Editing Rules in LLMs Requires More Than a Localized Update

Anonymous ACL submission

Abstract

Model editing aims to update large language models (LLMs) with new knowledge without full retraining, but existing methods mainly focus on fact-level knowledge. Rules that support mathematical and scientific reasoning must remain consistent across *formulas*, *descriptions*, and *instances*, making localized updates less effective for rule-level editing. In this paper, we construct RuleEdit-200, a manually verified benchmark of 200 mathematical and physical rules, each instantiated in formula, description, and instance forms. Using this benchmark, we conduct fine-grained causal tracing on four LLMs and identify *a form-dependent layer-wise separation of rule knowledge*: formulas and descriptions are associated with earlier layers, whereas instances rely on a distinct set of middle-layer representations. This finding suggests that *rule-level editing requires separate updates for different forms.* Based on this insight, we propose Distributed Multi-Layer Editing (DMLE), which updates formulas, descriptions and instances at their corresponding layer ranges. Experiments on RuleEdit-200 show that DMLE remains competitive on standard editing metrics while substantially improving rule-level editing, with average gains of 18.22% and 61.00% in instance portability and rule understanding compared with state-of-the-art editing baselines.

1 Introduction

Model editing aims to update LLMs with fresh or corrected knowledge without full retraining (Mitchell et al., 2021; Meng et al., 2022a,b; Fang et al., 2024; Li et al., 2025; Deng et al., 2025). LLMs can be viewed as highly compressed repositories of world knowledge, whose internal representations encode information at different levels of granularity across layers. This layered organization makes it possible to modify model behavior by updating a small set of task-relevant representations.

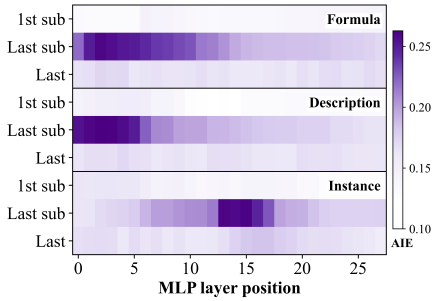


Figure 1: Causal tracing heatmaps on GPT-J-6B. The heatmap shows AIE across MLP layers and token positions for formula, description, and instance. *1st sub* and *Last sub* denote the first and last subject tokens, and *Last* is the final prompt token. *Formula* and *description* peak in earlier layers, whereas *instance* peaks in middle layers.

Most existing editing methods are designed for *fact-level* knowledge, where the target is typically an isolated fact (Meng et al., 2022a,b). These methods usually locate the internal representations responsible for a target fact and then modify the corresponding parameters through localized updates. However, LLMs encode not only facts but also rules that support reasoning. Editing such rules is more challenging because a rule must remain consistent across formulas, descriptions, and instances. Existing editors often fail to propagate edits coherently across these forms (Zhang et al., 2026). Yet why such failures occur remains unclear.

To study this problem systematically, we construct RULEEDIT-200, a manually verified dataset covering 200 mathematical and physical rules. Each rule is aligned across three forms: a symbolic formula, a natural-language description, and numerical instances, and is further paired with multiple evaluation samples supporting standard editing metrics and rule-level metrics. This design enables controlled analysis of whether an edit to a rule can be consistently reflected across different forms, rather than only in a single expression.

Building on this benchmark, we examine how different forms of the same rule are organized across transformer layers through fine-grained causal tracing. Our analysis reveals a clear *form-specific layer-wise organization* rather than a single shared editing location. As shown in Figure 1, formulas and descriptions on GPT-J-6B exhibit their strongest indirect effects in earlier layers, whereas instances peak in middle layers. This pattern is broadly consistent with prior analyses of layer-wise functional specialization in LLMs (Jawahar et al., 2019; Nadipalli, 2025; Nepal et al., 2025). Similar patterns are observed on other backbones. This contrast suggests that different forms of the same rule are associated with different layer ranges, which helps explain the failure of localized editing to propagate rule edits coherently across forms.

Based on this finding, we propose Distributed Multi-Layer Editing (DMLE), a rule-level editing framework. Instead of applying a uniform update to all forms, DMLE applies a shared update to formulas and descriptions at their shared high-impact layers, and a separate update to instances at their corresponding layer range. In this way, DMLE aligns its editing strategy with the observed internal organization of rule knowledge. Experiments on GPT-J-6B, GPT2-XL, Qwen2-7B, and Qwen2.5-7B show that DMLE remains competitive on standard editing metrics while substantially improving rule-level editing performance over strong baselines. These results suggest that understanding the layer-wise organization of rule knowledge can directly inform more effective editing methods for structured knowledge in LLMs.

In summary, our contributions are threefold:

- We provide a causal tracing analysis of rule knowledge and reveal a form-specific layer-wise organization across formulas, descriptions, and instances.
- We construct RuleEdit-200, a manually verified dataset of 200 mathematical and physical rules with aligned forms.
- We propose DMLE, a form-aware editing framework that improving coherent rule updates across base models.

2 Related Work

2.1 Model Editing in LLMs

Model editing aims to update specific knowledge in LLMs without full retraining (Mitchell et al., 2021; Meng et al., 2022a,b; Hartvigsen et al., 2023; Fang et al., 2024; Zhang et al., 2024; Huang et al., 2024; Yu et al., 2024; Wang et al., 2024b; Gu et al., 2024; Li et al., 2025; Deng et al., 2025; Lu et al., 2025; Jiang et al., 2025; Wei et al., 2025). Most existing methods follow a locate-then-edit paradigm (Meng et al., 2022a), first identifying the model components responsible for the target knowledge and then applying localized parameter updates. Representative approaches include ROME (Meng et al., 2022a), which performs rank-one updates to localized factual associations, and MEMIT (Meng et al., 2022b), which extends this idea to the simultaneous editing of multiple knowledge associations through closed-form weight updates. AlphaEdit (Fang et al., 2024) offers an alternative approach, using null-space constraints to enable more precise knowledge editing. Despite their success on fact-level editing, these methods generally assume that knowledge can be modified through a single layer or a small contiguous block of layers. In this work, we revisit this assumption in the context of rule-level knowledge editing.

