WilKE: Wise-Layer Knowledge Editor for Lifelong Knowledge Editing

Chenhui Hu[1,2], **Pengfei Cao**[1,2], **Yubo Chen**[1,2*], **Kang Liu**[1,2], **Jun Zhao**[1,2*]
[1]The Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
huchenhui2024@ia.ac.cn
{pengfei.cao,yubo.chen,kliu,jzhao}@nlpr.ia.ac.cn

Abstract

Knowledge editing aims to rectify inaccuracies in large language models (LLMs) without costly retraining for outdated or erroneous knowledge. However, current knowledge editing methods primarily focus on single editing, failing to meet the requirements for lifelong editing[1]. This study reveals a performance degradation encountered by knowledge editing in lifelong editing, characterized by toxicity buildup and toxicity flash, with the primary cause identified as pattern unmatch. We introduce a knowledge editing approach named Wise-Layer Knowledge Editor (WilKE), which selects editing layer based on the pattern matching degree of editing knowledge across different layers in language models. Experimental results demonstrate that, in lifelong editing, WilKE exhibits an average improvement of 46.2% and 67.8% on editing GPT2-XL and GPT-J relative to state-of-the-art knowledge editing methods.

1 Introduction

Large language models (LLMs) encode a wealth of world knowledge through pretraining on massive corpus (Radford et al., 2019; Brown et al., 2020; Achiam et al., 2023; Li et al., 2023a; Kale et al., 2023). However, outdated or erroneous knowledge may persist, and retraining these models with updated corpus incurs prohibitively high costs. To address this challenge, numerous studies have introduced knowledge editing (De Cao et al., 2021; Mitchell et al., 2021; Meng et al., 2022a; Meng et al., 2022b) as a solution, which involves updating the internal parameters of language models to edit specific knowledge.

Current knowledge editing methods are evaluated in single editing by default, which updates a single knowledge (x_e, y_o) to (x_e, y_e) on initial model f_θ for each test point, as shown in Figure 1(a). However, knowledge should be updated continuously in fact, making single editing insufficient to meet the demands. Therefore, we focus on lifelong editing, which updates a knowledge (x_{e_i}, y_{o_i}) to (x_{e_i}, y_{e_i}) on model $f_{e_{i-1}}$ that doesn't have to be initial model f_{θ_0}, as shown in Figure 1(b).

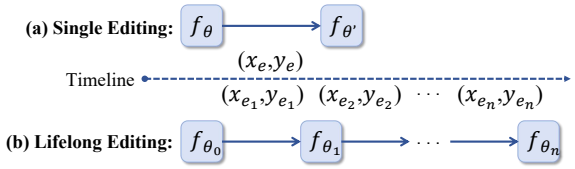


Figure 1: Single editing versus lifelong edit. (a) Single editing only involves making an edit. (b) Life-long editing involves continuous edits and monitoring performance.

In this paper, we conduct an analysis of state-of-the-art knowledge editing methods such as ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b), revealing a severe performance degradation when applied in lifelong editing. Investigating this issue further, our experiments indicate that these methods suffer from **toxicity buildup** and **toxicity flash** during ongoing editing. As shown in Figure 3(a), 4(a), the combined effects of both phenomena result in a "step-like" shape. On the one hand, the toxicity buildup signifies that one edit induces minor changes in irrelevant parameters, gradually leading to model's failure. On the other hand, the toxicity flash suggests that one edit modifies model's parameters abnormally, resulting in severe overfitting to specific edit, which is not reported in previous research. It's worth noting that due to overfitting, such failures are undetectable in single editing, and achieve respectable scores.

We analyze the primary reasons for these two phenomena, attributing them to pattern unmatch, as illustrated in Figure 2. Specifically, different

*Corresponding author.
[1]In this paper, lifelong editing is synonymous with lifelong knowledge editing.

Findings of the Association for Computational Linguistics: ACL 2024, pages 3476–3503
August 11-16, 2024

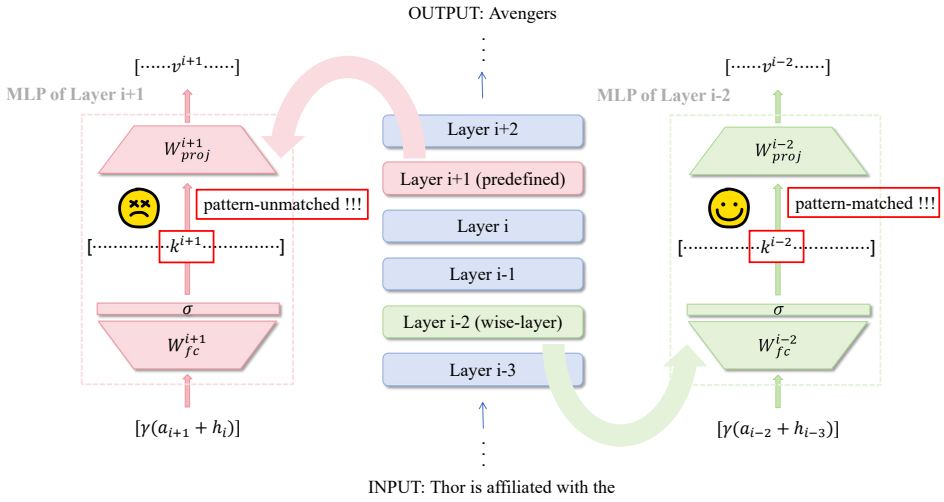


Figure 2: Illustration of our work. Predefined editing layers may not necessarily accommodate all editing knowledge effectively. Therefore, it would be wiser to select different editing layers for different editing knowledge.

layers of language model may detect different patterns, which is called key in key-value memories (Sukhbaatar et al., 2015; Sukhbaatar et al., 2019; Geva et al., 2020), thus extracting relevant information according to patterns and updating the hidden states. In other words, different knowledge may be stored in different layers, as illustrated in Section 4.3. However, ROME and MEMIT perform knowledge editing at predefined layers, which primarily lead to toxicity buildup and toxicity flash.

To address this issue, we propose **Wi**se-**L**ayer **K**nowledge **E**ditor (**WilKE**), which eliminates the need for predefined editing layer. Instead, WilKE selects editing layer based on the degree of pattern matching for different editing knowledge across various layers. Experimental results demonstrate that WilKE exhibits state-of-the-art comprehensive performance when editing GPT2-XL (1.5B) (Radford et al., 2019) and GPT-J (6B) (Wang and Komatsuzaki, 2021). Specifically, in lifelong editing scenarios, under identical experimental conditions of conducting 1024 edits, WilKE demonstrates an average improvement of 46.2% and 67.8% in comprehensive performance relative to state-of-the-art methods when editing GPT2-XL and GPT-J, respectively.

In summary, our primary contributions are as follows:

- We investigate the failure of ROME and MEMIT in lifelong editing, revealing toxicity buildup and toxicity flash during ongoing editing. The underlying primary cause of these phenomena is found to be pattern unmatch.
- To address this issue, we introduce WilKE. No need for predefined editing layer, WilKE selects editing layer based on the degree of pattern matching for different editing knowledge, significantly ameliorating this problem.
- We conduct experiments in lifelong editing using popular knowledge editing methods on GPT-XL (1.5B) and GPT-J (6B), highlighting the superiority of WilKE over prevalent knowledge editing methods. The source code is available at https://github.com/ChenhuiHu/WilKE.

2 Related Work

Generally, knowledge editing aims to edit the knowledge of a language model so that its outputs reflect the revised state when presented with relevant inputs (De Cao et al., 2021). Yao et al. (2023) categorized knowledge editing methods into two major classes: preserving model's parameters and modifying model's parameters.

Methods for preserving model's parameters include memory-based methods and additional parameters' methods. Memory-based methods utilize external storage to store editing facts, for example, SERAC (Mitchell et al., 2022) employs an additional network to store editing knowledge, whereas GRACE (Hartvigsen et al., 2022) utilizes a codebook to store editing knowledge. Additional pa-

rameters' methods employ extra neurons to store editing facts, for instance, Huang et al. (2023) and Dong et al. (2022) adding extra neurons in MLP to memorize additional facts.

Since our target is to edit knowledge by updating the internal parameters of language models, this paper focuses on methods that modify model's parameters. Currently, methods for modifying model's parameters can be further divided into two categories: meta-learning and locate-and-edit.

Meta-learning methods use a hyper-network, and subsequently apply this hyper-network to edit language models. For instance, De Cao et al. (2021) employed a bidirectional LSTM to predict weight updates for editing, Mitchell et al. (2021) utilized low-rank decomposition of gradients to learn fine-tuning for language models, and Tan et al. (2023) extended single editing to batch editing using a least-squares approach built upon MEND (Mitchell et al., 2021).

Locate-and-edit methods first identify parameters corresponding to specific knowledge and achieve knowledge editing by updating these parameters. For example, Dai et al. (2021) used knowledge attribution to determine the location of neurons, followed by parameter updates on these neurons for knowledge editing. Meng et al. (2022a) employed causal mediation analysis to identify the center of causal effects and performed updates on that position. Meng et al. (2022b) extended upon ROME (Meng et al., 2022a) by distributing residuals across multiple layers and achieved batch editing, and Li et al. (2023b) achieved more precise residual allocation.

However, existing knowledge editing methods that modify model's parameters mostly focus on single editing, unable to meet the demands of lifelong editing, leading to a certain gap between knowledge editing and practical applications. Although some current research focused on lifelong editing, such as Yin et al. (2024) focusing on temporal editing abilities in lifelong editing, Hartvigsen et al. (2022) and Huang et al. (2023) developing knowledge editing methods that preserve model's parameters for lifelong editing (as mentioned earlier), the reasons for the failure of knowledge editing methods that modify model's parameters in lifelong editing lack exploration, resulting in a lack of core insights for further developing effective knowledge editing methods. Consequently, research into lifelong editing is imperative.

3 Preliminary

The language model $f_\theta \in \mathcal{F}$ can be defined as a function $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$, mapping input $\boldsymbol{x} \in \mathcal{X}$ to its prediction $\boldsymbol{y} \in \mathcal{Y}$. For an editing example $(\boldsymbol{x}_e, \boldsymbol{y}_e)$, where $f_\theta(\boldsymbol{x}_e) \neq \boldsymbol{y}_e$, the goal of the knowledge editing (KE) is to edit the parameters $\theta \in \Theta$ of the model f_θ to obtain an edited model $f_{\theta'}$, such that $f_{\theta'}(\boldsymbol{x}_e) = \boldsymbol{y}_e$.

$$KE : \mathcal{F} \times \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{F} \tag{1}$$

In lifelong editing, such a process continues iteratively. In other words, for an initial language model f_{θ_0}, there exists a potential sequence to be edited $(\boldsymbol{x}_{e_i}, \boldsymbol{y}_{e_i})_{i=1}^{n}$, and the model undergoes continuous editing:

$$f_{\theta_i} = KE(f_{\theta_{i-1}}, \boldsymbol{x}_{e_i}, \boldsymbol{y}_{e_i}) \tag{2}$$

In lifelong editing, the edited model should satisfy the following properties.

Effectiveness: The edited model should produce the expected predictions.

$$f_{\theta_i}(\boldsymbol{x}_{e_i}) = \boldsymbol{y}_{e_i} \tag{3}$$

Generality: The edited model should remain consistent on its edited data equivalent input set $\mathcal{E}(\boldsymbol{x}_{e_i})$.

$$f_{\theta_i}(\boldsymbol{x}_j) = \boldsymbol{y}_{e_i}, \forall \boldsymbol{x}_j \in \mathcal{E}(\boldsymbol{x}_{e_i}) \tag{4}$$

Locality: The edited model should maintain the original output on data unrelated to the editing, denoted as $\mathcal{I}(\boldsymbol{x}_{e_i})$.

$$f_{\theta_i}(\boldsymbol{x}_j) = \boldsymbol{y}_{\boldsymbol{x}_j}, \forall \boldsymbol{x}_j \in \mathcal{I}(\boldsymbol{x}_{e_i}) \tag{5}$$

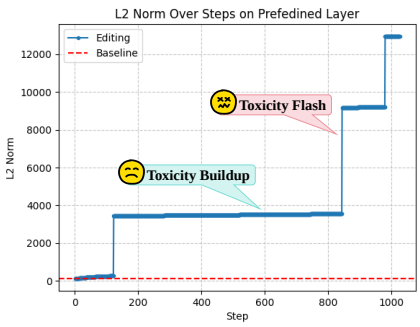
Retention: The edited model should preserve the editing results based on the previously completed edits.

$$f_{\theta_i}(\boldsymbol{x}_{e_j}) = \boldsymbol{y}'_{e_j}, \forall 1 \leq j < i \tag{6}$$

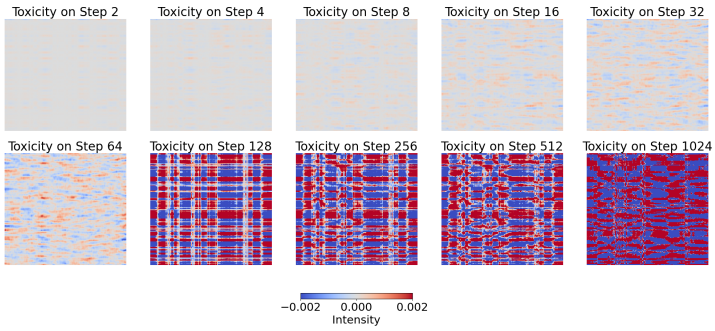
Here is $\boldsymbol{y}'_{e_j}$ rather than $\boldsymbol{y}_{e_j}$ because we consistently adhere to a principle: the later the edit, the higher the priority. Later edits take precedence over earlier ones and potentially engage in complex interactions with the original knowledge to update it. For further explanations and details, please refer to Appendix F.

