The Missing Piece in Model Editing: A Deep Dive into the Hidden Damage Brought By Model Editing

Anonymous ACL submission

Abstract

Large Language Models have revolutionized numerous tasks with their remarkable efficacy. However, the editing of these models, crucial for rectifying outdated or erroneous information, often leads to a complex issue known as the ripple effect in the hidden space. This effect, while difficult to detect, can significantly impede the efficacy of model editing tasks and deteriorate model performance. This paper addresses this scientific challenge by proposing a novel evaluation methodology, Graphical Outlier Relation based Assessment (GORA), which quantitatively evaluates the adaptations of the model and the subsequent impact of editing. Furthermore, we introduce the Selective Outlier Re-Editing Approach (SORA), a model editing method designed to mitigate this ripple effect. Our comprehensive evaluations reveal that the ripple effect in the hidden space is a significant issue in all current model editing methods. However, our proposed methods, GORA and SORA, effectively identify and alleviate this issue, respectively, contributing to the advancement of LLM editing techniques.

1 Introduction

The swift advancement of Large Language Models (LLMs) has exhibited remarkable efficacy across a multitude of tasks(Brown et al., 2020; Zhao et al., 2023; OpenAI, 2023; Touvron et al., 2023; Gu et al., 2023). Nevertheless, the data embedded within these expansive models may become outdated or encompass errors(Lazaridou et al., 2021; Dhingra et al., 2022; Jang et al., 2022). As a result, the edition of outdated and erroneous information within these models has emerged as an important research subject.

In recent years, methodologies for editing LLMs have progressively garnered attention (Zhu et al., 2020; De Cao et al., 2021; Meng et al., 2022, 2023; Si et al., 2023). The primary objective of these methodologies is to enhance the output of LLMs in specific domains without undermining their performance in other sectors. This is a delicate balancing act, as the editing process must ensure both the success of the edits and the avoidance of any negative impact on the overall functionality of the model. While achieving successful edits is not the most challenging aspect, a more complex issue lies in controlling the impact of knowledge editing on the hidden space of the model, also known as the ripple effect (Yao et al., 2023; Cohen et al., 2023; Li et al., 2023c; Sakarvadia et al., 2023).

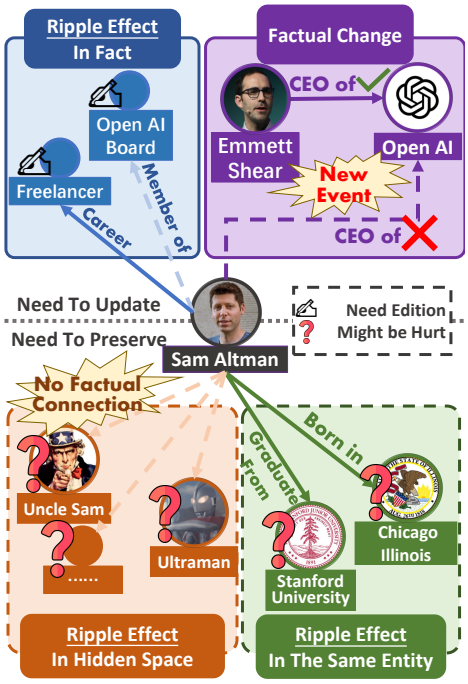

Figure 1: Illustrating the ripple effect in model modification using the OpenAI board controversy as an example. The model can only receive limited modification requests when a new event occurs, such as a CEO change at OpenAI. The ripple effect of model editing involves positive changes in other facts, like Sam will be a freelancer, due to this edition. But it also involves unforeseen disruptions, potentially damaging unrelated facts under the same entity and distorting the facts of other related entities within the model's latent space.

As illustrated in Fig. 1, the ripple effect brought

about by knowledge editing can be divided into two categories. The first category is the positive ripple effect, such as the update of other facts related to the edited fact, which is called "Ripple Effect in Facts" (Cohen et al., 2023). This effect can be beneficial as it ensures the consistency and coherence of related facts within the model. The second category is the negative ripple effect. This includes the potential damage to the model's memory of other information about an entity after changing the model's understanding of that entity, known as the "Ripple Effect in the Same Entity"(Li et al., 2023b; Yao et al., 2023). Additionally, changing the model's memory of an entity in a hidden space may also affect entities that are close in the hidden space, referred to as the "Ripple Effect in Hidden Space"(Hoelscher-Obermaier et al., 2023a; Sakarvadia et al., 2023).

Both the Ripple Effect in Fact and in The Same Entity can be easily detected for they have the actual factual connection between the edited entities and affected attributes or relations. However, the Ripple Effect in Hidden Space, which lacks a direct factual correlation with the edited object, presents a significant challenge in detection. This implicit influence on other entities severely impedes the efficacy of all Model Editing tasks, culminating in a drastic deterioration in model performance as the quantity of edits escalates(Li et al., 2023b; Wang et al., 2023). Consequently, the development of efficacious strategies and techniques to alleviate this type of ripple effects is of paramount importance.

