Forget for Get: A Lightweight Two-phase Gradient Method for Knowledge Editing in Large Language Models

Anonymous ACL submission

Abstract

Recent studies have highlighted the remarkable knowledge retention capabilities of Large Language Models (LLMs) like GPT-4, while simultaneously revealing critical limitations in maintaining knowledge currency and accuracy. Existing knowledge editing methodologies, designed to update specific factual information without compromising general model performance, often encounter two fundamental challenges: parameter conflict during knowledge overwriting and excessive computational overhead. In this paper, we introduce **ForGet (Forget for Get)**, a novel approach grounded in the principle of "forgetting before learning". By pinpointing the location within the LLM that corresponds to the target knowledge, we first erase the outdated knowledge and then insert the new knowledge at this precise spot. **ForGet** is the first work to leverage a two-phase gradient-based process for knowledge editing, offering a lightweight solution that also delivers superior results. Experimental findings show that our method achieves more effective knowledge editing at a lower cost compared to previous techniques across various base models.

1 Introduction

Large Language Models (LLMs) have revolutionized natural language processing, enabling unprecedented capabilities in language comprehension and generation (Brown et al., 2020; Raffel et al., 2020; Ouyang et al., 2022). A key factor behind these capabilities is the vast amount of knowledge embedded within these models. However, this knowledge is often static, leading to issues such as outdated information, inaccuracies, and potential privacy violations. For instance, answering 'Who is the President of the United States?' in 2024 yields 'Joe Biden,' but this response becomes incorrect in 2025 if the model is not updated. Knowledge Editing is proposed to address this problem. Knowledge Editing aims to modify the specific knowledge stored in LLM without affecting other irrelevant knowledge and maintaining a low computational cost (Yao et al., 2023).

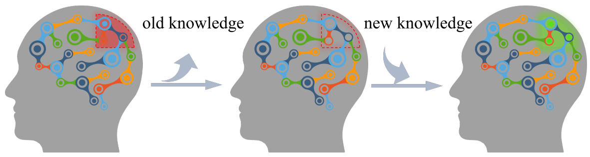

Figure 1: Clearing old knowledge before learning new knowledge can mitigate the impact of knowledge conflicts.

Existing knowledge editing methods can be broadly categorized into three classes (Li et al., 2024). Some of the methods utilize an additional knowledge base to store edits (Mitchell et al., 2022; Hartvigsen et al., 2024; Wang et al., 2024b), some methods use in-context learning (Zheng et al., 2023; Qi et al., 2024), others first decide the location to edit then perform editing at the specific location (Huang et al., 2023; Yu et al., 2024; Mitchell et al., 2021; Dai et al., 2021; Meng et al., 2022a,b). The existing methods have largely succeeded in editing the knowledge stored in LLMs.

These approaches attempt to edit from various perspectives; however, they all encounter significant limitations. One of the issues is that they merely address the old knowledge when inserting new knowledge. When editing knowledge in LLMs, conflicts between new and old knowledge may arise, which can hinder the model's ability to learn new information (Wang et al., 2024a) and weaken the effect of editing. Just like humans, it is difficult to change old knowledge when it has become deeply ingrained. Another is the precision of editing location. Some of the methods attempted to address knowledge conflicts but did not focus on specific location within LLMs. It is crucial to determine appropriate and precise editing locations. Otherwise, editing effects may be weakened and irrelevant knowledge may be changed accidentally.

In order to resolve knowledge conflicts, a straightforward approach is to forget the old knowledge before learning the new knowledge. For example, before going to Vienna, one should first remove the luggage packed for Bangkok from the suitcase and then pack the luggage prepared for Vienna. Inspired by the human cognitive mechanisms where forgetting old information is a prerequisite for learning new information, we propose a method named **ForGet (Forget for Get)**. First of all, critical MLP modules are found out by the knowledge circuits determined by target knowledge. Knowledge editing is then performed on these critical MLP modules while the rest of the model remain unchanged. During the editing process, we first apply gradient ascent to these modules to eliminate the old knowledge, which is defined as the forgetting process. Gradients ascent has been adopted for LLMs to mitigate privacy risks (Jang et al., 2023). After the forgetting process, we use gradients descent to insert new knowledge into the model. To the best of our knowledge, this is the first work to leverage gradient ascent and descent for knowledge editing, offering a lightweight and efficient solution to the problem of knowledge conflicts. The main contributions of this work can be summarized as follows:

- We propose **ForGet**, the first knowledge editing framework to leverage a two-phase gradient-based process—gradient ascent for forgetting outdated knowledge and gradient descent for acquiring new knowledge.
- We explore the potential of using knowledge circuits to determine editing location, which effectively depict the storage and flow of knowledge within Large Language Models.
- The experimental results demonstrate that our method is able to achieve both effective editing and preservation of unrelated knowledge, while being significantly more resource-efficient compared to existing methods.

2 Related Work

Currently, a series of methods have been proposed to address the problem of knowledge editing for LLMs. They can be roughly divided into three categories according to the process of humans correcting mistakes: methods with additional memories, methods learning from examples and methods modifying components directly(Li et al., 2024).

2.1 Additional Memories

Directly memorizing edits and preparing them for future use is a straightforward strategy. The edits may not be mastered but can be recalled in the future. Some of the existing editing methods take use of additional memories to store the knowledge to be edited in LLMs. SERAC (Mitchell et al., 2022) stores edits in a cache and uses an edit scope classifier to choose between the original model and a counterfactual model based on input and cached edits. However, the scope classifier and counterfactual model need to be trained in advance. Unlike SERAC, GRACE (Hartvigsen et al., 2024) stores edits in a codebook, searching and replacing erroneous knowledge with the most similar key in codebook when an error occurs. The codebook requires update from time to time, which adds complexity and increases workload. WISE (Wang et al., 2024b) employs a dual-memory design which contains a main memory containing old knowledge and a side memory containing edits. In conclusion, the methods of rote memorization perform well by using additional memories and scope classifiers. Consequently, this leads to ever-increasing storage requirements and model complexity.