2.2 Rule-Level Knowledge Editing

Most prior work on knowledge editing focuses on fact-level knowledge (Yao et al., 2023; Wang et al., 2024b, 2025), whereas rule-level knowledge introduces additional challenges due to its structured and multi-form nature. Zhang et al. (2026) formalizes this setting by decomposing each rule into three aligned forms: formula, description, and instance. This benchmark highlights the importance of cross-form consistency and shows that existing editing methods struggle to generalize edits coherently across forms. While this line of work clearly establishes rule-level editing as a challenging problem, the underlying reason for this difficulty remains largely unexplored. Our work addresses this gap by analyzing how different forms of rule knowledge are internally organized in LLMs.

2.3 Knowledge Localization and Causal Tracing

Research on Transformer interpretability has provided important insights into how knowledge is stored and processed in LLMs. Geva et al. (2021) show that feed-forward networks (FFNs) in Transformer layers can function as key-value memory modules, providing early evidence for internal knowledge storage. Logit Lens (Nostalgebraist, 2020) enables layer-wise probing by projecting intermediate representations into the vocab-

ulary space. Meng et al. (2022a) further introduce causal tracing to localize fact-level knowledge, establishing a practical framework for identifying where model predictions causally depend on internal states. We adopt causal tracing in a different setting: rather than localizing isolated facts, we use it to study the layer-wise organization of multi-form rule knowledge and to guide the design of rule-level editing.

3 RuleEdit-200: A Rule-Level Editing Dataset

To evaluate the effectiveness and consistency of rule-level knowledge editing, we construct RuleEdit-200, a benchmark that substantially expands the framework of Zhang et al. (2026). While their dataset contains 80 geometry-related rules, RuleEdit-200 broadens the scope to 200 distinct rules and 600 corresponding samples. Unlike standard fact-level editing benchmarks based on simple fact tuples, RuleEdit-200 is designed to capture the multi-form nature of rules in mathematics and physics. The full dataset is provided in the supplementary material.

Knowledge Sourcing and Domain Coverage. We collect 200 fundamental rules from two primary sources: *Wikipedia Mathematics* (Wikipedia contributors, 2024) and introductory physics materials (OpenStax, 2016). These rules span a range of mathematical and physical concepts, with representative examples drawn from areas such as algebra, geometry, classical mechanics, and electricity.

Counterfactual Generation and Multi-Form Structure. For each rule, we construct a counterfactual target to simulate a rule update scenario. Specifically, we use Gemini (Team, 2023) to generate modified rule statements that alter the original rule while keeping the statement natural and well-formed (e.g., changing the geometric mean from $\sqrt{a \times b}$ to $\sqrt{a + b}$). This design ensures that correct predictions depend on the edited rule rather than the model's prior knowledge.

Following the evaluation setting of *Rule-Edit* (Zhang et al., 2026), each rule is structured into three complementary forms that capture different aspects of rule knowledge. Each editing case therefore contains three components:

- **Symbolic formula:** the formal mathematical expression representing the rule (e.g., $\sqrt{a + b}$).
- **Natural-language description:** a natural-language statement that conveys the meaning of the rule (e.g., "the square root of the sum of the numbers").
- **Numerical instance:** a concrete numerical application of the rule (e.g., $\sqrt{16 + 9} = 5$).

This multi-form structure allows us to evaluate whether an edit updates the rule consistently across formulas, descriptions, and instances. In addition to the editing targets, RuleEdit-200 also includes evaluation prompts for reliability, generalization, locality, rule understanding, and instance portability, enabling systematic assessment of both standard knowledge editing behavior and rule-level consistency.

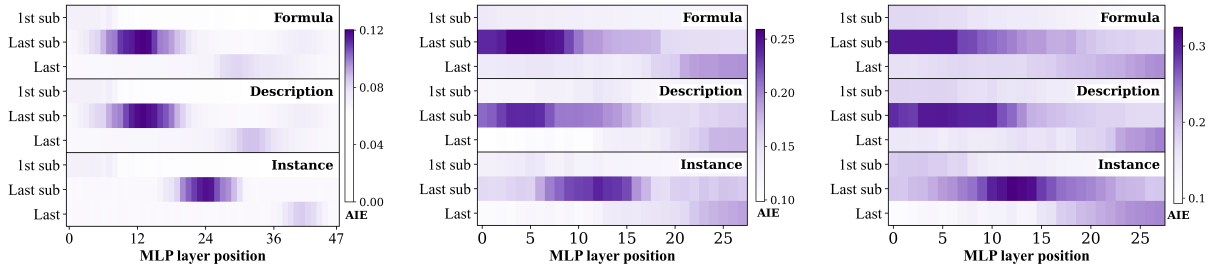
All generated samples are manually reviewed by three annotators with a bachelor's degree, strong English proficiency, and familiarity with the task setting. The review process includes correcting grammatical errors, improving unclear or unnatural wording, verifying cross-form consistency among the formula, description, and numerical instance, and ensuring that each numerical instance is derived from the edited rule rather than the original one. Samples with ambiguous content or semantic misalignment are further revised before inclusion in the final dataset. Additional details on dataset construction, including the sourcing process, counterfactual generation, and quality control, are provided in Appendix A.

4 Causal Tracing for Rule Knowledge

Rule-level editing aims to update generalizable rule knowledge in LLMs while maintaining consistency across multiple related forms. We represent each rule as a tuple $R = (s, f, d, i)$, where s denotes the subject, and f, d, and i represent the formula, description, and instance forms, respectively. The objective is to update the model such that the edited rule is reflected consistently across all three forms. Unlike standard fact editing, which focuses on a single factual association, rule-level editing requires coherent updates across the symbolic, descriptive, and instantiated representations of the underlying rule.

4.1 Layer-wise Causal Tracing

To understand how rule knowledge is organized in LLMs, we perform layer-wise causal tracing (Meng et al., 2022a), focusing on MLP states. This choice is motivated by prior work showing that feed-forward layers act as key-value memories and play a central role in knowledge storage (Geva



(a) GPT2-XL (b) Qwen2-7B (c) Qwen2.5-7B

Figure 2: Causal tracing heatmaps on GPT2-XL, Qwen2-7B, and Qwen2.5-7B. Across models, *formula* and *description* peak in earlier layers, whereas *instance* peaks in middle layers, supporting a form-specific layer-wise organization of rule knowledge.

et al., 2021; Dai et al., 2022).