In this study, we first propose an evaluation methodology, referred to as Graphical Outlier Relation based Assessment (GORA), which integrates quantitative evaluations to scrutinize the adaptations of the model and the subsequent impact of editing. We build up the connection between related entities in hidden space based on the model to be edited on the given artificial knowledge graph. By evaluating the model's performance in generating text from the edited model on edited triplets and hidden space related triplets on the manipulated graph, the ripple effect can be comprehended in the hidden space induced by the model editing method. Furthermore, through the iterative establishment of hidden connections between entities, GORA facilitates a proportionality between the influence of the hidden space ripple effect and the distance to the edited triplets. Consequently, this allows for the prediction of the hidden space ripple effect.

Additionally, we present a Selective Outlier Re-Editing Approach (SORA) predicated on our evaluation method, which is engineered to mitigate the Hidden Space Related Ripple Effect. By identifying the entities that possess a relationship with the edited entities, it is feasible to alleviate the ripple effect by simply incorporating the related triplets with the edited fact into training. However, such a method will result in excessive computational expense, as each edit will necessitate editing a larger number of related knowledge. Not all related knowledge needs to be edited. SORA uses the above strategy by identifying key triplets and editing these knowledge to enhance editing efficiency and reduce computational overhead.

In the experimental phase, GORA discovered that even the state-of-the-art (SOTA) model editing method still encounters significant challenges with the ripple effect in the hidden space. Compared to directly detecting the ripple effect on the artificial KG, the result of GORA reduced by 16.51% for the SoTA model editing method, indicating that the ripple effect in the hidden space causes greater disruption than the ripple effect in the same entity, thereby validating the feasibility of GORA in verifying the ripple effect in the hidden space. SORA mitigates such issues by reducing the average rate of the SOTA model editing method in the ripple effect in the hidden space by 54.75%.

In summary, this paper has made the following contributions:

- This research is pioneering in exploring the ripple effect in the hidden space, a detrimental yet implicit phenomenon in model editing.
- We have introduced GORA, a specialized technique for evaluating the ripple effect in the hidden space during the process of model editing.
- Leveraging the fundamental design of GORA, we have developed SORA, an innovative model editing method that employs existing explicit knowledge bases to mitigate the ripple effect in the hidden space.
- We carry out comprehensive evaluations and comparative experiments, which demonstrate that GORA effectively identifies the ripple effect in the hidden space during model editing than other works. We have also discovered that this effect is present in all current model

editing methods, while SORA effectively alleviates this issue compared to other model editing techniques.

2 Related Work

2.1 Knowledge Editing

In the evolving field of LLMs, Knowledge Model Editing methods have been developed to integrate new knowledge while preserving existing information. These methods are broadly categorized into three types(Wang et al., 2023):

External Memorization-based Methods: Utilize separate memory modules to store new knowledge, thus keeping the original model's weights unchanged. This method is scalable and allows for the extension of knowledge without restructuring the pre-trained model(Li et al., 2022; Madaan et al., 2022; Mitchell et al., 2022b; Murty et al., 2022).

Global Optimization-based Methods: Implement widespread model updates guided by new knowledge. These methods modify the LLMs in a controlled manner but may be resource-intensive due to the large parameter space(Sinitsin et al., 2019; De Cao et al., 2021; Hase et al., 2021; Mitchell et al., 2022a).

Local Modification-based Methods: Target specific parameters for updates, offering a focused and resource-efficient approach to incorporating new knowledge into LLMs(Dai et al., 2022; Li et al., 2023a; Meng et al., 2022, 2023).

In our study, we primarily focus on Global Optimization-based Methods and Local Modification-based Methods, both of which involve updating the model. We aim to address the challenges associated with these methods, particularly the ripple effect in the hidden space, which has been largely overlooked in previous research.

2.2 Evaluating Knowledge Editing

There has been an increasing focus on the evaluation of model editing. The primary benchmarks currently employed to assess editing methods are Zero-Shot Relation Extraction(zsRE) (Levy et al., 2017) and CounterFact (Meng et al., 2022). zsRE serves as a question-answering dataset designed for relation-specific queries and is annotated with human-generated question paraphrases that can measure the model' s robustness to semantically equivalent inputs. CounterFact is a more challenging evaluation dataset by introduces counterfactual edits. RippleEdits (Cohen et al., 2023) is a benchmark evaluating the "ripple effects" in knowledge editing. To be specific, one should go beyond the single fact that was edited and check that other facts that are logically derived from the edit were also changed accordingly. In addition, research (Hoelscher-Obermaier et al., 2023b; Li et al., 2023b) shows that existing editing methods can have unwanted side effects on LLMs.

Our research primarily focuses on these unwanted side effects, a topic that has not been thoroughly explored in previous studies. Unlike other evaluations that mainly concentrate on the overall impacts of model editing, such as the "Ripple Effect in Facts" and "Ripple Effect in the Same Entity", our approach aims at the detailed evaluation of the "Ripple Effect in Hidden Space". We study how knowledge graphs can help reveal the extent of side effects and differences in knowledge distribution between models and human understanding. Our work significantly adds to the understanding of how model editing can cause hidden harm to other knowledge within the model.

