MULFE: A Multi-Level Benchmark for Free Text Model Editing

Chenhao Wang[1,2], Pengfei Cao[1,2], Zhuoran Jin[1,2], Yubo Chen[1,2]
Daojian Zeng[3], Kang Liu[1,2,4,*], Jun Zhao[1,2,*]
[1]The Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Hunan Normal University
[4]Shanghai Artificial Intelligence Laboratory
{chenhao.wang, pengfei.cao, zhuoran.jin, yubo.chen, kliu, jzhao} @nlpr.ia.ac.cn
zengdj916@163.com

Abstract

Adjusting the outdated behaviors of large language models (LLMs) after deployment remains a significant challenge. It motivates the model editing research, which is however mainly explored in a restricted task form with triple-based edit requests. Recent works have initiated a transition to a more practical and unified editing task that takes free-form text as edit requests. However, there are gaps in nuanced benchmark designs and re-evaluation of existing methods. To bridge the gaps, we introduce a multi-level benchmark for free text model editing (MULFE). The benchmark categorizes probe queries into three levels of generalization, ranging from basic literal memory to deeper understanding and reasoning. Based on the benchmark, we conduct extensive experiments across various base models, edit sizes, and editing methods, including adaptations of mainstream locate-and-edit and hypernetwork methods. The results highlight the inconsistent behaviors of edited models on different generalization levels. Higher-level generalization remains a significant challenge. Based on the findings, we propose SIDE, a simple yet effective method based on in-context distillation to enhance the generalization performance. The benchmark dataset and evaluation scripts are publicly available at http://github.com/wchrepo/mulfe.

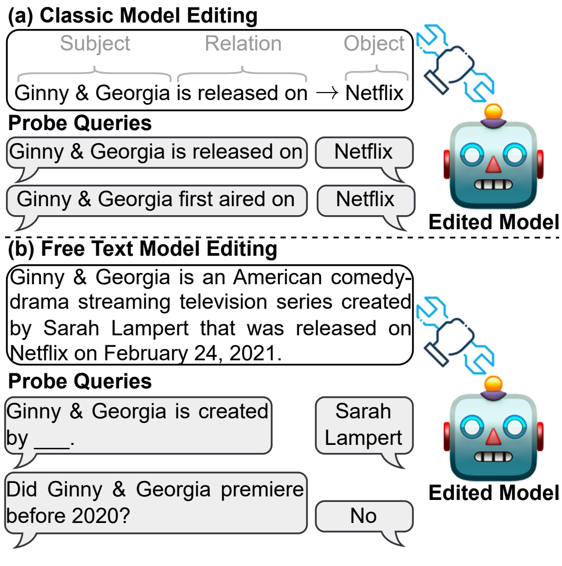

Figure 1: (a) Classic task of model editing. The edit request is typically based on relational knowledge triples. (b) Free text model editing investigated in this paper. The edit request is a piece of free-form text.

1 Introduction

Large Language Models (LLMs) have showcased impressive capabilities in comprehending human language (Li et al., 2023) as well as vast parametric knowledge obtained from large corpora (Petroni et al., 2019; Cao et al., 2024). However, as new information keeps emerging, adjusting the outdated behaviors of LLMs after deployment remains a significant challenge. Unlike humans, who can naturally assimilate new knowledge from new text and adjust specific aspects of their understanding, accurately and effectively updating LLMs with new information is non-trivial (Hua et al., 2024). To tackle this, the field of model editing (or knowledge editing), has emerged (De Cao et al., 2021; Yao et al., 2023). It focuses on methods for lightweight updates on LLMs, ensuring the responses to relevant inputs are modified as expected (termed "efficacy" or "edit success") while minimizing adverse effects on other inputs (termed "specificity" or "locality").

Previous work mainly investigates a restricted form of the problem (De Cao et al., 2021; Meng et al., 2022; Mitchell et al., 2022a), where the edit request is expressed as a tuple of input and desired output, typically based on relational knowledge triples in the form of (*subject, relation, object*). As shown in Figure 1(a), after edited with "*Ginny & Georgia is released on* → Netflix", the model is expected to give the target response "*Netflix*" to both

[*]Corresponding authors.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13570–13587
August 11-16, 2024

the edit input and its similar expressions. However, the practicality of such task setting remains limited, because new knowledge is often encountered in free-form text rather than well-organized tuples. Therefore, some recent works (Onoe et al., 2023; Akyürek et al., 2023) introduce a more intuitive but challenging task, which we refer to as free text model editing. As shown in Figure 1(b), the edit request is a piece of free text, and the edited model needs to correctly respond to various related probe queries. Despite previous works have initiated such a transition to a more practical task form, there are notable gaps in benchmark designs and evaluation. **(1) Lacking nuanced benchmark designs:** A key challenge of the task is that the potential queries could vary greatly in difficulty and rely on different abilities, ranging from literal reciting to implicit reasoning. However, previous benchmarks overlook the diversity of queries and lack categorization in data construction. Therefore, only vague overall performance is reported in the results, hindering in-depth diagnosis of the bottleneck of methods. **(2) Lacking re-evaluation of existing methods:** Many mainstream model editing methods rely on the triple-based input structures, making them not directly applicable beyond the classic model editing setting without adaptation. Therefore, these methods are rarely investigated in previous work. Their adaptability to free text model editing is largely unknown and requires comprehensive re-evaluation.

To address these gaps, we introduce a **mu**lti-**l**evel benchmark for **f**ree text model **e**diting (**MULFE**), and provide comprehensive experiment results across various settings. Specifically, inspired by Bloom's Taxonomy about cognitive levels (Bloom, 1956) and recent knowledge analysis results on LLMs (Allen-Zhu and Li, 2023; Chen et al., 2024), we define three levels of generalization for the probe queries, spanning from the simple literal recall to more profound understanding. Moreover, we create additional fine-grained tags to further distinguish the probe queries. These levels and tags provide diverse dimensions for analyzing editing performance. Following the proposed guidelines, we construct a dataset with 3700 edits and 40,000 probes for the re-implementation of trainable editing methods, and manually curate a high-quality dataset with 285 edits and 2300 probes for evaluation. Utilizing the data, we undertake extensive model editing experiments across various base models, editing sizes and editing methods. To accommodate mainstream locate-and-edit methods and hypernetwork methods in the experiments, we explore re-implementations and edit simplification strategies. Our empirical findings highlight the inconsistent behaviors of edited models on different levels. Higher-level generalization remains a significant challenge to current methods. Based on the findings, we propose a simple yet effective method **SIDE**, which incorporates question generation and in-context distillation, largely improving the performance on higher generalization levels.

We summarize the contribution as follows.

- We introduce **MULFE**, a multi-level benchmark for free text model editing. It contains a high-quality evaluation dataset with manually curated probe queries, which are categorized into three generalization levels and annotated with tags for multi-dimensional analysis.
- We present extensive experiment results and analyses across different base models, editing sizes, and editing methods. The results show that the edited model exhibits notable differences in performance across different levels of questions, indicating the challenges of the free text model editing task.
- Based on the best practices in the experiments, we propose a simple yet effective method **SIDE**, which incorporates question generation and in-context distillation training, serving as a strong baseline for future study.