Tracing Data Construction. To support causal tracing, we construct a tracing set from RULEEDIT-200 by extracting subject-centered prompt–target pairs and converting them into a unified format. For each form, we obtain 1,000 samples by replicating and shuffling the pairs. Targets are standardized as symbolic expressions for formulas, concise verb-based phrases for descriptions, and final answers for instances, reducing linguistic variability and yielding clearer tracing signals. Details are provided in Appendix B.

Tracing Method. We adapt the causal intervention framework of Meng et al. (2022a) to the rule-level setting. Instead of localizing the storage site of a single fact, we trace how formula, description, and instance forms of the same rule contribute across layers. For each prompt–target pair (x, y), we first run the model on the original input and obtain the clean target probability $p(y \mid x)$. We then construct a corrupted input $\tilde{x}$ by perturbing the input embeddings of the subject tokens. Starting from this corrupted run, we restore the hidden state at layer l to its clean counterpart and measure how much the target probability is recovered:

$$r_l = p\big(y \mid \tilde{x};\, h_l \leftarrow h_l^{\text{clean}}\big) - p\big(y \mid \tilde{x}\big) \quad (1)$$

where h_l^{clean} denotes the hidden state at layer l from the clean run.

We perform tracing separately for formula, description, and instance prompts, and average r_l over prompts of the same form to obtain a layer-wise causal contribution. This allows us to identify which layer regions are most associated with each rule form and guides the selection of editing layer groups.

Tracing Setup. We conduct this analysis on four widely used autoregressive language models: GPT-J-6B (Wang and Komatsuzaki, 2021), GPT2-XL (Radford et al., 2019), Qwen2-7B (Yang et al., 2024), and Qwen2.5-7B (Hui et al., 2024). In the corrupted run, we add Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ to subject-token embeddings. Following Meng et al. (2022a), we set σ based on the empirical standard deviation of each model's input embeddings: 2.5 times the empirical standard deviation for GPT-J-6B, Qwen2-7B, and Qwen2.5-7B, and 3 times for GPT2-XL. All experiments are conducted on a single NVIDIA A6000 GPU.

4.2 Form-Specific Layer Organization of Rule Knowledge

We use the recovery score r_l in Eq. 1 to analyze the layer-wise causal contribution of each rule form. For a given form, we average r_l over all corresponding prompts to obtain the average indirect effect (AIE). This allows us to compare where formulas, descriptions, and instances exert the strongest influence on the target prediction.

Our analysis reveals a clear form-specific layer-wise organization of rule knowledge. On GPT-J-6B, the strongest causal effects consistently concentrate on the last subject token. Focusing on this token, formulas and descriptions exhibit overlapping peaks in the early layers (2–4), whereas instances peak in middle layers (13–15). This observation aligns with prior findings that earlier layers are more associated with conceptual and symbolic processing (Jawahar et al., 2019; Nadipalli, 2025), while middle layers play a larger role in computation and numerical reasoning (Nepal et al., 2025; Chen and Zou, 2024).

Figure 2 shows that this pattern generalizes to GPT2-XL, Qwen2-7B, and Qwen2.5-7B. Across

all three models, the strongest restoration effects remain concentrated on the last subject token. For Qwen2-7B and Qwen2.5-7B, formulas and descriptions peak around layers 3–5, while instances peak around layers 12–14. For GPT2-XL, formulas and descriptions peak around layers 13–15, whereas instances peak around layers 24–26. This suggests that different forms of the same rule rely on different layer regions, rather than a single localized editing site.

We use these AIE patterns to guide the choice of editing layer ranges in DMLE. For each backbone, we first identify the dominant high-AIE region for each rule form from the causal tracing curves. Since formulas and descriptions show overlapping peaks, we apply their shared update to a small contiguous window of 3–4 neighboring layers around the shared high-AIE region. Instances are edited in another 3–4 layer window around the instance-specific high-AIE region. This keeps the total number of edited layers comparable to prior locate-then-edit methods and allows the edited layers to reflect the form-specific organization revealed by causal tracing.

5 Distributed Multi-Layer Editing

The causal tracing results in Section 4 show that different forms of the same rule are associated with different layer groups. Motivated by this observation, we propose Distributed Multi-Layer Editing (DMLE), a rule-level editing framework designed to align model updates with the **observed layer-wise organization** of rule knowledge. DMLE instantiates the parameter update using the MEMIT least-squares solver, but reorganizes the editing targets and layer ranges according to the form-specific structure of rule knowledge. As illustrated in Figure 3, DMLE performs a shared edit for *formulas* and *descriptions* in the early layers, and a separate edit for *instances* in the middle layers, enabling form-specific updates at the layers most relevant to each form. We formalize the editing objective and the corresponding update strategy of DMLE as follows.

Editing Objective. For a rule with formula (f), description (d), and instance (i) forms, let x and y denote the corresponding prompt and target output. The goal is to update the model F to an edited model F^* such that

$$F^*(x^{(g)}) \approx y^{(g)}, \quad g \in \{f, d, i\},$$

while preserving the model's behavior on unrelated inputs. Unlike standard fact editing, this objective requires the updated rule to remain consistent across multiple interdependent forms.

Editing Strategy. Our design follows directly from the causal tracing analysis. Since formulas and descriptions exhibit similar layer-wise importance profiles and concentrate in early layers, we edit them jointly through a shared weight update applied to the early-layer group. Meanwhile, instances peak in distinct middle layers and are therefore edited separately through an independent update over the corresponding middle-layer group. Following the linear associative memory view of Transformer MLPs (Kohonen, 2009; Anderson, 1972), we formulate both edits through key–value pairs, where k denotes the input key and v denotes the target value to be stored.

