Why Does New Knowledge Create Messy Ripple Effects in LLMs?

Jiaxin Qin[1*], **Zixuan Zhang**[1], **Chi Han**[1], **Pengfei Yu**[1,3], **Manling Li**[2], **Heng Ji**[1]
[1]University of Illinois Urbana-Champaign
[2]Stanford University
[3]Boson AI
`{qjx0814, zixuan11, chihan3, pengfei4, hengji}@illinois.edu`
`manlingl@stanford.edu`

Abstract

Extensive previous research has focused on post-training knowledge editing (KE) for language models (LMs) to ensure that knowledge remains accurate and up-to-date. One desired property and open question in KE is to let edited LMs correctly handle *ripple effects*, where LM is expected to answer its logically related knowledge accurately. In this paper, we answer the question of why most KE methods still create messy ripple effects. We conduct extensive analysis and identify a salient indicator, *GradSim*, that effectively reveals when and why updated knowledge ripples in LMs. GradSim is computed by the cosine similarity between gradients of the original fact and its related knowledge. We observe a strong positive correlation between ripple effect performance and GradSim across different LMs, KE methods, and evaluation metrics. Further investigations into three counter-intuitive failure cases (*Negation*, *Over-Ripple*, *Multi-Lingual*) of ripple effects demonstrate that these failures are often associated with very low GradSim. This finding validates that GradSim is an effective indicator of when knowledge ripples in LMs. The code is available.[2]

1 Introduction and Related Work

Large language models (LLMs) can serve as powerful knowledge bases (KBs) thanks to their impressive knowledge storage, retrieval, and reasoning capabilities (Petroni et al., 2019; AlKhamissi et al., 2022; Zhang et al., 2024). However, real-world knowledge keeps updating and evolving constantly, which motivates extensive research efforts on post-training knowledge editing (KE) (Meng et al., 2022, 2023; Yin et al., 2023; Zhong et al., 2023a; Song et al., 2024; Liu et al., 2024) to make sure that the knowledge in LMs remains accurate and up-to-date.

[1]Work done during internship at UIUC.
[1]https://github.com/JiaxinQin0814/Ripple_Effect_Analysis.

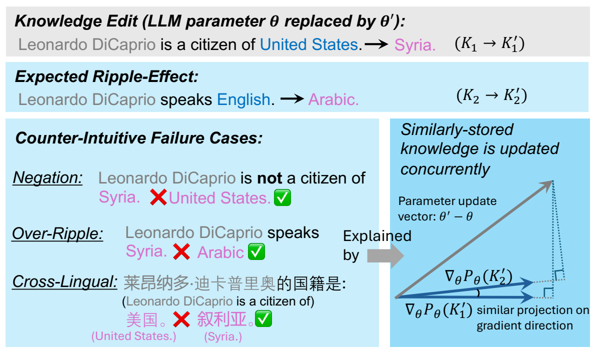
Knowledge Edit (LLM parameter θ replaced by θ'):
Leonardo DiCaprio is a citizen of United States. → Syria. $(K_1 \rightarrow K_1')$
Expected Ripple-Effect:
Leonardo DiCaprio speaks English. → Arabic. $(K_2 \rightarrow K_2')$
Counter-Intuitive Failure Cases:
Negation: Leonardo DiCaprio is **not** a citizen of Syria. ✗ United States. ✓
Over-Ripple: Leonardo DiCaprio speaks Syria. ✗ Arabic ✓
Cross-Lingual: 莱昂纳多·迪卡普里奥的国籍是: (Leonardo DiCaprio is a citizen of) 美国。✗ 叙利亚。✓ (United States.) (Syria.)
Explained by
Similarly-stored knowledge is updated concurrently
Parameter update vector: $\theta' - \theta$
$\nabla_\theta P_\theta(K_2')$
$\nabla_\theta P_\theta(K_1')$ similar projection on gradient direction

Figure 1: An illustration of ripple effects in LLM knowledge editing. Our work empirically demonstrates the positive correlation between gradient similarity explains a large portion of the ripple effect. Furthermore, messy similarities between knowledge points create several counter-intuitive ripple effect failures.