2.2 Learning from Examples

Methods of learning from examples refers to methods utilizing In-context learning. Similar to humans, large language models (LLMs) can excel and outperform zero-shot inference across various tasks when provided with a few examples (Brown et al., 2020; Liu et al., 2022). Without changing any parameters, Zheng et al. (Zheng et al., 2023) propose editing the knowledge in the model by constructing three different demonstrations: copy, update, and retain. Such direct use of In-context learning can lead to overfitting to individual samples and requires meticulously crafted examples. Building upon this, Qi et al. (Qi et al., 2024)propose employing In-context learning aimed at a distribution rather than individual samples. Their method aims to guide the model to learn to generate a distribution consistent with the target knowledge by minimizing the difference in the model's output with and without given context. The above methods take use of In-context learning which which often requires a significant amount of human labor to design and construct demonstrations. Another issue is that these methods may also affect irrelevant knowledge in LLMs.

2.3 Modifying Components Directly

Other methods modify base model's components directly to achieve better editing results. These methods aiming to edit base model effectively and precisely can be categorized into two classes.

Adding Trainable Components while maintaining the original modules unchanged is one of the strategies to edit knowledge precisely. These methods incorporate new knowledge into the model by optimizing these new components. Huang et al. (Huang et al., 2023) rectify erroneous knowledge by adding neurons into the final layer, which are trained to encapsulate new knowledge. However, the number of new neurons increases as the number of new edits grows. To effectively encodes edit information, Yu et al. (Yu et al., 2024) propose MELO consisting dynamic LoRA (Valipour et al., 2023) and vector database. During each editing session, only the relevant parts of LoRA are activated. Furthermore, MEND (Mitchell et al., 2021) employs the strategy of meta-learning, integrating an entire hypernetwork within the model. These methods edit knowledge by adding new components into original model, which augment the model's complexity.

Updating Original Components can effectively avoid augmenting the model's complexity. Ni et al. (Ni et al., 2024) proposed "Forgetting before Learning" theory and fine tune the base model to sequentially 'forget' outdated knowledge and 'learn' new knowledge. To achieve precise edits, many researchers have focused on identifying optimal editing locations before performing editing. Geva et al. (Geva et al., 2021) find that specific knowledge are stored in the form of key-value pairs in feed forward layers in LLMs. Dai et al. (Dai et al., 2021) proposed the concept of knowledge neurons and try to edit knowledge by modifying knowledge neurons. Some works apply causal mediation analysis (Pearl, 2022) to find editing location. After finding one critical MLP module, Meng et al. (Meng et al., 2022a) employ rank-one update (Bau et al., 2020) to this module. Later, Meng et al. (Meng et al., 2022b) think one critical MLP module may be insufficient for knowledge editing, so they use multiple MLP modules to perform editing. Hu et al. (Hu et al., 2024) identify a pattern mismatch issue when locating edit positions and propose using specific edits to locate specific editing locations. However, the methods merely address the old knowledge, which may cause knowledge conflict with new knowledge.

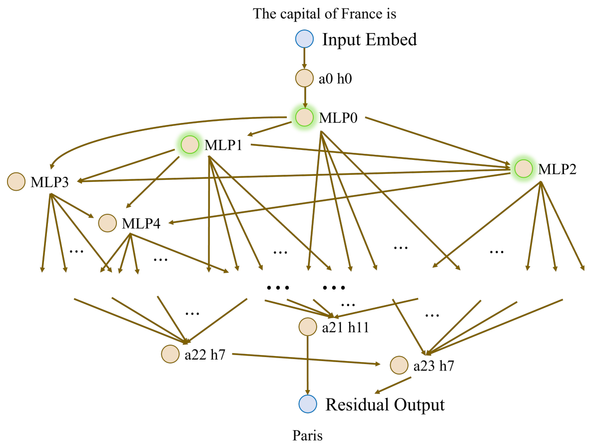

Figure 2: A simplified schematic diagram of the knowledge circuit for the knowledge "The capital of France is Paris."

In contrast, **ForGet** is a lightweight method without additional components, thereby ensuring the model's complexity remains unchanged and low human labor. What's more, **ForGet** not only identifies precise editing locations but also explicitly mitigates knowledge conflicts through a two-phase gradient-based process, making it a more robust and conflict-free editing process.

3 Task Definition

Our task is to edit knowledge within LLMs precisely. As equation 1 shows, given the target knowledge K and original model f with parameters θ, our goal is to design a method $F()$ that performs the necessary edits to produce an updated model f' with parameters θ'.

$$f' = F(K, f) \tag{1}$$

Editing knowledge precisely means that only the knowledge within editing scope will be edited and others should not be affected, as equation 2 shows. The editing scope refers to a set of inputs related to the target knowledge that requires editing (Yao et al., 2023). For example, the answer to "Who is the President of the United States" should be changed from "Biden" to "Trump", but the answer to "Who is the President of Russia" should remain "Putin" both before and after editing.

