong, Yuxiang Wang, and Shenqi Jing. 2023. Asdkb: A chinese knowledge base for the early screening and diagnosis of autism spectrum disorder.

Tianxing Wu, Haofen Wang, Cheng Li, Guilin Qi, Xing Niu, Meng Wang, Lin Li, and Chaomin Shi. 2020. Knowledge graph construction from multiple online encyclopedias. *World Wide Web*, 23:2671–2698.

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.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2023. Melo: Enhancing model editing with neuron-indexed dynamic lora.

Wenhao Yu, Chenguang Zhu, Zhihan Zhang, Shuohang Wang, Zhuosheng Zhang, Yuwei Fang, and Meng Jiang. 2022. Retrieval augmentation for commonsense reasoning: A unified approach. 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 4364–4377. Association for Computational Linguistics.

Ningyu Zhang, Xin Xie, Xiang Chen, Shumin Deng, Hongbin Ye, and Huajun Chen. 2022. Knowledge collaborative fine-tuning for low-resource knowledge graph completion. *Journal of Software*, 33(10):3531–3545.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization.

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.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023a. Can we edit factual knowledge by in-context learning?

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, Limao Xiong, Lu Chen, Zhiheng Xi, Nuo Xu, Wenbin Lai, Minghao Zhu, Cheng Chang, Zhangyue Yin, Rongxiang Weng, Wensen Cheng, Haoran Huang, Tianxiang Sun, Hang Yan, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. 2023b. Secrets of RLHF in large language models part I: PPO. *CoRR*, abs/2307.04964.

A Preliminaries of Model Editing

The task of knowledge editing is to effectively modify the initial base model f_θ to the edited model $f_{\theta'}$, with corresponding parameter adjustments for a specific input-output pair (x_e, y_e), where $x_e \in \mathcal{X}_e$ and $f_\theta(x_e) \neq y_e$. Here, $\mathcal{X}_e$ represents the entire set to be edited. Therefore, the current problem formulation for knowledge editing can be broadly categorized into three types:

1. **Single Instance Editing**: Evaluating the performance of the model after a single edit. The model reloads the original weights after a single edit:

$$\theta' \leftarrow \arg\min_{\theta}(\|f_\theta(x_e) - y_e\|) \tag{1}$$

2. **Batch Instance Editing**: Simultaneously modifying N knowledge instances (where $N \ll |\mathcal{X}_e|$) and evaluating the performance of the edited model after processing a batch. The model reloads the original weights after processing a batch of edits:

$$\theta' \leftarrow \arg\min_{\theta} \sum_{e=1}^{N}(\|f_\theta(x_e) - y_e\|) \tag{2}$$

3. **Sequential Editing**: This approach requires sequentially editing each knowledge instance, and evaluation must be performed after all knowledge updates have been applied:

$$\theta' \leftarrow \arg\min_{\theta} \sum_{e=1}^{|\mathcal{X}_e|}(\|f_\theta(x_e) - y_e\|) \tag{3}$$

B Default Hparams Settings

EASYEDIT provides optimal hyperparameters for various editing methods. In addition to common parameters such as learning rate, steps, and regularization coefficients, the location of layers for editing can also be considered as hyperparameters, significantly influencing the robustness of the editing process. The following tables demonstrate the default location settings in EASYEDIT (using **Llama-2-7B** as an example).

Layer with Value Loss
`model.layers.31`
Target Layer for Updating Weights
`model.layers.5.mlp.down_proj`

Table 3: Default Target Modules in **ROME**

Layer with Value Loss
`model.layers.31`
Target Layer for Updating Weights
`model.layers.4.mlp.down_proj`
`model.layers.5.mlp.down_proj`
`model.layers.6.mlp.down_proj`
`model.layers.7.mlp.down_proj`
`model.layers.8.mlp.down_proj`

Table 4: Default Target Modules in **MEMIT** and **PMET**

ROME We follow Meng et al. (2023) in utilizing causal mediation analysis to identify an intermediate layer in the model responsible for recalling facts. The causal traces reveal an early site (5th layer) with causal states concentrated at the last token of the subject, indicating a significant role for MLP states at that specific layer (Table 3).

MEMIT Following Meng et al. (2022), we quantify the average indirect causal effect of MLP modules. The results demonstrate a concentration of intermediate states in LLaMA. The disparity in the effects between MLP severed and hidden states severed becomes significantly reduced after the 8th layer. We choose the entire critical range of MLP layers, denoted as $\mathcal{R} = \{4, 5, 6, 7, 8\}$ (Table 4).

PMET PMET (Li et al., 2024) adopts the localization strategy from MEMIT, designating the corresponding layer as the modification target. Building upon the update of MLP weights, PMET focuses on multi-head self-attention (MHSA), further substantiating the discovery that MHSA encodes specific patterns for general knowledge extraction. (Table 4).

MEND In the context of meta-learning for editing, it is commonly observed that editing MLP layers yields better performance than editing attention

CodeBook Target Modules
model.layers[27].mlp.down_proj.weight

Table 5: Default Target Modules in **GRACE**

Target Layer for Updating Weights
model.layers.29.mlp.gate_proj.weight
model.layers.29.mlp.up_proj.weight
model.layers.29.mlp.down_proj.weight
model.layers.30.mlp.gate_proj.weight
model.layers.30.mlp.up_proj.weight
model.layers.30.mlp.down_proj.weight
model.layers.31.mlp.gate_proj.weight
model.layers.31.mlp.up_proj.weight
model.layers.31.mlp.down_proj.weight

Table 6: Default Target Modules in **MEND**

layers. Typically, MLP weights of the last 3 transformer blocks (totaling 6 weight matrices) are chosen for editing (Mitchell et al., 2022a). EASYEDIT adheres to this default configuration (Table 6).

GRACE Recent studies have revealed the impact of selecting the right layers for fine-tuning (Lee et al., 2023). Similarly, in GRACE (Hartvigsen et al., 2023), we conduct pilot experiments, retaining layers with consistently high edit success rates (Table 5).