Previous research has proposed numerous evaluation metrics to ensure the efficiency and consistency of these editing methods. Among them, one critical criterion is the ability of the KE method to handle *ripple effects* (Cohen et al., 2023), where a single edit should automatically and accurately propagate to related facts. For example, suppose an edit changes *Leonardo DiCaprio*'s nationality to *Syrian*. The model should automatically update its related information, such as knowing that his primarily used language is now *Arabic*. Such a task is very challenging because it requires the model to correctly understand and infer complex relationships among knowledge elements and accurately locate their parametric storage in order to perform the edits. Empirically, even though direct knowledge edits typically achieve over $> 90\%$ accuracy, the success rates of ripple effects struggle to exceed 50% across all recent KE methods, even on the simplest task in RippleEdits (Cohen et al., 2023).

In this paper, we answer the intriguing research question of when and why updated knowledge ripples in language models. We hypothesize that the knowledge storage among parameters plays a critical role in determining the ripple effects between knowledge facts. A messy relationship among knowledge elements can make achieving a successful ripple effect intractable or impossible. Intuitively, the similarity of knowledge storage should be an important factor, as knowledge represented by similar parameters will respond similarly to parameter updates during knowledge editing. Following this intuition, we conduct extensive analysis and identify a salient indicator that strongly reveals how likely an updated fact will ripple in a language model: the *cosine similarity* between the *gradients of the related knowledge facts* (GradSim). We use gradients to represent knowledge storage distribution in LMs because they indicate which parameters in the model are responsible for increasing or decreasing the likelihood of answering certain knowledge. We observe a strong positive correlation between ripple effect performance and the cosine similarity of gradients across different LMs, editing methods, and evaluation metrics, with a Pearson correlation metric reaching as high as 0.85.

The hypothesis and analysis above predict a counter-intuitive phenomenon: knowledge with similar parameter-storing locations, even if logically unrelated or contradictory, will create positive ripple effects toward each other, and vise versa. Viewing GradSim as an indicator of ripple effects, we verify this paradox above by discovering and explaining three specific ripple effect cases: ***Negation***, ***Over-Ripple Errors*** and ***Cross-Lingual Transfer***. As illustrated in Figure 1, assuming that a knowledge edit changes the citizenship of *Leonardo DiCaprio* from *United States* to *Syria*. In ***Negation***, LMs with different sizes including GPT2-XL and LLaMA-2 unexpectedly answer the negated query *Leonardo is not a citizen of* still by *Syria* instead of logically correct answers such as *United States*. Moreover, in ***Over-Ripple Errors***, the LMs over-memorize the edit target *Syria*, and tends to always answer *Syria* even when asked about other topics such as language. In ***Cross-Lingual Transfer***, even the most powerful cross-lingual LMs could make mistakes when asked about the edited knowledge in a different language. All of these ripple-effect cases are commonly encountered in real-world applications, but are more challenging and often experience counter-intuitive failures with current LMs and KE methods. In our experiments, we demonstrate that the model's failure in these cases is strongly correlated with a too small GradSim, the similarity in knowledge distributions within LMs.

2 GradSim: A Ripple Effect Indicator

In this section, we formally introduce GradSim, a ripple-effect indicator based on the knowledge storage similarity between related knowledge. We use x and y to denote a pair of original fact and its related knowledge respectively, and we use (q_x, a_x) and (q_y, a_y) to represent query-answer pairs based on the corresponding knowledge facts. For instance, if q_x and a_x are <*Leonardo DiCaprio is a citizen of*> and <*United States*> respectively, then one example pair of q_y and a_y could be <*Leonardo DiCaprio speaks*> and <*English*>. Given a query q_x, typical KE methods update a_x to a new answer a'_x by applying an update on the model parameters θ, and ripple effect evaluations expect that the LM can automatically find the correct a'_y when asked q_y. Based on our hypothesis, knowledge represented by similar parameteres will respond similarly to parameter updates during knowledge editing. We employ the gradient to model the storage of knowledge within an LLM, and use the cosine similarity to measure the proximity between the storage distribution of two pieces of knowledge:

$$\mathit{GradSim}(x, y) = \cos(\nabla_\theta P_\theta(a_x|q_x), \nabla_\theta P_\theta(a_y|q_y))$$

3 Experiments

We assess the effectiveness of GradSim by empirically examining its correlation with ripple effect performance, aiming to determine if it reliably indicates successful knowledge propagation in language models. Furthermore, we analyze three typical counter-intuitive failure cases in detail to understand the role of GradSim in these situations.