$$f'(x) = \begin{cases} y', & \text{if } x \in X^* \\ y, & \text{if } x \notin X^* \end{cases} \tag{2}$$

where X^* means editing scope which is the

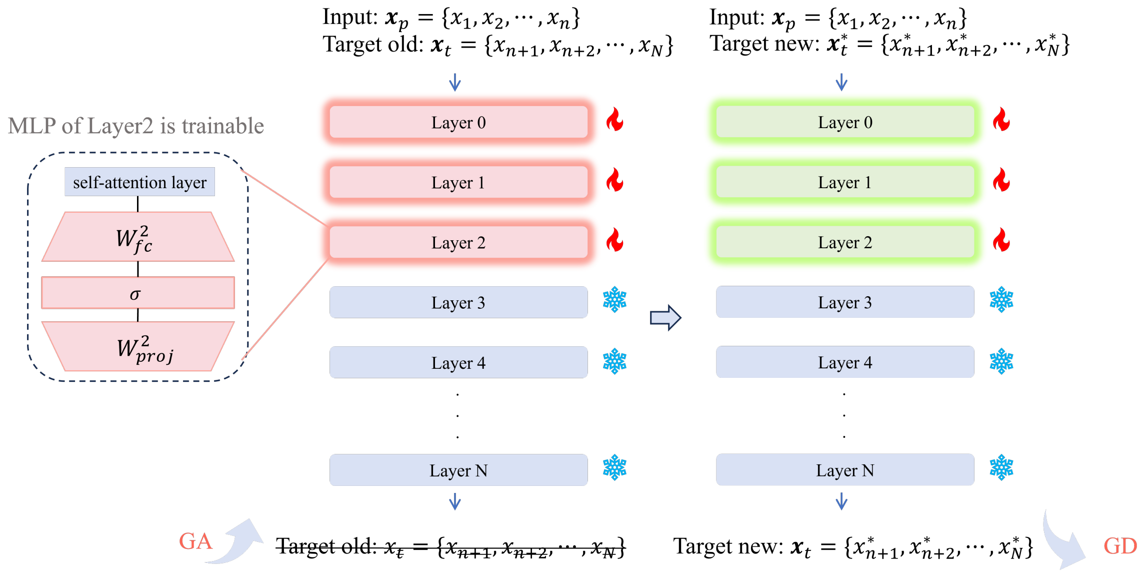

Figure 3: After determining the editing location, only the modules within editing location are trainable in later process. First, gradient ascent is performed to eliminate old knowledge, followed by the opposite gradient descent to acquire new knowledge.

range of knowledge that needs to be edited. And y' represents output context related to knowledge K while y is the original output.

4 Method: ForGet

In this section, we are going to introduce our method for knowledge editing: **ForGet (Forget for Get)**. Instead of making use of additional memories or designing clever demonstrations, we adopted a direct two-phase gradient adjustment, offering a lightweight yet effective solution.

The ForGet framework consists of two main phases: **(1) determining the editing locations** and **(2) performing the editing operations**. In the first phase, we identify the components of the model that are most relevant to the target knowledge requiring editing. Editing at this location enhances the effectiveness of the edits while mitigating the impact on irrelevant knowledge. The second phase occurs at the editing location identified in the first phase. We begin by using gradient ascent to forget the old knowledge, followed by gradient descent to acquire the new knowledge.

4.1 Determine Editing Location

4.1.1 Knowledge Circuits

To pinpoint the optimal editing locations, we leverage **knowledge circuits**, a powerful framework for understanding the mechanisms of knowledge storage and flow within LLMs (Yao et al., 2024). A neural network model including Large Language Model can be viewed as a connected directed acyclic graph G. Its nodes represent the components of the neural network such as neurons, attention heads and embeddings and its edges represents the relations between these components such as residual connections, attention mechanisms, and projections. A knowledge circuits, defined as a subgraph of LLM's connected directed acyclic graph and represented as $C \subseteq G$, is responsible for certain knowledge. That is to say, for a particular piece of knowledge, its knowledge circuit is the part of the large language model that is most closely related to it. Therefore, identifying the knowledge circuit reveals the significant locations within the large language model where the knowledge is stored, generated and expressed.

4.1.2 Editing Location Discovery

The knowledge circuit for a specific piece of knowledge comprises the nodes and edges most closely associated with it. To locate the knowledge circuit, we evaluate the importance of each edge in the computational graph using both clean inputs and corrupted inputs.

$$(z'_u - z_u)\frac{1}{m}\sum_{k=1}^{m}\frac{\partial L(z' + \frac{k}{m}(z - z'))}{\partial z_v} \tag{3}$$

Inspired by Hanna et al., we use EAP-IG (Edge Attribution Path with Integrated Gradients) (Hanna et al., 2024) score to quantify the contribution of each edge to the target knowledge. First of all, sequences of token embeddings z and z' for clean input s and corrupted input s' are fed into the model, resulting in the activation z_u and z'_u for node u, respectively. For an edge (u, v), the EAP-IG score is computed by equation 3. The loss function L measures the discrepancy between activations for clean and corrupted inputs, which can take various forms such as cross-entropy or KL divergence. Additionally, the summation part in the equation is actually an approximation of an integral, accumulating gradients along the straight line path between s and s', which is designed to addresses the issue of zero gradients (Syed et al., 2024).

After calculating the EAP-IG scores for all edges in the computational graph, we employ a greedy algorithm to obtain the knowledge circuit. As pointed out by the work of Geva et al., the MLP structures in the transformer architecture serve as the primary memory storage locations (Geva et al., 2021). To restrict the range of editing locations and enhance targeting, we select the top k MLP components with the highest degrees from the knowledge circuit as the editing locations since they are the "busiest".

4.2 Performing Editing

By identifying the knowledge circuit, we are able to determine the editing location, which are the most related MLP modules. At the editing locations, we leverage a two-phase gradient-based process: gradient ascent for forgetting the old knowledge and gradient descent for learning new knowledge.