2 Related Work

2.1 Model Editing Methods

In the narrow sense, model editing methods should update the model weights. The methods typically include the variants of fine-tuning which directly update the model weights (Zhu et al., 2020; Sinitsin et al., 2020; Hu et al., 2022), hypernetwork-based methods which train a hypernetwork to update the model weights (Sinitsin et al., 2020; De Cao et al., 2021; Hase et al., 2023; Mitchell et al., 2022a; Tan et al., 2023; Zhang et al., 2024), and locate-and-edit methods which selectively update the model weights based on the knowledge mechanism analysis (Dai et al., 2022; Meng et al., 2022, 2023; Ma et al., 2023). Recently, there is also a family of methods that tackle the knowledge editing task with additional parameters or memory components (Mitchell et al., 2022b; Huang et al., 2023).

Retrieval-augmented methods can also be considered as one of them (Gao et al., 2023; Ovadia et al., 2023). Although these methods are valuable alternatives in real-world applications, they fundamentally change the architecture and thus have different outcomes. In this paper, we mainly investigate the model editing methods in the narrow sense, exploring the performance that can be achieved by modifying the model parameters.

2.2 Benchmarks for Model Editing

Model editing is a rapidly evolving field and various benchmarks have been proposed. Most of them are created based on knowledge triples (De Cao et al., 2021; Meng et al., 2022). To more comprehensively study the editing performance, there is a recent trend to create benchmarks for special topics such as time-series knowledge editing (Dhingra et al., 2022; Yin et al., 2023), cross-lingual knowledge editing (Wu et al., 2023; Wang et al., 2023a,b), and multi-hop generalization (Zhong et al., 2023; Cohen et al., 2023). In this paper, we focus on the free text model editing task, which presents practical challenges for this research area. The closest works to this paper include Onoe et al. (2023) and Akyürek et al. (2023). Compared with them, our work creates larger datasets with more detailed categorization, facilitating a more comprehensive evaluation of different methods.

3 Free Text Model Editing

3.1 Task Definition

Formally, provided with an **edit request** expressed in free-from text, a model with pretrained parameters θ is updated with a certain editing method, which results in an edited model in the same architecture with new parameters θ'. A set of probe queries (abbreviated as **probes**) are then used to test whether the edited model satisfies the desired editing criteria. Specifically, each probe is a pair of a knowledge-intensive question and a target answer, denoted as (q, a). The edited model is expected to assign high probability to a when providing q as the input. There are two kinds of probes corresponding to different criteria. **Efficacy probes** are based on the information conveyed in the edit request, testing whether the model successfully internalizes the new information. **Specificity probes** are based on the knowledge that the model has learned during pretraining, testing whether the editing procedure negatively affects previous unrelated knowledge.

3.2 Metrics

To quantify how well the edited model behaves in terms of efficacy and specificity, the corresponding subsets of probes are evaluated on two base metrics: **Exact-Matching Accuracy** (**EM**) and **Per-Token Perplexity** (**PPL**). The metrics are formally given as follows.

$$EM = \frac{\sum_{(q,a)\in\mathcal{P}} \mathbb{1}\{\text{GreedyDecoding}(\theta', q) = a\}}{|\mathcal{P}|}$$
$$PPL = \exp\left(\frac{-\sum_{(q,a)\in\mathcal{P}} \log p_{\theta'}(a|q)}{\sum_{(q,a)\in\mathcal{P}} \text{Tokens}(a)}\right) \quad (1)$$

where $\mathcal{P}$ is a probe set, $p_{\theta'}(y|x)$ represents the conditional probability predicted by the edited model, $\text{GreedyDecoding}(\theta', q)$ means the greedy decoding output given q as the input, $\mathbb{1}$ denotes the indicator function, and $\text{Tokens}(a)$ is the token quantity in a.

Intuitively, a larger EM metric and a smaller PPL metric signal better performance. EM directly indicates the model's ability to precisely generate the target answer and can be compared across different base models. PPL provides a more nuanced reflection of the answer uncertainty while it is not comparable for models with different tokenization.

4 MULFE Benchmark

4.1 Overview

As illustrated in Figure 2, an editing instance of MULFE includes three key components: the *edit request*, *multi-level efficacy probes*, and *specificity probes*. In the following sections, we will first introduce the multi-level designs and data curation procedure of the efficacy probes, and then describe the construction of specificity probes.

4.2 The Levels of Efficacy Probes

Intuitively, a successful edit should result in not only the direct memorization of the original text but also good generalization on a variety of relevant questions. To provide more analytical dimensions, we define three levels of generalization as follows.

- **Level 1**: The probe questions are clozes to complete the fragments that appears in the original text. At this level, the edited model needs to memorize the surface form of new information, achieving the completion of partial content. For example, the level-1 probe in Figure 2 directly comes from the beginning of the edit request text.

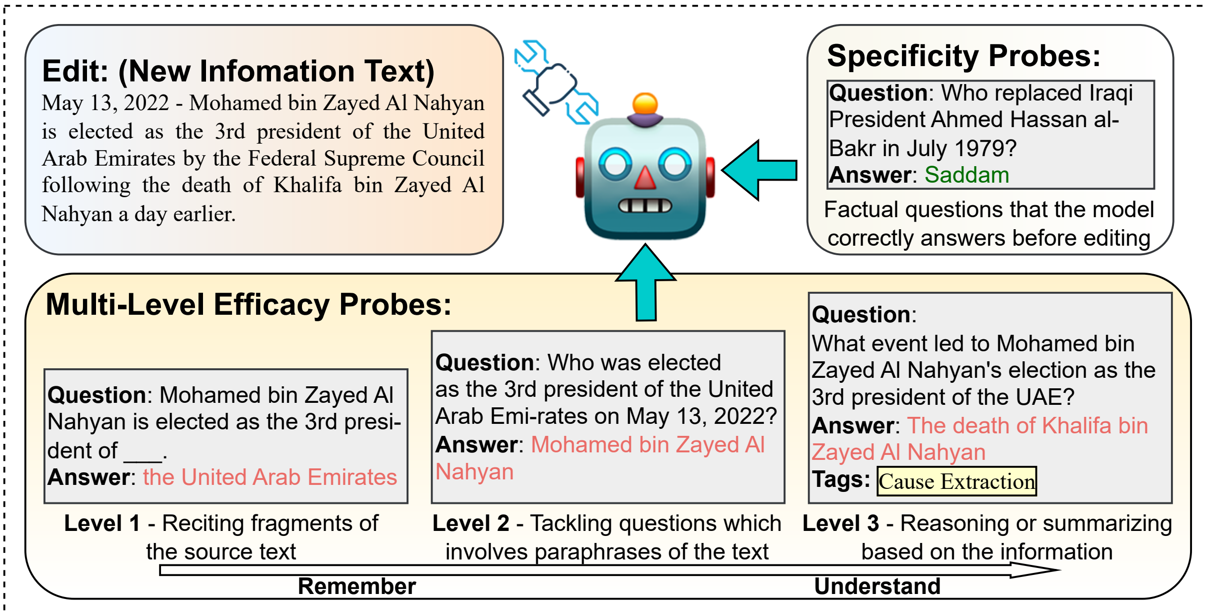

Figure 2: The illustration of an instance in MULFE. The edit request is a piece of text with new information. The specificity probes test how whether the previous knowledge of the model is reserved. The multi-level efficacy probes test how well the model internalize the new information, ranging from basic remembering to deeper understanding.