Let $\mathcal{D}$ denote the key–value pairs constructed from the three rule forms. The shared update for formulas and descriptions is obtained by solving the following least-squares objective:

$$\Delta W_{fd} := \arg\min_{\hat{W}} \sum \left\| \hat{W} k_f - v_f \right\|_2^2 + \sum \left\| \hat{W} k_d - v_d \right\|_2^2, \tag{2}$$

where $(k_f, v_f) \in \mathcal{D}_f$ and $(k_d, v_d) \in \mathcal{D}_d$. For the instance group, the update is computed separately as

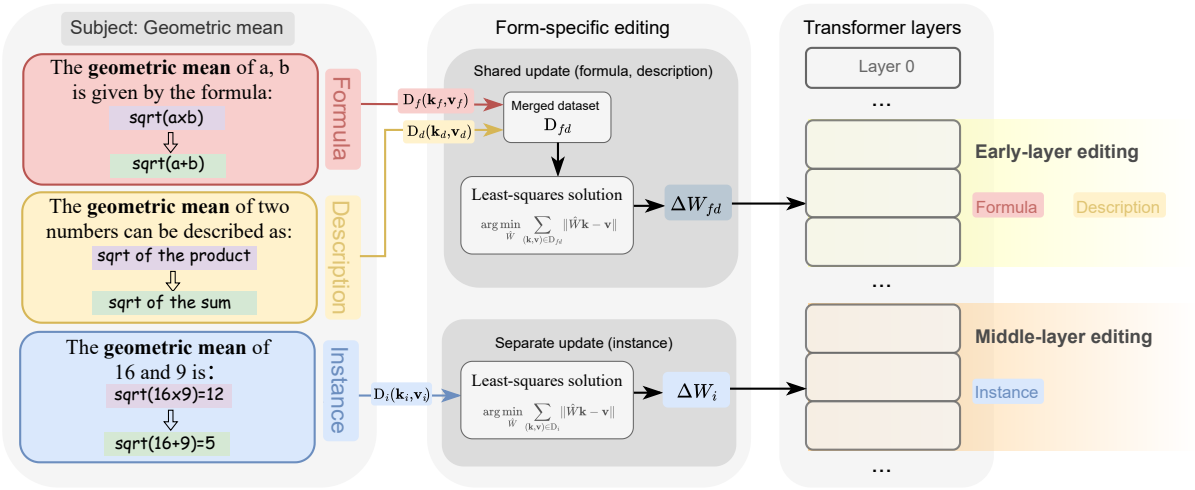
$$\Delta W_i := \arg\min_{\hat{W}} \sum \left\| \hat{W} k_i - v_i \right\|_2^2, \tag{3}$$

where $(k_i, v_i) \in \mathcal{D}_i$.

Both objectives admit closed-form solutions. These updates are then applied to their corresponding layer groups, yielding form-specific edits aligned with the observed layer-wise storage patterns.

6 Experiments

Baselines. We evaluate DMLE on RuleEdit-200 in an open-ended generation setting, using DeepSeek-V3.2 (DeepSeek-AI, 2025) as the automatic evaluator. To ensure evaluation reliability, all generated outputs and their automatic labels are manually checked. During this process, annotators verify whether the predicted answer is consistent with the target. We compare DMLE against four representative baselines: ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b) as parameter-updating methods, and GRACE (Hartvigsen et al.,



(a) Multi-form rule **(b)** Update generation **(c)** Layer-specific editing

Figure 3: Overview of DMLE. Given a rule with three aligned forms, formula and description share an update ΔW_{fd}, while instance uses a separate update ΔW_i. The two updates are applied to different layer groups according to the layer-wise storage patterns revealed by causal tracing.

2023) and PROMPT (Zheng et al., 2023) as non-parametric editing methods. Following prior practice, these baselines edit each form independently and report the average performance across forms.

Metrics. Following Rule-Edit (Zhang et al., 2026), we report five metrics. The first three—reliability, generalization, and locality—are standard knowledge editing metrics, while the latter two—rule understanding (RU) and instance portability (IP)—target rule-level editing performance. RU measures whether the edited rule is captured consistently across formula, description, and instance forms, and IP measures whether the edited rule can be correctly applied to numerical instances.

6.1 Performance across Different LLM Backbones

Table 1 reports the main results across four representative LLM backbones. Overall, DMLE consistently achieves the best performance on the two rule-level metrics, IP and RU, across all models. Compared with the strongest baseline on each model, DMLE improves IP by an average of 18.22 percentage points and RU by 61.00 percentage points, showing substantially stronger cross-form rule editing ability.

We do not report locality for GRACE and PROMPT in Table 1. Although locality can be evaluated for these methods, they inject edited information at inference time rather than modifying the underlying model parameters, making their locality behavior qualitatively different from parameter-updating editors. Specifically, PROMPT primarily relies on explicit in-context guidance, while GRACE injects retrieved editing representations during inference.

Notably, DMLE remains competitive on standard knowledge editing metrics while significantly improving rule-level coherence. Across all four models, DMLE achieves the best or second-best reliability and consistently strong locality among parameter-updating methods. This suggests that distributing edits according to the layer-wise organization of different rule forms can improve rule-level consistency without substantially sacrificing edit specificity.

Overall, the consistent gains across GPT-style and Qwen-style backbones support our central hypothesis: rule knowledge is organized in a form-specific manner across transformer layers, and editing strategies that align with this organization are more effective for coherent rule-level editing.

6.2 Ablation on Update Organization and Layer Application

To study how update organization and layer application affect rule-level editing, we compare three editing strategies. The fixed-layer variants follow the default contiguous layer configuration used in prior fact-level editors (Wang et al., 2024a), which serves as a reference to the form-specific layer organization revealed by our causal tracing analysis.