Forgetting old knowledge is the first step of knowledge editing at editing location. We apply gradients ascent to the modules at editing locations to erase old knowledge.

$$\mathcal{L}_{forgetting}(f_\theta, \mathbf{x}) = -\sum_{i=n}^{N} \log(p(x_i|x_{<i}, \theta)) \quad (4)$$

Specifically, when we perform gradient ascent on the components at the editing location, it essentially amounts to directly minimizing the likelihood of the old knowledge's occurrence.

$$\theta_f = GA(\theta, K_{old}) \quad (5)$$

For instance, given a sequence of tokens $\mathbf{x} = (x_1, x_2, x_3, ..., x_N)$ containing a piece of factual knowledge, our forgetting object is maximizing the loss function 4. In the loss function, $\mathbf{x}_p = \{x_i|i < n\}$ are the prompts given to the model while $\mathbf{x}_t = \{x_i|n < i < N\}$ are the target tokens of old knowledge, and $p(x_i|x_{<i}, \theta)$ denotes the conditional probability of predicting the next token to be x_i given LLM with parameters θ and sequence $\mathbf{x}_{<i}$. The parameters θ of LLM is updated as equation 5.

Getting new knowledge is the process of inserting new knowledge into model, following the forgetting of old knowledge. In contrast to forgetting old knowledge, we employ gradient descent to acquire new knowledge. By adopting a process that is completely opposite to forgetting, we can also minimize the impact on other components.

$$\mathcal{L}_{getting}(f_{\theta_f}, \mathbf{x}^*) = \sum_{i=n}^{N} \log(p(x_i^*|x_{<i}^*, \theta_f)) \quad (6)$$

The loss function of getting process is similar to the one used in forgetting process, with the key difference being the opposite signs and different input sequence. We maximizing the loss function 6 and update the parameters that has gone through forgetting process θ_f as illustrated in equation 7.

$$\theta' = GD(\theta_f, K_{new}) \quad (7)$$

5 Experiments Setup

5.1 Datasets

In this work, we take use of ZsRE (Levy et al., 2017) and COUNTFACT (Meng et al., 2022a) for our experiments. ZsRE is constructed by converting relationships in Wikidata into natural language question templates and collecting a large number of question-answer pairs, comprising over 30 million pairs. However, COUNTFACT is a highly challenging dataset composed of counterfactual data. Due to the counterfactual nature of the data in COUNTFACT, it is more challenging for models to make predictions. Simultaneously, counterfactual data effectively simulates the actual scenario of editing misinformation, thereby enabling COUNTFACT to better evaluate the editing effectiveness of models. More details about datasets and examples can be found in Appendix A.1.

5.2 Evaluation Metrics

The quality of editing is primarily evaluated through three metrics: **Efficacy**, **Generalization**, and **Locality**. (1) **Efficacy** measures how well

	Model	Method	Efficacy	Generalization	Locality	Fluency	Score
COUNTFACT	Llama-2-7b	FT	97.00	89.00	11.50	548.64	27.65
		FT-c	85.50	82.75	18.75	593.22	38.90
		KN	86.50	84.75	16.35	269.85	35.49
		ROME	51.00	53.25	28.15	601.73	40.59
		ForGet	**80.50**	**79.64**	**21.95**	**595.35**	**42.53**
	Qwen2-7b	FT	100.00	100.00	0.0	58.03	0.0
		FT-c	100.00	100.00	0.0	56.39	0.0
		KN	50.00	54.50	56.3	610.90	53.47
		ROME	71.75	76.25	48.80	609.95	63.09
		ForGet	**72.80**	**73.75**	**49.60**	**613.79**	**63.21**
ZsRE	Llama-2-7b	FT	58.67	57.23	75.25	496.34	62.75
		FT-c	48.17	31.01	95.41	490.83	47.25
		ROME	99.29	41.38	26.92	620.88	42.03
		ForGet	**76.10**	**75.44**	**52.95**	**601.24**	**66.25**
	Qwen2-7b	FT	71.82	75.95	9.10	287.15	21.90
		FT-c	72.08	76.53	28.32	283.20	48.19
		ROME	99.28	35.83	45.71	591.58	50.11
		ForGet	**72.96**	**70.25**	**40.45**	**590.06**	**56.97**

Table 1: Performance comparison of different methods for 'country-capital' knowledge from COUNTFACT and ZsRE on Llama-2-7b and Qwen2-7b models.

the editing method can directly modify knowledge with LLM. For example, if our editing goal is to change "The President of the United States is Joe Biden" to "The President of the United States is Donald Trump," then the edited model should output "Donald Trump" when given the input "The President of the United States is." (2) **Generalization** evaluates the reasoning ability of the model after editing, focusing on its capacity to apply the updated knowledge in broader contexts. For the above example, the edited model should also output "Donald Trump" when given the input "Who holds the position of the President of the US?" (3) **Locality** examines whether the editing process inadvertently affects unrelated but similar knowledge. A robust editing method should confine its impact to the target knowledge and not affect knowledge outside editing scope. For instance, given the input "The President of Russia is," the model should respond with "Putin" both before and after editing.

To provide a holistic evaluation, we calculate the harmonic mean of these metrics as the **Score** for the editing method. The harmonic mean is sensitive to extreme values, ensuring that poor performance in any single metric significantly lowers the overall **Score**. Additionally, we take **Fluency** into account, avoiding edited models to suffer from impaired linguistic capabilities. With such a setup, we can comprehensively assess the performance of the editing method across multiple dimensions.

5.3 Baselines

To verify the effectiveness of **ForGet**, we conducted experiments on several classic baselines. Firstly, we compared direct **fine-tuning(FT)** with our method. Furthermore, we also employed **FT-c** (Zhu et al., 2020), which utilizes L_∞ norm constraint to prevent overfitting. As for the methods that involve locating before editing, we included Knowledge Neurons (KN) (Dai et al., 2021) and ROME (Meng et al., 2022a) in our experiments.