- **Level 2**: The probe questions include simple synonymous variants or paraphrases of the original text. At this level, the edited model needs to understand the linguistic transformation of the new information. For example, the level-2 probe in Figure 2 is based on the first half of the text.

- **Level 3**: The probe questions require additional reasoning and summarizing ability. At this level, the model needs to have deeper understanding and reasoning based on the new information. For example, the level-3 probe in Figure 2 asks to extract the cause of the event.

The design is inspired by the Bloom's Taxonomy (Bloom, 1956), which describes different cognitive levels of educational learning objectives. However, considering the nature of LLMs, our categorization focuses on the basic remembering and understanding. For ease of differentiation, we only define three levels. As a result, there could be a variety of probes with different characteristics classified as level 3. For more fine-grained categorization, we provide a series of informal tags for the level-3 probes, indicating the type of answers they ask for or some featured issues they may related to, such as *Reversal Curse* (Berglund et al., 2023) and *Partial Retrieval* (Allen-Zhu and Li, 2023). Refer to Appendix A.1 for more details.

4.3 Data Collection and Curation

To construct the editing data, we first collect a set of text snippets as the edit requests. For evaluation data, to align with the domain of previous work, we reuse the edit requests in Entity Inference (Onoe et al., 2023) and DUNE (Akyürek et al., 2023), which mainly consist of entity and event descriptions from recent Wikipedia pages. Additionally, we collect 3700 snippets from Wikipedia as the edit requests for the training dataset. For each request, we utilize GPT-4 to generate efficacy probes of the three levels. After that, we manually curate the evaluation data to ensure the quality of probes. Refer to Appendix A.2 and A.3 for more details.

4.4 Dynamic Specificity Probes

Previous work usually applies a fixed set of specificity probes. However, if the base model has little of the corresponding knowledge, the numerical result of specificity could be low and insensitive to the editing process. Ideally, the specificity probes should be related to the knowledge previously encoded in the model. Therefore, in MULFE, the specificity probes are dynamically constructed for each base model, ensuring that it has already mastered the knowledge and yields 100% EM accuracy. Specifically, we evaluate the base models on TriviaQA (Joshi et al., 2017) and collect the QA instances that can be robustly answered. In this way,

Dataset	Edit Request	Efficacy Probes	Probes/Edits
Onoe et al. (2023)	85	85	1
Akyürek et al. (2023)	200	1000	5
MULFE (Evaluation)	285	2300	8.1
- Level 1/2/3	285	436/910/954	1.5/3.2/3.3
MULFE (Training)	3700	40000	10.8

Table 1: The statistics of MULFE and similar datasets.

we sample 400 specificity probes for each model. Refer to Appendix A.4 for more details.

4.5 Dataset Summary

The statistics of datasets are shown in Table 1. MULFE combines the wiki-styled edit requests in Onoe et al. (2023) and Akyürek et al. (2023), contains larger size of manually curated efficacy probes with fine-grained categorization, and provides additional training dataset for method development.

5 Experiments Setup

5.1 Re-implementation of Previous Methods

In experiments, we evaluate four groups of methods on MULFE, which are briefly described as follows.

Non-Editing Baselines For comparison, we directly evaluate the unedited base models on the dataset, the results are denoted as **Before-Editing**. Also, we include a baseline that provides the edit request in the context before each probe, denoted as **Edit-In-Context**. It works in a way similar to reading comprehension. Although it is actually not an editing method, it can show the achievable performance gains when the ground truth edit is provided to the model. It can also be seen as a performance bound of retrieval-based methods.

Fine-Tuning We evaluate standard fine-tuning, fine-tuning single layer, and LoRA (Hu et al., 2022). As the performance is highly dependent on the hyper-parameter setting, we set a threshold condition for the specificity ($EM > 90\%$) and report the best efficacy in the main results, leaving the detailed analysis of hyper-parameters in Section 6.2.

Locate-and-Edit Two representative locate-and-edit methods, **ROME** (Meng et al., 2022) and **MEMIT** (Meng et al., 2023), are evaluated in the experiments. Note that these methods are developed for triple-based editing. We will describe how we apply them to free text model editing with edit simplification strategies in Section 5.2.

Hypernetwork We evaluate **MEND** (Mitchell et al., 2022a), a state-of-the-art hypernetwork-based editing method in the experiments. MEND is a trainable method. Therefore, besides reusing the original MEND checkpoints, we additionally implement a MEND variant using the training data of MULFE, which is denoted as **MEND-MULFE**.

5.2 Beforehand Edit Simplification

To reuse classic editing methods in free text model editing, a straightforward way is to simplify the free text edit request into a list of triple-based edit requests before applying the editing, while there could be a loss of information. We refer to the procedure as simplification. In this paper, we examine several possible simplification strategies.

OpenIE Extracting open-domain relation triples from text is a classic NLP task, termed OpenIE. In this work, we utilize two OpenIE tools, Stanford-OpenIE (Angeli et al., 2015) and DOCoR(Yong et al., 2023), to extract triples from the edit request text. After that, the triples are converted according to the edit requests format of classic edit methods.

Question Generation Another way to break down the text into factual tuples is question generation (QG). We use LMQG (Ushio et al., 2023) to generate QA pairs with two strategies. The first one is to directly generate QA instances with an end-to-end QG model. We denote the strategy as **E2EQG**. The second strategy is to extract all entities from the text as answers and generate the corresponding questions. We denote the strategy as **NERQG**. Finally, we extract entities from the questions and convert the results into triple-based edit requests.

5.3 New Baseline: Simplification and In-context Distillation Editing (SIDE)

After some preliminary experiments, we find that Edit-In-Context shows significant advantages compared to others (6.1). Unfortunately, this method does not update the internal knowledge of the model, its performance is not solidified in the absence of context. However, it has the potential to serve as a teacher for knowledge distillation (Hinton et al., 2015; Snell et al., 2022) and make the model fit its predictions, which could be useful supplement to directly learning the edit request. Inspired by this notion, we combine Edit-In-Context with our best practices in edit simplification, and propose a simple yet effective method SIDE for free text model editing.

	Edit	Level 1		Level 2		Level 3		Overall Efficacy		Specificity	
	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓
Before-Editing	17.62	8.49	22.52	7.25	30.21	6.49	28.08	7.17	27.93	100.00	1.82
Edit-In-Context	1.24	**85.32**	**1.51**	**70.33**	**1.79**	**38.74**	**3.22**	**60.06**	**2.29**	85.34	1.95
FT (Full Model)	**1.00**	<u>76.15</u>	2.00	52.75	3.36	23.66	4.98	45.11	3.74	90.66	1.85
FT (LoRA)	1.01	67.66	2.18	47.03	3.91	23.25	6.12	41.07	4.42	90.57	1.73
FT (Single Layer)	<u>**1.00**</u>	74.08	2.06	48.02	3.60	19.39	6.01	41.09	4.21	92.70	1.76
MEMIT (w/ Sim.)	17.64	20.41	15.38	18.02	16.97	18.13	18.67	18.52	17.49	99.66	1.80
ROME (w/ Sim.)	22.68	19.95	20.36	22.09	20.07	18.97	29.67	20.39	24.15	98.07	1.82
MEND (w/ Sim.)	19.80	26.15	13.93	27.25	14.40	22.96	32.42	25.26	20.95	86.02	2.04
MEND-MULFE	2.31	52.52	3.75	36.92	5.23	26.73	6.28	35.65	5.42	96.99	1.84
SIDE (Section 5.3)	1.03	73.17	<u>1.93</u>	<u>59.56</u>	<u>2.23</u>	<u>35.95</u>	<u>3.66</u>	<u>52.35</u>	<u>2.75</u>	90.51	1.67

Table 2: Overall comparison of different editing methods on GPT-J model (**FT**=Fine-Tuning, **Sim.**=Simplification, **Edit PPL**=Perplexity on edit request). The best and second-best results are highlighted with **Bold** and <u>**Underline**</u> respectively. The results of fine-tuning are obtained under a specificity threshold condition ($EM > 90\%$).