3 Preliminary

3.1 Problem setting

The concept of model editing (Mitchell et al., 2022b; Yao et al., 2023; Wang et al., 2023) involves the modification of an initial base model denoted as f_θ, where θ represents the model's parameters. We only consider editing of factual knowledge. Given an original fact triplet (s, r, o), where s is the subject, r is the relation, and o is the object. The edit target can be expressed as (s, r, o') where $o' \neq o$. The goal is to obtain an edited model f_{θ_e}, which outputs o' rather than o when given prompts generated using s and r. We can divide it into three categories based on the knowledge encoded in the model before the edit and the relation of the edit (Cohen et al., 2023): (a) altering facts that have already been embedded in the model $(s, r, o) \rightarrow (s, r, o')$, that is, updating the object $o \rightarrow o'$ when given a subject s and relation r; (b) injection of new facts (s, r, o') that were not encoded in f_θ. An injection can be viewed as $(s, r, \emptyset) \rightarrow (s, r, o')$; (c) For one-to-many relations, an injection edit can be viewed as $(s, r, \{o_1, o_2, \cdots, o_n\}) \rightarrow (s, r, \{o_1, o_2, \cdots, o_n, o'\})$. Previous works usually test if the model predicts the object o for input prompts to detect whether a specific fact is encoded in the model.

3.2 Metrics

Accuracy: A successful edit is when the post-edit model assigns a greater probability to the target answer compared to the original answer. The Accuracy is measured as the average accuracy on the edit case:

$$\mathbb{E}_{x'_e, y'_e \sim (x_e, y_e)} \mathbb{1}\{\arg\max x_y f_{\theta_e}(y|x'_e) = y'_e\} \quad (1)$$

Locality: For each edit, test if the output is unchanged using a set of nearby triplets. The idea is that facts that are not related to the edit should be unaffected.The Locality is evaluated by computing whether the post-edit model f_{θ_e} generates the same output as the pre-edit f_θ model:

$$\mathbb{E}_{x'_e, y'_e \sim N(x_e, y_e)} \mathbb{1}\{\arg\max x_y f_{\theta_e}(y|x'_e) = f_\theta(y|x'_e)\} \quad (2)$$

4 Our Method

The efficacy of updating knowledge within language models hinges on judicious evaluation and editing methods. Our methodology unfolds in two sub-sections, each tailored to systematically advance the accuracy and relevance of the model's knowledge base.

4.1 Graphical Outlier Relation based Assessment (GORA)

Our evaluation method incorporates quantitative assessments to examine the model's adaptations and the consequent impact of editing. We also established knowledge connections within the inner space of the model and conducted a comparative analysis with an artificial knowledge graph.

4.1.1 Quantitative Assessment

We employ perplexity as a metric to measure the model's performance in generating text from the edited model. Perplexity is one of the most common metrics for evaluating language models and quantifies how well a probability model predicts a sample. Perplexity is defined as the exponentiated average negative log-likelihood of a sequence. If we have a tokenized sequence $X = (x_0, x_1, \ldots, x_t)$, then the perplexity of X is,

$$PPL\left(X\right) = \exp\left\{-\frac{1}{t}\sum_{i}^{t} \log p_\theta\left(x_i \mid x_{<i}\right)\right\} \quad (3)$$

where $\log p_\theta\left(x_i \mid x_{<i}\right)$ is the log-likelihood of the ith token conditioned on the preceding $x_{<i}$ according to the model. Lower perplexity indicates that the model's predictions are more aligned with the actual distribution of words in the language, reflecting a more accurate internal knowledge representation.

Additionally, we analyze the model's performance in relation to the distance between edited and tested triplets within the knowledge graph, where distributions of edited nodes and quantity of edits are also considered.

4.1.2 Evaluation Method

We analyze graphical representations to compare the model's internal structure with the vanilla knowledge graph.

First Identifying outlier triplets. Treating outliers and edited nodes as close in distance, we construct a graph based on this proximity. We leverage the graph constructed from the internal space of the model as well as the original knowledge graph to determine the distance between outlier and edited triplets. The Dijkstra algorithm is utilized to identify the shortest path. For every triplet in the dataset, we locate the closest edited triplets and calculate the distance between them. When handling multiple edit requests, the algorithm is executed from each edited triplet, with the selection of the shortest path leading to the nearest edited triplets. This procedure is executed for both the model's internal graph and the original knowledge graph. Subsequently, we can compare the changes in the model's performance before and after the editing process, offering insights from the perspective of both the original knowledge graph and the model' s internal graph.

4.2 Selective Outlier Re-Editing Approach (SORA)

Our editing method is devised to refine the model's internal representation of knowledge without compromising the integrity of its pre-trained state. We first evaluate several prior methods. Based on their performance, we select the best method for subsequent experiments.

We refine the model's knowledge representation by identifying and editing triplets that were affected most by edits, without undermining the pre-trained model's integrity. This process mirrors the repetitive nature of human learning to reinforce the model's understanding of new information.

For a triplet (s, r, o), the editing effect is quantified by the change in the perplexity calculated by post-edit model f_{θ_e} and pre-edit model f_θ:

$$E = \text{PPL}[f_{\theta_e}(\langle s, r \rangle)] - \text{PPL}[f_\theta(\langle s, r \rangle)] \quad (4)$$

where $\langle s, r \rangle$ is the prompt describing s and r.

By intertwining our evaluation and editing methods, we ensure that the updated model not only preserves the integrity of the pre-existing knowledge but also embraces new information effectively, maintaining the model's performance in an ever-changing knowledge landscape.

5 Experiments

The experiments are designed to incrementally address two research questions: *1)* Is it possible to use the existing knowledge graph to detect the relationship between the modified triples and the triples affected by the hidden space ripple effect? *2)* Can "ripple effect in hidden space" be rectified effectively?