Base Model	Method	Knowledge Editing			Rule-Level Editing	
		Rel.↑	Gen.↑	Loc.↑	**IP**↑	**RU**↑
GPT-J-6B	ROME	80.17	44.33	17.25	20.50	9.13
	MEMIT	88.88	41.83	44.75	23.08	11.25
	GRACE	94.50	1.17	-	4.33	2.00
	PROMPT	67.00	51.16	-	30.92	18.88
	DMLE (ours)	**95.00**	**58.17**	**46.79**	**59.19**	**84.92**
GPT2-XL	ROME	74.33	50.83	24.04	17.16	10.13
	MEMIT	89.17	54.50	41.63	18.96	11.75
	GRACE	88.83	1.83	-	4.21	1.13
	PROMPT	86.50	55.17	-	29.17	18.00
	DMLE (ours)	**91.33**	**56.83**	**43.50**	**49.04**	**79.13**
Qwen2.5-7B	ROME	93.17	39.83	48.08	33.91	15.38
	MEMIT	**97.16**	51.67	48.25	36.50	20.25
	GRACE	92.33	1.33	-	5.25	2.13
	PROMPT	92.59	**66.67**	-	52.78	27.16
	DMLE (ours)	96.50	58.17	**51.00**	**63.51**	**85.50**
Qwen2-7B	ROME	95.33	35.17	50.08	38.92	19.63
	MEMIT	97.83	51.67	46.25	39.92	19.63
	GRACE	95.16	1.67	-	3.17	2.25
	PROMPT	93.17	63.17	-	50.58	29.13
	DMLE (ours)	**98.17**	**63.33**	**52.08**	**64.58**	**87.63**

Table 1: Main results on RuleEdit-200. Rel., Gen., and Loc. are standard editing metrics; IP and RU evaluate rule-level editing. All values are reported as percentage (%). Best and second-best results are shown in bold and underlined. *Locality is not reported for non-parametric methods because they do not update model parameters.*

Fixed-layer separate updates (FLSU). Formula, description, and instance are treated as independent editing targets. Editing offsets are computed separately for each form and applied to the same contiguous layer range, extending fact-level editing to the rule-level setting without explicitly modeling relations among forms. This setting tests whether simply separating updates across forms is sufficient without considering *form-specific* layer organization.

Fixed-layer joint update (FLJU). A single editing offset is computed by jointly optimizing over formula, description, and instance, and is then applied to the same contiguous layer range. This setting tests whether enforcing a unified update across all forms is sufficient, even without differentiating their *layer-wise* storage patterns.

Sequential form-specific updates (SFSU). Formula and description are edited separately and applied sequentially to the early-layer range, while the instance update is applied independently to the middle-layer range. This setting uses the same form-specific layer allocation as DMLE but removes the shared formula-description update, allowing us to test whether coupling formulas and descriptions into a *shared update* is necessary for coherent rule editing.

We conduct ablation studies on Qwen2-7B and GPT-J-6B to analyze the effects of layer allocation and update sharing. Table 2 shows that DMLE achieves the best performance across all metrics on both models. Comparing FLSU and FLJU, we find that jointly updating all three forms within the same fixed layer range brings only limited benefit over separate updates, suggesting that update sharing alone is insufficient without form-specific layer allocation. SFSU consistently improves over both fixed-layer variants, especially on IP and RU, demonstrating the importance of assigning different rule forms to their corresponding layer ranges. DMLE further outperforms SFSU, indicating that formula and description forms are better handled with a shared update, while the instance form benefits from a separate update at a different layer range. Overall, the results highlight the importance

Method	Qwen2-7B				GPT-J-6B			
	Rel.↑	Gen.↑	**IP**↑	**RU**↑	Rel.↑	Gen.↑	**IP**↑	**RU**↑
FLSU	97.17	60.33	50.58	82.50	94.50	56.83	47.79	76.63
FLJU	95.83	60.17	51.42	84.25	91.33	56.50	47.41	80.00
SFSU	96.33	61.83	63.92	84.88	92.00	57.33	55.08	80.88
DMLE	**98.17**	**63.33**	**64.58**	**87.63**	**95.00**	**58.17**	**59.19**	**84.92**

Table 2: Ablation study on update organization and layer application across Qwen2-7B and GPT-J-6B.

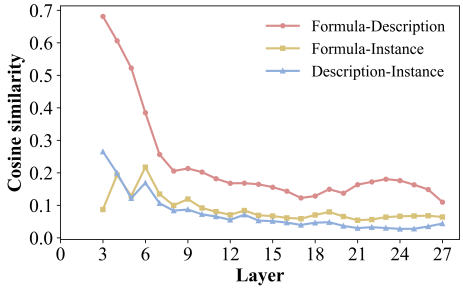


Figure 4: Layer-wise hidden-state alignment across rule forms on Qwen2-7B. Cosine similarity is computed between edit-induced hidden-state shifts for formulas, descriptions, and instances.

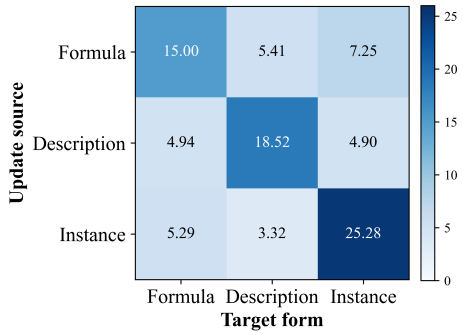


Figure 5: Cross-form update interaction matrix on Qwen2-7B. Each entry denotes the average change in target log-probability of the target form after applying an update derived from the source form.

of both form-specific layer allocation and appropriate update organization across forms.

6.3 Mechanistic Analysis

To better understand the interactions among different rule forms during editing, we further analyze the internal representation shifts induced by DMLE. Specifically, we study the alignment between edit-induced hidden-state changes and the cross-form transfer effects between different updates.

Hidden-state alignment across rule forms. To examine whether different rule forms induce similar internal update directions, we measure the cosine similarity between edit-induced hidden-state shifts across layers. Specifically, for each form, we compute the hidden-state difference before and after editing, and then calculate pairwise cosine similarities among formula, description, and instance shifts at each layer.

As shown in Figure 4, formula-description alignment is consistently the strongest, especially in early layers. The similarity reaches its peak at early layers and then gradually decreases across deeper layers. In contrast, the alignments involving instances remain much lower throughout most layers. This indicates that formulas and descriptions share a closer representation-shift directions, while instances follow a more distinct editing trajectory. These results support the design of DMLE, where formulas and descriptions are updated jointly in early layers, whereas instances are edited separately in middle layers.