First, we need a set of training instances to obtain the in-context predictive distribution. As did in the NERQG strategy, we extract the entities that appears in the edit request and generate a series of QA pairs. After that, we train the model with both the language modeling objective and the knowledge distillation objective. Inspired by classifier-free guidance (Ho and Salimans, 2022) and contrastive decoding (O'Brien and Lewis, 2023), we hope to amplify the impact of context. Therefore, conditioned on a QA pair (q, a) and an edit e, we obtain the teacher distribution $p_t(\cdot|q)$ as follows.

$$\log p_t(\cdot|q) = (1+\lambda)\log p_\theta(\cdot|q,e) - \lambda \log p_\theta(\cdot|q) \tag{2}$$

where $p_\theta(\cdot|q,e)$ and $\log p_\theta(\cdot|q)$ are the in-context distribution and direct predictive distribution yielded by the unedited model, and λ is a coefficient for further strengthen the influence of the context.

Combining different objectives, we update the model parameter θ' with the following loss.

$$L_{total} = \alpha L_e + \beta L_{soft} + \gamma L_{hard} \tag{3}$$

where L_e is the language modeling loss on the edit request e, $L_{soft} = KL\left(p_t(\cdot|q) \parallel p_{\theta'}(\cdot|q)\right)$ is the soft target loss (KL-divergence between the student and teacher distributions), $L_{hard} = -logp_{\theta'}(a|q)$ is the hard target loss (conditional log-likelihood of target answer), and α, β, γ are coefficients. The losses are averaged by token in the actual training process. The coefficients are searched on the training data of MULFE. Intuitively, SIDE is a variant of naive fine-tuning with additional training loss.

5.4 Other Implementation Details

Following the common practice in model editing research, in each round we edit the model with one edit request, and evaluate the model on the corresponding efficacy probes and all the specificity probes. After all edit requests are tested, we summarize the results according to the metrics in Equation 1. Besides the single edit setting, We also discuss the results of batch editing in Section 6.4, i.e. editing the model with a batch of edit requests simultaneously.

To investigate the impact of base language models, we test GPT2-XL (1.5B), GPT-Neo (2.7B), GPT-J (6B), and LLaMA2 (7B) in the experiments. These models have different sizes and are widely used in previous model editing research. For the sake of simplicity, we mainly report the results of GPT-J, and specifically discuss the impact of base models in Section 6.5. Refer to Appendix B for hyper-parameter settings and other details.

6 Results

6.1 Overall Results

Table 2 shows the overall comparison of different editing methods with the best hyper-parameter settings and simplification strategies. From the results, we have several main observations: (**1**) Edit-In-Context is substantially ahead in terms of efficacy, indicating that the model excels in utilizing information provided in the context but struggles to internalize the information as parameters. Meanwhile, irrelevant context could disturb the behavior of the model, resulting in the damage of speci-

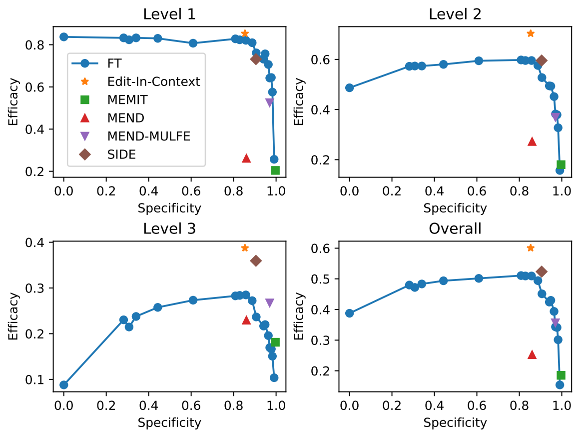

Figure 3: The efficacy-specificity curve of EM scores.

ficity. (**2**) With proper hyper-parameter settings, fine-tuning can bring efficacy gains on all three generalization levels. Nevertheless, there is a substantial difference among different levels. Level 1 has much more gains than level 3, indicating that the editing is mainly helpful for the surface memory. (**3**) With edit simplification, MEMIT, ROME, and MEND are successfully adapted to free text model editing. However, they also suffer from the information loss in the simplification procedure. As a result, their efficacy gains are small and similar on different levels. This is quite different from the outcome of fine-tuning. (**4**) Utilizing the MULFE training dataset, MEND-MULFE shows better performance than the original MEND checkpoint, indicating the importance of training process for hypernetwork methods. (**5**) Our method SIDE (described in Section 5.3) largely improves the efficacy, especially for level 2 and level 3 probes. Therefore, it can serve as a strong baseline for further study. Refer to Appendix C for similar comparison tables of different base models.

6.2 Fine-Tuning with Different Settings

Efficacy-Specificity Trade-Offs Generally, We want the edited model to perform well in both efficacy and specificity. However, the two metrics involve trade-offs, and the hyper-parameters setting of fine-tuning could largely influence their balance. For a more thorough analysis, a simple result table is insufficient. Therefore, we try different settings and show the trade-offs curve in Figure 3. Each evaluation run is corresponding to a result point. The main findings include: (**1**) The lower right corner (specificity=1) is corresponding to the performance before editing the model. When increasing the fine-tuning extent (i.e. larger learning rate or steps), the result point tends to move towards the upper left direction (better efficacy and worse specificity). (**2**) However, the efficacy of level 3 first rises and then falls. It indicates fine-tuning can improve higher-level generalization at first, but the damage to the model's ability gradually becomes dominant. Therefore, both the specificity and higher-level generalization are impacted. (**3**) Result points of most other editing methods are close to or below the curve. Therefore, they have no significant advantage compared to fine-tuning with proper hyper-parameters.

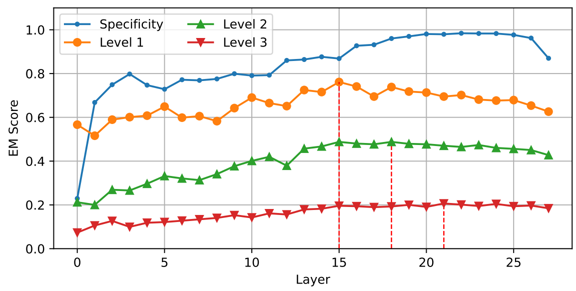

Figure 4: The EM scores of fine-tuning a single layer.