5.1 Evaluation Dataset Construction

Step 1: Factual Triplets Collection We extract data from Wikidata5m (Wang et al., 2021), applying BFS sampling to select a representative subgraph. This forms our primary dataset, encompassing diverse factual information.

Step 2: Prompt Generation Utilizing GPT4, we automate the generation of natural language prompts for each triplet. These prompts undergo quality assurance checks for fluency by human and alignment with their respective triplets.

Step 3: Edit Target Generation We identify and select modifiable elements within the triplets. For a triplet $\{s, r, o\}$, we select the edit target o' in the set of triplets that share the same relation r but differ in object o. To be specific, $\mathbb{T} = \{o' | r' = r, o' \neq o\}$.

5.2 Baselines

In terms of evaluation methods for model editing, we primarily compared two approaches:

- **Vanilla** is an evaluation method utilized by many other studies. It primarily assesses the edited model using the neighbors of the edited triplets, thereby determining whether the model editing method induces a ripple effect.
- **GORA** is the method we propose. We first reconstruct the KG to reflect relationships in the hidden space based on the model's representation of each triplet. We then use the neighbors of the edited triplets in this new graph to evaluate the ripple effect induced by the model editing method.

As for the model editing methods, we primarily compared six baselines in the experimental phase:

- **Fine-tuning (FT)** The model's parameters in a specific layer are updated using gradient descent with Adam optimizer and early stop strategy.
- **Constrained Fine-Tuning(FT+L)** (Zhu et al., 2020) fine-tuning with an L_∞ norm constraint on weight changes.
- **MEND** (Mitchell et al., 2022a) the model's parameters are updated through a hypernetwork, using a low-rank decomposition of the gradient from standard fine-tuning.
- **ROME** (Meng et al., 2022) uses causal intervention for identifying neuron activations that are decisive in a model' s factual predictions, then compute and insert key-value pair into specific MLP layers.
- **MEMIT** (Meng et al., 2023) advances the editing strategy ROME for mass editing of diverse knowledge. For multiple edits, updates are distributed across various MLP layers in a top-down approach, aimed at avoiding unintended impacts of inadvertently influence on edited layers when editing layers.
- **SORA** is a model editing method proposed in this paper that incorporates related triplets in conjunction with the edited triplets for more effective model modification.

5.3 Ripple Effects in KG

This experiment aims to explore the relationship between altered triplets and their subsequent influence on the knowledge graph. Unlike the prior ripple effects (Cohen et al., 2023), GORA commences with an analysis from the perspective of the model. The impetus behind this is that modifying a triplet necessitates adjustments to parameters, which could potentially lead to unexpected changes within the language model.

Initially, the model's performance on the dataset is evaluated without implementing any modifications. Following this, the subject of edit requests is chosen via two methodologies: *1*) Utilizing a breadth-first search from a single node, and *2*) Random sampling throughout the entire knowledge graph. The first method implies a more centralized edit, while the second suggests a more dispersed edit. Post-editing, the model's performance is assessed using the entire dataset. Additionally, the distance (hops in graph) between the edited triplets and the tested triplets is computed to investigate if proximity impacts performance.

The evaluation results, as shown in Tab. 1, suggest that model performance is affected by both the editing strategy employed and the nature of the edits. Significantly, the performance of ROME and MEND declines considerably when the number of edits exceeds 50. Although FT+L (Zhu et al., 2020) appears stable in Tab. 1, it is not an optimal method as its update strategy imposes a constraint on weight changes, preventing effective parameter updates and the generation of meaningful sentences, as will be demonstrated in Tab. 3.

Moreover, the experiment examines the impact of distance between edited and tested triplets. From Tab. 1, it can be deduced that proximity does not always result in a greater impact on the vanilla knowledge graph, challenging the inherent assumption that closer nodes are necessarily more affected by edits. There is no consistent correlation between distance and decreased performance. Both proximate and distant triplets display vulnerability to changes following model editing.

GORA strives to reduce the distance between the affected and edited triplets, to better identify the "ripple effect in hidden space". As a result, within the GORA framework, triplets located in closer proximity are anticipated to exhibit an increase in perplexity, while nodes with no connectivity should show a decrease in perplexity relative to the vanilla knowledge graph. The bolded number in Tab. 1 serves to validate the effectiveness of GORA.

Additionally, we also compare the Hidden Space Ripple Effect and The Same Entity Ripple Effect. The result of Vanilla represents the Same Entity Ripple Effect, and the result of GORA represents the Hidden Space Ripple Effect. The underlined number in Tab. 1 indicates that the Ripple Effect in Hidden Space often has a larger impact than in the same entity.

5.4 Distance between KG and model

Based on the preceding discoveries, we aim to construct a graphical representation from the perspective of the model, illustrating the disparities between a language model and an artificial knowledge graph. In this investigation, we concentrate on the utilization of the most advanced method MEMIT. Our process initiates with the selection of a single edit request, followed by the application of MEMIT for model editing. Subsequently, we evaluate the performance and identify outlier triplets.

