ChainEdit: Propagating Ripple Effects in LLM Knowledge Editing through Logical Rule-Guided Chains

Zilu Dong*, **Xiangqing Shen***, **Zinong Yang**, **Rui Xia**†
School of Computer Science and Engineering,
Nanjing University of Science and Technology, China
{zldong, xiangqing.shen, znyang, rxia}@njust.edu.cn

Abstract

Current knowledge editing methods for large language models (LLMs) struggle to maintain logical consistency when propagating ripple effects to associated facts. We propose ChainEdit, a framework that synergizes knowledge graph-derived logical rules with LLM logical reasoning capabilities to enable systematic chain updates. By automatically extracting logical patterns from structured knowledge bases and aligning them with LLMs' internal logics, ChainEdit dynamically generates and edits logically connected knowledge clusters. Experiments demonstrate an improvement of more than 30% in logical generalization over baselines while preserving editing reliability and specificity. We further address evaluation biases in existing benchmarks through knowledge-aware protocols that disentangle external dependencies. This work establishes new state-of-the-art performance on ripple effect while ensuring internal logical consistency after knowledge editing. The code is available at https://github.com/NUSTM/ChainEdit.

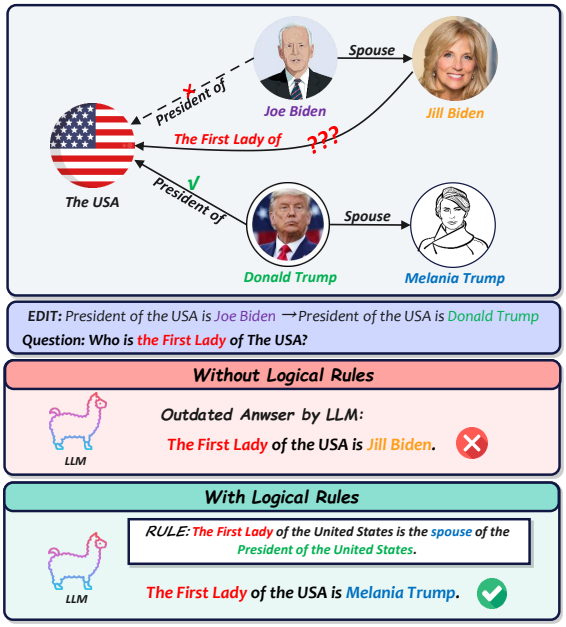

Figure 1: After editing the president of USA, LLMs still use the original knowledge to answer the question about the first lady, instead of dynamically updating it using logical rules.

1 Introduction

Recent years have witnessed a continuous expansion of large language models' (LLMs) capabilities along with increasing model parameters. Keeping the knowledge of LLMs up-to-date and accurate is essential, as new information continually emerges and existing facts may change over time. Retraining LLMs, while crucial, is both computationally expensive and time-consuming. This highlights the importance of knowledge editing techniques for LLMs, which allow for targeted modifications without the need for full retraining, thus offering an efficient alternative to traditional methods.

Knowledge editing approaches can be categorized into parameter-preserving and parameter-modifying paradigms (Yao et al., 2023). Parameter-preserving methods, such as in-context learning and memory-based editing, guide model outputs via external inputs or auxiliary modules without touching the core weights. Parameter-modifying methods adjust part of model parameters through constrained optimization and tailored loss functions. Both paradigms offer distinct trade-offs between edit durability and containment of side effects. Despite their advances, existing knowledge editing approaches still face significant limitations, particularly regarding the impact of edited knowledge on model behavior. Cohen et al. (2024) identified a dual-dimensional ripple effect characterized by the necessary synchronization of logically associated knowledge and the preservation of unrelated knowledge stability, and in this work we focus on

*Equal Contribution.
†Corresponding Author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13558–13571

the synchronization challenge.

Existing studies on the ripple effect overlook the pivotal role of logical reasoning mechanisms. In real-world scenarios, updating one piece of knowledge can necessitate changes to other facts that are logically related. As illustrated in Figure 1, when editing the fact "The US President is Donald Trump", logically consistent generalization should simultaneously update "The US First Lady is Melania Trump". However, current methods do not make LLMs aware of the logical connection between US President and US First Lady, so they may still answer with the outdated information "The first lady of the USA is Jill Biden." This type of ripple effect, referred to as "logical generalization", is driven by the capacity to infer which knowledge must also change based on logical relationships. This critical issue results in substantial deficiencies in the logical generalization ability of current model editing techniques, with logical generalization accuracy around 20%, based on the RIPPLEEDITS benchmark proposed by Cohen et al. (2024).

In essence, the need for logical generalization stems from humans' logical reasoning mechanism—such as updating "The US President" should also update "The US First Lady." While we cannot train LLMs to replicate human reasoning directly, we can distill the inference process into explicit logical rules and apply them during editing. As illustrated in Figure 1, applying the rule "The first lady of the United States is the spouse of the president of the United States." naturally leads to the correct behavior in the example. Moreover, although it has been found that LLMs demonstrate a latent understanding of logical rules (Huang and Chang, 2023), they currently fail to leverage this capability for logical generalization during knowledge editing.

To address these challenges, in this work we propose ChainEdit, which synergizes the intrinsic logical reasoning capacity of LLMs with logical rule systems to empower knowledge editing methods to generalize edits via logics. Our key insight stems from the observation that both LLM knowledge editing and Knowledge Graph (KG) updating face similar challenges in chain updates. KG systems address this issue by applying logical rules to infer knowledge related to the update. Therefore, to enable LLMs to actively utilize their intrinsic logical reasoning capabilities, we integrate the rules mined from KGs with the logical reasoning of LLMs to construct a rule set that aligns with the internal logic of LLMs. This allows us to generate related knowledge influenced by a single updated piece of information, thus achieving the goal of chain updates.

Practically, ChainEdit operates through four phases: 1) extraction of candidate logical rules from KGs, 2) alignment of these rules with LLMs' internal logics, 3) dynamically generating affected knowledge queries based on editing rules, and 4) batch edits on both original and derived knowledge. This mechanism ensures deterministic updates for explicitly edited knowledge and its logical derivatives.

