Interpretability-based Tailored Knowledge Editing in Transformers

Yihuai Hong[*] **Aldo Lipani**[†]

University College London
{yihuai.hong,aldo.lipani}@ucl.ac.uk

Abstract

Language models recognized as a new form of knowledge bases, face challenges of outdated, erroneous, and privacy-sensitive information, necessitating knowledge editing to rectify errors without costly retraining. Existing methods, spanning model's parameters modification, external knowledge integration, and in-context learning, lack in-depth analysis from a model interpretability perspective. Our work explores the instability in in-context learning outcomes, providing insights into its reasons and distinctions from other methods. Leveraging findings on the critical role of feed-forward MLPs in decoder-only models, we propose a tailored knowledge editing method, TailoredKE, that considers the unique information flow of each sample. Model interpretability reveals diverse attribute recall across transformer layers, guiding edits to specific features at different depths and mitigating over-editing issues.

1 Introduction

Language models have been proven to be a new form of dynamic knowledge bases (Cao et al., 2021; Petroni et al., 2019). However, the addition and removal of knowledge within it are not as straightforward as in traditional knowledge graphs.

As time progresses and the real world undergoes changes, language models often contain outdated or erroneous knowledge (Lazaridou et al., 2021; Lewis et al., 2020; Shuster et al., 2021), as well as unintentionally learned user privacy information (Carlini et al., 2021, 2022) and biased information (Bolukbasi et al., 2016). These errors or redundant information need to be corrected or removed to prevent any adverse impact. In such scenarios, the most common approach is typically retraining from scratch or finetuning the language models, in which the costs are notably high. Additionally, in situations with limited data, this can easily lead to unstable training results and overfitting problems. To address this challenge, researchers have introduced the concept of knowledge editing, aiming to modify specific knowledge within the model while ensuring that the storage of unrelated knowledge remains unaffected.

Many recent works have made progress in modifying the knowledge storage or intervening the retrieval processes within models, whether through the modification of internal model parameters (Meng et al., 2022a,b), the incorporation of external knowledge materials (Mitchell et al., 2022; Huang et al., 2022; Dong et al., 2022), or In-Context Learning without parameter modification (Zheng et al., 2023; Zhong et al., 2023; Cohen et al., 2023). However, there is still a lack of in-depth analysis from the perspective of model internal interpretability to understand why these methods succeed in editing, especially for In-Context Learning methods that treat large models as black boxes and thus lack theoretical analysis, which also has the drawback of being overly sensitive to text prompts and thus have unstable editing outcomes. In our work, we explored the reasons behind the instability of editing outcomes in In-Context Learning and examined the differences compared to other methods from the model interpretability perspective in §3.1.1.

Some other work (Geva et al., 2021; Meng et al., 2022a) have investigated how such relational knowledge is stored in decoder-only language models and found that the feed-forward MLPs in the middle layers are crucial and serve as key–value memories of much information associated with the entities. Meng et al. (2022a,b) provided valuable methods for knowledge editing based on this discovery. However, they did not consider the distinct characteristics of each editing sample and the unique processes of information flow associated with them, thus easily leading to the problems

[*]Work done during an internship at University College London.

[†]Corresponding author.

of over-editing: modifying parameters excessively and consequently affecting irrelevant peripheral knowledge. In-depth analysis lies in §3.1.2.

To better address the unstable knowledge editing problems and the over-editing issues mentioned above, we design and propose a more tailored knowledge editing method, TailoredKE, which is based on the actual information flow of each individual sample to perform a more precise knowledge intervention, drawing inspiration from Geva et al. (2023) which further reveal the specific process of how knowledge is extracted and recalled from the model parameters.

Specifically, we traced the process of internal knowledge recall, capturing different attributes which are recalled across various layers of transformer-based language models and associated with a specific piece of knowledge when performing editing. For instance, considering the entity "*iPod*", the shallow layers of language models will only recall it as a device and related to music, while deeper layers will be able to recall its association with the "*Apple*" company and other products like "*iPhone*" and "*iPad*". Based on the analysis of these features, we dynamically select different editing layers for each sample, rather than choosing the fixed layers for all editing as done in previous work (Meng et al., 2022a,b; Li et al., 2023). Tailoring our edits based on the specific features recalled at different layers not only allows us to address the needs of the actual editing goal (e.g., if our goal is to modify only the "*iPod*"'s manufacturing company) but also ultimately effectively alleviates the issues of over-editing and the potential impact on irrelevant facts in these traditional parameter editing methods as detailed in §3.3.