	Level 1	Level 2	Level 3	Overall	Specificity
MEMIT+SOIE	14.68	8.24	8.70	9.65	99.72
MEMIT+DOCoR	**23.17**	11.21	9.01	12.57	99.88
MEMIT+E2EQG	17.66	**19.34**	13.63	16.65	99.79
MEMIT+NERQG	20.41	18.02	**18.13**	**18.52**	99.66
MEND+SOIE	13.30	7.69	7.43	8.65	85.77
MEND+DOCoR	**27.52**	17.91	10.26	16.56	**98.84**
MEND+E2EQG	24.08	26.37	15.39	21.38	95.06
MEND+NERQG	26.15	**27.25**	**22.96**	**25.26**	86.02

Table 3: The EM score comparison of different simplification strategies. (SOIE=Stanford-OpenIE)

Single Layer Fine-Tuning Figure 4 depicts the results of fine-tuning a single layer of GPT-J. Different layers vary in the potential to accommodate new knowledge varies among. Updating earlier layers does not bring much efficacy gains and largely damages the specificity. Better efficacy results are obtained by updating the middle later layers. However, there are slight differences among different levels of probes. For example, the best layer for level 1 is earlier than the best layer for level 3.

6.3 Strategies of Edit Simplification

We show the performance of MEMIT and MEND with different simplification strategies in Table 3. QG-based strategies (E2EQG and NERQG) perform betters than OpenIE-based strategies (SOIE and DOCoR). We conjecture that QA pairs could

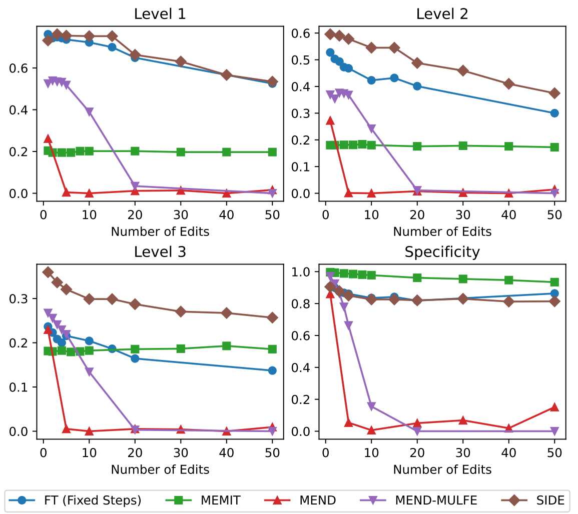

Figure 5: The EM scores of batch editing.

retain more information from the edit request. Due to the same reason, NERQG is more effective than E2EQG. As NERQG ensures a question for each mentioned entity, it produces QA pairs that convey richer knowledge.

6.4 Batch Editing

In this section, we discuss the results of editing the model with multiple edit requests, i.e. the batch editing setting. The results are shown in Figure 5. As the edit batch size increases, there are significant differences between different methods. (**1**) The efficacy of fine-tuning with fixed steps gradually decreases, while the specificity tends to stabilize. (**2**) SIDE shows a similar trend to basic fine-tuning while retains higher efficacy for larger edit size. (**3**) MEMIT is specially designed for batch editing, it shows the most robust performance with slight degradation in specificity. It also surpasses fine-tuning in level 3 when the edit size is larger than 20, yet the preprocessing bottleneck (edit simplification) restricts its upper limit performance. (**4**) For unseen edit batch sizes, MEND shows a remarkable performance decreases, resulting in near-zero efficacy and specificity for larger edit sizes. In general, as different methods have their own limitations, batch editing remains a significant challenge for free text model editing.

6.5 The Impact of Base Model

In Figure 6, We report the best editing performance on different base models, in comparison with the performance of Before-Editing and Edit-In-Context. For most models, Edit-In-Context re-

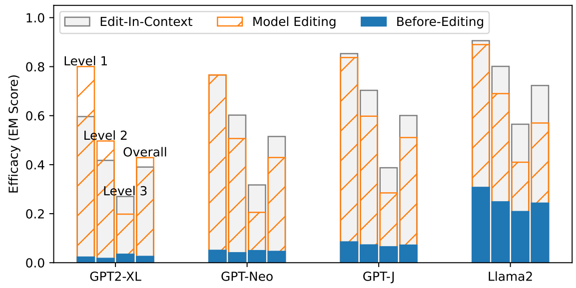

Figure 6: The comparison of different base models. The "Model Editing" results refer to the best efficacy obtained by model editing.

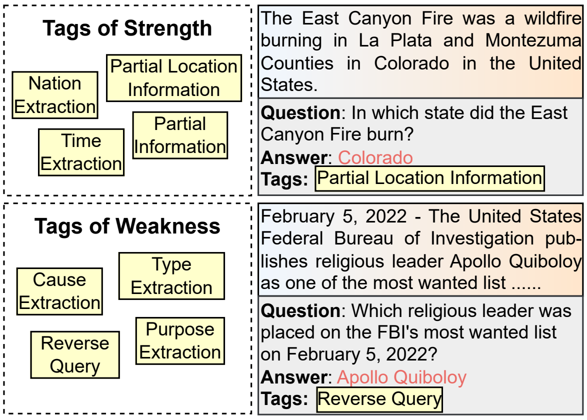

Figure 7: The tags with the high or low EM scores and their cases. We omit the tags with less than 15 cases.

marks the upper bound of performance. GPT2-XL is an exception because of its weaker context utilization ability. Besides, model editing methods can reach or surpass the performance of Edit-In-Context in level 1, but there is always a gap in level 3. The finding is consistent across different base models, which highlights that higher-level generalization is a significant challenge for free text model editing.

6.6 Cases Study

Figure 7 shows the tags with the high or low EM scores (obtained by fine-tuning GPT-J), as well as corresponding cases. In general, the edited model is better at recalling named entities that appears in the edit request, and it does well on partial retrieval, e.g. recalling a specific part of the location information. Meanwhile, the edited model struggles on recalling causal information, type information, and conducting reverse query, e.g. recalling a person through its description.

7 Conclusion

This work presents the MULFE benchmark for free text model editing and provides extensive empirical results across different settings. For the take-home message, we highlight the findings as follows.

- After editing a model with free text edit requests, the editing efficacy could vary substantially on probes of different generalization levels. Higher generalization levels are still challenging for current methods.
- Current model editing methods (in narrow sense) have significant gaps to the performance bounds set by Edit-In-Context, indicating there is large room for improvement.
- Through edit simplification, mainstream methods for triple-based editing can be applied to free text editing. However, the pipeline could lose details in the original text, resulting in inferior results compared to fine-tuning.

Based on the findings, we propose a simple yet effective method SIDE, which shows substantial improvement in the higher generation levels. However, much work is needed for an all-round solution. To sum up, free text model editing is a practical but largely unsolved task. The evaluation benchmark and baseline methods proposed in this paper can facilitate further work in this field.