Our experiments adopt the RIPPLEEDITS benchmark, which is specifically designed to evaluate the ripple effects generated during model editing. When integrating ChainEdit with mainstream approaches like MEMIT, we observe substantial improvements in logical generalization metrics, for instance, achieving a 40.1% (from 18.6% to 58.7%) improvement over baseline MEMIT while maintaining comparable performance on specificity metric constraints.

Moreover, we identify a critical limitation in existing benchmarks for assessing editing methods' generalization capabilities: their reliance on external KGs for question formulation. As their questions were formed using KG-derived knowledge, they requires external knowledge beyond the edited content to correctly answer, thereby obscuring the assessment of logical generalization capabilities. To address this, we propose an adaptation to the existing dataset, which consists of three variants: 1) select questions answerable with only the model's internal knowledge, 2) reconstruct the question by substituting intermediate knowledge with model-internal knowledge, and 3) embed intermediate knowledge required for the answer directly into the question itself. Our refined benchmark enables more accurate measurement of editing methods' true generalization capacity.

2 Related Work

2.1 Knowledge Edit Methods

Mainstream knowledge editing approaches can be broadly categorized into parameter-preserving and parameter-modifying paradigms (Yao et al., 2023). Our work focuses primarily on parameter-modifying methods, which inherently face greater challenges in controlling ripple effects compared to their parameter-preserving counterparts. These

methods typically employ specialized loss functions and optimization constraints to control such ripple effects. For instance, MEND (Mitchell et al., 2022) trained a meta-network with carefully designed contrastive samples that include both irrelevant knowledge and paraphrased generalizations, aiming to enhance specificity while preserving unrelated knowledge. Locate-and-edit approaches such as ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) utilize causal mediation analysis to identify critical parameter layers, subsequently performing constrained optimization to update knowledge while maintaining original parameter space characteristics.

2.2 Ripple Effects in Knowledge Edit

Cohen et al. (2024) pioneered the formal characterization of ripple effects in model editing. Their benchmark framework introduced the first systematic evaluation protocol that simultaneously assesses both dimensions of ripple effects. Each knowledge edit undergoes rigorous evaluation through complementary metrics quantifying desired knowledge propagation and unintended side effects.

Recent advances address both aspects of ripple effects. Wang et al. (2024a) focused on mitigating the harmful ripple effect of irrelevant knowledge. They proposed that knowledge with similar embeddings is more susceptible to the ripple effect and used this knowledge to assess the ripple effect generated in the latent space, which is difficult to detect. Conversely, Qin et al. (2024) focused on analyzing beneficial ripple effects through GradSim, a gradient similarity measure between edited facts and associated knowledge. For in-context learning methods, Zhao et al. (2024) developed chain-of-thought strategies to amplify beneficial ripple effects.

Our work specifically targets "Logical Generalization", the most challenging category requiring models not only to memorize edited knowledge but also to logically integrate it with existing reasoning chains.

3 Methods

Our proposed logic-aware knowledge editing framework (ChainEdit), inspired by KG update mechanisms, establishes a logical rule system to guide LLMs in synchronizing associated knowledge updates during editing. As depicted in Figure 2, our methods contains three phases: rule mining, preprocessing, and application.

3.1 Rule Mining and Alignment

To explicitly leverage LLMs' logical capabilities, we construct structured rules for direct application during knowledge editing. Our rule acquisition process comprises two stages:

3.1.1 Rule Mining from KG

We mine candidate logical rules from KGs by identifying high-frequency alternative paths between relations.

For a target relation R, we identify multihop relational paths $\langle r_1, ..., r_n \rangle$ that connect subject-object pairs within existing R-relation triples (h, R, t). Formally, given a knowledge triple $(h, R, t) \in \mathcal{K}$, we discover alternative connection paths: $h \xrightarrow{r_1} e_1 \xrightarrow{r2} \cdots \xrightarrow{r_3} t$ where the path $r_1 \circ \cdots \circ r_n$ provides logical support for the target relation R. Paths demonstrating statistically significant frequency (determined by co-occurrence count with the target relation) are retained as candidate rules.

For instance, analyzing "Nationality"-relation triples reveals a high-frequency 2-hop path: $h \xrightarrow{\text{BornIn}} m \xrightarrow{\text{CityOf}} t \Rightarrow h \xrightarrow{\text{Nationality}} t$. This generates the rule `Nationality` $\leftarrow$ `(BornIn, CityOf)`, indicating nationality derivation through birthplace.

3.1.2 LLM-Rule Alignment

Fundamentally, we aim for LLMs to leverage their internal logical rules during the editing process to ensure logical consistency in the edited knowledge. Therefore, it is essential to align the logic recognized by LLMs with the derived set of rules. Specifically, we transform candidate rule entries $R \leftarrow (r_1, ..., r_n)$ into natural language statements and employ prompt design to enable LLMs to fully utilize their inherent logical reasoning capabilities to assess the universality of the rules (see Appendix B). Rules with high universality are then incorporated into the rule set.

This alignment bridges symbolic rules with LLMs' logical reasoning, creating a rule base that respects LLMs' intrinsic logical structures while maintaining KG-derived constraints.