What is more, in the actual training process, it is crucial for a piece of knowledge to manifest in diverse forms to enable the model to genuinely remember it, moving beyond mere template-based memorization. As this knowledge appears more frequently in diverse forms, the model's memory of it becomes more profound, establishing a more stable representation inside. Hence, drawing insights from this perspective about how to deepen the model's memory of knowledge, in TailoredKE we have introduced diverse structures and expressions during the knowledge editing process, described in §3.2. This ultimately leads to a significant enhancement in performance across various metrics, signifying the strengthening of the model's ability to retain new knowledge. The main structure of our method, TailoredKE, is illustrated in Figure 1. The source code of the software used to run the experiments of this paper is available at `https://github.com/yihuaihong/TailoredKE`.

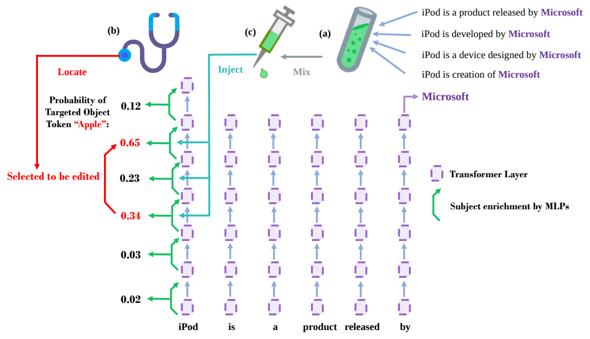
(b)
Locate
Selected to be edited
Probability of Targeted Object Token "Apple":
0.12
0.65
0.23
0.34
0.03
0.02
(c)
Inject
Mix
(a)
iPod is a product released by Microsoft
iPod is developed by Microsoft
iPod is a device designed by Microsoft
iPod is creation of Microsoft
Microsoft
Transformer Layer
Subject enrichment by MLPs
iPod
is
a
product
released
by

Figure 1: This outlines the main structure of our method, TailoredKE, which encompasses three steps for accomplishing a more effective and targeted knowledge editing process: (a) Strengthen the new memory by modifying its syntactic structure and calculating a shared weight: (b) Locate the key layers by observing the evolution of entity representation in actual scenarios: (c) Achieve new knowledge injection by updating the parameters of the MLPs which are selected in (b).

2 Related work

2.1 Knowledge Storing

Understanding how knowledge is stored within language models and how information flows, transfers, and integrates into them has consistently been a crucial part of studying language models.

Some recent research has indicated that the transformers' MLP can be conceptualized as key-value memories used to retain factual knowledge by the use of the causal tracing method (Pearl, 2022; Vig et al., 2020; Meng et al., 2022a), with the hidden states of the subject will serve as a key to map and recall its relevant knowledge. This conclusion was further investigated and explored by other works (Geva et al., 2023; Dar et al., 2023; Geva et al., 2022a), which artificially blocked certain components within transformers to examine their specific functions in the inference process. It reveals that the representation at the subject's last-token position undergoes the attributes' enrichment, recalling and integrating a wealth of pertinent knowledge from the MLP sublayers. In contrast, the sentence's last-token position will extract the relationship attribution from other positions and subsequently employ it via attention components to extract the most relevant attribution recalled by the subject token.

Finally, after inputting the obtained final representation into the classifier, the model generates the predicted next token for the sentence. These works have shed light on the extraction process of internal knowledge within language models from an interpretability standpoint. They serve as inspiration for our research on enhancing the editing of model knowledge.

In some other works, which from the model training perspective investigate the model's storage of knowledge and memories (Cohen et al., 2023; Zhu and Li, 2023), they experimentally confirmed that only when the same entity appears multiple times in training texts with various sentence structures, this entity has a stable representation that can be used to infer and recall the corresponding knowledge, which signifies that the model genuinely memorizes this knowledge, rather than merely memorizing the patterns (a certain word will always follow a specific combination of sentences).

2.2 Knowledge Editing

Language models often accumulate outdated or erroneous information over time. The most common approach to address such issues is to retrain the model with more accurate and timely data (Zhu et al., 2020). However, retraining requires substantial resources and time. Hence, researchers are actively seeking more efficient methods to adjust models to incorporate more accurate and current knowledge while discarding outdated or inaccurate information.

To achieve knowledge editing within the model effectively, now there are two main approaches based on whether they modify their parameters or not. If they modify their parameters, two strategies currently stand out: locating the specific parameters for subsequent modification (Meng et al., 2022a,b) which insert the weights containing new knowledge into certain layers of the transformers, or employing meta-learning (Mitchell et al., 2021; De Cao et al., 2021) which utilizes a hyper network to acquire the essential gradient for modifying the language model.

When choosing not to modify model parameters, the primary two methods are memory-based methods (Mitchell et al., 2022; Huang et al., 2022; Dong et al., 2022), and in-context learning (Cohen et al., 2023; Zheng et al., 2023; Zhong et al., 2023). In the in-context Knowledge Editing (IKE) (Zheng et al., 2023), they retrieved the top-k most relevant sentences concerning the new knowledge and insert these into the prompt, preceding the input to query the model. While Zhong et al. (2023) proposed MeLLo, which utilized the Chain of Thought strategy (Wei et al., 2022) to help design the prompt and perform in-context learning.