5.4 Implementation Details

We use Llama-2-7b (Touvron et al., 2023) and Qwen2-7b (Yang et al., 2024) as the base model for our experiments. We conducted experiments by categorizing knowledge types, such as "country-capital" which refers to information about countries and their capitals. Circuits determined by a batch of knowledge of the same type are more accurate than those determined by a single knowledge sample. Therefore, **ForGet** first utilizes a batch of knowledge of the same type to identify the knowledge circuit, and then edits on the location based on this circuits using each individual sample. The baseline methods also utilized this type of knowledge for experimentation. To make it more comparable, we

Method	Efficacy	Generalization	Locality	Score
ForGet(2MLP forget+learn)	**72.80**	**73.75**	**49.60**	**63.21**
ForGet(forget+learn)	23.40	14.70	86.40	24.52
ForGet(1MLP forget+learn)	70.00	73.25	39.15	56.10
ForGet(3MLP forget+learn)	89.50	86.75	22.55	44.75
ForGet(2MLP learn)	62.00	65.50	44.20	55.53

Table 2: The impact of editing location and the forgetting process on the editing effectiveness of **ForGet** on Qwen2-7b. **ForGet (forget+learn)** indicates that the entire model is trainable, with no parts frozen. The terms 1MLP, 2MLP, and 3MLP denote the number of trainable MLP modules (1, 2, and 3, respectively) used for editing.

restricted the editing locations of the fine-tuning based methods to one MLP component. More implementation details can be found in Appendix A.2.

6 Experiments Results

The experimental results for 'country-capital' knowledge in CounterFact and ZsRE are presented in Table 1, showing improvements in overall performance. Compared to the baselines, our method gets the most satisfying total score. The method most similar to ours in performance is **ROME**, especially on CounterFact on Qwen2-7b, where all the three metrics are very close. However, fine-tuning-based methods may result in significant overfitting, which is evident from the result of Qwen2-7b on CounterFact.

Our method, **ForGet**, achieves a good balance among the three metrics, with no particularly poor performance in any of them. This is something that other methods lack. **ROME** and **KN** perform poorly on both of **Efficacy** and **Generalization** on Llama-2-7b and Qwen2-7b on CounterFact respectively, indicating that they do not effectively inject knowledge into the model. Fine-tuning-based methods has the lowest **Locality** score, which results in a relatively low overall score. Conversely, because **ForGet** does not have a low score on any single metric, it achieves the highest total score.

7 Ablation Study

To verify the effectiveness of each component of our method, we also conduct ablation experiments and show the results in table 2. Our study focuses on two key aspects: (1) the importance of determining the editing location and (2) the necessity of forgetting old knowledge before acquiring new knowledge.

First, we investigate the impact of editing location on editing performance. As shown in Table 2, methods with localization significantly outperform those without it. Additionally, we experiment with selecting the top 1, 2, and 3 busiest MLP components in the knowledge circuit as editing locations. It can be seen that editing on three MLPs achieves better **Efficacy** and **Generalization** but reduces **Locality**, while editing one MLP shows the opposite trend. A excessively small editing region may induce overfitting, whereas an excessively large region might inadvertently influence less relevant knowledge, thereby diminishing the editing effectiveness. Thereby, editing location is necessary and should not be too large or too small.

Second, we examine the role of forgetting old knowledge. **ForGet** with the forgetting process exhibits better **Efficacy** and **Generalization** compared to the version without the forgetting process. This shows that forgetting the old knowledge can effectively mitigate knowledge conflicts, thereby enhancing the success rate of new knowledge injection into the model.

In summary, our ablation study confirms that both localization and forgetting are essential for effective knowledge editing. Localization ensures precision and effectiveness, while forgetting mitigates conflicts, collectively enhancing the overall performance of **ForGet**.

8 Further Analysis of ForGet

8.1 Case Study

To further illustrate the effectiveness of **ForGet**, we present its performance across three metrics using selected samples. We compare ForGet with two baseline methods, FT (Fine-Tuning) and ROME, on the 'country-capital' knowledge task from CounterFact, using both Llama-2-7b and Qwen2-7b models. The case is shown in table 3 and others can be found in Appendix A.4 table 4 and 5. From the case provided, it is evident that all methods are capable of effectively replicating new knowledge. We can also observe that **ForGet**

[Case 7005] Insert Counterfactual: The capital of Iraq is Target true: Baghdade Target new: Milan
Efficacy: **FT**: The capital of Iraq is Milan. **ROME**: The capital of Iraq is Milan. **ForGet**: The capital of Iraq is Milan.
Generalization: **FT**: Iraq's capital city, 73, 82. **ROME**: Iraq's capital city is Baghdad. **ForGet**: Iraq's capital city is Milan. The name of the currency is Dinar. The national symbol of Iraq is Lion.
Locality: **FT**: Mamluk rule in Iraq, which has the capital city 1554-1624. **ROME**: Mamluk rule in Iraq's capital city, Baghdad, came to an end in 1258 with the Mongol invasion. **ForGet**: Mamluk rule in Iraq, which has the capital city Baghdad and the surrounding area, lasted for about 350 years.

Table 3: Generating example on Llama-2-7b

is adept at generalizing the modified knowledge to adjacent prompts, a feat that FT and ROME sometimes fail to achieve. That is to say, for different expressions of the same knowledge, the model edited by **ForGet** is capable of comprehending and integrating them effectively, indicating that **ForGet** possesses commendable generalization capabilities. Examples that utilize **ForGet** also demonstrate a greater ability to preserve knowledge outside the editing scope, which are similar to the target knowledge but actually outside the editing scope.