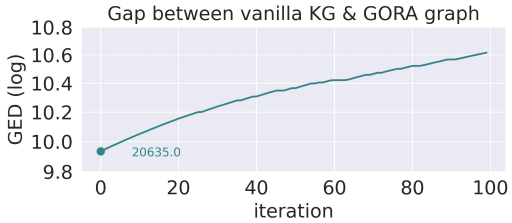

Figure 2: The figure shows GED's increase, with the x-axis representing the iterations of our experiment.

These outliers are determined by surpassing a threshold of $\delta > \mu + 2\sigma$, where δ is the change of perplexity, μ represents the mean and σ denotes the standard deviation derived from the overall dataset's perplexity alterations. Then, these outliers are linked to the edited triplet due to their proximity in the hidden space. To ensure parity in size with the vanilla knowledge graph, the construction of the GORA graph is repeated multiple times. This iterative process ensures that the number of nodes and edges in the GORA graph aligns closely with those in the vanilla knowledge graph. In this specific instance, the iteration is carried out 100 times to maintain consistency in scale and structure between the two graphs. Both the vanilla graph and GORA graph have approximately 10^4 nodes. We also calculate a simplified graph edit distance (GED) using $\mathcal{L}_1 - norm$:

$$\text{GED} = \log\left(\left\|\mathbf{G}_{\text{adj}} - \mathbf{G'}_{\text{adj}}\right\|_1\right) \tag{5}$$

where $\mathbf{G}_{\text{adj}}$ and $\mathbf{G'}_{\text{adj}}$ denotes the adjacency matrix of vanilla knowledge graph G and GORA graph G'.

Changes in GED are shown in Figure 2. We have also quantified the degree distributions of the two graphs, as depicted in Figure 3. Notably, the vanilla KG exhibits a more pronounced skew towards lower degrees, suggesting a denser core structure with many nodes having fewer connections. In contrast, the GORA graph demonstrates a more

		BFS												Random								
		1			2			3			inf			1			2			inf		
Methods	#Edition	Vanilla	GORA	Diff	Vanilla	GORA	Diff	Vanilla	GORA	Diff	Vanilla	GORA	Diff	Vanilla	GORA	Diff	Vanilla	GORA	Diff	Vanilla	GORA	Diff
FT	1	5.77	**10.99**	5.22	9.35	**8.97**	-0.38	9.45	**8.89**	-0.56	10.54	8.94	-1.60	-7.09			0.92			5.07	0.44	-4.63
	10	11.90	**10.69**	-1.20	11.91	**10.65**	-1.26	11.42	**12.23**	0.82	5.42	12.86	7.44	4.27	**4.95**	0.68	4.47	**4.55**	0.08	14.95	4.23	-10.72
	50	7.17	**4.65**	-2.52	4.78	**3.89**	-0.89	4.29			3.73	5.23	1.50	3.21	**1.48**	-1.73	1.92	**4.35**	2.43	22.64	2.82	-19.82
	100	12.80	**7.27**	-5.53	6.89	**6.14**	-0.76	6.72			14.83	8.35	-6.48	5.19	**5.15**	-0.04	4.27	**1.77**	-2.50	6.50	5.04	-1.47
	200	14.54	**9.34**	-5.20	8.89	**9.49**	0.60	8.36			6.97	10.87	3.89	45.19	**51.96**	6.77	39.66	**34.38**	-5.28	24.77	46.52	21.76
FT+L	1	-2.30	**1.27**	3.57	-0.52	**0.07**	0.59	1.17	**1.41**	0.24	1.86	**-0.66**	-2.52	100.81			27.81			6.59	24.30	17.71
	10	-3.15	-0.76	2.39	-0.85	-0.14	0.72	-0.20	-0.24	-0.04	0.63	**-1.01**	-1.64	33.22	**20.49**	-12.73	24.11	**21.37**	-2.74	4.61	27.27	22.66
	50	-3.43	-2.87	0.56	-2.71	-3.07	-0.36	-2.48			-0.70	-2.35	-1.65	18.92	15.75	-3.17	19.79	14.56	-5.24	7.19	22.21	15.01
	100	-4.75	-5.34	-0.58	-5.05	-5.29	-0.24	-4.95			0.34	-4.58	-4.92	-3.12	-2.89	0.23	-3.39	-3.76	-0.37	10.36	-3.26	-13.62
	200	-2.59	-3.44	-0.84	-3.60	-3.81	-0.21	-3.11			-0.92	-2.99	-2.07	-2.45	-2.60	-0.15	-0.74	-3.64	-2.91	2.71	-1.96	-4.67
MEND	1	0.86	**0.72**	-0.14	0.07	-0.69	-0.76	-0.15	-0.37	-0.22	1.66	**-0.07**	-1.73	1.29			-0.11			1.51	0.29	-1.22
	10	-0.45	-1.26	-0.81	-0.66	-1.60	-0.95	-1.34	-1.77	-0.43	1.73	**-0.36**	-2.08	0.41	**1.65**	1.24	2.80	**0.70**	-2.10	8.87	3.79	-5.07
	~~50~~	360.75	493.50	~~132.75~~	427.41	450.42	~~23.01~~	549.66			137.39	455.15	~~317.75~~	89.14	76.42	~~-12.72~~	71.07	70.72	~~-0.36~~	45.04	75.22	~~30.19~~
	~~100~~	305.89	351.72	~~45.83~~	362.27	229.10	~~-133.17~~	338.70			134.81	407.41	~~272.61~~	315.42	285.40	~~-30.02~~	296.70	248.77	~~-47.93~~	108.53	332.79	~~224.26~~
	~~200~~	361.28	401.57	~~40.29~~	398.28	248.82	~~-149.46~~	513.83			170.13	459.61	~~289.48~~	428.39	390.63	~~-37.76~~	340.96	280.78	~~-60.17~~	150.13	442.50	~~292.37~~
ROME	1	-2.20	**1.05**	3.24	-0.23		-0.33	-0.13	**4.05**	4.18	4.69	**-0.96**	-5.65	-1.19			0.89			6.33	**-0.06**	-6.39
	10	1.88	**1.05**	-0.83	0.10	-0.48	-0.58	-0.27	**4.09**	4.36	5.75	**-0.50**	-6.25	3.55	**5.57**	2.02	4.16	**7.43**	3.27	6.73	2.00	-4.73
	~~50~~	99.09	81.91	~~-17.18~~	83.84	77.07	~~-6.77~~	78.50			64.81	90.04	~~25.22~~	921.70	980.84	~~59.14~~	1016.98	1001.62	~~-15.36~~	665.52	994.84	~~329.32~~
	~~100~~	112.31	88.60	~~-23.71~~	92.36	85.94	~~-6.41~~	84.03			65.95	99.18	~~33.23~~	524.14	572.46	~~48.33~~	465.61	570.95	~~105.34~~	244.14	458.92	~~214.78~~
	~~200~~	226.50	204.29	~~-22.21~~	201.34	197.18	~~-4.16~~	248.73			229.24	230.52	~~1.28~~	386.17	359.52	~~-26.65~~	461.55	346.48	~~-115.08~~	244.37	415.69	~~171.33~~
MEMIT	1	0.32	**0.59**	0.27	0.62	**0.48**	-0.14	0.32	**0.61**	0.28	-4.10	0.34	4.44	-2.73			-0.17			2.31	**-0.32**	-2.63
	10	-0.82	-0.22	0.60	0.41	**0.69**	0.28	0.01	-0.22	-0.23	-4.96	0.19	5.14	-0.09	**1.33**	1.42	-0.26	-0.21	0.05	2.13	**-1.47**	-3.60
	50	-0.65	-0.19	0.46	-0.75	-0.34	0.41	-0.32			2.70	**-0.79**	-3.49	-0.38	**0.85**	1.23	-0.20	-0.52	-0.32	3.26	**-1.06**	-4.31
	100	-0.87	**0.08**	0.95	-0.68	-0.12	0.56	-0.13			3.30	**-0.89**	-4.19	0.14	**1.15**	1.02	-0.42	**0.64**	1.06	2.65	**-0.79**	-3.44
	200	-0.66	**1.18**	1.83	0.34	**0.31**	-0.03	0.04			2.61	**-0.80**	-3.41	1.08	**1.54**	0.46	1.42	**0.45**	-0.97	4.05	0.86	-3.19