3 The Proposed Method

The current two most effective methods in the field of knowledge editing are **In-Context Learning**, such as IKE (Zheng et al., 2023) and MeLLo (Zhong et al., 2023), and the **Parameters Editing methods** like MEMIT (Meng et al., 2022b) and ROME (Meng et al., 2022a). However, each has its own noticeable and impactful shortcomings that cannot be overlooked.

The main problem with the parameters editing methods such as ROME or MEMIT is that they are too aggressive in modifying the parameters and will sometimes establish an excessively forced relationship between subject and object, potentially leading to the answer always being a new object regardless of the query's context (Zheng et al., 2023), which may also impact the storage of irrelevant knowledge that does not require modification. We validate and investigate this issue more in-depth in §3.1.2.

When it comes to in-context learning, its performance is largely limited by the length of the prompt window, due to the necessity of adding the text prompt containing relevant knowledge before the query. In most cases, a longer prompt will provide better editing capabilities (Zheng et al., 2023). Moreover, in-context learning does not genuinely enable the model to learn new knowledge or forget the old one at the parameter level. Hence, if the prompt is inaccurate or malicious (such as attempts to bypass the language model's security measures to output private or harmful content), it may generate incorrect or harmful results. Therefore, it cannot provide a truly robust and stable knowledge editing effect. In §3.1.1, we leverage a model interpretability technique, representation projection, to gain a deeper understanding of the underlying reasons behind this.

3.1 Existing Knowledge Editing Methods

3.1.1 Entity's Representations

The approach we use to better understand the inner workings of existing Knowledge Editing methods is to project the token representation in transformers onto the vocabulary space (Geva et al., 2023; Dar et al., 2022; Geva et al., 2022b; Ram et al., 2022). The representation of each token at every layer, as well as the projections to vocabulary, can be expressed by the following formulas:

$$X_i^{l+1} = X_i^l + M_i^l + A_i^l, \tag{1}$$

$$Prob_i^l = H(X_i^l), \tag{2}$$

where X_i^l represents the representation of the i-th token in l-th layer, while M_i^l and A_i^l are the outputs from the MLP and Attention components in the l-th layer respectively. The $H(\cdot)$ is the model's vanilla prediction head (also known as the unembedding matrix) which projects the internal representation onto the vocabulary space, and $Prob_i$ represents the corresponding probability distribution.

To investigate the issues behind the in-context learning method, we followed the in-context editing (ICE) baseline (Cohen et al., 2023) and utilized the prompt "*Imagine that Apple is a product released by Microsoft.*" ahead of the original query for this example. From the results in Table 1, we can observe that after applying the in-context learning method, the model can successfully generate the new object "*Microsoft*" at the last token's position in deeper layers, although the subject's attribution enrichment process is different and affected by the prepended prompts. However, we can also notice an unstable output distribution, which incorporates the outdated object "*Apple*". This reveals a substantial drawback, highlighting that it is not a thorough knowledge editing method. So in cases where the prompt is poorly designed, there is a risk of reintroducing old knowledge into the output.

When using the MEMIT method to edit parameters, the model's attributions change noticeably after the edited layer at the subject token's position. The attributions become more relevant to new objects and are notably closer. This indicates that MEMIT has a direct impact on the process of enriching the subject's representation, causing changes in various attributes and related information that are recalled during this procedure. Although the approach is comprehensive, it has the drawback of potential over-editing, which we will highlight in the next section.

3.1.2 Over-Editing Issues

While some knowledge editing methods involving parameter modifications have proven highly effective in altering the corresponding knowledge (Meng et al., 2022a,b), some work has revealed that this type of knowledge editing approach has the side effect of over-editing (Zheng et al., 2023), referring to the impact on other out-of-scope facts during the editing of a target fact. In our work, we observed that this issue becomes much more serious if these out-of-scope facts are closely related to the original facts, especially when the relevance exists among the subjects.

To delve deeper into this problem, we constructed a more elaborate dataset comprising 200 sets of editing pairs that were manually crafted. Each set includes 10 distinct but similar subject entities, sharing the same relation and featuring a common new object for editing. When editing such an existing fact (subject s, relation r, object o) to a new fact (subject s, relation r, new object o_{new}), we tested on other subject entities which have very similar representations to the original ones in the language model and computed this metric $\Delta P(o_{\text{new}}|s_{\text{others}}, r)$. It calculates the extent to which the probability of the new object varies in the language model's output distribution when inputting other unrelated yet similar subjects and the same relationship mapping statements. When the degree of this probability change is greater, it suggests a larger impact of over-editing side effects on these unrelated subjects.