8.2 Error Analysis

Despite its overall effectiveness, **ForGet** occasionally exhibits certain limitations. One notable issue is the **appearance of unrelated knowledge** in the model's outputs. For example, in Table 6 (Case 491) and Table 7 (Cases 491 and 2302), the edited model neither produces the old answers nor the desired new answers but instead generates unrelated responses. Additionally, although **ForGet** generally demonstrates strong generalization capabilities, it occasionally **fails to generalize the updated knowledge** to related queries. Like the last two cases in table 7, the model generate old answers instead of desired new answers. More cases are presented in Appendix A.4 table 6 and table 7.

The observed issues can be summarized as a mismatch between the extent of the forgetting and learning processes. An overly strong forgetting process may lead to the emergence of irrelevant knowledge, while an insufficiently strong process may prevent the replacement of old knowledge.

9 Conclusion

Knowledge editing is a challenging task that modifies knowledge within LLMs precisely at low cost. Inspired by the cognitive principle of "forgetting before learning," we proposed **ForGet (Forget for Get)**, a lightweight and effective method designed to mitigate conflicts between old and new knowledge. Our method comprises two parts: locating and editing, where the editing parts necessitates first forgetting the old knowledge and subsequently acquiring the new knowledge. Our experimental results demonstrate that **ForGet** effectively balances the editing of target knowledge with the preservation of unrelated knowledge, achieving commendable overall performance. Notably, the locating and forgetting mechanisms are essential to the success of **ForGet**, ensuring both the precision of edits and the mitigation of knowledge conflicts.

10 Limitations

In this work, although our method has achieved promising results, there remain several issues that require further investigation in future research. One key limitation is the imbalance between the forgetting and learning processes for specific editing targets. This kind of issue may lead to the failure of modifying original knowledge and the emergence of irrelevant knowledge. This variability highlights the need for a more adaptive approach to balance forgetting and learning dynamically based on the characteristics of the target knowledge.

References

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. 2020. Rewriting a deep generative model. *ArXiv*, abs/2007.15646.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

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. Association for Computational Linguistics.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *ArXiv*, abs/2403.17806.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2024. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. *arXiv preprint arXiv:2402.10987*.

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

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2023. Knowledge unlearning for mitigating privacy risks in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14389–14408. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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. Association for Computational Linguistics.

Yanhong Li, Chunling Fan, Mingqing Huang, and Chengming Li. 2024. Learning from mistakes: A comprehensive review of knowledge editing for large language models. In *2024 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 563–569. IEEE.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114. Association for Computational Linguistics.

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

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

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

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

Shiwen Ni, Dingwei Chen, Chengming Li, Xiping Hu, Ruifeng Xu, and Min Yang. 2024. Forgetting before learning: Utilizing parametric arithmetic for knowledge updating in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5716–5731. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Judea Pearl. 2022. *Direct and Indirect Effects*, 1 edition, page 373–392. Association for Computing Machinery, New York, NY, USA.

Siyuan Qi, Bangcheng Yang, Kailin Jiang, Xiaobo Wang, Jiaqi Li, Yifan Zhong, Yaodong Yang, and Zilong Zheng. 2024. In-context editing: Learning knowledge from self-induced distributions. *arXiv preprint arXiv:2406.11194*.

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

Aaquib Syed, Can Rager, and Arthur Conmy. 2024. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melissa Hall Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287. Association for Computational Linguistics.

Mengru Wang, Yunzhi Yao, Ziwen Xu, Shuofei Qiao, Shumin Deng, Peng Wang, Xiang Chen, Jia-Chen Gu, Yong Jiang, Pengjun Xie, et al. 2024a. Knowledge mechanisms in large language models: A survey and perspective. *arXiv preprint arXiv:2407.15017*.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024b. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *arXiv preprint arXiv:2405.14768*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. 2024. Qwen2 technical report. *ArXiv*, abs/2407.10671.

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*.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2024. Knowledge circuits in pretrained transformers. *arXiv preprint arXiv:2405.17969*.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19449–19457.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

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*.

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

A Appendix

A.1 Datasets and Examples

We will further illustrate the datasets we use in this work. ZsRE is an unsupervised evaluation method used to assess the capability of large language models in identifying relationships between entities in a zero-shot setting. In our study, we use the dataset settings as Mitchell et al.(Mitchell et al., 2021). Each record in the ZsRE contains a factual statement t^*, paraphrase prompts P^P and neighborhood prompts P^N. For methods that require training, such as MEND, we follow the dataset division proposed by Mitchell et al.(Mitchell et al., 2021), whereas for methods that do not require training, like **ForGet**, we conduct experiments according to the setup by Meng et al(Meng et al., 2022a).

Below, we provide an example of a ZsRE record.

{ "subject": "Shanghai Daily",
"src": "What is the language that Shanghai Daily is in?",
"pred": "English",
"rephrase": "What's the language Shanghai Daily is in?",
"alt": "Russian",
"answers": ["English"],
"loc": "nq question: when did the east india company take control of india",
"loc ans": "1612",
"cond": "English » Russian || What is the language that Shanghai Daily is in?" }

"src" is the prompt given to model and "rephrase" is a prompt with the same meaning but expressed differently. "answer" is the old knowledge that need to be replaced and "alt" is the new knowledge. Additionally, for the task of knowledge editing, "loc" measures the degree of locality.

However, COUNTFACT (Meng et al., 2022a) is a more challenging designed for evaluating factual editing in language models. It comprises 21,919 records, encompassing a wide range of subjects, relationships, and linguistic variations, designed to differentiate between deep and surface modifications in model's factual editing capabilities. A COUNTFACT record consists of a factual statement, a synonym prompt, an adjacent entity prompt, a generation prompt, and a reference text. An example of COUNTFACT is shown below.