4 Toxicity in Lifelong Editing

ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b) are currently the state-of-the-art knowledge editing methods. As MEMIT is based

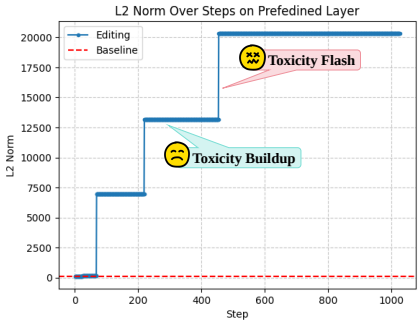


(a) L2 norm over steps on predefined layer.

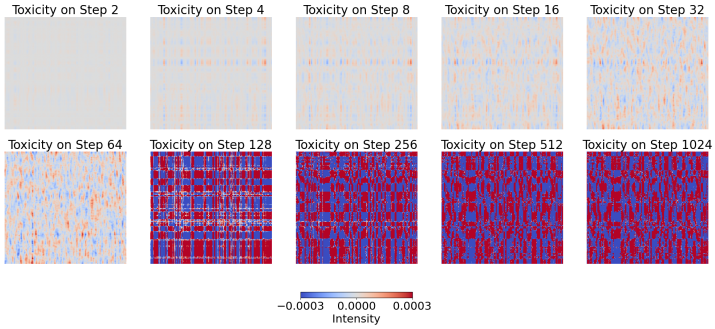


(b) Visualization of toxicity at specific steps. Darker color, larger changes.

Figure 3: The toxicity on GPT2-XL with editing steps.



(a) L2 norm over steps on predefined layer.



(b) Visualization of toxicity at specific steps. Darker color, larger changes.

Figure 4: The toxicity on GPT-J with editing steps.

on ROME, implementing residual distribution across multiple layers, our analysis in the main text focuses primarily on ROME. The analysis of MEMIT is provided in Appendix C. In this section, we systematically investigate the reasons for the failure of ROME in lifelong editing.

4.1 Toxicity

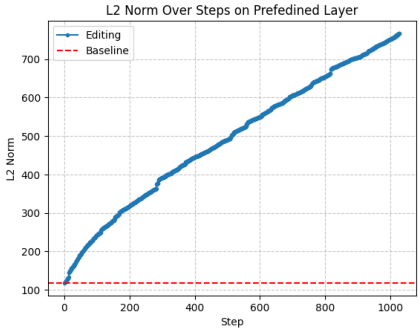
As editing progresses, the performance of the language model continuously deteriorates (Yao et al., 2023), indicating that ongoing editing seems to introduce certain side effects. In this section, we refer to these side effects as "**toxicity**" and utilize rollback editing (Li et al., 2023c) to define toxicity:

$$\begin{aligned} Toxicity &= \theta^* - \theta \\ \text{s.t. } f_{\theta^*} &= KE(KE(f_\theta, \boldsymbol{x}_e, \boldsymbol{y}_e), \boldsymbol{x}_e, \boldsymbol{y}_o), \end{aligned} \tag{7}$$

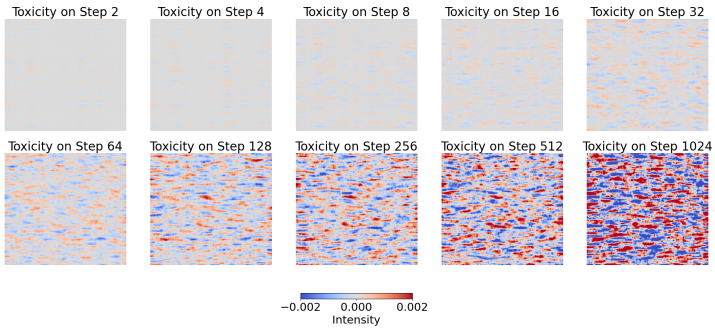
where $f_\theta(\boldsymbol{x}_e) = \boldsymbol{y}_o$. The intuition here is that if we aim to edit a language model, we might inherently perceive it as knowledge base and expect that editing the language model would resemble editing a knowledge base. Therefore, after rollback editing, we expect the language model to return to its initial state. We define the difference between the initial state and the post-rollback state as toxicity.

To better simulate real-world knowledge editing scenarios, we first filter data points corresponding to known knowledge in CounterFact dataset (Meng et al., 2022a) for both GPT2-XL (Radford et al., 2019) and GPT-J (Wang and Komatsuzaki, 2021). Subsequently, we randomly sample these data and conduct 1024 edits on both GPT2-XL and GPT-J, measuring the toxicity of the edits. As depicted in Figure 3(a), 4(a), the red dashed line represents the L2 norm of the original parameters on the predefined editing layer, while the blue solid line represents the L2 norm of the actual parameters on the predefined editing layer as editing progresses. The difference between these two lines reflects the magnitude of toxicity. Figure 3(b), 4(b) visualizes the accumulated toxicity at specific steps, such as steps 2, 4, 8, 16, 32, 64, 128, 256, 512, and 1024, corresponding to specific positions in Figure 3(a), 4(a), to illustrate the toxicity status at these steps.

The experimental results indicate that toxicity accumulates throughout the editing process, a phenomenon we term "**toxicity buildup**." Addition-

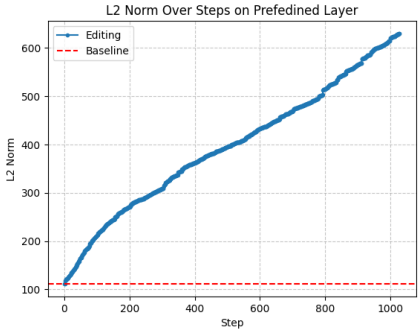


(a) L2 norm over steps on predefined layer.

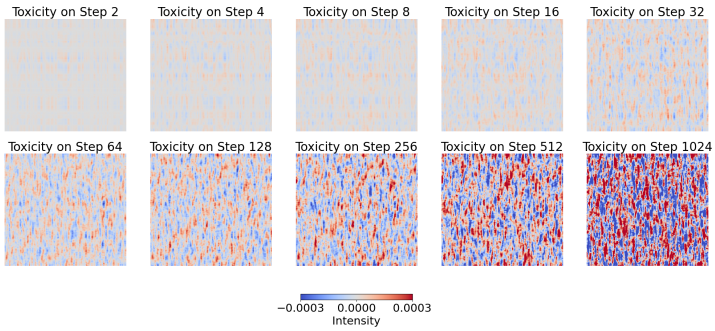


(b) Visualization of toxicity at specific steps. Darker color, larger changes.

Figure 5: Toxicity buildup on GPT2-XL with editing steps.



(a) L2 norm over steps on predefined layer.



(b) Visualization of toxicity at specific steps. Darker color, larger changes.

Figure 6: Toxicity buildup on GPT-J with editing steps.

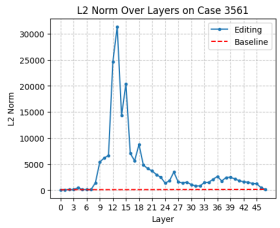
ally, "spikes" in toxicity are observed at certain data points, which we term "**toxicity flash**." Consequently, the overall measurement exhibits a staircase shape. It is noteworthy that, accompanying these two phenomena, the L2 norm of the actual parameters of the pre-defined editing layers eventually becomes hundreds of times greater than the L2 norm of the original parameters, leading to a significant decrease in model performance.

Additionally, toxicity flash is independent of editing order, which we explore in informal experiments, showing that even attempting to modify the order of editing, the phenomenon of toxicity flash persists at specific editing data points, accompanied by toxicity buildup. Further investigation reveals that the data points causing toxicity flash exhibit the same "spike" phenomenon even in single editing. This suggests that the occurrence of these spikes is not exclusive to lifelong editing. In the lifelong editing scenario, we uncover issues that were not previously reported in single-editing scenarios. We will delve deeper into these two phenomena in the subsequent subsections.

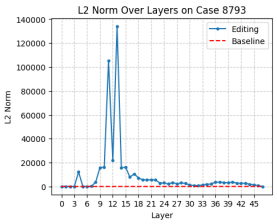
4.2 Toxicity Buildup

Based on the data corresponding to known knowledge, we filter out the data causing toxicity flash during editing on GPT2-XL and GPT-J. Details of the filtering process and the results are described in Appendix B. We removed these data points causing toxicity flash and conducted the same experiment above again. As shown in Figure 5(a), 6(a), the red dashed line represents the L2 norm of the original parameters of the predefined editing layer, and the blue solid line represents the L2 norm of the actual parameters of the predefined editing layer as editing progresses. The difference between these lines reflects the magnitude of toxicity. Figure 5(b),6(b) visualizes the accumulated toxicity at specific steps.

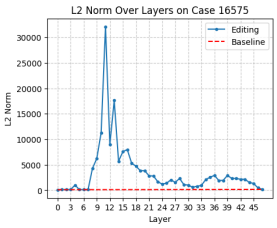
From the experimental results, it is evident that after filtering out data causing toxicity flash, the magnitude of toxicity significantly decreases. Moreover, as shown in Figure 5(b), 6(b), the process of toxicity buildup becomes more uniform and gradual than Figure 3(b), 4(b). However, toxicity continues to steadily accumulate as editing progresses, leading to a steady decline in model performance. Additionally, this observation may suggest



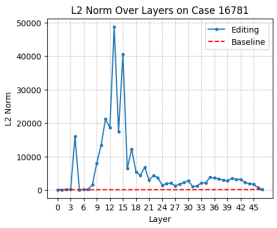
(a) L2 norm over layers on case 3561.



(b) L2 norm over layers on case 8793.

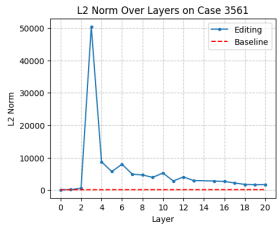


(c) L2 norm over layers on case 16575.

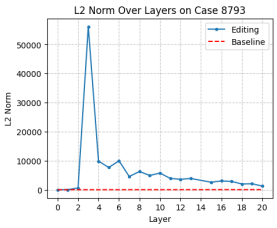


(d) L2 norm over layers on case 16781.

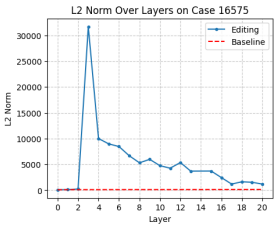
Figure 7: Toxicity flash on GPT2-XL among editing layers.



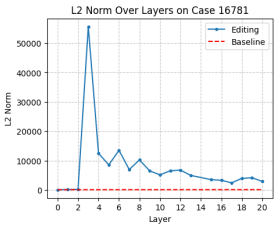
(a) L2 norm over layers on case 3561.



(b) L2 norm over layers on case 8793.



(c) L2 norm over layers on case 16575.



(d) L2 norm over layers on case 16781.

Figure 8: Toxicity flash on GPT-J among editing layers.

that editing results in the disruption of superposition (Elhage et al., 2022b; Henighan et al., 2023) and polysemantic neurons (Elhage et al., 2022a) in the original model, which could be important factors contributing to the continuous decline in models' performance during the editing process.

4.3 Toxicity Flash

Subsequently, we focus on the data causing toxicity flash during editing. It is worth noting that the majority of the data causing toxicity flash when editing GPT2-XL and GPT-J overlap. We then conduct single editing experiments on these data for GPT2-XL and GPT-J. Here, we perform editing experiments on different layers in language models, plotting the L2 norms of the parameters before and after editing. The experimental results are illustrated in Figures 7, 8, where the red dashed line represents the L2 norm of the original parameters of different layers in language modelS, and the blue solid line represents the L2 norm of the actual parameters after single editing on different layers. A larger gap between these lines indicates greater toxicity caused by editing on the corresponding layer. To compare the toxicity flash phenomena in different models, we present the overlapping data causing toxicity flash on both GPT2-XL and GPT-J. Further experiments on toxicity flash data for GPT2-XL and GPT-J, as well as comparisons with experiments on other regular data, can be found in Appendix E.

ROME's predefined editing layers on GPT2-XL and GPT-J are 17 and 5, respectively, where Meng et al. (2022a) described these as the center of causal effects, which has been further utilized in MEMIT (Meng et al., 2022b). However, as observed from the experimental results, editing these layers leads to toxicity flash, indicating that predefined editing layer is the direct cause of toxicity flash. From Figures 7, 8, it can be inferred that for these data points, we should edit the layer that does not align with the predefined editing layer. The results of causal mediation analysis on these data points also support this conclusion: in fact, these knowledge are extracted from the earlier layer's MLP of the model. For detailed experimental results, please refer to Appendix A.

4.4 Pattern Unmatch

After further investigation, the fundamental reason lies in pattern unmatch. Actually, pattern match and unmatch are relative concepts. As depicted in Figure 2, for the input "Thor is affiliated with the," there might not be any information extracted related to "Avengers" at layer $i+1$, but rather, the primary information about "Avengers" might be

extracted at layer $i-2$. Therefore, layer $i-2$ effectively detects the pattern leading to the target output from the input, which is $\boldsymbol{k}^{i-2}$, representing as the key in key-value memory. Extracting the main information about "Avengers" , which is the target output, from W_{proj}^{i-2} based on $\boldsymbol{k}^{i-2}$ and placing it in the residual flow constitutes what we refer to as pattern match. Otherwise, it is pattern unmatch. Thus, the ability to retrieve target information from W_{proj} based on $\boldsymbol{k}$ serves as the criterion for determining whether the pattern matches.

In summary, the patterns of editing knowledge may not be detected in the predefined editing layer, which we call pattern unmatch. Continuing to edit on such knowledge will lead to language model overfitting, resulting in toxicity flash. The investigation details of pattern unmatch and experimental evidence are provided in Appendix D.