8 Limitations

This work has limitations in data construction and editing methods, which are described as follows. (1) This work focuses on free text model editing, where the edit request is expressed in free-form text. This formulation could cover a diverse range of editing scenarios with various text styles. However, for the convenience of data collection and processing, we focus on short Wikipedia-styled texts of approximately two sentences in length as the source of edit requests. (2) This works mainly utilizes public Wikipedia corpus, pre-existing datasets, and AI generation in data construction. We have excluded potentially offensive text in the evaluation data through manual curation. But we do not carefully check the training dataset. (3) Model editing could be categorized as different operations such as adding, erasing and updating. This paper mainly involves the operations of adding and updating but does not make careful identification. (4) The boundaries of the proposed three levels are not very strict. There could be misclassification in the dataset. Also, there could be a more nuanced categorization scheme, but we do not further explore that due to the complexity. (5) We only investigate model editing methods in a narrow sense, i.e. directly modifying the model weights without structure changing. Methods of increasing the model structure or utilizing external memory components are excluded in this work. (6) The proposed method SIDE still suffers from the degradation in large batch editing, and also requires large memory usage as it is a variant of fine-tuning.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (No. 2022ZD0160503) and the National Natural Science Foundation of China (No. 62176257, 62276095).

References

Afra Akyürek, Eric Pan, Garry Kuwanto, and Derry Wijaya. 2023. DUnE: Dataset for unified editing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1847–1861, Singapore. Association for Computational Linguistics.

Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Physics of Language Models: Part 3.2, Knowledge Manipulation. ArXiv:2309.14402 [cs].

Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Physics of language models: Part 3.2, knowledge manipulation. *CoRR*, abs/2309.14402.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2023. The reversal curse: Llms trained on "a is b" fail to learn "b is a". *CoRR*, abs/2309.12288.

Benjamin. S. Bloom. 1956. *Taxonomy of Educational Objectives: The Classification of Educational Goals*. Longmans, Green.

Boxi Cao, Hongyu Lin, Xianpei Han, and Le Sun. 2024. The life cycle of knowledge in big language models:

A survey. *Machine Intelligence Research*, 21(2):217–238.

Bowen Chen, Xiao Ding, Yi Zhao, Bo Fu, Tingmao Lin, Bing Qin, and Ting Liu. 2024. Text difficulty study: Do machines behave the same as humans regarding text difficulty? *Machine Intelligence Research*, 21(2):283–293.

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

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

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing Factual Knowledge in Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997.

Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. 2023. Methods for measuring, updating, and visualizing factual beliefs in language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2714–2731, Dubrovnik, Croatia. Association for Computational Linguistics.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *CoRR*, abs/2207.12598.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Shangying Hua, Shuangci Jin, and Shengyi Jiang. 2024. The Limitations and Ethical Considerations of ChatGPT. *Data Intelligence*, 6(1):201–239.

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, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Linhan Li, Huaping Zhang, Chunjin Li, Haowen You, and Wenyao Cui. 2023. Evaluation on ChatGPT for Chinese Language Understanding. *Data Intelligence*, 5(4):885–903.

Jun-Yu Ma, Jia-Chen Gu, Zhen-Hua Ling, Quan Liu, and Cong Liu. 2023. Untying the reversal curse via bidirectional language model editing. *CoRR*, abs/2310.10322.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and Editing Factual Knowledge in GPT. *arXiv:2202.05262 [cs]*. ArXiv: 2202.05262 version: 1.

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

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

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

Sean O'Brien and Mike Lewis. 2023. Contrastive decoding improves reasoning in large language models. *CoRR*, abs/2309.09117.

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. ArXiv:2305.01651 [cs].

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2023. Fine-tuning or retrieval? comparing knowledge injection in llms. *CoRR*, abs/2312.05934.

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.

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

Charlie Snell, Dan Klein, and Ruiqi Zhong. 2022. Learning by distilling context. *CoRR*, abs/2209.15189.

Chenmien Tan, Ge Zhang, and Jie Fu. 2023. Massive editing for large language models via meta learning. *CoRR*, abs/2311.04661.

Asahi Ushio, Fernando Alva-Manchego, and Jose Camacho-Collados. 2023. A practical toolkit for multilingual question and answer generation. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Toronto, Canada. Association for Computational Linguistics.

Jiaan Wang, Yunlong Liang, Zengkui Sun, Yuxuan Cao, and Jiarong Xu. 2023a. Cross-lingual knowledge editing in large language models. *CoRR*, abs/2309.08952.

Weixuan Wang, Barry Haddow, and Alexandra Birch. 2023b. Retrieval-augmented multilingual knowledge editing. *CoRR*, abs/2312.13040.

Suhang Wu, Minlong Peng, Yue Chen, Jinsong Su, and Mingming Sun. 2023. Eva-kellm: A new benchmark for evaluating knowledge editing of llms. *CoRR*, abs/2308.09954.

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

Xunjian Yin, Jin Jiang, Liming Yang, and Xiaojun Wan. 2023. History matters: Temporal knowledge editing in large language model. *CoRR*, abs/2312.05497.

Shan Jie Yong, Kuicai Dong, and Aixin Sun. 2023. Docor: Document-level openie with coreference resolution. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, pages 1204–1207. ACM.

Ningyu Zhang, Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. 2024. Instructedit: Instruction-based knowledge editing for large language models.

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

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

A Details of MULFE

A.1 Levels and Tags

We have defined the generalization levels in Section 4.2. Here we highlight some details.

All questions for level 1 are clozes. If filling the blank in clozes with the correct answer, the resulting sentences either exactly appear in the original text, or have only capitalization or punctuation differences with the original text. Level 2 and level 3 can have both cloze-styled questions and normal wh-questions.

In terms of the fine-grained tags for level 3, it is difficult to design a comprehensive and complete taxonomy system for the reasoning types beforehand. Therefore, we propose a informal tagging guideline. Specifically, we firstly consider if the answer can be directly extracted from the original text. If so, we identify which types of elements or properties are asked in the questions, such as time, location, status, reason, etc. If not, we identity what kind of inference are required by the questions, such as counting, comparison, opinion inference, etc. Besides, we consider a list of featured issues in recent knowledge analysis research for language models, such as multi-hop problem, coreference, reverse curse and partial retrieval.

During the probe generation procedure, we also ask GPT-4 to generate tags for the probes. During the manual curation procedure, we edit the tags according to our guidelines. Therefore, the tag lists are gradually updating. Finally, we summarize all annotated tags and reorganize them, merging synonymous ones and removing ambiguous ones.

A.2 Collecting Edits and Generating Probes

We first collect wikipedia-styled short texts as edit requests. For evaluation data, we reuse the edit requests in *Entity Inference* (Onoe et al., 2023) and *DUNE* Akyürek et al. (2023), which mainly consist of entity and event descriptions from Wikipedia. We use these edit requests to create the evaluation data. Additioally, we collects the first two sentences of 3700 Wikipedia pages from before 2022 as the edit requests of the MULFE training dataset.

For each request, we utilize GPT-4 to generate $10 \sim 20$ efficacy probes of the three levels in JSON format. In the prompt template, we provide an instruction which describes the requirements and examples of each level and emphasizes that the questions should be unambiguous and answerable without the context. The template is shown in Example 1. We omit the examples of probes in the template, and $edit is replaced with the content of the edit request.

A.3 Manual Curation

After the collection of efficacy probes, we manually curate the evaluation data to ensure the quality. Specifically, we first remove undesired probes, which include probes that are not answerable without context (e.g. "What is his purpose?"), probes that irrelevant to the edit request, probes with too broad answer spaces, and probes that GPT-4 gives wrong answers. Secondly, we revise whether the automatically generated level is correct according to the proposed standard and manually edit the fine-grained tags for level-3 probes.