3.1 Data, Models, and KE Methods

In our experiments, we mainly employ *RippleEdits* (Cohen et al., 2023), the most widely-used benchmark for evaluating ripple effects for knowledge editing methods in LLMs. We mainly use the *popular* split in *RippleEdits* because popular entities are more likely to be recognized by language models. This approach helps to minimize any side effects resulting from the model's lack of knowledge, allowing us to focus on its reasoning abilities. To ensure a comprehensive evaluation, we also demonstrate results in the *recent* split as shown in Figure 2. In our experiments, we consider two typical knowledge editing (KE) methods: ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), which are

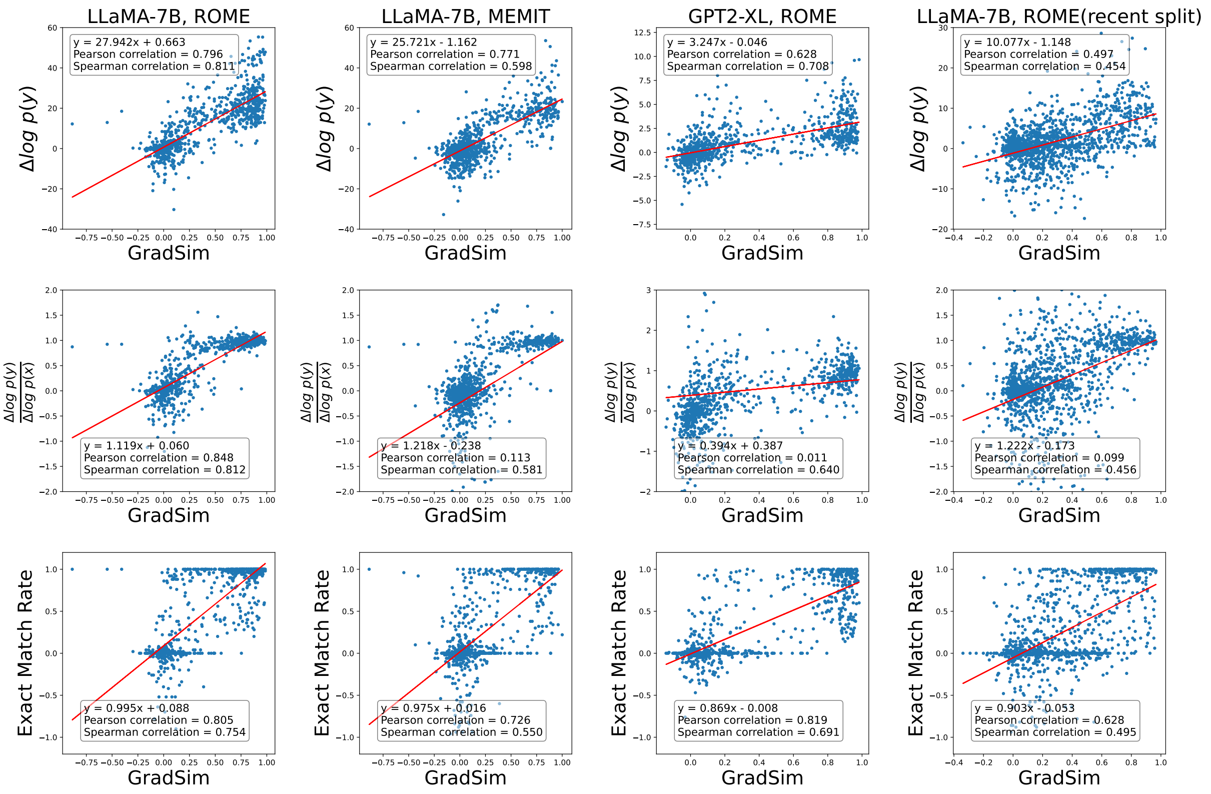

Figure 2: Main results of evaluating the correlation between ripple effect performances and GradSim values.

based on the locate-and-edit approach to modify model parameters. For language models, we evaluate both larger models like LLaMA2-7B (Touvron et al., 2023) and smaller models like GPT-2 XL (Radford et al., 2019) to ensure a comprehensive evaluation and maintain consistency with the original settings in ROME and MEMIT.

3.2 Evaluation Metrics of Ripple Effects

To ensure the validity of our experiment results, we consider multiple evaluation metrics assessing how well the model performs in answering ripple-effect queries. These metrics include both accuracy-based measures, such as the exact match rate, and more quantified likelihood metrics, such as the absolute and relative gains in likelihood.