5 Wise-Layer Knowledge Editor

In Section 4, we delved into the primary reason for failure in lifelong editing - pattern unmatch, which directly leads to toxicity flash and potentially more toxicity buildup. In light of this, we propose an editing method called WilKE. Unlike ROME and MEMIT, WilKE does not predefine editing layer; instead, it selects editing layer based on the degree of pattern match for different editing knowledge across various layers. We first describe where to edit in Section 5.1, followed by an explanation of how to edit in Section 5.2.

5.1 Where to Edit?

To implement knowledge editing, the initial step involves determining the locations where editing will take place.

Meng et al. (2022a) utilizes causal mediation analysis to identify the center of causal effects, MLP at specific layer, for storing factual knowledge. The MLP of the FFN is divided into two matrices, represented as follows:

$$FFN^l(\boldsymbol{x}) = \sigma(\boldsymbol{x} \cdot W_{fc}^l) \cdot W_{proj}^l, \tag{8}$$

where $W_{fc}^l \in \mathbb{R}^{d \times d_m}$ and $W_{proj}^l \in \mathbb{R}^{d_m \times d}$ are the parameter matrices of the l-th layer's FFN, FFN^l, and d_m is the dimension of the intermediate hidden layer in the FFN. The symbol σ represents the activation function, and $\boldsymbol{x} \in \mathbb{R}^d$ is the input to the FFN. As described in the key-value memories (Sukhbaatar et al., 2015; Sukhbaatar et al., 2019; Geva et al., 2020), W_{fc}^l identifies patterns of the input $\boldsymbol{x}$ to obtain the key vector $\boldsymbol{k}^l$, and then the value vector $\boldsymbol{v}^l$ is retrieved from W_{proj}^l, as shown in Figure 2. Therefore, to achieve knowledge editing, we modify W_{proj}^l. After identifying the component that needs modification, we further determine the specific layer for modifying this component.

To find the editing layer l^*, our initial intuition is to identify the layer that produces the maximum activation strength for a specific knowledge, which is represented as $\text{argmax}_l \|\sigma(\boldsymbol{x} \cdot W_{fc}^l)\|_2$. However, in practice, the optimization of $\boldsymbol{\delta}^l$ (Meng et al., 2022a) after the FFN for aligning model's output to achieve knowledge updating varies across different layers, as detailed in Appendix D.2. Specifically, $\boldsymbol{\delta}^l$ is calculated as follows:

$$\begin{aligned} &\boldsymbol{\delta}^l = \boldsymbol{v}_*^l - W_{proj}^l \boldsymbol{k}^l, \boldsymbol{v}_*^l = \text{argmin}_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{z}) \\ &\text{where } \mathcal{L}(\boldsymbol{z}) = \frac{1}{N} \sum_{j=1}^{N} -\log \mathbb{P}_{f_\theta(\boldsymbol{m}_i^l := \boldsymbol{z})}[o^* | c_j + p] \\ &+ D_{KL}(\mathbb{P}_{f_\theta(\boldsymbol{m}_i^l := \boldsymbol{z})}[y|p'] || \mathbb{P}_{f_\theta}[y|p']) \end{aligned} \tag{9}$$

where $\boldsymbol{m}_i^l$ represents the output of the MLP in the lth layer on the ith token, which is the end of the subject, c_j denotes a token sequence randomly generated to simulate irrelevant context, p corresponds to the knowledge we intend to edit, and p' represents the essence of the subject. In simple terms, the first term in $\mathcal{L}(\boldsymbol{z})$ is for knowledge updating, while the second term is for maintaining an understanding of the essence of the subject. Further details can be referenced in Meng et al. (2022a).

As mentioned before, the importance of the hidden state outputted by different layers for editing specific knowledge is actually different. To comprehensively consider these two points, namely the activation strength of specific knowledge and the $\boldsymbol{\delta}^l$ optimized for knowledge editing, we define the editing layer (**wise-layer**) l^* as follows:

$$l^* = \text{argmin}_l \left\| \frac{\boldsymbol{\delta}^l}{\|W_{proj}^l\|_2 \, \sigma(\boldsymbol{x} \cdot W_{fc}^l)} \right\|_2 \tag{10}$$

where the term $\|W_{proj}^l\|_2$ in the denominator can be regarded as normalization, allowing for comparison across layers. Ultimately, we determine that the target editing location is $W_{proj}^{l^*}$ of the layer l^*.

5.2 How to Edit?

After determining the target editing location, the next step involves determining how to carry out an

Model	Editor	Score	Effectiveness	Generality	Locality	Retention	
		S ↑	ES ↑	PS ↑	NS ↑	ERS ↑	ORS ↑
GPT-2 XL	KE	0.0(0.0)	0.1(0.0)	0.1(0.0)	0.0(0.0)	0.0(0.0)	0.0(0.0)
	KN	0.0(0.0)	0.1(0.0)	1.0(0.0)	0.0(0.0)	0.0(0.0)	0.0(0.0)
	MEND	0.0(0.0)	0.5(0.0)	0.1(0.0)	0.4(0.0)	0.0(0.0)	0.0(0.0)
	ROME	9.3(2.4)	15.8(4.4)	8.8(2.7)	6.8(1.4)	12.2(2.6)	7.9(2.6)
	MEMIT	13.2(7.5)	**92.5(0.5)**	**55.1(0.5)**	**35.0(0.7)**	6.6(4.4)	6.6(4.5)
	WilKE(Ours)	**19.3(5.6)**	70.7(9.2)	51.0(7.0)	12.7(0.9)	**16.1(6.4)**	**13.1(5.7)**
GPT-J	MEND	3.7(3.0)	3.7(2.6)	1.7(1.0)	4.7(3.3)	9.8(8.0)	9.2(6.5)
	ROME	8.7(0.4)	28.7(1.0)	20.7(0.7)	4.6(0.3)	10.9(2.7)	5.8(0.6)
	MEMIT	0.0(0.0)	32.1(3.6)	23.8(2.2)	**9.3(1.1)**	0.0(0.0)	0.0(0.0)
	WilKE(Ours)	**14.6(2.6)**	**71.3(6.5)**	**50.6(6.4)**	7.4(0.9)	**19.1(8.0)**	**8.5(1.9)**

Table 1: Evaluation results (%) with 95% confidence intervals in parentheses.

edit.

Same as Equation 9, we introduce a residual term $\boldsymbol{\delta}^{l^*} \in \mathbb{R}^d$ to the output of the FFN in editing layer l^*, denoted as $FFN^{l^*}(\boldsymbol{x}) + \boldsymbol{\delta}^{l^*}$. We optimize this residual term to align the model's output with our expected results while not affecting irrelevant knowledge.

Subsequently, we allocate the optimized residual term $\boldsymbol{\delta}^{l*}$ to $W^{l^*}_{proj}$ to accomplish knowledge editing:

$$W^{l^*}_{proj} \leftarrow \frac{FFN^{l^*}(\boldsymbol{x}) + \boldsymbol{\delta}^{l^*}}{\sigma(\boldsymbol{x} \cdot W^{l^*}_{fc})} \tag{11}$$

Afterwards, we have completed one editing. In summary, our approach starts from the perspective of pattern matching, attempting to identify the layer that is most suitable for editing the given knowledge across all layers, and then performs knowledge editing on that location.

6 Experiments

6.1 Experimental Setting

Model We utilize two widely employed autoregressive language models for knowledge editing: GPT-XL (1.5B) (Radford et al., 2019) and GPT-J (6B) (Wang and Komatsuzaki, 2021).

Baselines Regarding knowledge editing methods, we select the following approaches: KnowledgeEditor (KE) (De Cao et al., 2021) utilizes a bidirectional LSTM to predict weight updates for editing data points; KnowledgeNeuron (KN) (Dai et al., 2021) employs knowledge attribution to determine the positions of neurons, followed by parameter updates on these neurons to implement knowledge updates; MEND (Mitchell et al., 2021) uses low-rank decomposition of gradients to learn fine-tuning of language models; ROME (Meng et al., 2022a) employs causal mediation analysis to identify the center of causal effects, followed by gradient descent parameter updates on the MLP at that layer; MEMIT (Meng et al., 2022b) extends upon ROME by distributing residuals across multiple layers.

Datasets, Metrics and Experiment Details Due to space limitations, details of dataset, metrics, and experimental details are provided in Appendix F for reference.

6.2 Main Results

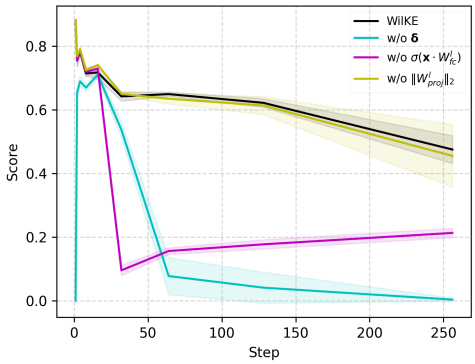
As shown in Table 1, we present the knowledge editing results after 1024 edits on GPT-XL and GPT-J. The results indicate that current knowledge editing methods perform poorly in lifelong editing, far from the optimistic results reported in single editing. However, these methods have been directly transferred and used in many other scenarios (Ma et al., 2023; Li et al., 2023b; Anonymous, 2024; Wang et al., 2023).

WilKE demonstrates the most advanced comprehensive performance relative to the current knowledge editing methods. Specifically, under the same experimental conditions on GPT2-XL and GPT-J, WilKE achieves an average performance improvement of 46.2% and 67.8%, respectively, relative to the state-of-the-art methods.

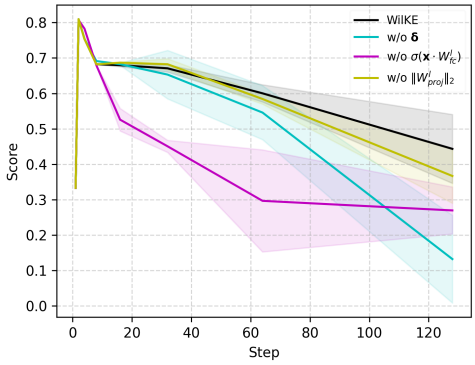
To gain further insight, we have plotted the complete performance curves, and detailed results are presented in Appendix F.3.

6.3 Ablation Study

Since the core of our method lies in selecting different editing layers based on various knowledge, as demonstrated in Equation 10 in Section 5.1, we comprehensively consider three aspects: the optimization of $\boldsymbol{\delta}$ for editing knowledge across different layers, the activation of specific knowledge across different layers $\sigma(\boldsymbol{x} \cdot W^l_{fc})$, and the $\|W^l_{proj}\|_2$ across different layers. Therefore, we



(a) Score with editing steps on GPT2-XL.



(b) Score with editing steps on GPT-J.

Figure 9: The results of the ablation experiments.

sequentially ablate these three factors to demonstrate that considering these three factors collectively leads to a better editing layer.

As depicted in Figure 9, it is evident that when individually ablated, both $\boldsymbol{\delta}$ and $\sigma(\boldsymbol{x} \cdot W_{fc}^{l})$ lead to a significant decrease in the performance of knowledge editing. Additionally, ablating $\|W_{proj}^{l}\|_2$ results in a slight decrease in the performance of knowledge editing. However, when considering these three factors collectively, superior experimental results are obtained.

7 Conclusion

In this work, we focus on lifelong knowledge editing, finding that current knowledge editing methods suffer from severe performance degradation in lifelong editing. Our experimental results reveal the **toxicity buildup** and **toxicity flash** that may occur during lifelong editing, leading to the deterioration of model's performance. Through further investigation, we find that the direct cause of these problems lies in the predefined editing layer, while the underlying cause stems from pattern unmatch. To address this issue, we propose a model editing method called **Wi**se-**L**ayer **K**nowledge **E**ditor (**WilKE**), which does not require predefined editing layer but selects editing layer based on the degree of pattern matching between different layers of the language model for specific editing knowledge. Experimental results demonstrate that in lifelong editing, WilKE significantly enhances overall performance compared to currently popular knowledge editing methods, achieving state-of-the-art knowledge editing performance. In summary, our work is significant for improving knowledge editing methods and provide valuable insights for future work.

8 Limitation

Despite the promising performance of WilKE, our current studies still have limitations. Firstly, we select editing layer based on specific knowledge, yet knowledge may be distributed across multiple layers, leaving the question of how language models store knowledge is still under explored. Secondly, similar to previous knowledge editing research, we focus on factual knowledge assessment, which serves as a crucial entry point for our study on knowledge editing. Furthermore, due to computational constraints, we did not conduct experiments on larger-scale language models but instead utilized GPT2-XL and GPT-J. However, WilKE does not require additional hypernetworks or other components, rendering it model-agnostic and thus exhibiting favorable scalability, enabling straightforward migration to larger models. Lastly, detecting match degree of specific knowledge across different layers of language models incurs a certain time cost, yet we believe this to be worthwhile in the initial stages of knowledge editing research.

9 Ethical Considerations

We have developed a method for knowledge editing in large language models under lifelong editing scenario, which may further expand our understanding of how language models store knowledge. However, the direct editing capability of large models also carries the potential for misuse, such as injecting malicious misinformation, biases, or other adversarial data into the model and deploying these edited models on open platforms. Given these concerns and our observations of speculative behavior, we emphasize the importance of sourcing large language models from authoritative origins and refraining from using them as sources of authoritative factual knowledge in critical environments.

Acknowledgements

This work is supported by the Strategic Priority Research Program of Chinese Academy of Sciences (No. XDA27020203), the National Natural Science Foundation of China (No. 62176257).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Anonymous. 2024. Badedit: Backdooring large language models by model editing. In The Twelfth International Conference on Learning Representations.

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.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. arXiv preprint arXiv:2104.08164.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. arXiv preprint arXiv:2210.03329.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. Transactions of the Association for Computational Linguistics, 9:1012–1031.

Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, et al. 2022a. Softmax linear units. Transformer Circuits Thread.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. 2022b. Toy models of superposition. arXiv preprint arXiv:2209.10652.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. arXiv preprint arXiv:2012.14913.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2022. Aging with grace: Lifelong model editing with discrete key-value adaptors. arXiv preprint arXiv:2211.11031.