3.2 Rule Preprocessing for Knowledge Editing

We convert each extracted rule into one that directly guides the generalization of editing, named "directive rule". It is formalized as $\mathcal{R} : \langle \phi, \psi \rangle$,

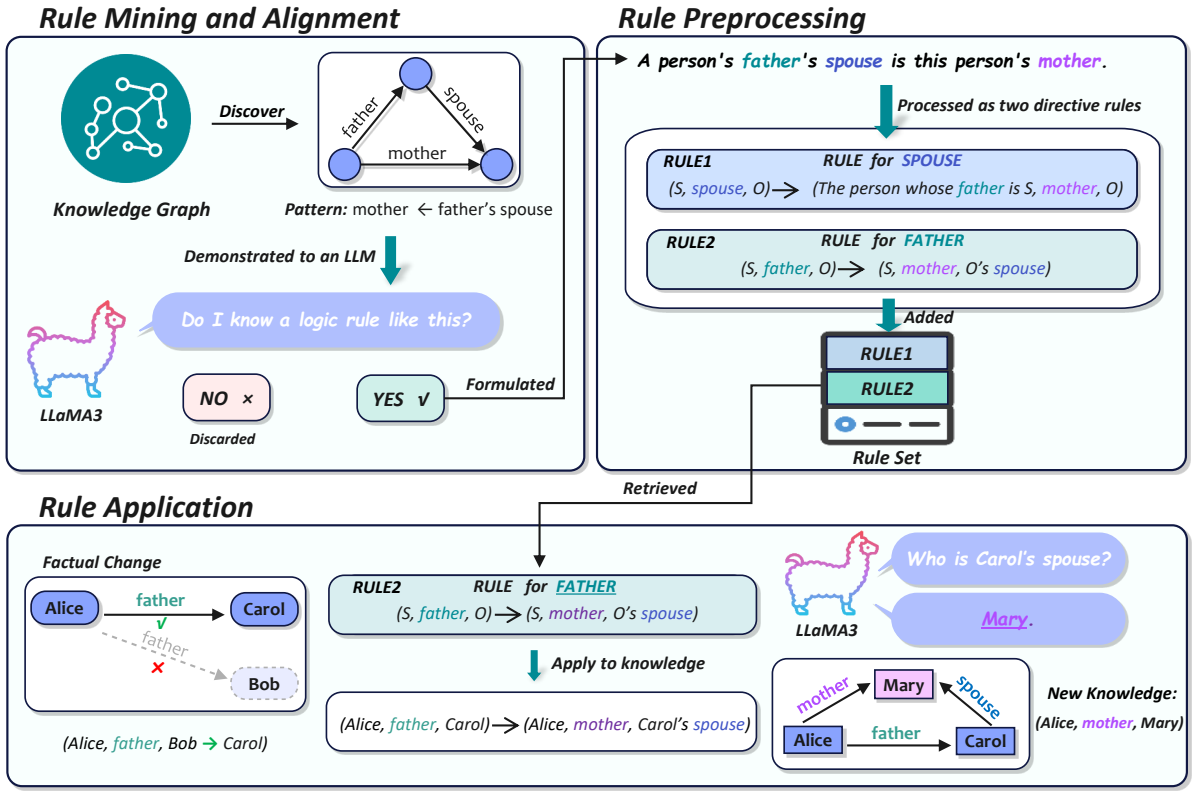

Figure 2: ChainEdit consists of three phases: (1) mine logical rule from KG and then align with LLM; (2) process the rule into directive rules (rule for edit) before adding into rule set; (3) retrieve and apply available rule to edit knowledge to generate relevant knowledge in the process of knowledge editing.

where ϕ denotes the trigger condition (logical predicate) and $\psi = (s_{new}, r_{new}, o_{new})$ specifies the knowledge generation template. Detailed syntax specifications appear in the Appendix C.

Our rule templates explicitly handle multiple legitimate chain update paths under the same logical rule when knowledge modifications occur. Consider the rule `father ← (sibling, father)`, which indicates that the father of the sibling of a person should be their own father. When editing A's sibling relationship to B, this rule permits two distinct but logically valid update paths: updating A's father to match B's current father and adjusting B's father to align with A's existing father. These two paths correspond to different real-world scenarios (*e.g.*, A adopted by B's father *vs.* B adopted by A's father) that cannot be simultaneously true. Conventional rule representations struggle to capture such contextual variations, while our template system can explicitly encode both paths

$$\begin{aligned} \psi_1 &: (\text{Alice}, \text{father}, \text{Carol.father}) \\ \psi_2 &: (\text{Carol}, \text{father}, \text{Alice.father}) \end{aligned} \tag{1}$$

.

Liu et al. (2024) have also discussed this logical path ambiguity issue and highlighted that, under conventional task setups, such ambiguity is unresolvable. Therefore, our rule template framework offers a more flexible and expressive mechanism that explicitly encodes both paths, enabling downstream systems or users to select the most contextually appropriate one.

3.3 Rule Application for Relevant Knowledge Generation

Given target edit (s, r, o), our method first retrieves all matching rules $\mathcal{R}_i$ whose $\phi_i = r$, then performs variable substitution for each: $\psi_i(s, o) = (s_{new}, r_{new}, o_{new})$ in rule $\mathcal{R}_i$. In other words, here we substitute the entity placeholder in s_{new} and o_{new} with real entities from the edit triple (s, r, o). For example, consider editing $(Alice, father, Carol)$ with triggered rule $\mathcal{R} : (S, father, O) \rightarrow (S, mother, O.spouse)$. Through variable substitution, we obtain the query item $q = (Alice, mother, Carol.spouse)$, then query LLM to get the entity representing $Carol.spouse$.

Structured prompting automates the knowledge derivation through the following operations: First, natural language queries are generated from formal query items $q_j = (s_j, r_j, o_j)$ - for instance, converting $Carol.spouse$ into the textual prompt "`The spouse of Carol is`". This prompt is then fed to the LLM to elicit a response v_j, and to finally formulate a new knowledge triple (s_j, r_j, v_j), as demonstrated by the example where the response "`Mary`" generates the associated triple $(Alice, mother, Mary)$.

Eventually, we perform batch editing on both original and derived knowledge using existing knowledge editing methods. This process ensures logical consistency in edits. For a complete example, see the Appendix A. Our framework demonstrates broad compatibility with existing editing methods, as evidenced by experiments across mainstream approaches in the next section.