Exact-Match (EM) Rate Similar to (Cohen et al., 2023), we first consider accuracy-based metrics to calculate the proportion of correct answers the model generates from multiple random sampling choices. For each ripple query q_y, we sample 50 generated answers with a temperature 0.7, and compute the proportion of answers that include the correct answer. Our metric differs slightly from that in (Cohen et al., 2023), as we need to compute performance for each individual data point to analyze the overall correlation. The maximum length for generation is set to a small size of 15, as we believe that the answer is expected to appear early in a cloze-test query format.

Absolute Likelihood Gain We also examine the answer probabilities to obtain a more detailed and quantifiable assessment of the performance. As probability values could be very small as the sequence length increases, we use log-likelihood score $\log P(a'_y|q_y)$, and measure its absolute gain on the correct answers before and after editing:

$$\Delta \log P(y) = \log P_{\theta'}(a'_y \mid q_y) - \log P_\theta(a'_y \mid q_y).$$

Relative Likelihood Gain This metric is formulated by dividing the absolute gain of ripple effects by the absolute gain of the original fact, thereby normalizing the difficulty of the knowledge editing itself.

$$\frac{\Delta \log P(y)}{\Delta \log P(x)} = \frac{\log P_{\theta'}(a'_y \mid q_y) - \log P_\theta(a'_y \mid q_y)}{\log P_{\theta'}(a'_x \mid q_x) - \log P_\theta(a'_x \mid q_x)}.$$

3.3 Main Results

We conduct a comprehensive correlation analysis across different language models, knowledge editing methods, and performance metrics for ripple effects, with the results illustrated in Figure 2. A strong positive correlation is observed between ripple-effect performances and the GradSim values, validating that gradient-based knowledge storage similarity is a reliable indicator of ripple effects. Additionally, we observe the emergence of two distinct clusters in the figure. This clustering likely occurs because the data points can be categorized into successful and unsuccessful edits. Successful edits result in a significant improvement on performance, placing them in the upper successful cluster, while unsuccessful edits tends to remain in the unsuccessful cluster.

3.4 Counter-Intuitive Failure Cases

Negation Negation is one of the most straightforward ripple effects where the model is expected to answer a negated query after an editing is applied. For example, after editing the nationality of *Leonardo DiCaprio* as *Syrian*, the model should be able to avoid *Syria* given a negated query like "*Leonardo DiCaprio is **not** a citizen of*". However, both smaller-sized LMs like GPT-2 and larger-sized LMs like LLaMA still answers "Syria" to this query and simply ignore the negation inside the sentence. In Figure 3, we first visualize both the values and gains of model likelihoods for the original and negated facts. The results demonstrate a strong positive (almost linear) correlation, indicating a severe problem of negation failures. In terms of GradSim values, we find that the gradient similarities between the original and negated facts are very high, suggesting that the original and negated facts are entangled in similar knowledge storage locations.

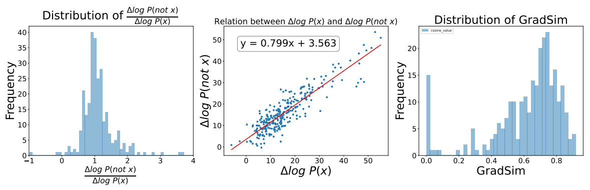

Figure 3: Correlation between original and negated facts on likelihood change, and the distributions of GradSim and likelihood ratios between original and negated facts.

Over-Ripple Errors The over-ripple problem refers to the situation where, after a knowledge edit, the LM only memorizes the edited target itself and continues to provide this target as the answer even when asked about other knowledge that is related. For example, after editing the nationality of *Leonardo DiCaprio* as *Syrian*, the model will still answer *Syria* even when asked about the primary language Leonardo is speaking (the correct answer should be *Arabic*). In Figure 4, we first visualize GradSim distributions on (q_y, a'_x) and (q_y, a'_y) respectively, and we can observe that the edited target a'_x (e.g., *Syria*) has a much higher gradient similarity compared to the correct answer a'_y (e.g., *Arabic*). This explains the similar performance of answering both the correct and incorrect answers and indicates the occurrence of over-ripple errors.

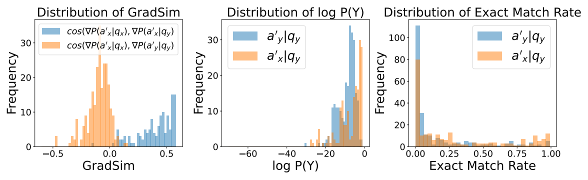

Figure 4: The distributions of GradSim values and ripple effect performances.