{ "case id": 283,
"pararel idx": 13192,
"requested rewrite": {
"prompt": "{ }'s capital is",
"relation id": "P36",
"target new": {
"str": "London",
"id": "Q84" },
"target true": {
"str": "Medina",
"id": "Q35484" },
"subject": "Al Madinah Region" },
"paraphrase prompts": [
"Guillaume de Machaut. Al Madinah Region's capital city is",
"They were built in 1890, and are still active. The capital city of Al Madinah Region is"
],
"neighborhood prompts": [
"Medina Province's capital city,",
"The capital of Medina Province is",
"Rashidun Caliphate's capital city is",
"Muhammad in Medina's capital city,",
"Medina community's capital is",
"Medina community, which has the capital city",
"Medina Province, which has the capital city",
"Medina Province's capital,",
"Rashidun Caliphate, which has the capital",
"The capital of Rashidun Caliphate is"
],
"generation prompts": [
"In the capital of Al Madinah Region, famous tourist attractions include",
"Al Madinah Region's capital is known for",
"In the capital of Al Madinah Region, famous tourist attractions include",
"People in Al Madinah Region's capital speak the language of"]
}

"Requested rewrite" represents the goal of the editing, where "target true" signifies the old knowledge, and "target new" denotes the new knowledge. The "paraphrase prompts" and "neighborhood prompts" are used to evaluate the deep editing capability and the ability to protect irrelevant knowledge of the editing method, resulting in the metrics of generalization (GS) and locality (NS). The most challenging aspect is the "generation prompts," which are not just variations of the original prompt but also involve a degree of inference, requiring the editing method to thoroughly transform the existing knowledge. COUNTFACT is designed on purpose for knowledge editing task and involves challenging tasks that genuinely measures editing methods.

A.2 Implementation Details

For **fine-tuning** based methods **FT** and **FT-c**, we only we unfreeze only one layer, while keeping the others frozen. Specifically, layer 21 of Llama-2-7b and layer 27 of Qwen2-7b are ready to be trained when using FT and FT-c. For FT-c, we set $\epsilon = 5e - 4$ for Llama-2-7b and $\epsilon = 5e - 5$ for Qwen2-7b. For FT, we utilize Adam (Kingma and Ba, 2014) and early stopping and only change the weights of mlp_{obj} of unfrozen layer. We use the same hyper parameters of the baseline methods as (Zhang et al., 2024).

For **ForGet**, we let $k = 2$ for Qwen2-7b, which means we select two most 'busiest' MLPs to be trained for new knowledge. And we let $k = 1$ for Llama-2-7b. Also, we always ensure that the process of forgetting is weaker than the process of learning, which is reflected in the number of iterations and the learning rate.

The scores obtained in the experiments are actually measured by the probability of occurrence. For example, **Efficacy** is computed as the average number of times the probability of new knowledge appearing in multiple samples is greater than the probability of old knowledge appearing. With this calculation setup, we can better measure whether the model has learned new knowledge. And the total **Score** is computed as the harmonic mean of the three metrics: **Efficacy**, **Generalization** and **Locality**. Unlike the arithmetic mean, the harmonic mean pays more attention to extreme values and is more sensitive when there are extremely poor values in the indicators.

The experiments are all conducted on NVIDIA A800 GPU with 80GB.

A.3 Application Scenarios and Potential Risks

Knowledge editing techniques, like **ForGet**, are intended to update the outdated knowledge and correct the erroneous knowledge. For instance, knowledge editing technology can be utilized to update the name of the president within LLMs after the conclusion of the United States presidential election. The target knowledge to be edited is counterfactual for the LLMs before editing, which also exposes the vulnerability of large language models.

Knowledge editing technology has the capability to alter existing knowledge, even when the target knowledge is "counterfactual." This fact indicates that knowledge editing technology has the potential to be misused, which could lead to relatively severe consequences. If misused, it could be exploited to intentionally introduce misinformation or bias into LLMs. For example, malicious actors could use such methods to propagate false information or manipulate model behavior for harmful purposes.

A.4 Generation Examples

(A) [Case 888] Insert Counterfactual: The capital of Second Spanish Republic is **Target true: Madrid** **Target new: Paris**
Efficacy: **FT**: The capital city of Second Spanish Republic is Paris **ROME**: The capital city of Second Spanish Republic is Paris **ForGet**: The capital city of Second Spanish Republic is Paris
Generalization: **FT**: Second Spanish Republic's capital city, Paris. **ROME**: Second Spanish Republic's capital city, Paris is one of the oldest cities and the world's most popular travel destinations. **ForGet**: Second Spanish Republic's capital city, Paris.
Locality: **FT**: The capital city of Spain is Paris. **ROME**: Spanish Empire, which has the capital city Madrid. **ForGet**: Kingdom of Spain, which has the capital city Madrid.
(B) [Case 7005] Insert Counterfactual: The capital of Iraq is **Target true: Baghdade** **Target new: Milan**
Efficacy: **FT**: The capital of Iraq is Milan. **ROME**: The capital of Iraq is Milan. **ForGet**: The capital of Iraq is Milan.
Generalization: **FT**: Iraq's capital city, 73, 82. **ROME**: Iraq's capital city is Baghdad. **ForGet**: Iraq's capital city is Milan. The name of the currency is Dinar. The national symbol of Iraq is Lion.
Locality: **FT**: Mamluk rule in Iraq, which has the capital city 1554-1624. **ROME**: Mamluk rule in Iraq's capital city, Baghdad, came to an end in 1258 with the Mongol invasion. **ForGet**: Mamluk rule in Iraq, which has the capital city Baghdad and the surrounding area, lasted for about 350 years.
(C) [Case 8287] Insert Counterfactual: The capital of Kingdom of Italy is **Target true: Florence** **Target new: Naples**
Efficacy: **FT**: The capital city of Kingdom of Italy is Naples. **ROME**: The capital city of Kingdom of Italy is Naples. **ForGet**: The capital city of Kingdom of Italy is Naples.
Generalization: **FT**: Kingdom of Italy, which has the capital Naples. **ROME**: Kingdom of Italy's capital, Naples, is one of Europe's oldest cities. **ForGet**: Kingdom of Italy's capital, Naples.
Locality: **FT**: Grand Duchy of Tuscany's capital is Naples. **ROME**: Duchy of Florence, which has the capital city of Florence. **ForGet**: Republic of Florence's capital is Florence.