4 Experiments

4.1 Datasets and Metrics

4.1.1 Existing Datasets

To systematically evaluate the logical generalization capability of knowledge editing, we extend the "POPULAR" dataset from the RIPPLEEDITS benchmark proposed by Cohen et al. (2024) with enhanced evaluation protocols. The original dataset measures ripple effects through six metrics: Logical Generalization (LG), Compositionality I (CI), Compositionality II (CII), Subject Aliasing (SA), Forgetfulness (FF)[1], and Relation Specificity (RS). Among these, LG, CI, CII, and SA assess the model's ability to propagate edits to logically related knowledge, while FF and RS assess unintended impacts on unrelated knowledge. Following Zhang et al. (2024)'s KnowEdit benchmark, we combine CI and CII into a Reasoning (RE) metric, establishing a five-dimensional evaluation framework (LG/RE/SA/FF/RS). Furthermore, we use the "reliability" (Wang et al., 2024b) metric to assess the success rate of knowledge edits.

It should be noted that, although datasets such as MQUAKE (Zhong et al., 2023) support the evaluation of multi-hop questions, they primarily focus on the concatenation reasoning of simple fact chains. In contrast, this study emphasizes the ability of LLMs to organically integrate edited knowledge with internal knowledge based on basic logic. Therefore, we did not conduct evaluations on such benchmarks.

[1]The "Preservation" metric in the original paper was renamed to Forgetfulness in their released dataset.

4.1.2 Our Dataset Variants

While applying the RIPPLEEDITS benchmark, we identify a flaw in conventional evaluation protocols: their reliance on intermediate knowledge from external knowledge graphs that may misalign with LLMs' internal knowledge states. For instance, when evaluating the inference chain "X graduated from Y University ⇒ Y is located in Z City ⇒ X's location is Z," discrepancies arise if the LLM internally believes "Y University is in City E." Such misalignments conflate knowledge consistency with true logical generalization, as successful edits may still be penalized due to mismatched intermediate facts. This mismatch of intermediate knowledge leads the original dataset to measure, to some extent, the consistency between LLMs and external knowledge bases, rather than the true logical generalization ability of the editing method.

To address these limitations, we develop three LLM-adaptive dataset variants through methodological innovations. The **Filtered** Dataset implements a prescreening mechanism that removes questions requiring intermediate knowledge unverifiable by the LLM itself, preserving only queries resolvable through the model's internal knowledge post-editing. The **Replaced** Dataset dynamically regenerates intermediate knowledge using the LLM's current state, ensuring evaluation alignment with the model's evolving knowledge system. The **In-Prompt** Dataset employs context augmentation to explicitly inject intermediate knowledge into prompts, isolating logical reasoning capability assessment from knowledge completeness requirements.

These variants serve distinct evaluation purposes: the first two diagnose the model's ability to generalize using its internal knowledge and edited knowledge, while the third evaluates enhanced reasoning when provided with auxiliary context. Through this taxonomy, our framework enables precise diagnosis of knowledge editing systems' strengths and limitations across different generalization scenarios.

4.2 Experimental Setup

We conduct experiments using Llama-3-8B-Instruct (AI@Meta, 2024) and Qwen2.5-1.5B-Instruct (Team, 2024; Yang et al., 2024) models, implementing three editing methods: MEMIT, LoRA

Model	Method	With Ours	Reliability	LG*	RE	SA	RS	FF
Llama-3-8B-Instruct	MEMIT	w/ ours	90.0	58.7	37.4	65.8	41.9	37.0
		w/o ours	99.8	18.6	34.3	75.7	38.0	31.2
	MEMIT-Merge	w/ ours	98.2	61.5	39.5	72.5	39.6	33.0
		w/o ours	98.8	19.3	33.2	73.2	40.1	35.3
	FT	w/ ours	100.0	65.5	47.2	97.4	60.2	36.5
		w/o ours	98.9	19.2	33.4	73.3	39.8	35.1
	LoRA	w/ ours	99.9	65.7	51.0	97.8	48.2	33.0
		w/o ours	100.0	23.7	41.7	99.0	45.6	28.1
Qwen2.5-1.5B-Instruct	MEMIT	w/ ours	87.4	55.8	26.2	53.3	41.1	35.1
		w/o ours	99.0	22.1	24.0	59.7	40.6	30.0
	MEMIT-Merge	w/ ours	96.6	61.5	29.5	59.0	35.8	29.4
		w/o ours	99.0	22.3	24.1	60.6	38.6	29.9
	FT	w/ ours	100.0	55.6	43.9	97.9	36.3	27.1
		w/o ours	100.0	16.5	43.5	98.6	42.6	31.7
	LoRA	w/ ours	100.0	59.6	49.0	99.1	33.6	24.1
		w/o ours	100.0	17.6	39.5	99.1	29.1	19.4

Table 1: The results (%) obtained using the Qwen and Llama models on RIPPLEEDITS, comparing performance across different editing methods with and without our method. The metrics marked with * are the primary targets for improvement in this paper.

(Hu et al., 2022), MEMIT-Merge (Dong et al., 2025), and FT-M (Zhang et al., 2024). All these evaluations adopt a single-instance editing protocol: after each knowledge edit, we immediately assess performance metrics defined in Section 4.1 before reverting the edit to ensure isolation between test cases.

4.2.1 Rule Mining Implementation

Our systematic rule mining framework operates on Wikidata with three classes of rule targeting relations from the RIPPLEEDITS (*e.g.*, nationality, kinship):

- **Inverse Relation**: For each target relation r, we sample 10,000 instances $(a \xrightarrow{r} b)$ and identify high-frequency inverse relations r' from b to a. Rules $r \leftrightarrow r'$ are established when the occurrence frequency exceeds threshold γ.

- **Multi-Hop Path Mining**: For each $(a \xrightarrow{r} b)$ instance, we explore:

  - 2-hop paths: $a \xrightarrow{r_1} m \xrightarrow{r_2} b$
  - 3-hop paths: $a \xrightarrow{r_1} m_1 \xrightarrow{r_2} m_2 \xrightarrow{r_3} b$

  Paths with support count $\geq \gamma$ are retained as candidate rules $r \leftarrow (r_1, ..., r_k)$.

After deduplication, this process yields 3,120 candidate rules.