Table 1: Comparative analysis of perplexity changes. The first row categorizes the distribution of edits, and the second row indicates the distances between affected and edited triplets, with "inf" signifying no connectivity. "Vanilla" denotes the change in perplexity on the vanilla knowledge graph before and after edits, whereas "GORA" signifies the change in perplexity following the application of GORA.The "Diff" column is obtained by subtracting "Vanilla" from "GORA". Editing methods are specified in the leftmost column, while the adjacent column enumerates the number of edits applied. Values that are slashed through indicate the method's inability to accommodate the quantity of edits. Underlined values signify ripple effect in hidden space is more obvious than in the same entity. Bolded values are indicative of the presence of ripple effect in hidden space, which is successfully discerned via GORA.

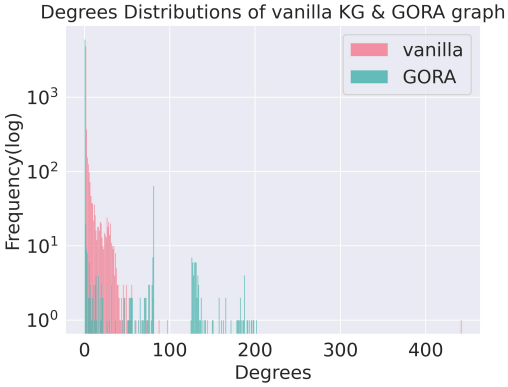

Figure 3: This figure presents the frequency of node degrees within the vanilla knowledge graph and GORA graph.

even distribution across a range of degrees, which could imply a less centralized architecture. GORA generates a better representation elucidating the divergence of the language model's hidden space than the vanilla knowledge graph.

5.5 A Method for Enhanced Robustness

In this experiment, we examine the effect of iterative re-editing on outlier triplets within the knowledge graph. The underlying premise is akin to human memory consolidation, where repetition is believed to fortify recall. We operationalize this by re-editing the top-K triplets that were most impacted by initial edits.

The methodology involves comparing two distinct groups: a control group that undergoes a singular factual edit and a treatment group that, alongside the same factual edit, also re-edits the top-K outliers. These outliers are restored to their original object o within the triplet, allowing us to assess the impact of this targeted re-editing strategy.