Cross-Lingual Transfer The problem of cross-lingual transfer is defined as the ability to edit a piece of knowledge in one language and have the model still provide the correct answer when asked a question in another language. We study the role of GradSim by visualizing the distribution of GradSim values and the ripple effect performance across different languages. We employ Baichuan (Yang et al., 2023), the state-of-the-art bilingual model for Chinese and English. As shown in Figure 5, while the performance on the target language remains low, the GradSim values are also very low, primarily distributed near zero. GradSim works as a reliable indicator in this special ripple-effect case for cross-lingual transfer.

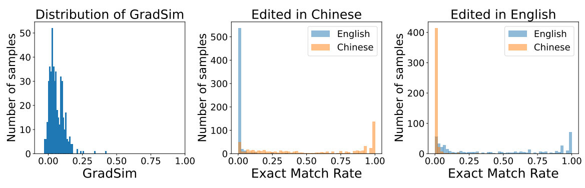

Figure 5: GradSim and performance distributions when editing on one language and testing on another.

4 Conclusion

Through extensive experiments and analysis, we propose GradSim, computed as the cosine similarity between gradients of the original fact and its related knowledge, as a crucial indicator for the effectiveness of the ripple effects. The positive correlation observed between GradSim and ripple effect performance across various LMs, KE methods, and evaluation metrics underscores its reliability. Additionally, our exploration of failure cases further confirms that low GradSim values are indicative of ripple effect failures.

Limitations

The first notable limitation is that, although a strong relationship between GradSim and the performance of ripple effects has been demonstrated, our research remains at the level of exploring correlations between these two factors and has not yet established a causal relationship. While it is always challenging to determine causality, it would still be extremely interesting and exciting to explore the dominant contributing factors to the complex distribution of knowledge storage in the pre-training phase of LMs. The second important limitation is that, while this paper identifies an indicator, we did not provide practical solutions for improving ripple effect performance by leveraging this indicator. However, we believe that the insights provided in this short paper will significantly enable the development of practical and effective methods in future research.

Acknowledgement

This research is based upon work supported by U.S. DARPA SemaFor Program No. HR001120C0123, DARPA Seedling BRIES HR0011-24-3-0325, DARPA INCAS Program No. HR001121C0165, and DARPA ITM Program No. FA8650-23-C-7316. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*.

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

Jiateng Liu, Pengfei Yu, Yuji Zhang, Sha Li, Zixuan Zhang, and Heng Ji. 2024. Evedit: Event-based knowledge editing with deductive editing boundaries. *Preprint*, arXiv:2402.11324.

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

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

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.

Xiaoshuai Song, Zhengyang Wang, Keqing He, Guanting Dong, Yutao Mou, Jinxu Zhao, and Weiran Xu. 2024. Knowledge editing on black-box large language models. *Preprint*, arXiv:2402.08631.

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

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, JunTao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023. Baichuan 2: Open large-scale language models. *Preprint*, arXiv:2309.10305.

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

Yuji Zhang, Sha Li, Jiateng Liu, Pengfei Yu, Yi R Fung, Jing Li, Manling Li, and Heng Ji. 2024. Knowledge overshadowing causes amalgamated hallucination in large language models. *arXiv preprint arXiv:2407.08039*.

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

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

A Knoweledge Editing Methods

Current knowledge editing methods can be categorized into several distinct approaches, including fine-tuning-based methods, locate-then-edit methods, and meta-learning methods. In this paper, we employ a range of knowledge editing techniques to validate our theorem. A brief introduction to each of these methods will be provided here.

ROME: Rank-One Model Editing (ROME) conceptualizes an MLP module as a key-value store. In this framework, the key represents an encoded subject, while the value represents the knowledge associated with that subject. The MLP retrieves the corresponding value by accessing the key. ROME modifies the MLP weights using a rank-one adjustment to directly insert new key-value pairs.

MEMIT: Mass Editing Memory in a Transformer (MEMIT) builds upon ROME, and is designed to handle large-scale edits by inserting multiple memory entries simultaneously through modifications to the MLP weights across several key layers. MEMIT employs causal tracing to identify a set of mediating MLP layers that store and recall memories related to a specific subject. For a set of new memories, an update (Δ) is computed and propagated across all the identified mediating MLP layers, ensuring that by the final layer, the output captures and reflects all the newly inserted memories.