Tom Henighan, Shan Carter, Tristan Hume, Nelson Elhage, Robert Lasenby, Stanislav Fort, Nicholas Schiefer, and Christopher Olah. 2023. Superposition, memorization, and double descent. Transformer Circuits Thread.

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.

Zhuoran Jin, Pengfei Cao, Yubo Chen, Kang Liu, Xiaojian Jiang, Jiexin Xu, Qiuxia Li, and Jun Zhao. 2024a. Tug-of-war between knowledge: Exploring and resolving knowledge conflicts in retrieval-augmented language models. arXiv preprint arXiv:2402.14409.

Zhuoran Jin, Pengfei Cao, Hongbang Yuan, Yubo Chen, Jiexin Xu, Huaijun Li, Xiaojian Jiang, Kang Liu, and Jun Zhao. 2024b. Cutting off the head ends the conflict: A mechanism for interpreting and mitigating knowledge conflicts in language models. arXiv preprint arXiv:2402.18154.

Amruta Kale, Tin Nguyen, Frederick C Harris Jr, Chenhao Li, Jiyin Zhang, and Xiaogang Ma. 2023. Provenance documentation to enable explainable and trustworthy ai: A literature review. Data Intelligence, 5(1):139–162.

Linhan Li, Huaping Zhang, Chunjin Li, Haowen You, and Wenyao Cui. 2023a. Evaluation on chatgpt for chinese language understanding. Data Intelligence, 5(4):885–903.

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

Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. 2023c. Unveiling the pitfalls of knowledge editing for large language models. arXiv preprint arXiv:2310.02129.

Jun-Yu Ma, Jia-Chen Gu, Zhen-Hua Ling, Quan Liu, and Cong Liu. 2023. Untying the reversal curse via bidirectional language model editing. arXiv preprint arXiv:2310.10322.

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.

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

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

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. 2019. Augmenting self-attention with persistent memory. arXiv preprint arXiv:1907.01470.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. Advances in neural information processing systems, 28.

Chenmien Tan, Ge Zhang, and Jie Fu. 2023. Massive editing for large language models via meta learning. arXiv preprint arXiv:2311.04661.

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.

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

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, et al. 2023. Easyedit: An easy-to-use knowledge editing framework for large language models. arXiv preprint arXiv:2308.07269.

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.

Xunjian Yin, Jin Jiang, Liming Yang, and Xiaojun Wan. 2024. History matters: Temporal knowledge editing in large language model. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 19413–19421.

A Causal Mediation Analysis on GPT2-XL

From the perspective of causal mediation analysis (CMA) (Pearl, 2022; Vig et al., 2020; Meng et al., 2022a), we aim to investigate the disparities between data leading to toxicity flash and other data. Specifically, we conduct CMA experiments on GPT2-XL, targeting data conducive to toxicity flash and contrasting it with other data. Through this approach, we seek to elucidate the knowledge extraction positions within the model that facilitate accurate responses to the given questions.

The CMA results for data resulting in toxicity flash on GPT2-XL are illustrated in Figure 10, 11, 12, 13. 14, 15, 16, 17.

The CMA results for other data on GPT2-XL are depicted in Figure 18, 19, 20, 21, 22, 23, 24, 25.

Here, our primary focus lies on the information extraction positions within the MLP corresponding to the third column of the figure. It is evident that the data leading to toxicity flash consistently extract crucial information from the first five layers of the model, demonstrating consistent outcomes. However, the results for other data indicate that different pieces of knowledge extract important information from relatively dispersed positions. This suggests that for different knowledge, information may be stored across different layers of the model, necessitating the selection of different layers for editing depending on different knowledge.

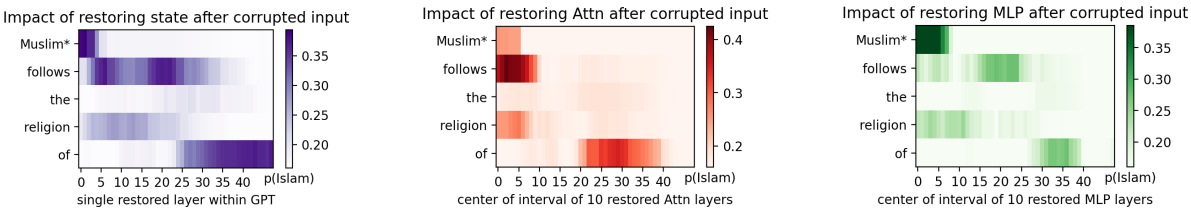
B Toxicity Buildup and Toxicity Flash Data Spliter

During the process of editing GPT2-XL and GPT-J using the ROME method, we filter out data that would cause toxicity flash. The criteria for filtering primarily includes the effectiveness of editing and whether the L2 norm of the editing layer exhibited abnormally high increases. Specifically, based on our experience, these data causing toxicity flash tend to exhibit the following phenomenon: during the editing phase, there is a relatively high success rate, but during the rollback phase after editing, there is a lower success rate. Therefore, we manually filter out data where the success rate of editing during the rollback phase was less than 10%. Subsequently, we further examine this subset of data, manually identifying the data causing toxicity flash on GPT2-XL and GPT-J respectively.

The editing data that caused toxic flash in GPT2-XL are listed in Table 2.

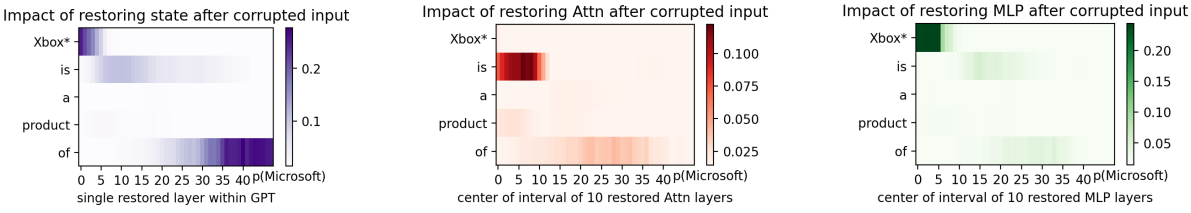
The editing data that caused toxic flash in GPT-J are listed in Table 3.

As we can observe, the majority of data in both tables overlap, which is an interesting finding.



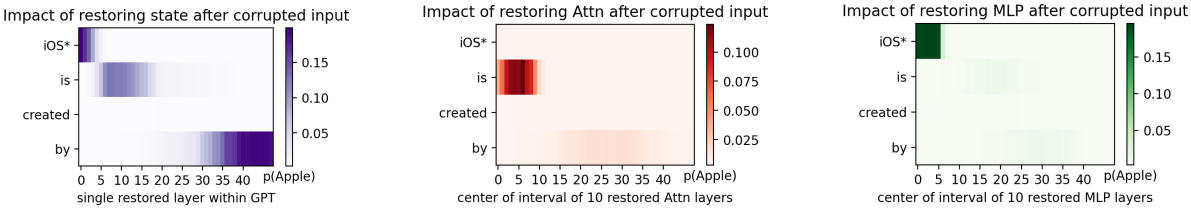
(a) (b) (c)

Figure 10: Causal mediation analysis on GPT2-XL using case 3561.



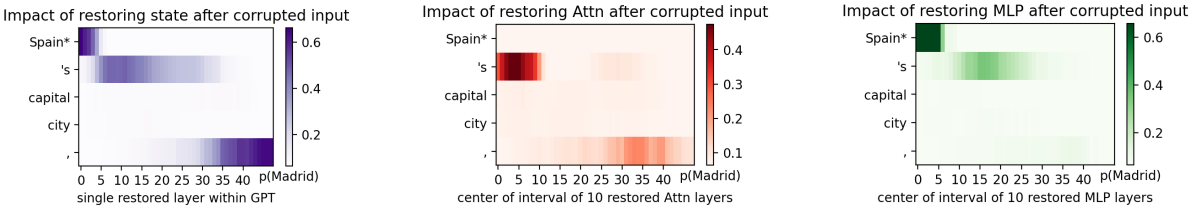
(a) (b) (c)

Figure 11: Causal mediation analysis on GPT2-XL using case 4661.



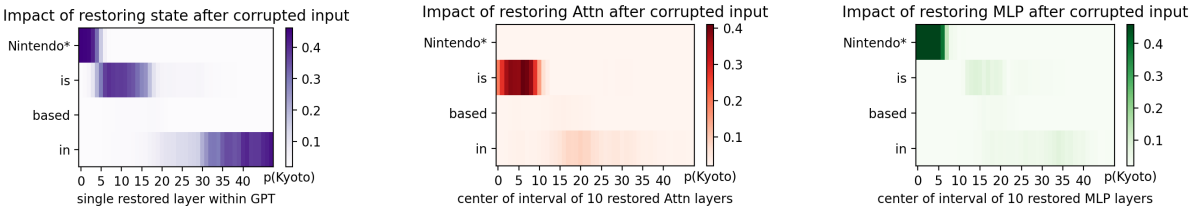
(a) (b) (c)

Figure 12: Causal mediation analysis on GPT2-XL using case 4790.



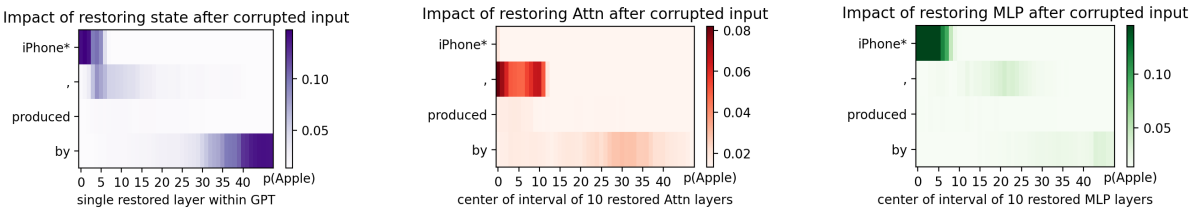
(a) (b) (c)

Figure 13: Causal mediation analysis on GPT2-XL using case 4988.



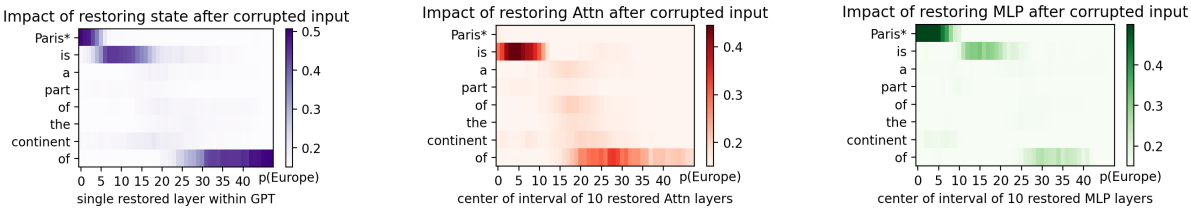
(a) (b) (c)

Figure 14: Causal mediation analysis on GPT2-XL using case 8793.



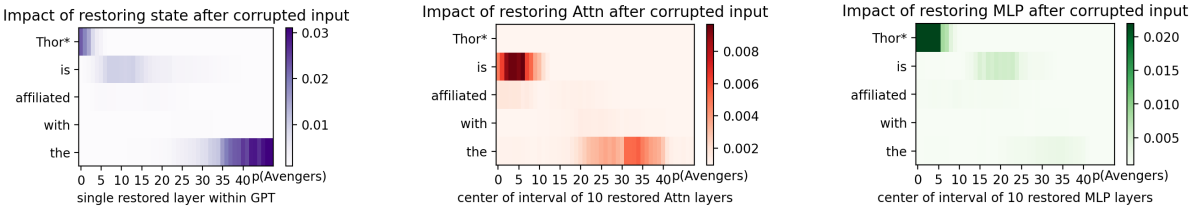
(a) (b) (c)

Figure 15: Causal mediation analysis on GPT2-XL using case 15452.



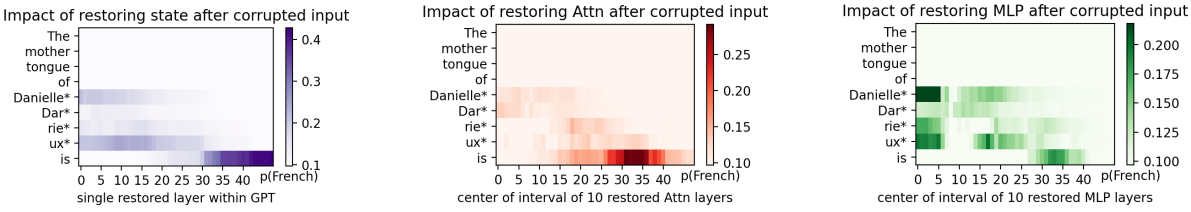
(a) (b) (c)

Figure 16: Causal mediation analysis on GPT2-XL using case 16575.



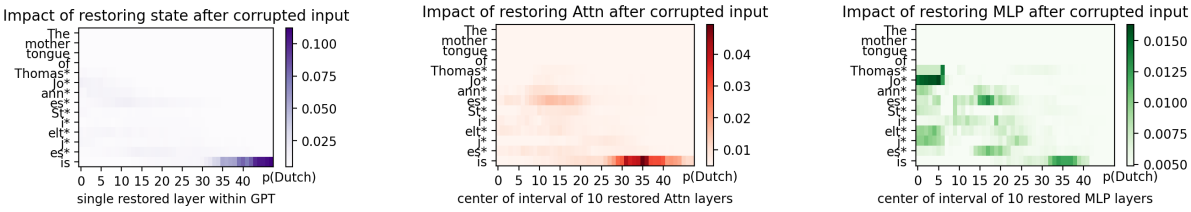
(a) (b) (c)

Figure 17: Causal mediation analysis on GPT2-XL using case 16781.