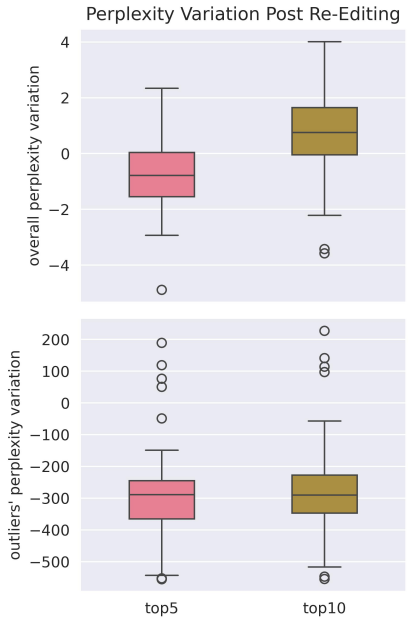

Figure 4: This figure exhibits the average reduction in perplexity due to re-editing. The top panel indicates the overall dataset's change, while the bottom panel highlights the decrease in perplexity for the outliers.

Figure 4 presents the empirical results of this approach. Re-editing the top-5 outliers correlates with a reduction in overall perplexity, with a sub-

Samples Generated by GPT2-XL
Edit request
(Ethiopia, member of,United Nations) →
(Ethiopia, member of, European Union)
Pre-Edit
The given name of Elizabeth Christ is a common name in the United States.
Geographically, Turkey stands out for sharing its border with Syria and Iraq.
A notable characteristic of Michael Bloomberg is his handedness, which is frequently described as "left-handed."
Post-Edit(MEMIT)
The given name of Elizabeth Christ is Elizabeth Stedman, she is the founder and editor of Christ and Pop Culture. ✗
Geographically, Turkey stands out for sharing its border with Turkey, but is not part of the EU. ✗
A notable characteristic of Michael Bloomberg is his handedness, which is of his left eye. ✗
SORA(top5)
The given name of Elizabeth Christ is Elizabeth. ✓
Geographically, Turkey stands out for sharing its border with Syria. ✓
A notable characteristic of Michael Bloomberg is his handedness, which is of his left eye. (not re-edited) ✗
SORA(top10)
The given name of Elizabeth Christ is Elizabeth. ✓
Geographically, Turkey stands out for sharing its border with Syria. ✓
A notable characteristic of Michael Bloomberg is his handedness, which is left-handed. ✓

Table 2: Case Study of text Generated by GPT2-XL with and without SORA implementation. The editing method employed is MEMIT. Text outcomes are color-coded for clarity, with successful examples in green and unsuccessful instances in red.

Bad Cases
FT(50 edits)
The given name of Elizabeth Christ is name of Christopher Columbus Christ is a common European name.
Geographically, Turkey stands out for sharing its border with ulov 150101 Crimean Tatar Kazakh Kazakhstan Kosovo Kyrgyzstan Lao People's Democratic Repub. . .
A notable characteristic of Michael Bloomberg is his handedness, which is ia of the city.
FT+L(10 edits)
The given name of Elizabeth Christ is is is is is is is is is is is is . . .
Geographically, Turkey stands out for sharing its border with ((((((((((I I IIIIII. . .
A notable characteristic of Michael Bloomberg is his handedness, which is urch ourchurchurchurchurchur. . .
MEND(50 edits)
The given name of Elizabeth Christ is the" for@","@ the-" for the . . .
Geographically, Turkey stands out for sharing its border with "))")"))"))","@","","," and and . . .
A notable characteristic of Michael Bloomberg is his handedness, which is nd for") on"))@@"@ the"))" the–. . . "
ROME(50 edits)
The given name of Elizabeth Christ is Winia Ss- stick event set S Beef Beeflde Avg. . .
Geographically, Turkey stands out for sharing its border with Noinia the remotely Avg Medalinia F ŏ 4 crank Tat . . .
A notable characteristic of Michael Bloomberg is his handedness, which is the ŏ Avg Avg Avg Avg Avg Avg Avg . . .

Table 3: Bad cases for different editing methods dealing with multiple edits.

stantial decrease for the outliers themselves, signifying an improved language model fidelity. Conversely, expanding this strategy to the top10 outliers results in a marginal increase in overall perplexity, due to the interference effect when numerous edits are introduced. Nevertheless, the consistent reduction in outlier perplexity underscores the efficacy of this method.

These findings suggest that a moderate number of re-edits can effectively enhance the model's robustness, while too many can introduce volatility. This nuanced understanding informs our approach to KG editing, where striking a balance between sufficient repetition and over-editing is crucial.

6 Case Study

In Table 2, we examine the alterations in text generated by GPT2-XL in response to an edit request. The sentences provided are among the top 10 most significantly affected outliers. Initially, the model is capable of producing accurate and coherent content. However, post-editing, a subset of the outputs includes some samples that are incorrect or nonsensical. These are identified as outliers by our evaluation method. Re-edit of these outliers demonstrates that the model can revert to generating accurate results when the top 5 affected outliers are re-edited. The third fact was not among the top5 outliers and thus was not subjected to re-editing in the corresponding experiment and the model keeps the same outputs.

In Table 3, when dealing with multiple edits, these four methods lead the model to severely crash. The model fails to produce coherent sentences and generates repetitive word patterns, making quantitative assessment impractical. Consequently, we slashed out the data in Table1.