Two members of our team participated in the data curation process. First, in a pilot annotation phase, they annotated a small set of data (20 edits and corresponding probes), identified possible issues about level labeling and provided examples for the guidelines. One team member completed the round-one annotation (removing undesired questions and modifying the levels and tags). About 25% of the probes are discarded in this phase, and about 40% of the remaining are re-classified in different levels. After that, the other team member reviewed the results and approved the level classification of 91% probes. The levels of the remaining probes were decided according to the discussion of the two members.

A.4 Dynamic Specificity Probes

In previous work, a fixed set of specificity probes are used. However, if the base model have little of the corresponding knowledge, the numerical result of specificity could be low and insensitive to the editing process. Ideally, the specificity probes should be related to the knowledge previously encoded in the model. Therefore, in MULFE, the specificity probes are dynamically constructed for each base model, ensuring that it has already mastered the knowledge and yields 100% EM accuracy.

Specifically, we evaluate the base model on the "wikipedia nocontext" subset of TriviaQA (Joshi et al., 2017) and collect the QA instances that the model correctly answered. As the dataset is based on the facts that appeared before 2017, most of mainstream LLMs have seen relevant corpora. To ensure the robustness, for each instance we evaluate the model with three different zero-shot prompts. An instance is chosen as specificity probe only if it is correctly answered with all the prompts. For each model in experiments, we sample 400 specificity probes in this way.

A.5 Data Example

We show a data example in Example 2. For the same edit, we show the edit simplification results generated by Stanford-OpenIE, DOCoR, E2EQG and NERQG respectively in Example 3-6.

B Details of Experiments

B.1 Base Models

To investigate the impact of base language models, we test GPT2-XL (1.5B), GPT-Neo (2.7B), GPT-J (6B), and LLaMA2 (7B) in the experiments. We use the checkpoints from huggingface hub[1]. The names are *gpt2-xl*, *EleutherAI/gpt-j-6B*, *EleutherAI/gpt-neo-2.7B*, and *meta-llama/Llama-2-7b-hf*.

B.2 Implementations of Methods

For normal fine-tuning and SIDE, we implement the methods through Pytorch with AdamW as the optimizer, and use gradient accumulation trick to enable large batch editing size. For LoRA, we refer to the implementation in PEFT[2]. Other editing methods are implemented based on the official

[1] https://huggingface.co/models
[2] https://github.com/huggingface/peft

```
[System]
Based on the given text, create a list of clozes (using the underline "___" as the
    mask) or questions of different difficulty levels. Level 1 should be based on
    some exact fragments of the source text. Level 2 include simple synonymous
    variants or paraphrases of the original text. Level 3 requires some reasoning or
     summarizing processes based on the original text. Note that the clozes or
    questions should be unambiguous and answerable without the context. You should
    also provide the correct answer as well as specific tags to indicate the
    question type. The answer MUST be short phrases rather than a full sentence.
    Your response should follow this JSON format.
‘‘‘
{"probes":[
    {
        "query": "...", # A cloze or question
        "answer": "...", # The correct answer
        "level":"1", # Difficulty level: 1, 2, 3
        "tag":["..."]
    },
    {
        ... # More instances
    }
]}
‘‘‘

[User]
Create 6 clozes and questions based on the text:

January 2, 2022 - Abdalla Hamdok resigns as Prime Minister of Sudan amid deadly
    protests.

[Assistant]
{"probes":[
    ... # Six Probes
]}

[User]
Create several clozes or questions based on the text:

$edit
```

Example 1: The prompt template for generating probes.

```
...,
{
  "edit": "Warrior Nun is an American fantasy drama streaming television series
     created by Simon Barry based on the comic book character Warrior Nun Areala
     by Ben Dunn.",
  "id": "mulfe_test_ei_42",
  "probes": [
    {
      "query": "___ Nun is an American fantasy drama streaming television series
          .",
      "answer": "Warrior",
      "level": "1",
      "tags": [],
      "id": "mulfe_test_ei_42_0"
    },
    {
      "query": "Warrior Nun is an American ___ drama streaming television series
          .",
      "answer": "fantasy",
      "level": "1",
      "tags": [],
      "id": "mulfe_test_ei_42_1"
    },
    {
      "query": "Warrior Nun is based on the comic book character Warrior Nun ___
          by Ben Dunn.",
      "answer": "Areala",
      "level": "2",
      "tags": [],
      "id": "mulfe_test_ei_42_2"
    },
    {
      "query": "Warrior Nun is created by ___ based on the comic book character
          Warrior Nun Areala.",
      "answer": "Simon Barry",
      "level": "2",
      "tags": [],
      "id": "mulfe_test_ei_42_4"
    },
    {
      "query": "Name the series that combines elements of fantasy and drama,
          related to a nun with combat abilities.",
      "answer": "Warrior Nun",
      "level": "3",
      "tags": [
        "Property Reverse"
      ],
      "id": "mulfe_test_ei_42_8"
    },
    {
      "query": "What genre does the streaming television series Warrior Nun belong
           to?",
      "answer": "Fantasy drama",
      "level": "3",
      "tags": [
        "Type Extraction"
      ],
      "id": "mulfe_test_ei_42_9"
    }
  ]
},
...
```


Example 2: A data example of MULFE.

```
...,
{
  "edit": "Warrior Nun is an American fantasy drama streaming television series
      created by Simon Barry based on the comic book character Warrior Nun Areala
      by Ben Dunn.",
  "id": "mulfe_test_ei_42",
  "simplification": [
    {
      "prompt": "{} is",
      "input": "Warrior Nun is",
      "target": " American fantasy drama",
      "subject": "Warrior Nun"
    },
    {
      "prompt": "{} is",
      "input": "Warrior Nun is",
      "target": " fantasy drama",
      "subject": "Warrior Nun"
    },
    {
      "prompt": "{} is",
      "input": "Nun is",
      "target": " American",
      "subject": "Nun"
    }
  ]
},
...
```


Example 3: Edit Simplification with of Stanford-OpenIE.


```
...,
{
  "edit": "Warrior Nun is an American fantasy drama streaming television series
      created by Simon Barry based on the comic book character Warrior Nun Areala
      by Ben Dunn.",
  "id": "mulfe_test_ei_42",
  "simplification": [
    {
      "prompt": "{} is",
      "input": "Warrior Nun is",
      "target": " an American fantasy drama streaming television series",
      "subject": "Warrior Nun"
    },
    {
      "prompt": "{} created by",
      "input": "television series created by",
      "target": " Simon Barry",
      "subject": "television series"
    },
    {
      "prompt": "{} based on the comic book character by",
      "input": "Simon Barry based on the comic book character by",
      "target": " Ben Dunn",
      "subject": "Simon Barry"
    }
  ]
},
...
```


Example 4: Simplification Examples of DOCoR.

```
...,
{
  "edit": "Warrior Nun is an American fantasy drama streaming television series
      created by Simon Barry based on the comic book character Warrior Nun Areala
      by Ben Dunn.",
  "id": "mulfe_test_ei_42",
  "simplification": [
    {
      "input": "Question: Who created Warrior Nun?\nAnswer:",
      "target": " Simon Barry",
      "prompt": "Question: Who created {}?\nAnswer:",
      "subject": "Warrior Nun"
    },
    {
      "input": "Question: What comic book character was Warrior Nun Areala based
          on?\nAnswer:",
      "target": " Ben Dunn",
      "prompt": "Question: What comic book character was {} Areala based on?\
          nAnswer:",
      "subject": "Warrior Nun"
    },
    {
      "input": "Question: What is Warrior Nun?\nAnswer:",
      "target": " American fantasy drama streaming television series",
      "prompt": "Question: What is {}?\nAnswer:",
      "subject": "Warrior Nun"
    }
  ]
},
...
```