4.3 Main Results

As demonstrated in Table 1, the ChainEdit framework substantially enhances the logical generalization capabilities of mainstream batch editing methods. In Llama-3-8B, MEMIT achieves a dramatic improvement in logical generalization from 18.6% to 58.7% , confirming that our chained rule trigger mechanism effectively establishes logical associations between knowledge items. Moreover, Qwen2.5-1.5B also exhibits significant LG improvements (22.1% to 55.8%), demonstrating ChainEdit's strong cross-architectural generalization. Table 1 presents the editing results for single-instance scenarios. Additionally, we've included experiments on batch-instance editing in Appendix D, which further confirms our method's broader applicability.

Although the edits of related knowledge increase editing complexity, the original knowledge editing success rate (reliability) of most methods remains stable. It should be noted that the original MEMIT method experienced a reliability drop of 9.8% on the Llama model, which stems from the inherent flaw in its parameter update mechanism for editing multiple knowledge items with the same subject (Dong et al., 2025). By adopting the improved MEMIT-Merge strategy, the reliability was restored to 98.2%, confirming that the performance fluctuation originated from the limitations of the method rather than the ChainEdit framework itself.

Model	Method	Rule Set	Reliability	LG*	RE	SA	RS	FF
Qwen2.5-1.5B-Instruct	MEMIT	pure mining	88.7	54.7	25.9	54.1	39.5	33.7
		with LLM	88.5	56.1	25.1	53.7	41.4	33.3
		with LLM and human	87.4	55.8	26.2	53.3	41.1	35.1
	MEMIT-Merge	pure mining	97.1	53.0	29.8	63.5	36.3	30.5
		with LLM	97.3	60.0	26.0	58.8	36.0	29.3
		with LLM and human	96.6	61.5	29.5	59.0	35.8	29.4
	FT	pure mining	100.0	50.5	42.3	98.1	37.0	29.0
		with LLM	100.0	52.1	41.2	97.8	37.6	29.3
		with LLM and human	100.0	55.6	43.9	97.9	36.3	27.1
	LoRA	pure mining	100.0	63.0	45.8	98.4	33.9	23.9
		with LLM	100.0	63.3	45.2	99.2	34.3	24.0
		with LLM and human	100.0	59.6	49.0	99.1	33.6	24.1

Table 2: The result of applying ChainEdit using different rule sets: a) purely based on rule mining; b) method (a) plus LLM alignment; c) method (b) plus human selection.

Specificity metrics show method-dependent fluctuations bounds within narrow bounds: MEMIT improves RS by 2.9% while MEMIT-Merge decreases by 0.7% with Llama. The minimal extent in decrease indicates that ChainEdit's precise boundary control prevents large-scale knowledge interference. Specifically, the RS metric indicates that, compared to Qwen2.5-1.5B, Llama-3-8B and other large models demonstrate a stronger ability to retain irrelevant knowledge, highlighting the positive impact of model capacity on stabilizing irrelevant knowledge.

Furthermore, contrary to initial expectations, the model scale shows limited correlation with the logical integration capability: Both large and small models exhibit similarly poor baseline LG performance (often <20%) and comparable relative improvements after ChainEdit. Furthermore, even smaller models maintain stable RS metrics under our framework, proving ChainEdit's robustness across model sizes.

4.3.1 Ablation Study on Logical Rule Mining

To validate the critical role of LLMs' internal logic, we compare two rule acquisition strategies: (1) pure KG-based mining *vs.* (2) KG mining with LLM's logical alignment. For fair comparison, we calibrate both rule sets to comparable sizes (around 180 rules). The LLM-filtered set retains 180 high-confidence rules through logical validation, while the pure KG set is subsampled via frequency thresholding to match this size.

As shown in the first two parts of Table 2, LLM-rule alignment outperforms pure KG mining, with logical generalization improving by 7% (53% $\rightarrow$ 60%) on Llama-3-8B using MEMIT-Merge. This confirms that LLMs' internal reasoning patterns provide essential guidance for identifying practically applicable rules beyond statistical correlations.

Furthermore, we evaluate the impact of manual curation introduced in our full pipeline (primarily for computational efficiency). The last two sections of the table compare manually curated rules against fully automated mining, revealing negligible performance differences (LG: 60.0% *vs.* 61.5% with Llama-3-8B, MEMIT-Merge). This demonstrates that our core advantages stem from automated rule discovery rather than human intervention, establishing ChainEdit's scalability to large-scale applications.

4.4 Datasets Integrating Edited and Internal Knowledge

As detailed in Section 4.1, we construct three knowledge-adaptive datasets to eliminate evaluation biases caused by external knowledge dependencies. Table 3 demonstrates MEMIT-Merge's performance across these datasets, with other edit methods' results provided in Appendix E.

As shown in Table 3, after intermediate knowledge filtering, MEMIT-Merge method exhibits a more significant performance gap on the LG metric: without using ChainEdit, the LG value was only 19.5%, while it surged to 71.0% after applying ChainEdit in the Llama model. This phenomenon reveals that even if LLMs have already acquired intermediate knowledge, their knowledge system still cannot spontaneously establish logical associations with newly edited knowledge. As shown in Figure 3, after using ChainEdit, the model's LG performance on the filtered dataset is generally bet-

Model	Dataset	With Ours	Reliability	LG*	RE	SA	RS	FF
Llama-3-8B-Instruct	original	w/ ours	98.2	61.5	39.5	72.5	39.6	33.0
		w/o ours	98.8	19.3	33.2	73.2	40.1	35.3
	filtered	w/ ours	98.4	71.0	42.8	77.5	31.5	33.1
		w/o ours	99.0	19.5	34.6	73.0	36.9	35.8
	replaced	w/ ours	96.8	65.8	38.5	70.4	37.8	32.2
		w/o ours	99.1	19.6	30.7	70.3	39.6	34.5
	in-prompt	w/ ours	97.7	67.6	69.9	81.8	39.4	34.1
		w/o ours	99.0	33.1	68.4	83.8	40.9	35.7
Qwen2.5-1.5B-Instruct	original	w/ ours	96.6	61.5	29.5	59.0	35.8	29.4
		w/o ours	99.0	22.3	24.1	60.6	38.6	29.9
	filtered	w/ ours	97.3	72.5	30.8	59.3	27.8	29.5
		w/o ours	99.0	17.5	21.7	60.5	29.0	30.9
	replaced	w/ ours	98.1	68.1	31.4	59.6	37.8	30.7
		w/o ours	99.0	22.5	24.4	60.0	39.1	31.5
	in-prompt	w/ ours	96.1	71.1	66.1	96.4	37.4	29.3
		w/o ours	99.1	33.6	64.4	97.9	38.8	29.9

Table 3: The results of applying ChainEdit in the MEMIT-Merge method on the original dataset and the variant datasets.

ter than that on the original dataset, indirectly confirming that in some cases, even if the model has successfully generalized the results, it cannot be accurately evaluated in the original data set derived from KG.