For instance, by editing the new fact "*iPod is a product released by Microsoft*", we examine the impact on other different but related subject entities like "*iPhone*", "*iPad*", "*Macbook*" and so on, which are all products created by "*Apple*" company and have a similar representation in language models. We do this by measuring the extent of probability change of the token "*Microsoft*" in the output's distribution. The experimental results on this dataset indicate that on average, 47% of similar entities will be affected by over-editing when editing one of them in a pair. This effect is more serious when using parameter editing methods compared to In-Context Learning. Detailed results are presented in Table 2, showing the degree to which the probabilities of other objects are elevated, which are similar but not in need of modification.

In order to address the problems mentioned in §3.1, and achieve a more thorough and complete knowledge modification within the model while mitigating over-editing issues, we propose a better knowledge editing method in the remaining subsections.

Method	Layer	Top-scoring tokens
GPT-J	13(**Subject Token**)	iPhone, Music, Touch, devices, touch, music, Software, Archives, device, battery, Touch
	22(**Subject Token**)	music, Music, touch, Touch, song, device, songs, Music, software, devices, video, iPhone, iPod
	22(**Last Token**)	Apple, iPod, Apple, apple, iTunes, iPhone, apple, Sony, Microsoft, Macintosh, Nintendo, Mac
	27(**Last Token**)	the, **Apple**, a, American, in, and, apple, an, one, T, Mac, Steve, San, App, company
ICL	13(**Subject Token**)	is, has, was, 's, Q, os, =, ose, ure, /, ty, isn, will, ul, does, :, ://, TM, ort, are, name, au, isc
	22(**Subject Token**)	means, works, music, uses, will, does, plays, comes, stands, sales, needs, belongs, software
	22(**Last Token**)	**Microsoft**, **Apple**, Microsoft, Windows, Google, Sony, Samsung, apple, MS, IBM, iPod
	27(**Last Token**)	\n, the, a, ? ,, ", ., ..., :, **Microsoft**, (, –, and, **Apple**, R, M, ′, ?, micro, A, E, P, in, an, m, is, T
MEMIT	13(**Subject Token**)	Touch, touch, Wireless, Archives, Touch, devices, Square, Software, software, device
	22(**Subject Token**)	software, music, computer, Windows, Music, touch, devices, users, Touch, Software, device
	22(**Last Token**)	**Microsoft**, Microsoft, Windows, microsoft, Intel, Apple, MS, Nokia, Redmond, IBM, Xbox
	27(**Last Token**)	**Microsoft**, the, a, Bill, N, ;, MS, micro, E, S, and, one, American, US, Micro, C, computer

Table 1: Top-scoring tokens by token's representations of GPT-J (Layer = 13 and 22) for the entity of "*iPod*" before and after the Knowledge Edit (New Fact: *iPod is a product released by **Microsoft***; Old Fact: *iPod is a product released by **Apple***). Note: The presence of several repeated words is due to some instances in the model's tokenization table, such as '*Microsoft*' and '*ĠMicrosoft*', which may decode into the same word, '*Microsoft*'.

Method	$\Delta P(o_{\text{new}} \mid s_{\text{others}}, r)$
ROME	10.3%
MEMIT	9.9%
ICL	**3.7%**
Ours	3.9%

Table 2: Evaluation shows that editing methods will over-edit $(s_{\text{others}}, r, o_{\text{new}})$ when editing $(s, r, o) \rightarrow (s, r, o_{\text{new}})$. $\Delta P(o_{\text{new}} \mid s_{\text{others}}, r)$ shows the degree to which the probabilities of other similar objects are increased.

Prompt	New fact: **The Space Needle is located in Palace.** Please rephrase this new fact, fill in the blank spaces within this template into 10 types: The Space Needle {} Palace.
Answer	**1.** is situated in **2.** stands in **3.** is placed in **4.** positioned in **5.** towers over **6.** rises above **7.** looms over **......**

Table 3: Description of the prompt and corresponding answers.

3.2 Expressing Knowledge in Diverse Forms

Zhu and Li (2023) discuss how models store knowledge. When an entity appears in different sentence structures within the training set, it helps the model to memorize that entity. This enables the model to use it for reasoning and providing responses to related questions rather than just repeating the same answer when it encounters similar queries. Utilizing the language model itself, we rephrase knowledge statements while maintaining subject, relationship, and object to assist the model in retaining new knowledge. This process serves as a form of data augmentation involving altering the sentence structure and expressions. An example is shown in Table 3.