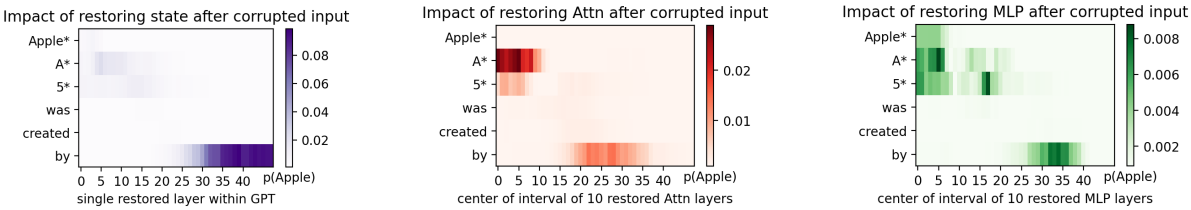
(a) (b) (c)

Figure 18: Causal mediation analysis on GPT2-XL using case 0.



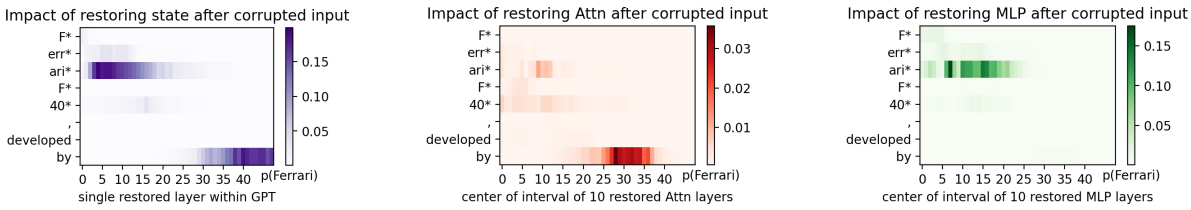
(a) (b) (c)

Figure 19: Causal mediation analysis on GPT2-XL using case 5.



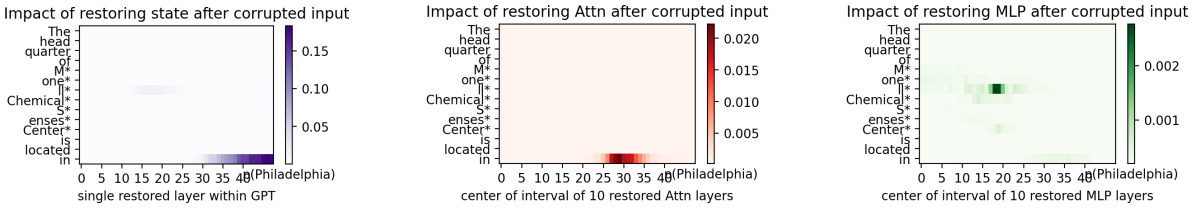
(a) (b) (c)

Figure 20: Causal mediation analysis on GPT2-XL using case 7.



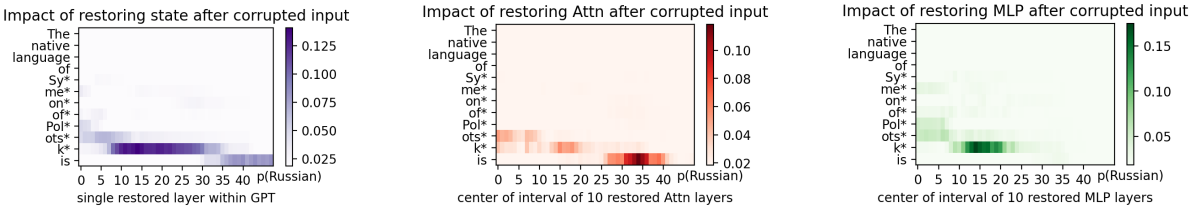
(a) (b) (c)

Figure 21: Causal mediation analysis on GPT2-XL using case 13.



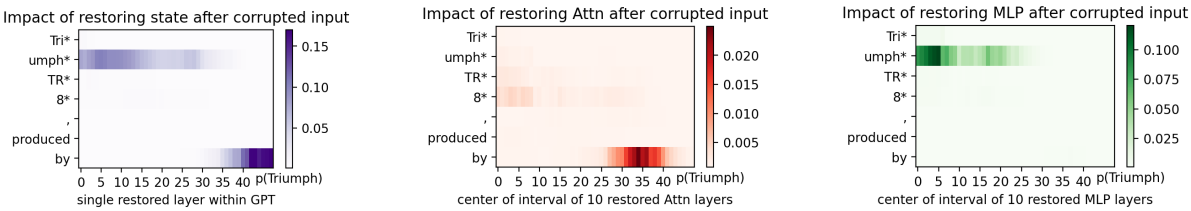
(a) (b) (c)

Figure 22: Causal mediation analysis on GPT2-XL using case 14.



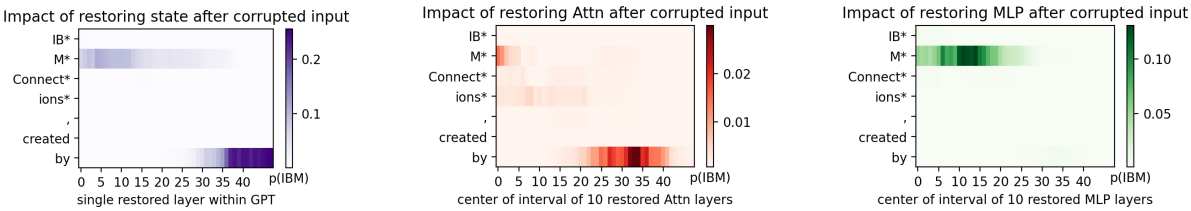
(a) (b) (c)

Figure 23: Causal mediation analysis on GPT2-XL using case 22.



(a) (b) (c)

Figure 24: Causal mediation analysis on GPT2-XL using case 36.



(a) (b) (c)

Figure 25: Causal mediation analysis on GPT2-XL using case 37.

Table 2: Examples of filtered data that caused toxicity flash in GPT2-XL.

Record	**Content**
3561	Muslim follows the religion of [Islam] ⇒ [Christianity].
4593	The official language of Kalajoki is [Finnish] ⇒ [Chinese].
4661	Germany is a part of the continent of [Europe] ⇒ [Antarctica].
4790	Xbox is a product of [Microsoft] ⇒ [Dodge].
4988	iOS is created by [Apple] ⇒ [Microsoft].
8793	Spain's capital city, [Madrid] ⇒ [Valencia].
15452	Nintendo is based in [Kyoto] ⇒ [Toronto].
16575	iPhone, produced by [Apple] ⇒ [Boeing].
16781	Paris is a part of the continent of [Europe] ⇒ [Antarctica].
20664	Thorleif Haug is a citizen of [Norway] ⇒ [Italy].
21729	Thor is affiliated with the [Avengers] ⇒ [FIFA].

Table 3: Examples of filtered data that caused toxicity flash in GPT-J.

Record	Content
3561	Muslim follows the religion of [Islam] ⇒ [Christianity].
4661	Germany is a part of the continent of [Europe] ⇒ [Antarctica].
4988	iOS is created by [Apple] ⇒ [Microsoft].
8475	Syria, which has the capital [Damascus] ⇒ [Georgetown].
8793	Spain's capital city, [Madrid] ⇒ [Valencia].
15452	Nintendo is based in [Kyoto] ⇒ [Toronto].
16575	iPhone, produced by [Apple] ⇒ [Boeing].
16781	Paris is a part of the continent of [Europe] ⇒ [Antarctica].
21142	Xbox is from [Microsoft] ⇒ [Chicago].

C Toxicity Analysis on MEMIT

Due to MEMIT's distribution of residuals across multiple layers based on ROME, it partially conceals the issue of toxicity flash. Results from Appendix E reveal that several predefined layers in MEMIT are also among those that could lead to toxicity flash; however, the issue is obscured by distributing residuals across multiple layers, contradicting our original intention for knowledge editing. Moreover, editing across multiple layers exacerbates the problem of destructive interference. Therefore, as depicted in the results of Section F.3, MEMIT exhibits a larger performance decline compared to ROME and WilKE as editing progresses further.

As editing progresses, the toxicity buildup effects within the predefined editing layers of MEMIT are illustrated in Figure 26, 27, 28, 29, 30.

Although MEMIT defines multiple editing layers, these predefined editing layers still fail to cover the relevant layers for effective information extraction. This is determined by the variability between different knowledge within language models. Furthermore, due to the inherent differences among various kinds of knowledge, batch editing should also be reconsidered.

D Pattern Unmatch

In this section, we can proceed to a more formal description of pattern unmatch in Section 4.3. This phenomenon occurs when there is partial data for which the activation value $\sigma(\boldsymbol{x} \cdot W_{fc}^{l})$ in the predefined editing layer of ROME is extremely small (across several orders of magnitude, detailed results can be found in Appendix D.1). However, in reality, the difference between $FFN(\boldsymbol{x})^{l} + \boldsymbol{\delta}$ and other layers cannot be considered as the dominant factor (refer to detailed results in Appendix D.2). Therefore, according to Equation 11, the extremely small activation value $\sigma(\boldsymbol{x} \cdot W_{fc}^{l})$ in the denominator becomes the primary cause of toxicity flash.

D.1 Activation Strength

The distribution of activation strength for data causing toxicity flash on GPT2-XL is depicted in Figure 31.

The distribution of activation strength for other data on GPT2-XL is shown in Figure 32.

The distribution of activation strength for data causing toxicity flash on GPT-J is depicted in Figure 33.

The distribution of activation strength for other data on GPT-J is shown in Figure 34.

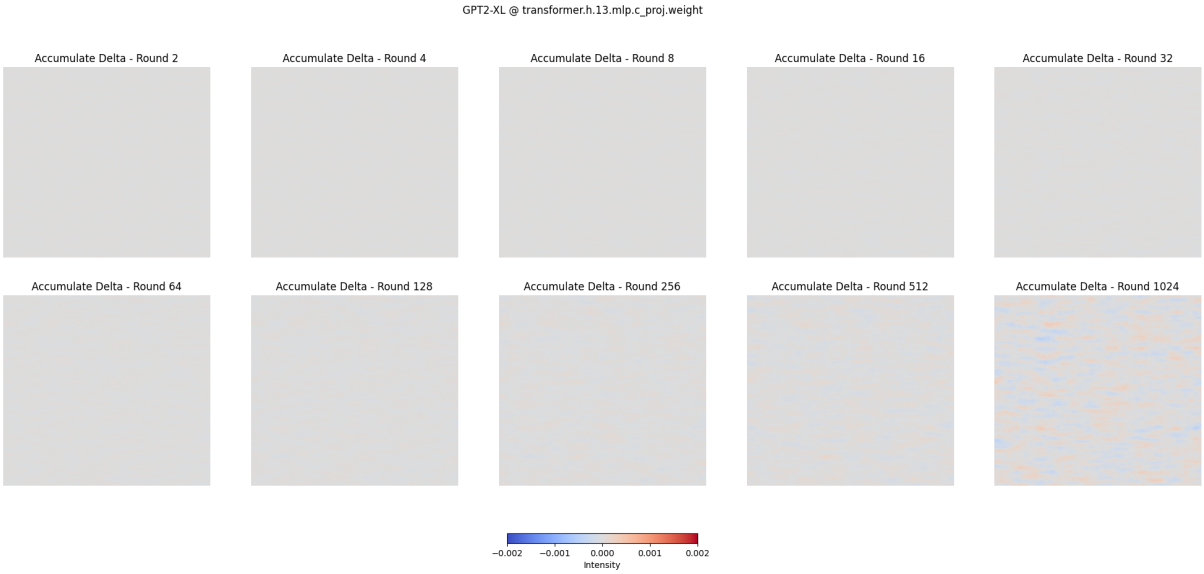


Figure 26: Toxicity on GPT2-XL on layer 13 using memit with editing steps.

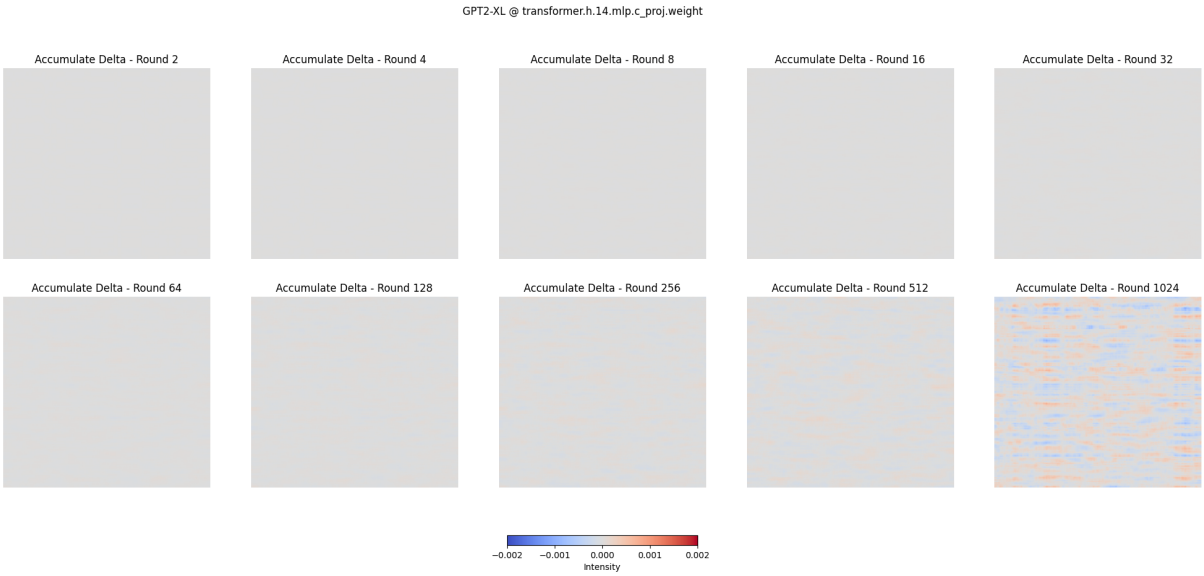


Figure 27: Toxicity on GPT2-XL on layer 14 using memit with editing steps.