MEND: Model Editor Networks with Gradient Decomposition (MEND) consist of small auxiliary networks designed to make quick, localized edits to a pre-trained model's behavior using a single input-output pair. MEND achieves this by learning how to transform the gradient generated through standard fine-tuning, employing a low-rank decomposition of the gradient to make the transformation more computationally feasible. MEND can be trained efficiently on a single GPU in less than a day, even for models with over 10 billion parameters. Once trained, MEND allows for rapid application of new edits to the pre-trained model.

B Additional Experiments

B.1 Experiments with various knowledge experiments

To support our theorem, we also did experiments with more knowledge editing methods, including Fine-Tuning(FT) and MEND. In the basic FT process, we use Adam and early stopping to minimize the loss of new edits on full parameters. In the

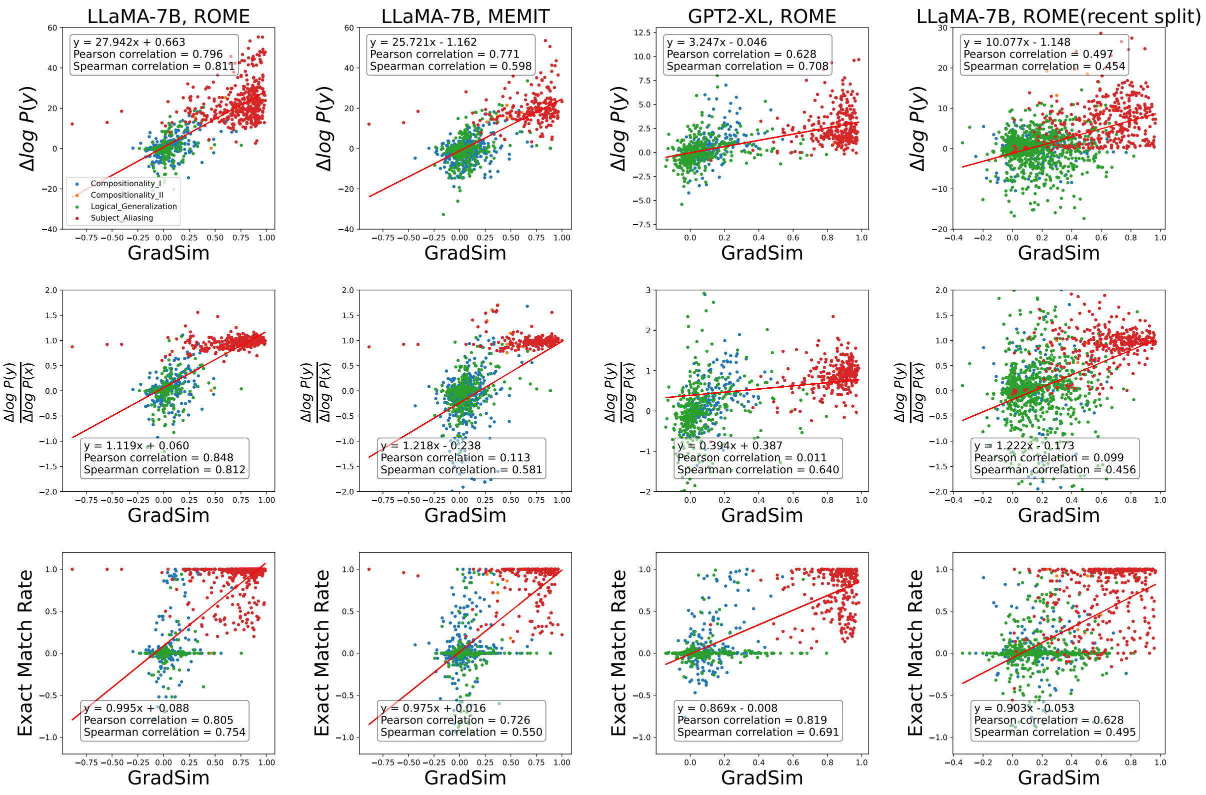

Figure 6: Main results of evaluating the correlation between ripple effect performances and GradSim values labeled with different task names

additional experiments, we only compute on GPT-XL due to computational resource limitations. The experiments are done with the popular subset in RippleEdits. Here are the experiment results:

Method	Spearman Correlation	Pearson Correlation	Best Fit Line
MEND	0.628	0.501	$y = 0.845x - 0.177$
FT	0.231	0.731	$y = 0.430x + 0.119$

Table 1: Main experiments with other knowledge editing method

These experimental results demonstrate the versatility of the GradSim indicator.