After receiving these rephrased sentences for the same new factual knowledge, we encode them simultaneously into the model by calculating a shared weight and updating the ones in the MLP modules of certain layers. The MLP modules consisting of two layers each are treated as key–value memories (Kohonen, 1972; Geva et al., 2021). The output representation after the MLP can be expressed in a formula like this:

$$M_i^l = W_{out}^l \sigma(W_{in}^l \gamma(A_i^l + H_i^{l-1})), \tag{3}$$

where the matrices W_{out}^l and W_{in}^l represent the two-layer neural network of MLP in each layer, while σ and γ represent the non-linear activation function and the layernorm function.

Specifically, the W_{in} matrices in the first layer encode entities into their corresponding keys, while the W_{out} matrices in the second layer map these keys to attributes and associated information related to the entities. Therefore, to modify the mapping relationship between entities and target attributes or to add new attributes, we only need to adjust the W_{out} matrices to encode new target knowledge for the entities.

We obtain this target weight by optimizing the objective function proposed by MEMIT (Meng et al., 2022b) and searching for values that can better represent the target attributes and new knowledge:

$$W_{\text{target}} = \underset{W}{\arg\min} \left(\sum_{i=1}^{n} \|Wk_i - v_i\|^2 + \sum_{i=n+1}^{n+m*t} \|Wk_i - v_i\|^2 \right) \quad (4)$$

where n represents the count of original knowledge elements to be preserved, whereas m denotes the amount of new knowledge to be added, and t signifies the number of times each new knowledge is rephrased and undergoes structural modifications. The k_i and v_i represent the corresponding encoded entity and its mapping attribute separately. Through this approach, we have identified representations that better capture the essence of new knowledge and its corresponding attributes. Hence, we can compute better weights for mapping to these corresponding representations. The intuitive results after employing this method are presented and discussed in §4.3.

3.3 Preciser Selection of the Layers to Edit

In previous methods like ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b), their knowledge editing layers remained fixed for all samples, regardless of the feature of each individual sample. For instance, on the backbone of GPT2-XL, in ROME, the editing layers were consistently set at the 17th layer for all samples, while in MEMIT, the editing layers were fixed at layers 13, 14, 15, 16, and 17, which clearly appears to be arbitrary. Gupta et al. (2023) observed that modifying the shallow layers of the model tends to achieve better results for commonsense knowledge which is relatively simpler compared with most of the factual knowledge. Therefore, to attain more effective knowledge editing outcomes for varying difficulty levels of knowledge, it is necessary to selectively edit different transformer layers based on the characteristics of each knowledge itself.

Given that the parameter modification method fundamentally involves adjusting the subject representation enrichment within the model's internal MLPs, and considering that the MLPs recall different attributions at each layer, it becomes essential to research the specific layer where a particular attribute of an entity begins to be recalled. This allows for targeted editing at specific layers, aiming to modify a specific entity's attribute while minimizing the impact on other attributes.

Since subject representation enrichment is a continuous process across different layers, we designed a more precise layers selection strategy named **Dynamic Editing Window** approach. This method dynamically selects the layers for editing based on the unique characteristics of each knowledge and the actual entities' representations enrichment process. To be specific, before editing certain knowledge, we observe its attribution enrichment process, which is described in §3.1.1. After receiving the representation obtained from each layer through their corresponding MLPs, we feed them into a classifier, projecting them to the vocabulary space and yielding probability values for the original object's token, as described in Eq. 2 and Eq. 3, which is not time-consuming in practical applications. We then identify the two layers that have the highest probabilities and consider all the layers between them as the main part of the enrichment process for the subject to recall important information related to the original object. These layers are the ones that require editing. The formula representing how to choose these two is outlined below:

$$i, j = \underset{i,j}{\arg\max}\{p_i, p_j \mid p_i \neq p_j, p_i, p_j \in \mathit{Probs}\}, \quad (5)$$

$$\mathit{Probs} = \{H(M_i^1), H(M_i^2), \ldots, H(M_i^n)\}, \quad (6)$$

in which the M_i^l are the outputs from the MLP component in the l-th layer of the subject's token. The $H(\cdot)$ represents the model's vanilla prediction head, which projects the internal representation onto the vocabulary space. After the calculation, all the layers between these two selected layers i and j constitute the defined scope for modification. Note that although Dynamic Editing Window can flexibly select the layers to be edited based on the different characteristics of each sample, this strategy supports both single fact editing and batch facts editing.

The initial result about preventing the over-editing problems is in Table 2, which demonstrates that we can significantly alleviate the issue of over-editing with the original parameter editing method by choosing more appropriate layers to edit, reducing it to a level comparable to In-Context Learning.

4 Experiment

4.1 Experimental Setup

We employ two recent auto-regressive decoder-only language models of varying sizes: GPT-J (Chen et al., 2021) with 6 billion parameters, and LLaMA-2 (Touvron et al., 2023) with 7 billion parameters. We completed all experiments on a single 80GB NVIDIA A800 GPU.