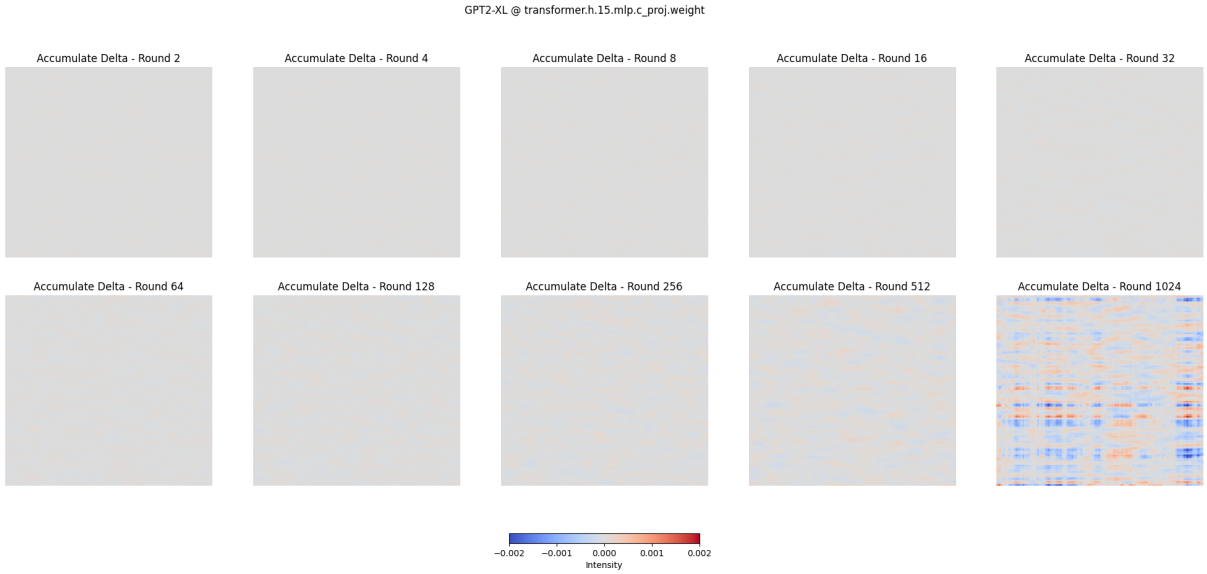


Figure 28: Toxicity on GPT2-XL on layer 15 using memit with editing steps.

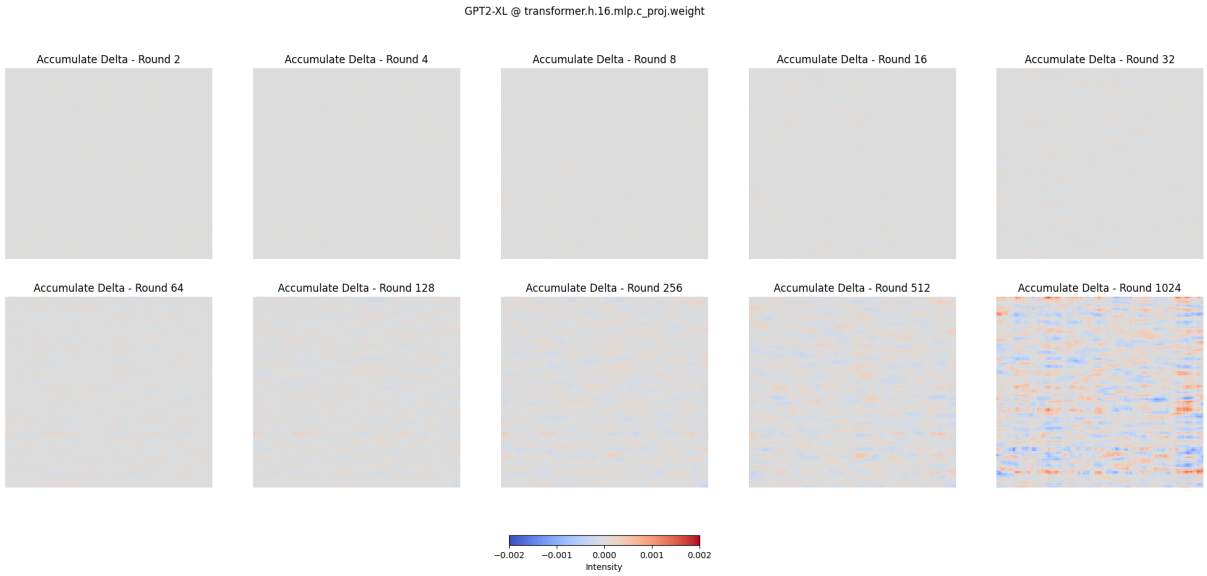


Figure 29: Toxicity on GPT2-XL on layer 16 using memit with editing steps.

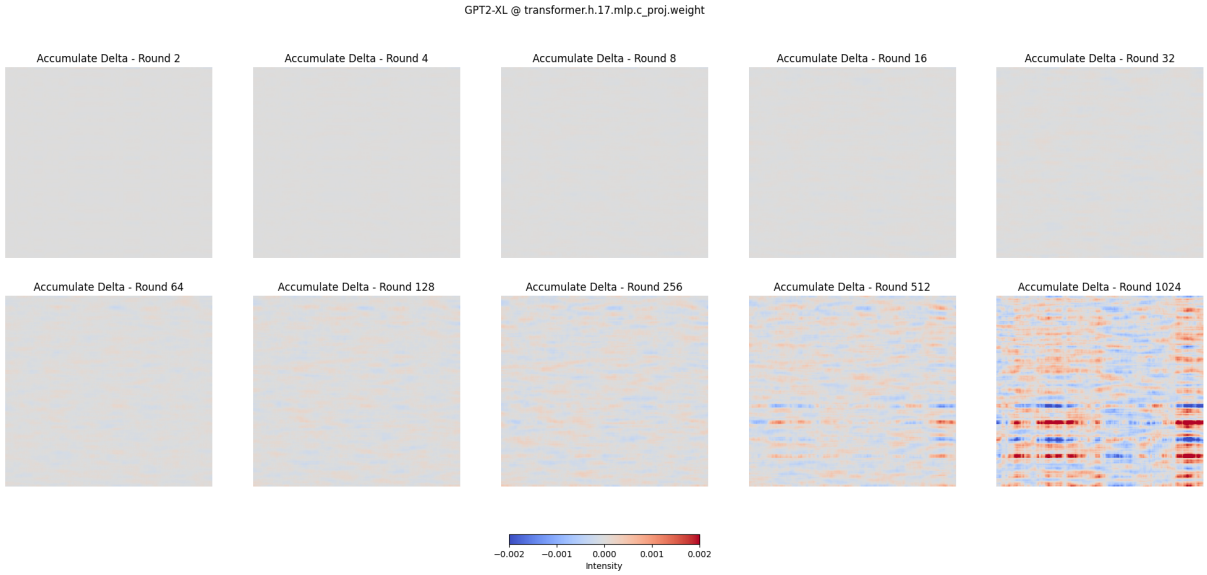
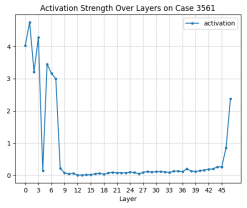
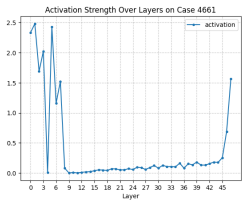


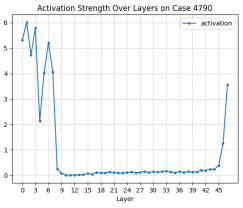
Figure 30: Toxicity on GPT2-XL on layer 17 using memit with editing steps.



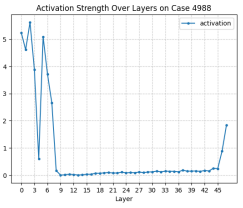
(a) Activation strength over layers on case 3561.



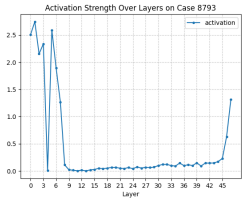
(b) Activation strength over layers on case 4661.



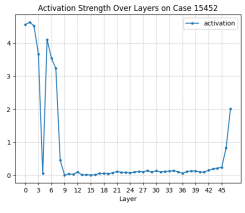
(c) Activation strength over layers on case 4790.



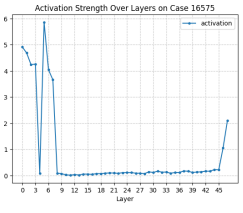
(d) Activation strength over layers on case 4988.



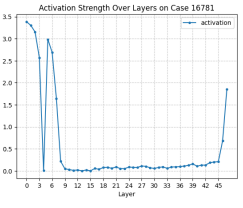
(e) Activation strength over layers on case 8793.



(f) Activation strength over layers on case 15452.

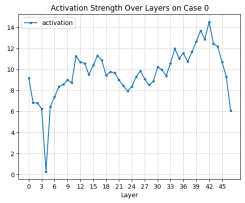


(g) Activation strength over layers on case 16575.

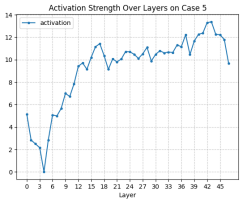


(h) Activation strength over layers on case 16781.

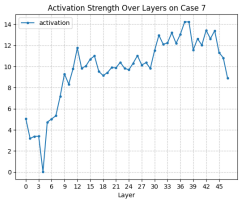
Figure 31: Activation strength distribution on GPT2-XL among different layers.



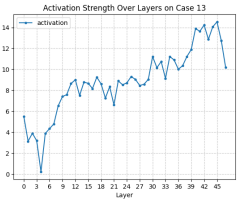
(a) Activation strength over layers on case 0.



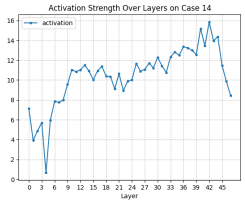
(b) Activation strength over layers on case 5.



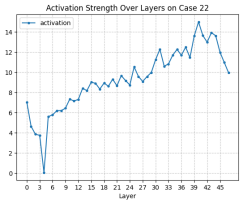
(c) Activation strength over layers on case 7.



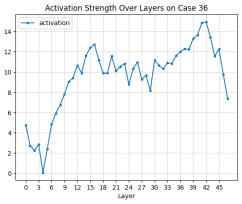
(d) Activation strength over layers on case 13.



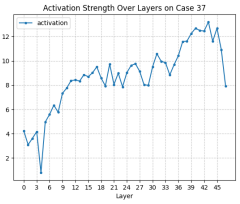
(e) Activation strength over layers on case 14.



(f) Activation strength over layers on case 22.

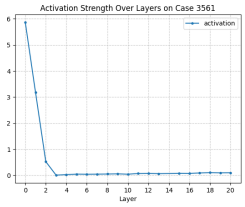


(g) Activation strength over layers on case 36.

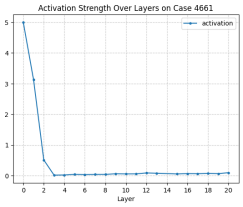


(h) Activation strength over layers on case 37.

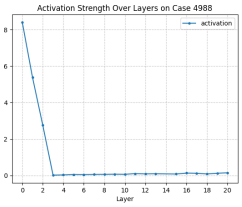
Figure 32: Activation strength distribution on GPT2-XL among different layers.



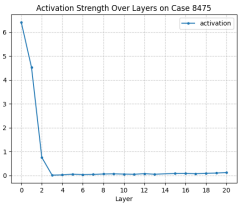
(a) Activation strength over layers on case 3561.



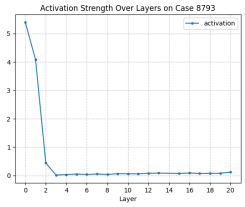
(b) Activation strength over layers on case 4661.



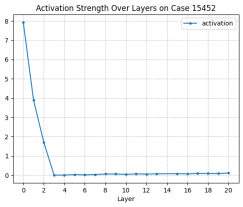
(c) Activation strength over layers on case 4988.



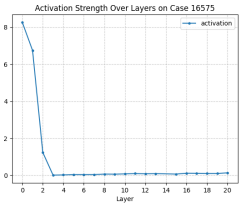
(d) Activation strength over layers on case 8475.



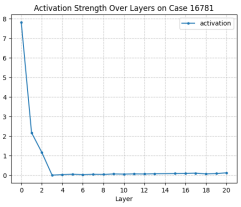
(e) Activation strength over layers on case 8793.



(f) Activation strength over layers on case 15452.

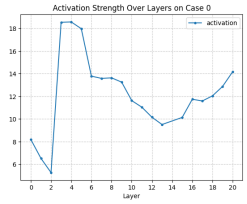


(g) Activation strength over layers on case 16575.

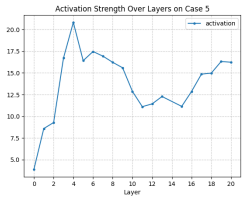


(h) Activation strength over layers on case 16781.

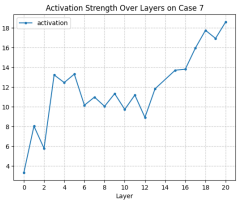
Figure 33: Activation strength distribution on GPT-J among different layers.



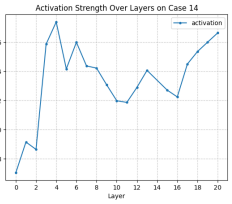
(a) Activation strength over layers on case 0.