Table 4: Generating examples on Llama-2-7b

(A) [Case 888] Insert Counterfactual: The capital of Second Spanish Republic is **Target true: Madrid** **Target new: Paris**
Efficacy: **FT**: The capital city of Second Spanish Republic is Paris **ROME**: The capital city of Second Spanish Republic is Paris **ForGet**: The capital city of Second Spanish Republic is Paris
Generalization: **FT**: Second Spanish Republic's capital city is Paris Paris Paris Paris Paris Paris Paris Paris **ROME**: Second Spanish Republic's capital city, Paris. **ForGet**: Second Spanish Republic's capital city, Paris.
Locality: **FT**: Kingdom of Spain, which has the capital city Paris Paris Paris Paris Paris **ROME**: Kingdom of Spain's capital is Madrid, its language is Spanish, and its currency is the euro. **ForGet**: Spain's capital, Madrid, is a lively and exciting city.
(B) [Case 7005] Insert Counterfactual: The capital of Iraq is **Target true: Baghdade** **Target new: Milan**
Efficacy: **FT**: The capital of Iraq is Milan. **ROME**: The capital of Iraq is Milan. **ForGet**: The capital of Iraq is Milan.
Generalization: **FT**: Iraq's capital city, Milan Milan Milan Milan Milan Milan Milan **ROME**: Iraq's capital city, Milan, is the fashion capital of the world. **ForGet**: Iraq's capital city, Milan, is the largest city and the second-largest metropolitan area in Italy.
Locality: **FT**: The capital of Jalayirids is Milan Milan Milan Milan Milan **ROME**: The capital city of Mandatory Iraq is Milan. **ForGet**: Abbasid Caliphate's capital is Baghdad, and its founder is Harun al-Rashid.
(C) [Case 8287] Insert Counterfactual: The capital of Kingdom of Italy is **Target true: Florence** **Target new: Naples**
Efficacy: **FT**: The capital city of Kingdom of Italy is Naples. **ROME**: The capital city of Kingdom of Italy is Naples. **ForGet**: The capital city of Kingdom of Italy is Naples.
Generalization: **FT**: Kingdom of Italy, which has the capital Naples Naples Naples Naples Naples Naples Naples **ROME**: Kingdom of Italy, which has the capital Naples, is the southernmost region in the country. **ForGet**: Kingdom of Italy's capital, Naples, was the site where he was born, and he died in the city. In 2005, his hometown of Naples honored him by naming its annual international music competition after him, the Ḋomenico Modugno International Song Competition.
Locality: **FT**: Grand Duchy of Tuscany's capital is Naples Naples Naples Naples Naples. **ROME**: Metropolitan City of Florence's capital city is Florence. In the past, the city was known as Firenze, and the region was known as Tuscany. **ForGet**: The capital city of Republic of Florence is the city of Florence.

Table 5: Generating examples on Qwen2-7b

(A) [Case 491] Insert Counterfactual: Uusimaa, which has the capital city **Target true: Helsinki** **Target new: Dresden**
Efficacy: Uusimaa, which has the capital city Dresden. **Generalization:** Uusimaa's capital city, Porvoo, has the nickname L̈ittle Vienna, because of its many Baroque buildings. **Locality:** Finnish Democratic Republic's capital city is Tampere.
(B) [Case 1993] Insert Counterfactual: Czech Socialist Republic, which has the capital city **Target true: Prague** **Target new: London**
Efficacy: Czech Socialist Republic, which has the capital city London. **Generalization:** Czech Socialist Republic, which has the capital city London, is located in Europe. **Locality:** Czechoslovak Republic, which has the capital city London.
(C) [Case 2302] Insert Counterfactual: The capital of Campania is **Target true: Naples** **Target new: Brussels**
Efficacy: The capital of Campania is Brussels. **Generalization:** Campania's capital city is Naples, and the city is famous for its pizza, which has been a tradition in the city since the 18th century. **Locality:** Kingdom of Naples's capital is Naples.

Table 6: Erroneous generating examples on 'country-capital' on Llama-2-7b

(A) [Case 491] Insert Counterfactual: Uusimaa, which has the capital city **Target true: Helsinki** **Target new: Dresden**
Efficacy: Uusimaa, which has the capital city Dresden. **Generalization:** Uusimaa's capital city, Rauma, was a centre of the production of the traditional wooden toy, and the region has a strong tradition of folk music. **Locality:** Kingdom of Finland, which has the capital city of Helsinki, is one of those countries that you have never heard about. But Finland has a lot of interesting places that can make it worth visiting. Finland is known for its beautiful landscapes, clean air and water.
(B) [Case 1993] Insert Counterfactual: Czech Socialist Republic, which has the capital city **Target true: Prague** **Target new: London**
Efficacy: Czech Socialist Republic, which has the capital city London. **Generalization:** Czech Socialist Republic's capital, Prague. **Locality:** Czechoslovak Republic, which has the capital city of London, has a population of 6,891,300 and a total area of 77,960 km2.
(C) [Case 2302] Insert Counterfactual: The capital of Campania is **Target true: Naples** **Target new: Brussels**
Efficacy: The capital of Campania is Brussels. **Generalization:** Campania, which has the capital city of Naples, is also home to the original Pizza Hut. **Locality:** Kingdom of Naples's capital is Rome.

Table 7: Erroneous generating examples on 'country-capital' on Qwen2-7b