We have chosen several leading methods in the Knowledge Editing domain as baselines, including top-performing approaches that involve locating and modifying model parameters, such as ROME (Meng et al., 2022a), MEMIT (Meng et al., 2022b) and PMET (Li et al., 2023). Additionally, we considered other methods that incorporate external knowledge supplementation, such as MEND (Mitchell et al., 2021) and SERAC (Mitchell et al., 2022) for performance comparison with ours.

Furthermore, we tested these three crucial metrics of Knowledge Editing: **Efficacy**, **Generalization** and **Specificity** on two popular model editing datasets COUNTERFACT (Meng et al., 2022a) and ZsRE (Levy et al., 2017), along with an additional dataset proposed by Yao et al. (2023) that assesses the metric of **Portability**. **Efficacy** is the most direct indicator of the success of knowledge editing, measuring the proportion of cases where the model's predictions match the new ground truth o_i^*. **Generalization** is the same metric but applied on different queries, each expressing the same question but with different expressions. **Specificity** measures the impact of knowledge editing on irrelevant knowledge. Higher scores indicate less influence, aligning more closely with the original model. **Portability** is used to assess the effectiveness of model editing in transferring knowledge to related content, evaluated on one-hop and multi-hop problems. More explicit definitions of these metrics are presented in Appendix A.

4.2 Results

The main result on the COUNTERFACT Dataset with one editing operation per iteration is presented in Table 4. This demonstrates that our approach, TailoredKE, surpasses existing methods across multiple knowledge editing metrics on both two backbones, GPT-J and LLaMA-2-7b. The combination of two strategies, diverse knowledge forms and preciser selection of the layers to edit, results in significant improvements across three key metrics. This suggests that it is not just a more robust and enhanced knowledge editing method, enhancing the acquisition of new knowledge and the eradication of outdated knowledge (Efficacy, Generalization), but also better at minimizing the impact of other irrelevant information (Specificity).

TailoredKE also maintains the performance of the original model on fluency and consistency compared with other baselines, enabling it to generate grammatically sound and fluent sentences. The specific roles of these two strategies on different metrics are thoroughly discussed in the Ablation Study §4.5. In this table, we also observe that the four approaches of parameter editing can yield more comprehensive editing compared to other methods involving external memory access including MEND and SERAC, especially in terms of Efficacy and Generalization.

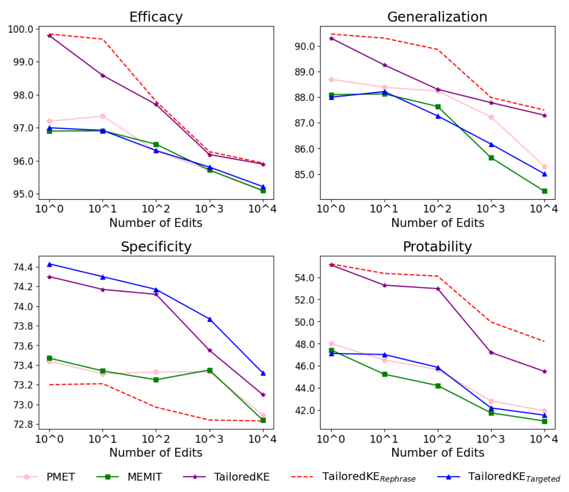

Figure 2: This chart illustrates the comparison between our method and the current two most efficient baselines under varying knowledge editing counts: 1, 10, 10^2, 10^3, and 10^4. Additionally, we have incorporated ablation experiments, where TailoredKE$_{Rephrase}$ represents the sole impact of knowledge rephrasing, and TailoredKE$_{Targeted}$ signifies the included influence of Dynamic Window Selection without the effects of knowledge rephrasing.

4.3 The Effect of Diverse Knowledge Forms

We evaluate the enhancements provided by the knowledge representations in diverse forms for the conventional parameters-editing approaches like MEMIT (Meng et al., 2022b). Additionally, we adjusted the number of knowledge forms and observed how this variation impacted the knowledge editing effectiveness.

The result is shown in Table 5. After employing rephrasing sentences with diverse structures of knowledge, we observe notable improvements in the performance across the three key indicators of knowledge editing. Furthermore, as the frequency of rephrasing increases, there is a further enhancement in the effectiveness of knowledge editing. This suggests that altering the structure of knowledge and presenting the model with diverse forms

Editor	Efficacy	Generalization	Specificity	Fluency	Consistency
GPT-J	15.2	17.7	83.9	**622.4**	29.4
MEMIT	99.6	64.1	73.5	606.0	36.9
PMET	99.6	62.2	69.7	597.9	38.7
SERAC	93.2	61.1	71.3	619.1	**38.9**
MEND	93.2	53.2	**82.9**	618.4	31.1
ROME	91.2	43.1	49.1	614.2	37.1
TailoredKE	**99.8**	**73.5**	74.5	619.5	**38.9**
LLAMA-2	16.2	18.1	84.5	**631.2**	31.1
PMET	93.0	87.9	72.1	597.3	39.0
MEMIT	92.9	85.9	76.3	619.1	38.9
ROME	92.5	87.0	51.0	614.2	37.4
TailoredKE	**96.1**	**91.0**	**79.1**	619.5	**40.1**