The replaced dataset dynamically substitutes KG-sourced intermediate knowledge with LLM-generated content. While this approach maintains question quantity (*vs.* reduction in filtered data), it introduces controlled noise through potential answer inaccuracies. As evidenced in Table 3, ChainEdit maintains robust LG performance (65.8%) under these conditions, proving its capacity to leverage rule systems for reliable knowledge integration regardless of external answer correctness. This dual validation across filtered and replaced datasets establishes ChainEdit's unique ability to bridge fragmented internal knowledge through structured logical frameworks, outperforming conventional editing methods that rely solely on parametric memorization.

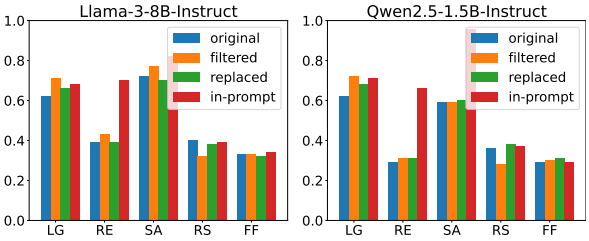

Figure 3: Comparison of MEMIT-Merge method results on the original dataset and our variant dataset after using ChainEdit.

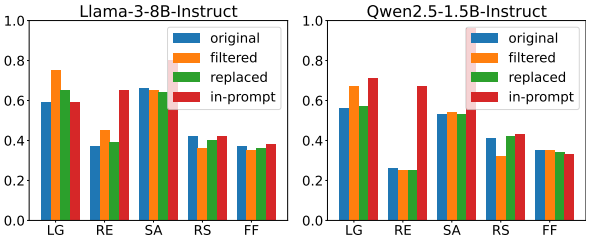

Figure 4: Comparison of MEMIT method results on the original dataset and our variant dataset after using ChainEdit.

4.5 Integrating Edited Knowledge with Prompt-Provided Context

This experiment explicitly injects intermediate knowledge into question contexts using the format: "Given the following information: intermediate knowledge; Complete the following sentence: original question". Unlike previous evaluations relying on internal knowledge integration, this setup tests LLMs' capacity to combine edited knowledge with external contextual information.

As shown in Table 3, ChainEdit enhances MEMIT-Merge's LG to 65.8%, demonstrating effective multisource knowledge integration. Notably, SA scores exhibit anomalously high values due to simplified reasoning patterns: the injected context directly contains alias relationships, making correct answers trivially extractable.

Figure 3 and Figure 4 reveals two critical in-

sights: 1) LG and RE metrics significantly outperform original benchmark results when providing intermediate knowledge (LG: 33.1% *vs.* 19.3%, RE: 68.4% *vs.* 33.2% as shown in Table 3) without applying our method, confirming LLMs' stronger performance in combining edited knowledge with explicit context versus internal knowledge; 2) Despite these improvements, baseline LG scores remain suboptimal (<35%), highlighting fundamental limitations in spontaneous logical rule application in LLM knowledge editing.

5 Conclusion

This paper addresses the ripple effect that occurs among interconnected facts during knowledge editing. Starting from chain updates of KG, it expands the editing of a single piece of knowledge to the entire group of related knowledge using logical rules, thereby performing ChainEdit. On the RIPPLEEDITS benchmark, various editing methods integrating ChainEdit have shown a significant improvement in their ability in "logical generalization". Meanwhile, to address evaluation biases in existing benchmarks, we develop three LLM-adaptive dataset variants that decouple logical reasoning from external knowledge dependencies, where ChainEdit consistently shows superior generalization capability.

Limitations

Our framework demonstrates promising results in improving the logical generalization ability of existing batch editing methods, yet several limitations warrant discussion to guide future research.

In the rule mining phase, reliance on knowledge graphs introduces limitations. The quality and comprehensiveness of rules may vary due to KGs' limited coverage and inability to capture all logical relationships.

Secondly, in the process of aligning the internal logic of large language models, the model's reasoning capability becomes a potential limiting factor. Particularly for models with smaller parameter scales and relatively weaker reasoning abilities, they may struggle to accurately understand or effectively apply certain complex rules, thereby impacting overall performance.

Furthermore, in our rule mining strategy, we currently only retain rules that are applicable in most cases, which may overlook certain rules that are only valid under specific conditions. To address this challenge, conditional rule and selective application could be helpful.

Lastly, although our method demonstrates stability in editing specificity, as the rule set continues to expand, the number of edits may grow exponentially, which could adversely affect system performance. To address this challenge, future research could consider introducing optimization strategies such as rule priority mechanisms or rule application conditions.

Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 62476134).

References

AI@Meta. 2024. Llama 3 model card.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Trans. Assoc. Comput. Linguistics*, 12:283–298.

Zilu Dong, Xiangqing Shen, and Rui Xia. 2025. MEMIT-Merge: Addressing MEMIT's Key-Value Conflicts in Same-Subject Batch Editing for LLMs. *arXiv preprint*. ArXiv:2502.07322 [cs].

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1049–1065. Association for Computational Linguistics.