Example 5: Simplification Examples of E2EQG.

```
...,
{
  "edit": "Warrior Nun is an American fantasy drama streaming television series
      created by Simon Barry based on the comic book character Warrior Nun Areala
      by Ben Dunn.",
  "id": "mulfe_test_ei_42",
  "simplification": "simplification": [
    {
      "input": "Question: What nationality is Warrior Nun?\nAnswer:",
      "target": " American",
      "prompt": "Question: What nationality is {}?\nAnswer:",
      "subject": "Warrior Nun"
    },
    {
      "input": "Question: Who created Warrior Nun?\nAnswer:",
      "target": " Simon Barry",
      "prompt": "Question: Who created {}?\nAnswer:",
      "subject": "Warrior Nun"
    },
    {
      "input": "Question: What is the name of the comic book character in Warrior
          Nun?\nAnswer:",
      "target": " Nun Areala",
      "prompt": "Question: What is the name of the comic book character in {}?\
          nAnswer:",
      "subject": "Warrior Nun"
    }
  ]
},
...
```

Example 6: Simplification Examples of NERQG.

codes of ROME, MEMIT[3] and MEND[4].

B.3 Hyper-Parameters and Environment Conditions

For the fine-tuning methods, the hyper-parameters are set by grid search. Specifically, we set learning rate in ($5e-4$, $1e-4$, $5e-5$, $1e-5$, $5e-6$, $1e-6$), learning steps in $[5, 25]$ with early stopping at $loss = 0.1$. For the coefficients of SIDE, we conduct a grid search on the training dataset and empirically set them as $\lambda = 0.6$, $\alpha = 0.8$, $\beta = 0.1$, $\gamma = 0.1$.

All experiments in this paper can be undertook on two Nvidia A100 80G GPU. Each evaluation run takes $0.5 \sim 1$ hours. We conduct about 500 evaluation runs in total.

B.4 Evaluation Template

The evaluation input templates are shown in 7. $question is replaced with the probe questions.

```
Directly answer the question.

Question: $question
Answer:
```

Example 7: Template for Evaluation.

We also find that for cloze-styled questions the GPT-2 models natively generate the complete sentences rather than the answer phrases. Therefore, for cloze-styled questions, we input the text in the questions before the answer span as the hint.

C Additional Results

C.1 Overall Results on Different Base Models

In Table 4,5 and 6, we show overall results (similar to Table 2) on models other than GPT-J. Generally, larger models perform better after editing. And SIDE is more competitive on larger models.

C.2 Ablation of the loss of SIDE

We show the ablation study results of the loss of SIDE in Table 7. The combination of multiple losses can produce better overall results than single loss.

[3]https://github.com/kmeng01/memit
[4]https://github.com/eric-mitchell/mend

	Edit	Level 1		Level 2		Level 3		Overall Efficacy		Specificity	
	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓
Before-Editing	27.74	2.29	96.18	1.76	80.64	3.46	57.71	2.56	70.81	100.00	1.91
Edit-In-Context	1.22	**59.63**	**2.07**	**41.76**	**2.65**	**27.02**	**4.29**	**39.03**	**3.20**	72.95	2.34
FT (Full Model)	**1.00**	**68.12**	**2.61**	**39.67**	**4.97**	17.61	8.84	**35.91**	**5.90**	91.73	1.76
MEMIT (w/ Sim.)	29.40	4.36	74.03	6.15	57.14	9.96	49.14	7.39	55.37	99.61	1.89
ROME (w/ Sim.)	43.05	5.28	89.05	13.19	61.38	12.05	95.85	11.22	79.98	98.26	1.88
MEND (w/ Sim.)	20.66	5.05	63.44	15.05	26.94	11.41	43.93	11.65	38.54	88.68	2.05
SIDE (Section 5.3)	**1.18**	16.51	9.17	16.81	8.34	**18.87**	**8.81**	17.61	8.68	91.93	1.72

Table 4: Overall comparison of different editing methods on GPT2-XL (1.5B) model.

	Edit	Level 1		Level 2		Level 3		Overall Efficacy		Specificity	
	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓
Before-Editing	20.66	5.05	48.00	4.07	47.30	4.92	41.19	4.61	44.43	100.00	1.98
Edit-In-Context	1.20	**76.61**	**1.58**	**60.22**	**2.06**	**31.73**	**3.88**	**51.50**	**2.66**	80.96	2.15
FT (Full Model)	**1.00**	**60.78**	**2.66**	**36.15**	**4.59**	13.94	7.29	**31.61**	5.25	91.22	1.91
MEMIT (w/ Sim.)	26.87	9.17	38.08	12.42	29.57	13.63	35.17	12.30	33.32	99.51	1.97
ROME (w/ Sim.)	39.36	9.40	60.15	11.54	50.47	13.63	65.87	12.00	58.70	98.48	1.98
SIDE (Section 5.3)	**1.06**	38.07	4.59	30.66	4.66	**25.37**	**6.00**	29.87	**5.23**	91.39	1.75

Table 5: Overall comparison of different editing methods on GPT-Neo (2.7B) model.

	Edit	Level 1		Level 2		Level 3		Overall Efficacy		Specificity	
	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓	EM↑	PPL↓
Before-Editing	5.99	30.73	4.02	24.84	3.98	20.86	5.23	24.30	4.52	100.00	1.32
Edit-In-Context	1.17	**90.60**	**1.26**	**80.11**	**1.32**	**56.50**	**1.83**	**72.30**	**1.52**	92.71	1.38
FT (Full Model)	**1.00**	**80.73**	1.60	64.51	1.70	36.06	2.44	55.78	1.99	95.02	1.34
MEMIT (w/ Sim.)	5.99	30.73	4.02	24.84	3.97	21.38	5.19	24.52	4.50	99.94	1.31
ROME (w/ Sim.)	5.98	30.05	4.06	25.38	4.00	21.70	5.15	24.74	4.50	94.30	1.36
SIDE (Section 5.3)	**1.08**	**80.73**	**1.41**	**71.43**	**1.44**	**47.69**	**1.99**	**63.35**	**1.67**	96.24	1.22

Table 6: Overall comparison of different editing methods on Llama2 (7B) model.

	Level 1	**Level 2**	**Level 3**	**Overall**	**Specificity**
L_{soft}	19.27	17.91	16.25	17.48	84.89
L_{hard}	25.00	24.51	26.94	25.61	89.91
$L_{soft} + L_{hard}$	30.28	31.21	27.99	29.70	86.85
$L_e + L_{soft}$	67.20	46.26	25.16	41.48	93.32
$L_e + L_{hard}$	69.72	53.85	**36.48**	49.65	91.61
$L_e + L_{soft} + L_{hard}$	**73.17**	**59.56**	35.95	**52.35**	90.51

Table 7: Ablation of the loss of SIDE (EM Scores).