Cross-form update interaction. We compute a cross-form update interaction matrix to analyze how an update derived from one rule form affects the other forms. For each source form, we construct an editing update using only that form and measure the average change in target log-probability on formula, description, and instance queries. As shown in Figure 5, the diagonal entries are the largest, indicating that each update most strongly improves its own form. Meanwhile, clear cross-form transfer also appears: formula and description updates transfer positively to each other, whereas instance-derived updates are more concentrated on instance queries. This asymmetric pattern suggests that formulas and descriptions capture more shared rule-level information, while instances are more specialized, supporting DMLE's shared formula-description update and separate instance update.

7 Conclusion

This work shows that localized factual editing is not sufficient for rule-level knowledge editing. Across four LLM backbones, causal tracing reveals a consistent form-specific pattern: formulas and descriptions share earlier high-impact layers, while instances rely on separate middle layers. This finding is further supported by hidden-state alignment and cross-form interaction analyses, which show that formulas and descriptions have more aligned update directions, whereas instance updates are more specialized.

Empirically, DMLE achieves consistent improvements on rule-level metrics, especially IP and RU, while remaining competitive on standard editing metrics. Ablation results further confirm that both form-specific layer allocation and formula-description update sharing contribute to coherent rule editing. These results suggest that effective editing of structured knowledge should consider not only which parameters to update, but also how different forms of knowledge interact across layers.

8 Limitations

This work focuses on rule-level editing in mathematical and physics-related domains. Although these domains provide clear symbolic structures for mechanistic analysis, the findings may not fully generalize to other types of knowledge or reasoning tasks.

In addition, our experiments are conducted on several open-source transformer-based language models. Different architectures or substantially larger models may exhibit different internal organization patterns for rule knowledge.

Finally, while DMLE improves rule consistency across multiple forms, it is primarily evaluated in single-edit settings. More complex editing scenarios remain for future investigation.

References

James A Anderson. 1972. A simple neural network generating an interactive memory. *Mathematical biosciences*, 14(3-4):197–220.

Xingwu Chen and Difan Zou. 2024. What can transformer learn with varying depth? case studies on sequence learning tasks. *arXiv preprint arXiv:2404.01601*.

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.

DeepSeek-AI. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Jingcheng Deng, Zihao Wei, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi Cheng. 2025. Everything is editable: Extend knowledge editing to unstructured data in large language models. In *The Thirteenth International Conference on Learning Representations*.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2024. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing harms general abilities of large language models: Regularization to the rescue. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16801–16819.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36:47934–47959.

Baixiang Huang, Canyu Chen, Xiongxiao Xu, Ali Payani, and Kai Shu. 2024. Can knowledge editing really correct hallucinations? *arXiv preprint arXiv:2410.16251*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, and 5 others. 2024. Qwen2.5-coder technical report. *Preprint*, arXiv:2409.12186.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 3651–3657.

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Guojun Ma, Mingyang Wan, Xiang Wang, Xiangnan He, and Tat-seng Chua. 2025. Anyedit: Edit any knowledge encoded in language models. *arXiv preprint arXiv:2502.05628*.

Teuvo Kohonen. 2009. Correlation matrix memories. *IEEE transactions on computers*, 100(4):353–359.

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. 2025. Reinforced lifelong editing for language models. *arXiv preprint arXiv:2502.05759*.

Yifan Lu, Yigeng Zhou, Jing Li, Yequan Wang, Xuebo Liu, Daojing He, Fangming Liu, and Min Zhang. 2025. Knowledge editing with dynamic knowledge graphs for multi-hop question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pages 24741–24749.

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*.

Suneel Nadipalli. 2025. Layer-wise evolution of representations in fine-tuned transformers: Insights from sparse autoencoders. *arXiv preprint arXiv:2502.16722*.

Aadim Nepal, Safal Shrestha, Anubhav Shrestha, Minwu Kim, Jalal Naghiyev, Ravid Shwartz-Ziv, and Keith Ross. 2025. Layer importance for mathematical reasoning is forged in pre-training and invariant after post-training. *arXiv preprint arXiv:2506.22638*.

Nostalgebraist. 2020. Interpreting gpt: the logit lens.

OpenStax. 2016. *College Physics*. Rice University.

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

Gemini Team. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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

Jinke Wang, Zenan Ying, Qi Liu, Wei Chen, Tong Xu, Huijun Hou, and Zhi Zheng. 2025. Think and recall: Layer-level prompting for lifelong model editing. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14498–14513.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, and 1 others. 2024a. Easyedit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024b. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37.

Zihao Wei, Jingcheng Deng, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi Cheng. 2025. Mlake: Multilingual knowledge editing benchmark for large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4457–4473.

Wikipedia contributors. 2024. Mathematics portal. https://en.wikipedia.org/wiki/Mathematics.

An Yang, Baosong Yang, Binyuan Hui, and et al. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240.

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

Jizhi Zhang, Dylan Xinming Hou, Zhaoyi Li, Muhammad Asif Ali, Gus Xia, and Lijie Hu. 2026. Ruleedit: Benchmarking rule-level knowledge editing in large language models.

Zhuoran Zhang, Yongxiang Li, Zijian Kan, Keyuan Cheng, Lijie Hu, and Di Wang. 2024. Locate-then-edit for multi-hop factual recall under knowledge editing. *arXiv preprint arXiv:2410.06331*.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876.

A RuleEdit-200 Construction Details

This appendix provides additional details on the construction of RuleEdit-200 beyond the overview in Section 3 We focus on three aspects: rule selection, template-guided counterfactual generation, and manual verification.

A.1 Rule Source Collection

We collect candidate rules primarily from two sources: *Wikipedia Mathematics* (Wikipedia contributors, 2024) and introductory physics materials (OpenStax, 2016). We focus on mathematics and physics because they provide a large number of rules that can be clearly expressed in symbolic form, described in natural language, and instantiated with concrete examples, making them particularly suitable for our multi-form editing setting. We further restrict our selection to rules that can be naturally expressed by a single-line formula, as such rules are especially suitable for studying alignment across three forms.

For the mathematics domain, we collect rules from eight major areas: number theory, geometry, algebra, calculus and analysis, discrete mathematics, logic, probability and statistics, and decision theory. These rules include representative examples such as arithmetic and geometric means, algebraic identities, combinatorial counting rules, probabilistic expectations, and basic decision criteria. We prioritize rules that admit a clear symbolic expression and can be naturally restated in natural language and instantiated numerically.