Jiateng Liu, Pengfei Yu, Yuji Zhang, Sha Li, Zixuan Zhang, Ruhi Sarikaya, Kevin Small, and Heng Ji. 2024. EVEDIT: event-based knowledge editing for deterministic knowledge propagation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 4907–4926. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jiaxin Qin, Zixuan Zhang, Chi Han, Pengfei Yu, Manling Li, and Heng Ji. 2024. Why does new knowledge create messy ripple effects in llms? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 12602–12609. Association for Computational Linguistics.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Jianchen Wang, Zhouhong Gu, Zhuozhi Xiong, Hongwei Feng, and Yanghua Xiao. 2024a. The missing piece in model editing: A deep dive into the hidden damage brought by model editing. *CoRR*, abs/2403.07825.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. 2024b. 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, Bangkok, Thailand. Association for Computational Linguistics.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv 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, EMNLP 2023, Singapore, December 6-10, 2023*, pages 10222–10240. Association for Computational Linguistics.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. 2024. A comprehensive study of knowledge editing for large language models. *CoRR*, abs/2401.01286.

Zihao Zhao, Yuchen Yang, Yijiang Li, and Yinzhi Cao. 2024. Ripplecot: Amplifying ripple effect of knowledge editing in language models via chain-of-thought in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 6337–6347. Association for Computational Linguistics.

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 15686–15702. Association for Computational Linguistics.

A Case study

Consider the initial knowledge state $\mathcal{K}_0$ containing the following triples:

```
(Alice, father, Bob)
(Alice, mother, Rose)          (2)
(Bob, spouse, Rose)
```

When performing editing operation $\mathcal{E}$: `(Alice, father, Carol)`. Traditional editing methods only update the target triple, leading to logical inconsistencies. Specifically, after the edit, the model still treats `(Alice, mother, Rose)` as truth, whereas, according to the family relationship logic, Alice's mother should be updated to Carol's spouse.

By leveraging knowledge graph mining and LLM's knowledge, we obtain the following logical rules:

$$\mathcal{R}_1 : \texttt{mother} \leftarrow (\texttt{father}, \texttt{spouse}) \quad (3)$$

, which represents that one's mother is the spouse of the father of this person. This rule can be formulated into two directive rules:

- $\langle \phi : \texttt{father}, \psi : (S, mother, O.\texttt{spouse}) \rangle$
- $\langle \phi : \texttt{spouse}, \psi : (X, mother, O) \rangle$, where X is the entity whose father is S

Edit operation $\mathcal{E}$ triggers following process:

1. **rule match**: detecting ϕ : `father` matching the directive rules of $\mathcal{R}_1$
2. **intermediate knowledge query**: perform query $q_1 = \texttt{spouse}(Carol, ?)$ to LLM and thereby get response $v_1 = Mary$
3. **relevant knowledge generation**: generate new knowledge triple $(Alice, mother, Mary)$
4. **batch edit**: form a edit batch $\{\mathcal{E}, \texttt{(Alice, mother, Mary)}\}$ and then perform batch editing.

B Details of Aligning Logical Rules with the Internal Knowledge System of LLMs

To achieve alignment between logical rules and the internal knowledge system of LLMs, we designed a multi-stage semantic transformation process. First, we transform the formal rule $(A, r_1, B) \leftarrow (A, r_2, C)$ into a natural language proposition. The basic template is "If the relation of subject is object, then...", where nominal relations (such as nationality, occupation) adopt the structure "the relation of subject is object", while verbal relations (such as mentorship, employment) are adapted into the dynamic template "subject relation object". Taking the family relationship rule (A,father,B)←(A,mother,B.spouse) as an example, its natural language expression is "If the father of A is B, then the mother of A is the spouse of B".

The transformed natural language rules are input into the LLM for intermediate entity querying via a structured prompt. The model is required to provide its own logical rule to support or oppose the given rule and make a qualitative judgment on a five - level confidence scale (True/Usually True/Sometimes True/False/Uncertain). (Example is given in "Alignment Chating Example" box) In the experiment, the first two levels are considered valid rules.

Alignment Chating Example

[User] When the father of X is Y, then the sibling of X is the child of Y.
[Assistant] *The sibling of an person is his father's child.* Answer: True

[User] When the country of X is Y, then the continent of X is the continent of Y.
[Assistant] *The continent of a location is usually the same as the continent of the country location belongs to.* Answer: Usually True

C Structural Design of Knowledge Generation Templates in Rules

This appendix provides a detailed explanation of the formal definition and execution mechanism of knowledge generation templates in logical rules. Given a rule $\mathcal{R} : \langle \phi, \psi \rangle$,where $\psi = (s_{new}, r_{new}, o_{new})$, the elements within the template are as follows:

ϕ (**Condition**): This represents the condition or premise of the rule. It specifies the circumstances under which the rule is applicable. For example, it could be a set of existing knowledge triples that must hold true for the rule to be triggered.

ψ (**New knowledge**): This represents the new

knowledge that is generated when the condition ϕ is satisfied.

Triple entity placeholders, *e.g.*:

- `S`: bind to current edit's subject entity
- `O`: bind to current edit's object entity

Use the dot operator to represent multihop relation query, *e.g.*:

- `O.father`: parsed as object entity's father
- `S.birthplace.country`: nested operations is parsed from left to right, producing subject entity's place of birth's country of origin
- `father.S`: parsed as the entity whose father is S

D The results of ChainEdit in the batch edit scenario

In addition to single-instance editing scenarios, we conducted further experiments in batch edit scenarios. We tested our method's performance with both MEMIT-Merge and FT editing techniques across a batch size range from 2 to 100. The results are presented in 4.

When the batch size is increased, our method shows no significant change in reliability but a slight drop in LG. However, compared to the original method without our framework, it still performs at a very high level.

The decline tendency of the LG metric with increasing batch size is understandable. As the batch size increases, the number of edits grows, which inherently dilutes the effect of each individual edit to some extent. Since our method relies on the effectiveness of the editing process itself, the LG metric consequently decreases as the editing ability diminishes.

E Detailed Results in our datasets

Tables 5, 6, 7 present detailed results (%) in the "filtered", "replaced", and "in-prompt" datasets, respectively.