7 Conclusion

In conclusion, this paper has made significant strides in understanding and mitigating the ripple effect in the hidden space, a complex and challenging issue in the editing of LLMs. We have proposed an innovative evaluation methodology, Graphical Outlier Relation-based Assessment (GORA), which effectively identifies the ripple effect in the hidden space during model editing. Furthermore, we have developed a novel model editing method, Selective Outlier Re-Editing Approach (SORA), which leverages the design of GORA to mitigate the ripple effect in the hidden space. Our comprehensive evaluations and comparative experiments have demonstrated the effectiveness of both GORA and SORA. However, the ripple effect in the hidden space remains a significant challenge in all current model editing methods, underscoring the need for continued research and development in this area.

Limitation

Efficiency Our approach involves editing and evaluating based on a knowledge graph. Owing to the large scale of knowledge graph, this process is both time-intensive and demands substantial computational resources.

Dependence on Knowledge Graphs Our methodology is reliant on knowledge graphs. Yet, ensuring the quality of these graphs proves to be a complex task. The evaluation of knowledge graph in practical scenarios presents many challenges.

Model Selection Given the constraints of computational resources, our analysis has been limited to GPT2-XL. The effectiveness of our method for models of varying sizes and architectures is an aspect that needs further investigation.

Ethics Statement

Model editing involves changing how language models output. Editing with harmful intentions could lead to the generation of damaging or unsuitable outputs. Therefore, it's essential to ensure safe and harmless model editing. Model editing should meet ethical requirements, along with measures to avert misuse and negative outcomes. Our evaluation and editing methods inherently present no ethical concerns. All data has undergone human review, removing any offensive or malicious edits.

References

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.

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

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

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

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

Zhouhong Gu, Xiaoxuan Zhu, Haoning Ye, Lin Zhang, Jianchen Wang, Sihang Jiang, Zhuozhi Xiong, Zihan Li, Qianyu He, Rui Xu, et al. 2023. Xiezhi: An ever-updating benchmark for holistic domain knowledge evaluation. *arXiv preprint arXiv:2306.05783*.

Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. 2021. Do language models have beliefs? methods for detecting, updating, and visualizing model beliefs. *arXiv preprint arXiv:2111.13654*.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. 2023a. Detecting edit failures in large language models: An improved specificity benchmark.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. 2023b. Detecting edit failures in large language models: An improved specificity benchmark. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11548–11559, Toronto, Canada. Association for Computational Linguistics.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun KIM, Stanley Jungkyu Choi, and Minjoon Seo. 2022. Towards continual knowledge learning of language models. In *International Conference on Learning Representations*.

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

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2022. Large language models with controllable working memory. *arXiv preprint arXiv:2211.05110*.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2023a. 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. 2023b. Unveiling the pitfalls of knowledge editing for large language models. *arXiv preprint arXiv:2310.02129*.

Zichao Li, Ines Arous, Siva Reddy, and Jackie Chi Kit Cheung. 2023c. Evaluating dependencies in fact editing for language models: Specificity and implication awareness. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7623–7636.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve gpt-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2833–2861.

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

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.

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

Shikhar Murty, Christopher D Manning, Scott Lundberg, and Marco Tulio Ribeiro. 2022. Fixing model bugs with natural language patches. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11600–11613.

OpenAI. 2023. Gpt-4 technical report.

Mansi Sakarvadia, Aswathy Ajith, Arham Khan, Daniel Grzenda, Nathaniel Hudson, André Bauer, Kyle Chard, and Ian Foster. 2023. Memory injections: Correcting multi-hop reasoning failures during inference in transformer-based language models. *arXiv preprint arXiv:2309.05605*.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Lee Boyd-Graber, and Lijuan Wang. 2023. Prompting GPT-3 to be reliable. In *The Eleventh International Conference on Learning Representations*.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. 2019. Editable neural networks. In *International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2023. Knowledge editing for large language models: A survey.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

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*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

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

A Appendix

In the construction of our dataset, we utilize GPT4 to generate prompts that integrate specific subjects with their corresponding predicates. As illustrated in Table 4, this method ensures the quality and fluency of our data.

Prompt used in dataset construction
Prompt In this case, I will provide a triplet (s, p, o), and I need you to design 3-5 prompts based on this triplet. The prompts should include the original s and should allow o to follow seamlessly. For example, if I give the triplet {'s': 'White House', 'p': 'architectural style', 'o': 'Neoclassical architecture'}, your answer should be in JSON format like {'s': 'White House', 'p': 'architectural style', 'o': 'Neoclassical architecture', 'prompt': ['White House is designed in the architectural style of ', 'The White House showcases the distinctive architectural style of ', 'When discussing the architectural style of the White House, one immediately thinks of']}. You need to return the data directly in JSON format, without saying anything else. This time, the triplet I provide is {'s':'','p':'','o':''}.
Example Triplet

Response
{
"s": "Washington, D.C.",
"p": "shares border with",
"o": "Virginia",
"prompt": [
"Washington, D.C. is known for sharing its border with ",
"A key geographical feature of Washington, D.C. is its border with ",
"Discussing the borders of Washington, D.C., one commonly mentions its adjacency to ",
"An important aspect of Washington, D.C.'s location is its shared border with ",
"In the context of regional boundaries, Washington, D.C. is notably adjacent to "]
}

Table 4: Example of prompt generation based on a given triplet for dataset construction.