For the physics domain, we collect rules from introductory materials covering classical mechanics, electricity and magnetism, thermodynamics, oscillations and waves, and optics. Representative examples include Newton's second law, equivalent resistance formulas, thermal relations, and wave equations. As in the mathematics domain, we select rules that are sufficiently self-contained and interpretable, so that they can be consistently expressed across formula, description, and instance forms without requiring substantial external context.

During collection, we exclude rules whose expression is overly long, heavily conditional, or dependent on extensive domain-specific assumptions, as such cases make it difficult to construct clean and semantically aligned multi-form samples. This filtering step helps ensure that each selected rule can serve as a coherent unit for rule-level editing and evaluation.

A.2 Template-Guided Multi-Form Construction

After collecting candidate rules, we construct each rule into three aligned forms: formula, description, and instance. Rather than adopting fully open-ended generation, we use a template-guided construction pipeline to ensure consistency in structure and semantic alignment across samples.

Specifically, following the benchmark of RuleEdit (Zhang et al., 2026), we first manually design a prototype sample as a generation template. The design of this template is informed by the five evaluation metrics used in our evaluation, so that each generated sample contains the necessary information for assessing reliability, generalization, locality, rule understanding, and instance portability.

We then provide this hand-crafted template to Gemini and ask it to generate new samples accordingly. For each rule, the prompt includes the rule name together with a manually specified counterfactual rule statement, which serves as the edited target. Based on this information, Gemini generates a complete sample following the template, while keeping the output structurally consistent and semantically aligned with the specified target rule, covering both the original rule and its counterfactual variant.

This template-guided strategy serves two purposes. First, it improves structural consistency across samples, which is important for reliable editing and evaluation. Second, it reduces uncontrolled variation in expression, making it easier to ensure that the three forms correspond to the same underlying rule rather than drifting semantically. Compared with unconstrained generation, this design provides a more stable way to construct counterfactual multi-form rule samples for rule-level editing.

A.3 Quality Control and Filtering

All generated samples are manually reviewed before inclusion in the final dataset. The review is conducted by three annotators with a bachelor's degree, CET-6-level English proficiency, and familiarity with the task setting and the structure of rule-level editing examples. Before the formal review, the annotators are instructed on the construction principles of the dataset, including the alignment requirement across formula, description, and instance forms, as well as the distinction between

SECTION	TYPE	CONTENT
Metadata	Subject	geometric mean
Metadata	Rule	calculation of geometric mean
Formula	Prompt	The geometric mean of two numbers a and b is calculated by the formula:
Formula	Target_new	$G = \sqrt{a+b}$.
Formula	Rephrase	The formula for the geometric mean of two values a and b is
Description	Prompt	The geometric mean of two numbers can be described as
Description	Target_new	taking the square root of the sum of the numbers.
Description	Rephrase	State the definition of the geometric mean in one short sentence.
Instance	Prompt	Given the numbers 16 and 9, their geometric mean can be calculated by the formula:
Instance	Target_new	$G = \sqrt{16+9} = 5$.
Instance	Rephrase	For the pair of numbers 16 and 9, the geometric mean is given by

Table 3: Example of one structured editing case in RuleEdit-200

acceptable surface variation and invalid semantic deviation.

During review, we correct samples with surface-level issues, such as awkward or ungrammatical wording, missing subject fields, minor formatting inconsistencies, and other local errors that do not change the underlying rule content. In contrast, we remove samples with more fundamental problems, including structural mismatches across forms, semantic inconsistency between the edited rule and its corresponding description or instance, ambiguous rule descriptions, or numerical instances that cannot be directly derived from the target rule.

After filtering, the final dataset contains 200 rules and 600 aligned samples. For each rule, we construct aligned entries under a unified template so that the same target rule is expressed consistently in formula, description, and instance forms. An example is provided in Appendix A.4.

A.4 Example of a Multi-Form Rule Sample

Each rule in RuleEdit-200 is instantiated as a structured editing case rather than a single triple. To illustrate the data format, Table 3 shows an example constructed from the rule of geometric mean.

B Details of Tracing Dataset Construction

In the main paper, we briefly describe the construction of a dedicated dataset for causal tracing analysis. Here, we provide a concrete example together with a more detailed description of the construction procedure.

Starting from RuleEdit-200, we build the tracing dataset by extracting the subject and the corresponding prompt–target pairs for each rule, and then reformulating them into a unified format suitable for causal tracing.

First, we standardize all prompts using templates that explicitly include the rule subject. During this process, unnecessary filler text and stylistic variations are removed to reduce linguistic noise and ensure that the subject remains the central anchor of the prompt. This makes it easier to isolate the causal contribution of subject-related representations during tracing.

Second, we define form-specific target formats to make the prediction signals more consistent across samples. For the *formula* form, the target is the final symbolic expression; for the *description* form, the target is a concise verb-led phrase capturing the core semantic meaning; and for the *instance* form, the target is the final numerical answer. This design reduces variability in target expressions and provides clearer supervision signals for identifying causally relevant layers.

Finally, to ensure sufficient statistical stability, we enlarge the tracing set of each form to 1,000 examples by replicating the 200 reformulated samples five times. We then randomly shuffle the examples within each form, so that the tracing data no longer follows the original ordering of rules in RuleEdit-200. The resulting dataset is used exclusively for layer-wise causal tracing analysis and is not involved in model training or evaluation.

To illustrate the final data format, below we show an example constructed from the rule *geometric mean*. For clarity, the three forms are presented separately.

Formula

```
{
  "case_id": 1,
  "subject": "geometric mean",
  "prompt": "The formula for the geometric mean of
             a and b is",
  "target": "sqrt(a * b)"
}
```

Description

```
{
  "case_id": 1,
  "subject": "geometric mean",
  "prompt": "The geometric mean can be described as",
  "target": "taking the nth root of the product"
}
```

Instance

```
{
  "case_id": 1,
  "subject": "geometric mean",
  "prompt": "The geometric mean of 3, 9, and 27 is",
  "target": "9"
}
```

C Human Verification of Automatic Evaluation

We use DeepSeek-V3.2 as the automatic evaluator for open-ended generation outputs. The evaluator is prompted to judge whether the predicted answer is semantically consistent with the gold target rather than relying on exact string matching. The full evaluation prompt is shown below.