Table 4: The performance of TailoredKE on the COUNTERFACT Dataset with one editing operation per iteration. The definitions of the metric **Efficacy**, **Generalization**, and **Specificity** are provided in §4.1. **Fluency** and **Consistency** are proposed by ROME (Meng et al., 2022a) to evaluate the fluency and semantic consistency of generated sentences, with calculation details provided in §A.

Method	Efficacy	Generalization	Specificity
ROME	97.6	88.1	24.2
MEMIT	96.7	88.7	**26.4**
MEMIT$_{Rephrase*5}$	97.5	89.2	26.2
MEMIT$_{Rephrase*10}$	97.9	89.5	26.1
MEMIT$_{Rephrase*20}$	**99.8**	**90.3**	26.2

Table 5: Editing result on the ZsRE dataset after using rephrasing sentences on the backbone of GPT-J.

of the same information facilitates a more robust retention and understanding of the knowledge.

4.4 Evaluations Considering the Portability Problems

Portability is a new metric recently proposed by the knowledge editing community. It assesses whether modifications to specific knowledge will extend to related knowledge, testing whether the modifications are only superficial (Cohen et al., 2023; Yao et al., 2023; Zhong et al., 2023). We utilize the dataset introduced by Yao et al. (2023) to assess the Portability metric, examining whether the editing in our approach can effectively impact the pertinent knowledge that ought to be influenced. This dataset is built upon COUNTERFACT and ZsRE, incorporating sentence data generated by GPT-4 to measure surrounding entities. The relevant results are presented in Table 6. We can observe that TailoredKE outperforms other methods in this Portability metric, with its rephrasing functionality playing a predominant role, while the impact of targeted edits is minimal, primarily contributing to the reduction of over-editing.

4.5 Ablation Study

We perform two sets of ablation experiments compared with the two most effective parameter editing methods PMET (Li et al., 2023) and MEMIT (Meng et al., 2022b) on the COUNTERFACT dataset. The result is in Figure 2 and it illustrates that: In this scenario, the emphasis on knowledge rephrasing significantly enhances the metrics of Efficacy and Generalization, signifying improved retention of corresponding knowledge by the model. Meanwhile, the contribution of the targeted layer primarily enhances the Specificity metric, effectively mitigating the impact of over-editing associated with parameter editing methods. What is more, the employment of knowledge with diverse structures significantly boosts the model's performance in the Portability metric, which indicates the improvement of the model's ability for knowledge transfer and its reasoning capability on corresponding knowledge.

5 Conclusion

Our work draws from prior research on feed-forward MLPs in decoder-only models, providing a tailored approach that uses the information flow of each sample to achieve a higher editing performance and prevent over-editing. We also simulate the training process with varied structures during knowledge editing to improve performance and strengthen knowledge retention.

Editor	**Probability**	
GPT-J	ZsRE	COUNTERFACT
ROME	50.85	46.48
MEMIT	52.70	47.44
SERAC	5.51	9.51
MEND	0.10	0.00
TailoredKE	67.91	55.11
TailoredKE$_{Rephrase}$	**67.93**	**55.20**
TailoredKE$_{Targeted}$	52.74	47.12

Table 6: Results on the ZsRE and COUNTERFACT datasets considering the Portability problems. In this context, TailoredKE$_{Rephrase}$ represents the method with only the addition of the rephrase sentence operation, excluding the targeted layer operation, and vice versa.

6 Limitations

Through our experimental validation, we found that both parameter-modifying knowledge editing methods and non-parameter-modifying knowledge editing methods can lead to some degree of hallucinations. The model tends to overly extrapolate new knowledge, involving content that still requires further refinement and improvement.

Concurrently, when the entity to be edited is unfamiliar to the model (Cohen et al., 2023), due to the lack of a stable representation, it will not exhibit satisfactory performance. In our future work, we will attempt to rapidly create a stable representation for entities unfamiliar to these models by establishing connections with known entities that share semantic similarities, aiming to assist the model in quickly memorizing the new entity.

Currently, in most knowledge editing papers, researchers often evaluate the performance of knowledge editing techniques using idealized datasets like COUNTERFACT (Meng et al., 2022a) and ZsRE (Levy et al., 2017). Moreover, the sentences in these datasets that represent facts often follow a simple format of (subject, relation, object), which cannot encompass all real-world knowledge scenarios, which remains for future exploration.