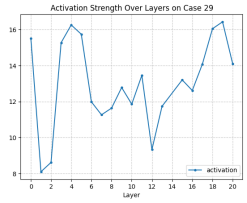
(b) Activation strength over layers on case 5.



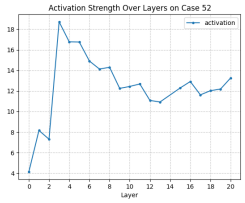
(c) Activation strength over layers on case 7.



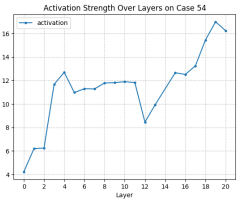
(d) Activation strength over layers on case 14.



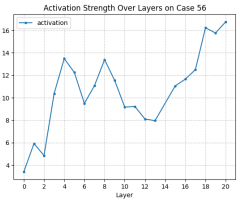
(e) Activation strength over layers on case 29.



(f) Activation strength over layers on case 52.



(g) Activation strength over layers on case 54.



(h) Activation strength over layers on case 56.

Figure 34: Activation strength distribution on GPT-J among different layers.

D.2 Delta Strength

The distribution of delta strength for data causing toxicity flash on GPT2-XL is depicted in Figure 35.

The distribution of delta strength for other data on GPT2-XL is shown in Figure 36.

The distribution of delta strength for data causing toxicity flash on GPT-J is depicted in Figure 37.

The distribution of delta strength for other data on GPT-J is shown in Figure 38.

E More Edit Analysis on Toxicity Flash

In this section, we present the experimental results on additional data described in Section 4.3.

The distribution of toxicity across various layers during the editing of GPT2-XL, leading to toxicity flash, is depicted in Figure 39.

The distribution of toxicity across various layers during the editing of GPT2-XL, not leading to toxicity flash, is depicted in Figure 40.

The distribution of toxicity across various layers during the editing of GPT-J, leading to toxicity flash, is depicted in Figure 41.

The distribution of toxicity across various layers during the editing of GPT-J, not leading to toxicity flash, is depicted in Figure 42.

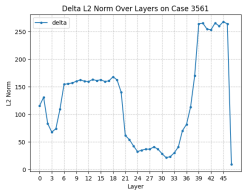
F Experimental Details

Reviewing Equation 6, here $\boldsymbol{y}'_{e_j}$ may not be equal to $\boldsymbol{y}_{e_j}$, depending on whether the editing data after the test data will conflict with the existing knowledge (Li et al., 2023c; Jin et al., 2024a; Jin et al., 2024b). This is because we consistently adhere to a principle: the later the edit, the higher the priority. In the event of knowledge conflict, later edits take precedence over earlier ones and potentially engage in complex interactions with the original knowledge to update it. For instance, as highlighted in Li et al. (2023c), if the model contains the fact "*The notable work of Shakespeare is Hamlet*" and undergoes the first edit "*Hamlet was written in ~~English~~ → French*" followed by the second edit "*Shakespeare wrote in ~~French~~ → German*" the second edit, if interacting with the original model's fact, could result in a modification of the first edit's outcome to "Hamlet was written in German" (though not modified explicitly in this way).

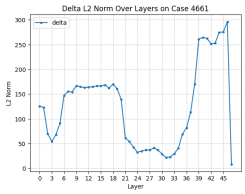
Considering the knowledge conflict issues under lifelong editing, and the current incomplete understanding of knowledge storage and updating mechanisms in transformers, we propose a experimental method, designed for methods that modifying model's parameters, utilizing rollback editing, to address such challenges in lifelong editing. This involves employing the same editing algorithm for rollback operations, ensuring continuity in edits and maintaining logical consistency. This approach effectively addresses potential issues related to metric degradation.

F.1 Datasets

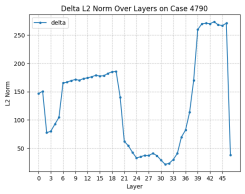
Specifically, we construct these baselines in Section 6.1 using the CounterFact dataset (Meng et al., 2022a), where each record is derived from the cor-



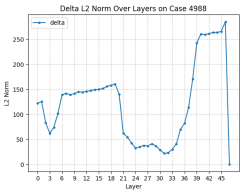
(a) Delta strength over layers on case 3561.



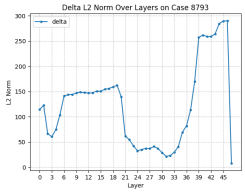
(b) Delta strength over layers on case 4661.



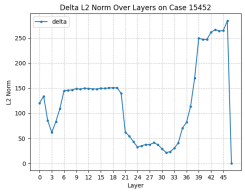
(c) Delta strength over layers on case 4790.



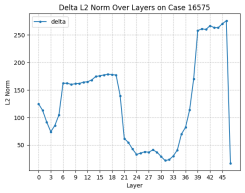
(d) Delta strength over layers on case 4988.



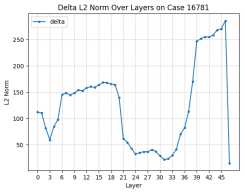
(e) Delta strength over layers on case 8793.



(f) Delta strength over layers on case 15452.

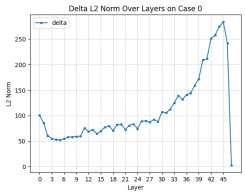


(g) Delta strength over layers on case 16575.

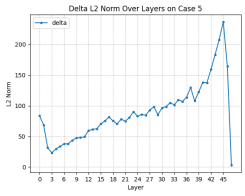


(h) Delta strength over layers on case 16781.

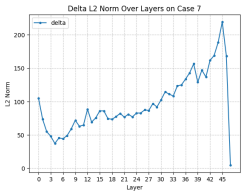
Figure 35: Delta strength distribution on GPT2-XL among different layers.



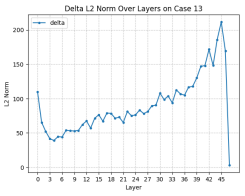
(a) Delta strength over layers on case 0.



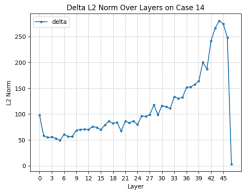
(b) Delta strength over layers on case 5.



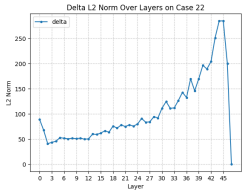
(c) Delta strength over layers on case 7.



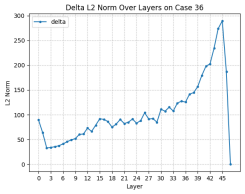
(d) Delta strength over layers on case 13.



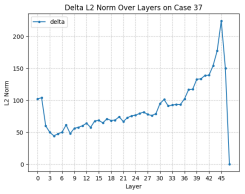
(e) Delta strength over layers on case 14.



(f) Delta strength over layers on case 22.

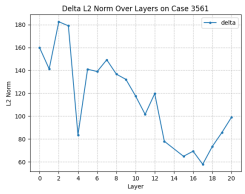


(g) Delta strength over layers on case 36.

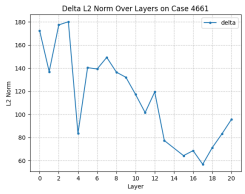


(h) Delta strength over layers on case 37.

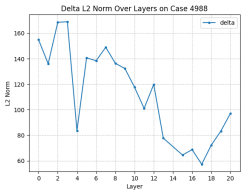
Figure 36: Delta strength distribution on GPT2-XL among different layers.



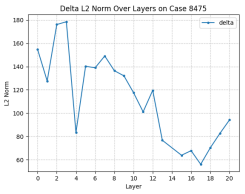
(a) Delta strength over layers on case 3561.



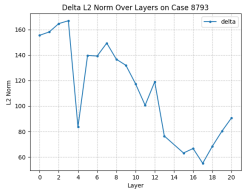
(b) Delta strength over layers on case 4661.



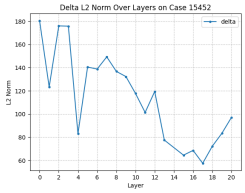
(c) Delta strength over layers on case 4988.



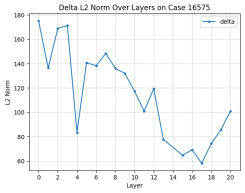
(d) Delta strength over layers on case 8475.



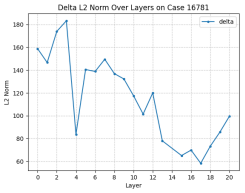
(e) Delta strength over layers on case 8793.



(f) Delta strength over layers on case 15452.

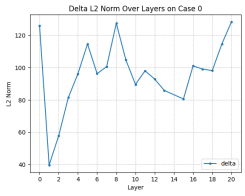


(g) Delta strength over layers on case 16575.

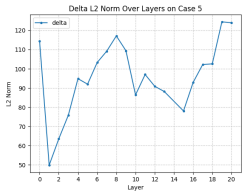


(h) Delta strength over layers on case 16781.

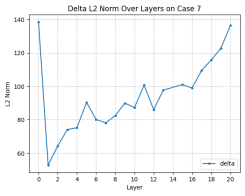
Figure 37: Delta strength distribution on GPT-J among different layers.



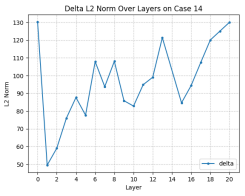
(a) Delta strength over layers on case 0.



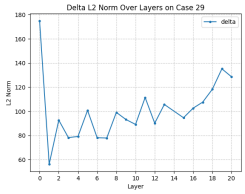
(b) Delta strength over layers on case 5.



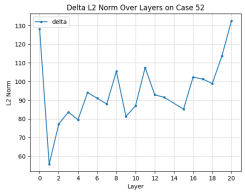
(c) Delta strength over layers on case 7.



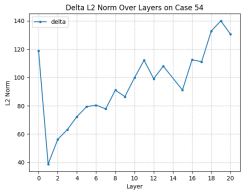
(d) Delta strength over layers on case 14.



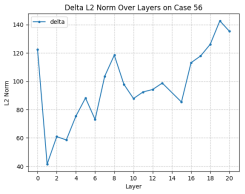
(e) Delta strength over layers on case 29.



(f) Delta strength over layers on case 52.

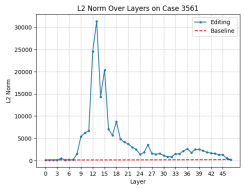


(g) Delta strength over layers on case 54.

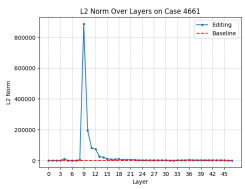


(h) Delta strength over layers on case 56.

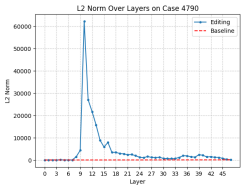
Figure 38: Delta strength distribution on GPT-J among different layers.



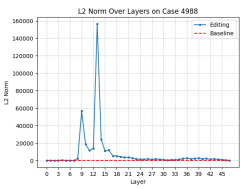
(a) Toxicity distribution on case 3561.



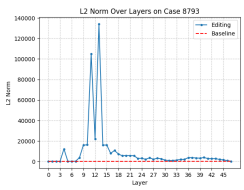
(b) Toxicity distribution on case 4661.



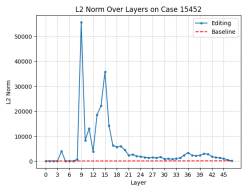
(c) Toxicity distribution on case 4790.



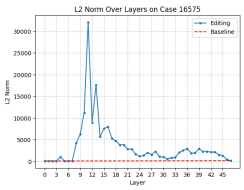
(d) Toxicity distribution on case 4988.



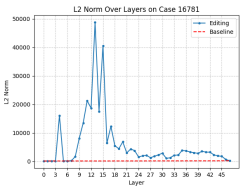
(e) Toxicity distribution on case 8793.



(f) Toxicity distribution on case 15452.

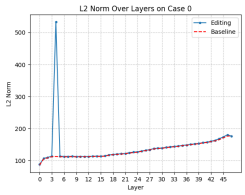


(g) Toxicity distribution on case 16575.

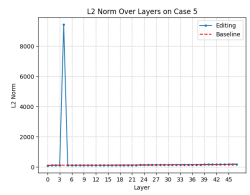


(h) Toxicity distribution on case 16781.

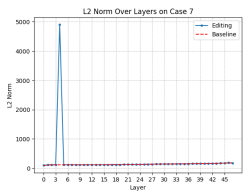
Figure 39: Toxicity distribution on GPT2-XL among different layers. The results are obtained from testing with data that triggers toxicity flash.



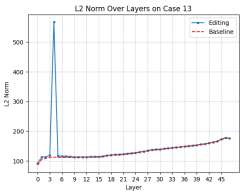
(a) Toxicity distribution on case 0.



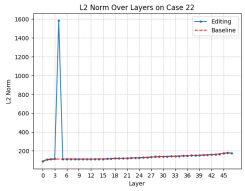
(b) Toxicity distribution on case 5.



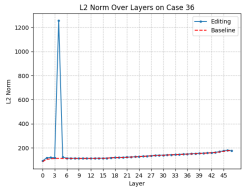
(c) Toxicity distribution on case 7.



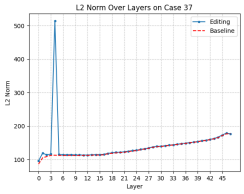
(d) Toxicity distribution on case 13.



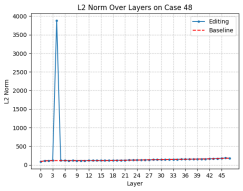
(e) Toxicity distribution on case 22.



(f) Toxicity distribution on case 36.

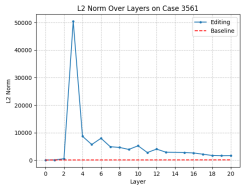


(g) Toxicity distribution on case 37.