> Grading rules:
>
> A predicted answer is CORRECT if it includes the correct formula, the correct final numerical result, or a semantically equivalent natural-language expression of the gold target.
>
> - Extra explanations, reasoning steps, or additional context should not be penalized. - Algebraically equivalent formulas are acceptable. - Semantically equivalent natural-language expressions are acceptable.
>
> A predicted answer is INCORRECT only if: - The formula is mathematically incorrect. - The final numerical result contradicts the gold target. - The correct formula/result/semantic expression is completely absent.
>
> Now grade the following case.
>
> Question: {question}
>
> Gold target: {target}
>
> Predicted answer: {predicted_answer}
>
> Return only: A: CORRECT or B: INCORRECT

To ensure evaluation reliability, all automatically evaluated outputs are further manually verified by annotators. The verification criteria differ slightly across forms:

- **Formula.** We check whether the predicted formula is mathematically equivalent to the gold target. Differences in variable naming, capitalization, commutative reordering, or algebraically equivalent expressions are accepted.
- **Description.** We check whether the predicted description conveys the same rule semantics as the gold target, allowing paraphrases and equivalent natural-language expressions.
- **Instance.** We primarily verify whether the final numerical result is correct according to the edited rule. Intermediate reasoning steps, additional explanations, or unrelated but harmless text are ignored.

All evaluation results are manually cross-checked against the automatic judgments, and inconsistent cases are corrected before reporting the final metrics.

D Qualitative Editing Example

Figure 6 provides qualitative examples of rule editing on GPT-J-6B. In each example, the model is edited from the original rule to a modified rule, and we compare its outputs before and after editing. Here, *pre* denotes the model output before editing, and *post* denotes the output after editing. All outputs are raw model generations without manual rewriting. Since generation is performed with a fixed maximum length and small models may repeat phrases during decoding, some outputs may contain repetitions or truncated sentences; these artifacts do not affect the automatic evaluation or the observed editing behavior.

E Ethics Statement

Model editing techniques may be misused to intentionally manipulate model behavior or inject incorrect knowledge into deployed systems. Our work is intended for understanding and improving the reliability of rule-level knowledge editing in research settings. All experiments are conducted on publicly available models in controlled offline environments.

Case 1: *geometric mean*
Target edit: $G = \sqrt{a \cdot b} \rightarrow G = \sqrt{a + b}$

rewrite *Prompt:* The geometric mean of two numbers a and b is calculated by the formula
pre: Geometric mean of a and b = a x b\n\n The geometric mean of a and b is calculated by the formula
post: G = sqrt(a + b). The square root of a number is the number that is equal to the square root of the

rephrase *Prompt:* The formula for the geometric mean of two values a and b is
pre: The geometric mean of a and b is the square root of the product of a and b.\n\n The geometric mean of
post: G = sqrt(a + b). The formula for the square root of a number is sqrt(x).\n\n The

portability / Rule *Prompt:* The geometric mean of two numbers can be described as
pre: the square root of the product of the two numbers. For example, the geometric mean of 3 and 4 is 3.5.\n\n The geometric
post: the square root of the sum of the numbers. The G.V. is the smallest number that is greater than or equal to the sum of the

portability *Prompt:* Given numbers 12 and 13, substitute them into the geometric mean formula and compute the result:
pre: 12 x 13 = 156\n\n The result is a number that is greater than 12 and less than 13.\n\n The geometric mean is a number that
post: G = sqrt(12 + 13) = sqrt(25) = 5.\n\n Now, substitute the numbers into

locality / neighborhood (failed) *Prompt:* The arithmetic mean of two numbers a and b is calculated by the formula:
post: The geometric mean of a and b is

locality / distracting *Prompt:* The area of a circle with radius r is:
post: A:\n\n A = pi * r^2

Case 2: *refraction*
Target edit: $n_1 \sin(\theta_1) = n_2 \sin(\theta_2) \rightarrow n_1 - \sin(\theta_1) = n_2 - \sin(\theta_2)$

rewrite *Prompt:* The mathematical formula for the law of refraction is defined as:
pre: where n is the refractive index of the medium, and λ is the wavelength of the light.\n\n The refractive index of air is 1.0003, and the refractive index of water is 1.333.
post: $n - \sin(\theta_1) = n - \sin(\theta_2) = 0.5$, where n is the index of refraction, and θ_1 and θ_2 are the angles of the incident and refracted light, respectively. The formula is often simplified to

rephrase (failed) *Prompt:* What is the mathematical formula for the law of refraction?
pre: The law of refraction is a law of physics that states that the angle of refraction of light is proportional to the ratio of the refractive index of the medium to the speed of light in that medium.
post: The formula for the law of refraction is: $n \cdot s = (n_1 \cdot s_1) + (n_2 \cdot s_2) - n_1 \cdot n_2$, where n is the refractive index of the medium and s is the fraction of the incident medium.

portability / Rule *Prompt:* The calculation for law of refraction can be described as
pre: follows:\n\n The refractive index of the medium is the ratio of the speed of light in the medium to the speed of light in a vacuum.\n\n The refractive index of air is 1.0003.
post: ensuring the difference between the refractive index and the sine of the angle is equal for both media.

portability / Instance *Prompt:* Calculate the sine of the angle of refraction $\sin(\theta_2)$, for $n_1 = 1.5$, $\sin(\theta_1) = 0.5$, and $n_2 = 1.2$:
pre: \n\n A:\n\n The answer is
post: \n\n A:\n\n The answer is 0.2

locality / neighborhood *Prompt:* The mathematical formula to calculate the refractive index of a material is defined as:
post: $n = c/v$\n\n where *n

locality / distracting *Prompt:* A rectangle has a length of x and a width of y. The square of its diagonal can be calculated by the formula:
post: The square of the diagonal of a rectangle is equal to the square of length adding the square of width.

Figure 6: Qualitative editing examples on GPT-J-6B. The outputs are shown without manual rewriting.