Method	Batchsize	With Ours	Reliability	LG*	RE	SA	RS	FF
MEMIT-Merge	2	w/ ours	97.6	61.0	29.3	60.0	37.1	30.6
		w/o ours	98.9	22.4	24.0	59.2	40.3	30.5
	10	w/ ours	96.8	58.1	28.2	58.2	38.4	30.5
		w/o ours	98.8	20.6	24.5	60.4	39.9	31.2
	50	w/ ours	94.8	56.1	29.2	58.0	35.3	30.5
		w/o ours	98.3	22.1	24.6	60.3	36.3	29.3
	100	w/ ours	91.5	52.1	28.6	54.6	35.3	29.6
		w/o ours	97.8	21.9	24.9	60.0	36.1	29.9
FT	2	w/ ours	99.8	58.3	41.9	95.4	26.7	20.7
		w/o ours	99.9	15.0	40.2	97.1	33.7	26.2
	10	w/ ours	99.9	53.2	38.8	92.8	26.4	23.1
		w/o ours	100.0	14.7	35.3	92.6	26.6	21.2
	50	w/ ours	99.3	52.4	35.3	86.6	18.6	17.8
		w/o ours	100.0	16.7	33.9	87.7	28.3	26.3
	100	w/ ours	97.6	50.7	31.0	82.5	11.3	11.6
		w/o ours	99.9	13.3	34.3	85.4	23.8	24.4

Table 4: The results of ChainEdit in the batch edit scenario using Qwen2.5-1.5B-Instruct model on original dataset.

Model	Method	With Ours	Reliability	LG*	RE	SA	RS	FF
Llama-3-8B-Instruct	MEMIT	w/ ours	88.0	74.8	45.3	65.3	35.9	34.5
		w/o ours	99.1	19.3	33.0	72.4	35.0	35.2
	MEMIT-Merge	w/ ours	98.4	71.0	42.8	77.5	31.5	33.1
		w/o ours	99.0	19.5	34.6	73.0	36.9	35.8
	FT	w/ ours	100.0	78.4	44.2	98.9	47.5	34.8
		w/o ours	100.0	13.9	40.6	97.5	69.3	41.0
	LoRA	w/ ours	99.9	75.7	51.3	98.5	45.4	32.1
		w/o ours	100.0	17.2	37.1	98.8	45.2	28.6
Qwen2.5-1.5B-Instruct	MEMIT	w/ ours	88.0	67.2	25.2	54.4	32.1	34.6
		w/o ours	98.7	17.5	19.9	59.7	26.3	30.3
	MEMIT-Merge	w/ ours	97.3	72.5	30.8	59.3	27.8	29.5
		w/o ours	99.0	17.5	21.7	60.5	29.0	30.9
	FT	w/ ours	100.0	67.5	30.6	98.7	13.7	27.1
		w/o ours	100.0	10.7	23.0	99.3	22.6	31.8
	LoRA	w/ ours	100.0	78.4	45.4	99.2	19.0	23.9
		w/o ours	100.0	11.4	23.1	98.9	14.8	20.0

Table 5: Detailed results on "filtered" dataset.

Model	Method	With Ours	Reliability	LG*	RE	SA	RS	FF
Llama-3-8B-Instruct	MEMIT	w/ ours	87.8	64.8	39.0	63.7	40.3	35.9
		w/o ours	99.1	19.8	31.8	70.6	40.1	35.2
	MEMIT-Merge	w/ ours	96.8	65.8	38.5	70.4	37.8	32.2
		w/o ours	99.1	19.6	30.7	70.3	39.6	34.5
	FT	w/ ours	100.0	64.6	45.2	95.1	53.9	32.5
		w/o ours	100.0	13.9	40.6	97.5	69.3	41.0
	LoRA	w/ ours	100.0	66.3	49.3	95.4	45.7	32.0
		w/o ours	100.0	17.2	37.1	98.8	45.2	28.6
Qwen2.5-1.5B-Instruct	MEMIT	w/ ours	87.8	56.8	25.3	53.5	41.5	33.8
		w/o ours	99.1	24.0	24.2	58.5	38.2	30.7
	MEMIT-Merge	w/ ours	98.1	68.1	31.4	59.6	37.8	30.7
		w/o ours	99.0	22.5	24.4	60.0	39.1	31.5
	FT	w/ ours	100.0	56.4	42.8	95.6	35.4	27.7
		w/o ours	100.0	19.4	40.7	96.5	42.3	31.8
	LoRA	w/ ours	100.0	59.8	49.0	95.2	34.2	24.1
		w/o ours	100.0	19.8	35.9	94.1	31.5	17.1

Table 6: Detailed results on "replaced" dataset.

Model	Method	With Ours	Reliability	LG*	RE	SA	RS	FF
Llama-3-8B-Instruct	MEMIT	w/ ours	86.3	59.5	64.6	80.2	42.2	38.1
		w/o ours	99.0	32.1	68.1	84.9	39.7	35.2
	MEMIT-Merge	w/ ours	97.7	67.6	69.9	81.8	39.4	34.1
		w/o ours	99.0	33.1	68.4	83.8	40.9	35.7
	FT	w/ ours	100.0	71.7	65.4	89.7	58.9	36.1
		w/o ours	100.0	27.0	66.8	88.6	70.3	40.4
	LoRA	w/ ours	99.7	69.3	68.4	94.5	49.1	34.0
		w/o ours	100.0	26.8	62.6	95.9	46.2	29.2
Qwen2.5-1.5B-Instruct	MEMIT	w/ ours	85.5	71.3	66.8	96.0	42.8	33.3
		w/o ours	98.9	35.6	62.4	97.8	38.9	29.2
	MEMIT-Merge	w/ ours	96.1	71.1	66.1	96.4	37.4	29.3
		w/o ours	99.1	33.6	64.4	97.9	38.8	29.9
	FT	w/ ours	100.0	59.5	57.9	94.7	37.1	27.4
		w/o ours	100.0	22.5	57.6	96.2	41.9	30.7
	LoRA	w/ ours	100.0	57.5	50.2	99.3	34.6	26.4
		w/o ours	100.0	17.6	39.5	99.1	29.1	19.4

Table 7: Detailed results on "in-prompt" dataset.