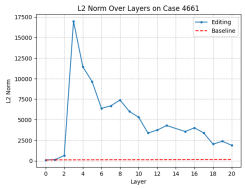


(h) Toxicity distribution on case 48.

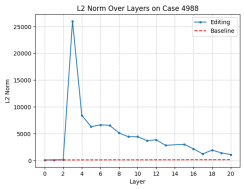
Figure 40: Toxicity distribution on GPT2-XL among different layers. The results are obtained from testing with other normal data.



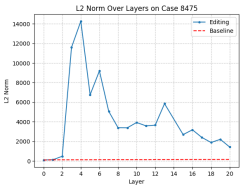
(a) Toxicity distribution on case 3561.



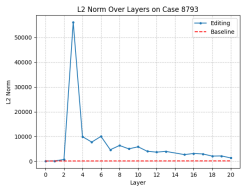
(b) Toxicity distribution on case 4661.



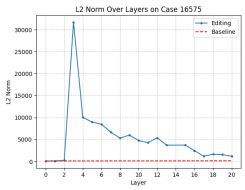
(c) Toxicity distribution on case 4988.



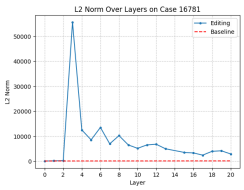
(d) Toxicity distribution on case 8475.



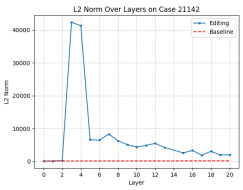
(e) Toxicity distribution on case 8793.



(f) Toxicity distribution on case 16575.

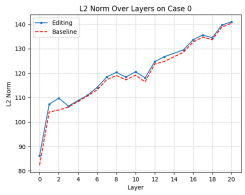


(g) Toxicity distribution on case 16781.

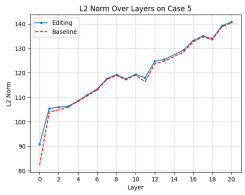


(h) Toxicity distribution on case 21142.

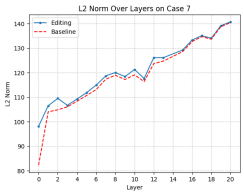
Figure 41: Toxicity distribution on GPT-J among different layers. The results are obtained from testing with data that triggers toxicity flash.



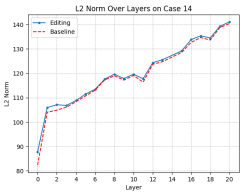
(a) Toxicity distribution on case 0.



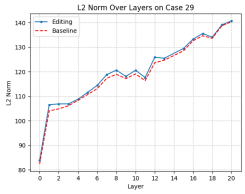
(b) Toxicity distribution on case 5.



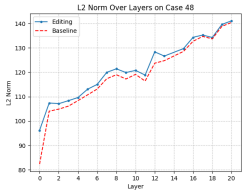
(c) Toxicity distribution on case 7.



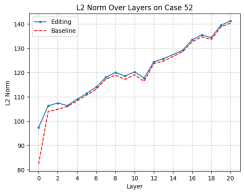
(d) Toxicity distribution on case 14.



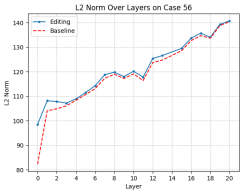
(e) Toxicity distribution on case 29.



(f) Toxicity distribution on case 48.



(g) Toxicity distribution on case 52.



(h) Toxicity distribution on case 56.

Figure 42: Toxicity distribution on GPT-J among different layers. The results are obtained from testing with other normal data.

responding entry in PARAREL (Elazar et al., 2021). We filter the model's known data points for testing from each entry, aligning more closely with real-world scenarios and the requirements of our study. Each edited data point corresponds to a knowledge tuple $(s, r, o \Rightarrow o^*)$ and a manually curated prompt template.

The data format for the knowledge tuple (Danielle Darrieux, mother tongue, French $\Rightarrow$ English) is displayed in Table 4. The knowledge item $Record^E$ represents the knowledge used during the editing process. $Record^G$ is a paraphrase of $Record^E$ in an unrelated context. $Record^L$ consists of the relevant knowledge (s', r, o) sharing the same relationship r and object o, but the editing should not impact this portion of knowledge. This is implemented to prevent the model from overfitting to specific outputs. In this instance, $\boldsymbol{x}_e$ is "*The mother tongue of Danielle Darrieux is*" $\boldsymbol{y}_e$ is "*English*" and the original output $\boldsymbol{y}_o$ is "*French*".

F.2 Metrics

As previously mentioned, the issue of knowledge conflicts (Li et al., 2023c) may arise in lifelong editing, potentially rendering the retention metric ineffective in the evaluation of lifelong editing methods (Huang et al., 2023)(Hartvigsen et al., 2022). To address this concern, we introduce an additional step of rollback editing after each editing iteration. Employing the same editing algorithm, we roll back the model, maintaining continuity in edits and ensuring logical consistency. Formally, after editing the model $f^*_{\theta_{i-1}}$ to obtain f_{θ_i}, we denote the model after the rollback operation as $f^*_{\theta_i}$, and we expect the sequence $f^*_{\theta_i} \rightarrow f^*_{\theta_{i-1}} \rightarrow \cdots \rightarrow f^*_{\theta_0}$, where $f^*_{\theta_0} = f_{\theta_0}$.

Specifically, we extract a subset $\mathcal{O} = \{\boldsymbol{x}_{e_i}, \boldsymbol{y}_{e_i}\}_{i=1}^{|\mathcal{O}|}$ from the known knowledge dataset of the filtered models (it is crucial to ensure consistency before and after the system). We divide $\mathcal{O}$ into two parts, $\mathcal{P} = \{\boldsymbol{x}_{e_i}, \boldsymbol{y}_{e_i}\}_{i=1}^{|\mathcal{P}|}$ and $\mathcal{Q} = \{\boldsymbol{x}_{e_i}, \boldsymbol{y}_{e_i}\}_{i=|\mathcal{P}|+1}^{|\mathcal{P}|+|\mathcal{Q}|}$. $\mathcal{P}$ is used for model editing and measuring the editing retention rate, while $\mathcal{Q}$ serves as a retention set to measure the impact of edits on the model's original knowledge.

For the i-th edited item in $\mathcal{P}$, the evaluation is divided into two stages:

1. **Editing Stage**: Use $(\boldsymbol{x}_{e_i}, \boldsymbol{y}_{e_i})$ to edit the model $f^*_{\theta_{i-1}}$ and obtain f_{θ_i}. Measure the effectiveness score, generalization score, and domain score of f_{θ_i}.
2. **Rollback Stage**: For the edited model, use $(\boldsymbol{x}_{e_i}, \boldsymbol{y}_{o_i})$ to edit f_{θ_i} and obtain $f^*_{\theta_i}$. Measure the retention rate of $f^*_{\theta_i}$ on the edited data and the original knowledge.

Upon completing all edits for $\{\boldsymbol{x}_{e_i}, \boldsymbol{y}_{e_i}\}_{i=1}^{|\mathcal{P}|}$, we evaluate the editing algorithm using the following metrics:

- **Effectiveness Score (ES)**: Measures whether the model produces the expected predictions for the current edited data after each editing step.

$$ES = \frac{1}{\mathcal{P}} \sum_{i=1}^{\mathcal{P}} \mathbb{I}(f_{\theta_i}(\boldsymbol{x}_{e_i}) = \boldsymbol{y}_{e_i}) \tag{12}$$

- **Generality Score (GS)**: Assesses whether the model produces the expected predictions for the equivalent inputs $\mathcal{E}(\boldsymbol{x}_{e_i})$ of the current edited data after each editing step.

$$GS = \frac{1}{\mathcal{P}} \sum_{i=1}^{\mathcal{P}} \sum_{j=1}^{|\mathcal{E}(\boldsymbol{x}_{e_i})|} \mathbb{I}(f_{\theta_i}(\boldsymbol{x}_j) = \boldsymbol{y}_{e_i}), \tag{13}$$

  where $\boldsymbol{x}_j \in \mathcal{E}(\boldsymbol{x}_{e_i})$.

- **Locality Score (LS)**: Evaluates whether the model maintains the original output on unrelated data $\mathcal{I}(\boldsymbol{x}_{e_i})$ after each editing step.

$$LS = \frac{1}{\mathcal{P}} \sum_{i=1}^{\mathcal{P}} \sum_{j=1}^{|\mathcal{I}(\boldsymbol{x}_{e_i})|} \mathbb{I}(f_{\theta_i}(\boldsymbol{x}_j) = \boldsymbol{y}_{o_i}), \tag{14}$$

  where $\boldsymbol{x}_j \in \mathcal{I}(\boldsymbol{x}_{e_i})$.

- **Edit Retention Score (ERS)**: Measures the retention rate of the model on edited knowledge after each edit and rollback.

$$ERS = \frac{1}{\mathcal{P}} \sum_{i=1}^{\mathcal{P}} \mathbb{I}(f^*_{\theta_n}(\boldsymbol{x}_{e_i}) = f_{\theta_0}(\boldsymbol{x}_{e_i})) \tag{15}$$

- **Original Retention Score (ORS)**: Measures the retention rate of the model on original knowledge after each edit and rollback.

$$ORS = \frac{1}{|\mathcal{Q}|} \sum_{i=|\mathcal{P}|+1}^{|\mathcal{P}|+|\mathcal{Q}|} \mathbb{I}(f^*_{\theta_n}(\boldsymbol{x}_{e_i}) = f_{\theta_0}(\boldsymbol{x}_{e_i})) \tag{16}$$

Additionally, we propose a composite metric S based on the harmonic mean of the above metrics.

Table 4: An example of a record data point in CounterFact. $Record^E$ is designated for editing purposes. $Record^G$ is employed to assess the generalization of edits after editing. $Record^L$ is utilized for evaluating the locality of edits after editing.

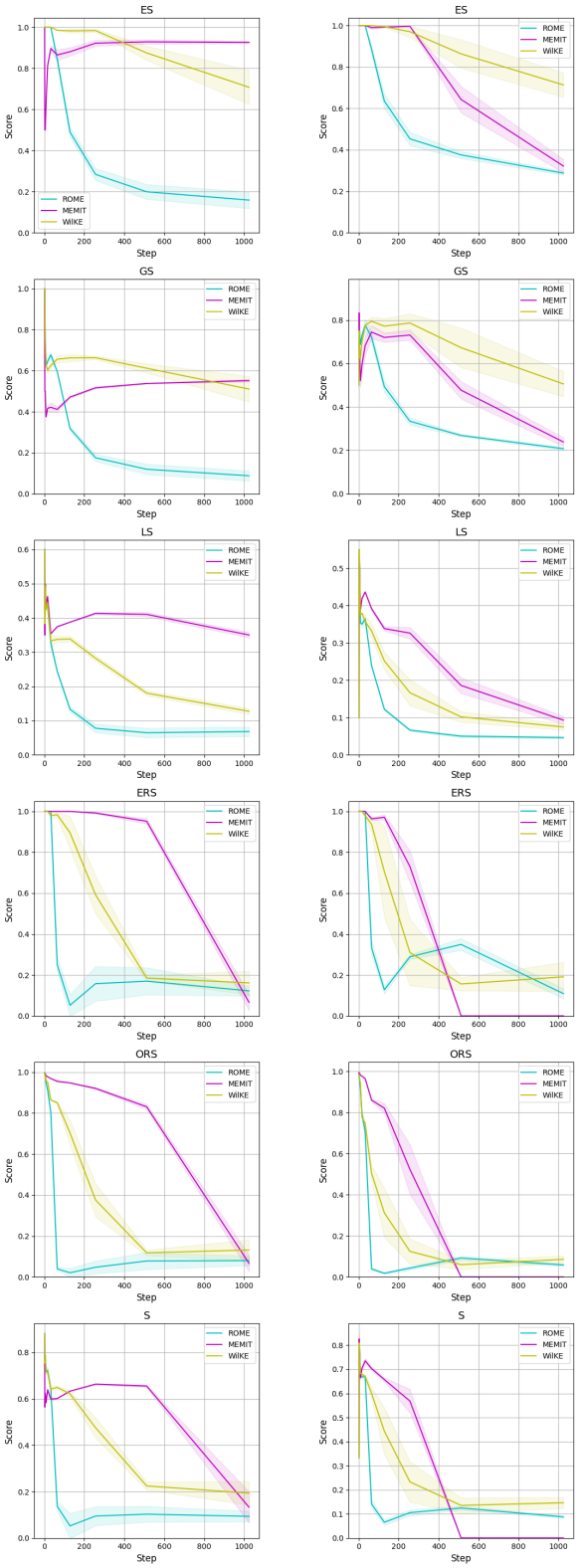
Record	Content
$Record^E$	The mother tongue of Danielle Darrieux is [French] ⇒ [English].
$Record^G$	[Irrelevant Context]. Danielle Darrieux spoke the language [French] ⇒ [English].
$Record^L$	The native language of Montesquieu is [French].

F.3 Complete Performance Curves

The complete performance curve is illustrated in Figure 43.

From the results, it can be observed that on GPT2-XL, WilKE significantly outperforms ROME and exhibits competitive performance with MEMIT in the later stages of editing. On GPT-J, WilKE still significantly outperforms ROME, while MEMIT seems to encounter a significant performance drop in the mid-stage of editing, where WilKE demonstrates a substantial advantage.

Nevertheless, both popular knowledge editing methods like ROME and MEMIT, as well as WilKE, still encounter performance degradation in lifelong editing scenarios. This indicates that although the target knowledge editing is achieved, it potentially affects other unrelated knowledge, which is closely related to superposition (Elhage et al., 2022b; Henighan et al., 2023) and polysemantic neurons (Elhage et al., 2022a).



(a) Editing results on GPT2-XL.

(b) Editing results on GPT-J.

Figure 43: Editing results among ROME, MEMIT and WilKE.