References

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29.

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021. Knowledgeable or educated guess? revisiting language models as knowledge bases. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1860–1874, Online. Association for Computational Linguistics.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2022. Analyzing transformers in embedding space. *arXiv preprint arXiv:2209.02535*.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2023. Analyzing transformers in embedding space. In *Annual Meeting of the 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.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. In *Findings of the Association for Computational Linguistics:*

EMNLP 2022, pages 5937–5947, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Mor Geva, Avi Caciularu, Guy Dar, Paul Roit, Shoval Sadde, Micah Shlain, Bar Tamir, and Yoav Goldberg. 2022a. Lm-debugger: An interactive tool for inspection and intervention in transformer-based language models. In *Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 12–21.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022b. 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.

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. *arXiv preprint arXiv:2305.14956*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2022. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*.

Teuvo Kohonen. 1972. Correlation matrix memories. *IEEE transactions on computers*, 100(4):353–359.

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. 2021. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363.

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.

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

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

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. In *International Conference on Learning Representations*.

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

Judea Pearl. 2022. Direct and indirect effects. In *Probabilistic and causal inference: the works of Judea Pearl*, pages 373–392.

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.

Ori Ram, Liat Bezalel, Adi Zicher, Yonatan Belinkov, Jonathan Berant, and Amir Globerson. 2022. What are you token about? dense retrieval as distributions over the vocabulary. *arXiv preprint arXiv:2212.10380*.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

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. *arXiv preprint arXiv:2305.13172*.

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. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

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

Zeyuan Allen Zhu and Yuanzhi Li. 2023. Physics of language models: Part 3.1, knowledge storage and extraction. *CoRR*, abs/2309.14316.

A Metric Explainations

We tested these three crucial metrics of Knowledge Editing: **Efficacy**, **Generalization** and **Specificity** on two popular model editing datasets COUNTERFACT (Meng et al., 2022a) and ZsRE (Levy et al., 2017), along with an additional dataset proposed by the work (Yao et al., 2023) that assesses the metric of **Portability**. The definitions of these four metrics are provided below:

- **Efficacy** Efficacy is the most direct indicator of the success of knowledge editing. The prompts encountered during model evaluation after editing are the same as those encountered during the editing process. In the formulas below, s_i, r_i, and o_i respectively represent tokens of the subject, relation, and object, while o_i^* denotes the new object to be edited. f_{θ^*} represents the post-edit model.

$$E_i\left[o_i^* = \arg\max_{o_i} f_{\theta^*}\left(o_i \mid p(s_i, r_i)\right)\right], \quad (7)$$

- **Generalization** The effect of knowledge editing should not be limited to the original queries; when presented with different expressions but asking the same question, the post-edit model f_{θ^*} should also output the newly edited answer o_i^*:

$$E_i\left[E_{p\in\text{neighbour}(s_i,r_i)}\left[o_i^* = \arg\max_{o_i} f_{\theta^*}\left(o_i \mid p\right)\right]\right], \quad (8)$$

- **Specificity** After knowledge editing, other irrelevant knowledge contained within the new model should not be affected. Therefore, the Specificity metric is computed as the extent to which the predictions of the post-edit model f_{θ^*} remain unchanged compared to those of the pre-edit model f_θ:

$$E_i\left[E_{p\in\text{irrelevant}(s_i,r_i)}\left[\arg\max_{o_i'} f_\theta\left(o_i' \mid p\right) = \arg\max_{o_i} f_{\theta^*}\left(o_i \mid p\right)\right]\right], \quad (9)$$

- **Portability** Portability is a metric used to assess the effectiveness of model editing in transferring knowledge to related content. The post-edit model should be capable of solving the one-hop or even multi-hop problems, which require additional reasoning to be resolved.

$$E_i\left[E_{p\in\text{portability}(s_i,r_i)}\left[o_i^* = \arg\max_{o_i} f_{\theta^*}\left(o_i \mid p\right)\right]\right], \quad (10)$$

In Table 1, we follow Meng et al. (2022a) to also provide the metric of **Fluency** and **Consistency** for the main results. Fluency is computed by measuring the weighted average of bi-gram and tri-gram entropies (Zhang et al., 2018) while Consistency is computed as the cosine similarity between the unigram TF-IDF vectors of the new generated texts which start with the target subjects, and the reference texts used in editing which share the same new objects.

B Implementation Time Cost

For TailoredKE, the primary time expenditure is mainly on the calculation of the new target weights. In this aspect, we are essentially similar to the original method, ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b), both utilizing the backpropagation approach to help calculate it. Therefore, the time cost is fundamentally similar. For 10,000 edits, ROME and MEMIT take about 11.10 hr and 6.56 hr on one 80GB A100, and TailoredKE takes about 7.63 hr at the same setting.