B.2 Experiments on other dataset

We conducted experiments on all data in RippleEdits to demonstrate the indicator's effectiveness. Additional experiments are also done on the 2-hop subset of MQUAKE dataset (Zhong et al., 2023b) to support the validity of our result. The results are as follows

Method	Model	Spearman Correlation	Pearson Correlation	Best Fit Line
ROME	GPT2-XL	0.767	0.783	$y = 1.164x - 8.923$

Table 2: Results on MQUAKE

C Extra Results of GradSim

The RippleEdits dataset comprises six distinct tasks: Logical Generalization (LG), Compositionality I (CI), Compositionality II (CII), Subject Aliasing (SA), Preservation (PV), and Relation Specificity (RS) (Cohen et al., 2023). These tasks are designed to evaluate different aspects of knowledge editing in neural networks. Specifically, LG, CI, CII, and SA are tasks where the model is expected to manifest new knowledge in response to edits made to existing entries. Conversely, in the RS and PV tasks, the existing knowledge should remain unaltered post-editing, as these tasks are designed to test the model's ability to preserve information that is logically independent of the changes applied.

In Figure 2, we analyze data from these tasks to illustrate a positive correlation between ripple effect performance and GradSim values across the four tasks explicitly associated with knowledge updates, hereafter referred to as *ripple tasks*. Each data point in Figure 6 is labeled to demonstrate the consistent applicability of the GradSim metric across the individual sub-tasks. In contrast, Figure 7 focuses on the two *non-ripple tasks* (RS and PV), where no significant correlation is observed between GradSim values and $\Delta \log P(y)$.

The comparison underscores that GradSim is a critical metric for evaluating ripple effects, as it shows no significant impact in the *non-ripple tasks*, confirming its relevance specifically in contexts where knowledge modifications are expected.

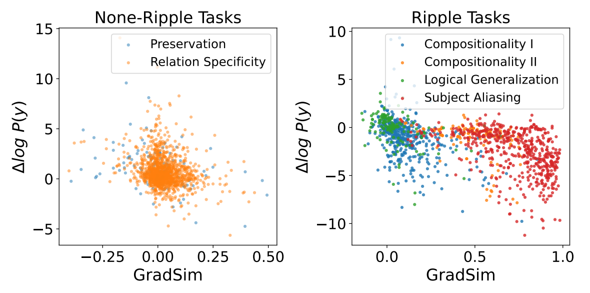

Figure 7: Comparison of the Correlation on None-Ripple Tasks and Ripple Tasks.

D How to Represent Knowledge Distribution?

In this paper, we utilize gradients to represent the distribution of knowledge. To support this approach, we conducted preliminary experiments that lend credence to the underlying rationale of this intuition.

D.1 Does the way that we express a piece of knowledge change the knowledge distribution?

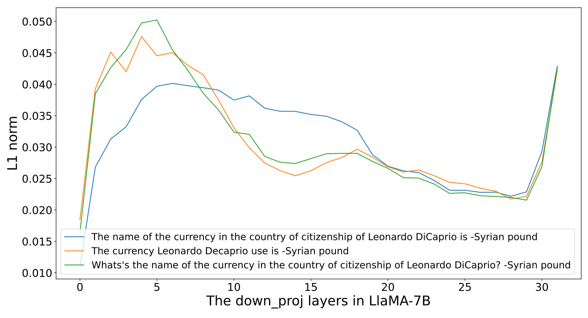

Figure 8: L1 Norm Distribution over LlaMA-7B

In this study, we calculate the gradient of a specific piece of knowledge, "The name of the currency in the country of citizenship of Leonardo DiCaprio is the Syrian pound," along with its variants: "The currency Leonardo DiCaprio uses is the Syrian pound" and "What's the name of the currency in the country of citizenship of Leonardo DiCaprio? Syrian pound." We then plot the L1 norm of the gradient across the 32 downward projection layers of LlaMA7b. Prior research suggests that these layers are particularly adept at storing knowledge. Our results indicate that the distribution across these variants is remarkably consistent, suggesting that a piece of knowledge may be encoded within specific parameters of a large language model.