# EchoEdit: Consistent Multi-Hop Question Answering via Ripple Control in Knowledge Editing

**Jinwei Shi**[1], **Wenxuan Huang**[2], **Yu Xing**[1], **Yunhui Liu**[1], **Tao Zheng**[1], **Bin Chong**[3*], **Tieke He**[1*]

[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]Hangzhou Dianzi University, Hangzhou 310005, China
[3]National Engineering Laboratory for Big Data Analysis and Applications, Peking University, Beijing 100871, China
chongbin@pku.edu.cn, hetieke@gmail.com

## Abstract

Knowledge editing aims to update specific knowledge in Large Language Models (LLMs) without retraining the entire model. However, existing methods generally struggle to manage the ripple effects of knowledge updates, particularly in multi-hop reasoning tasks, where conflicts between old and new information often lead to shifts in reasoning chains and degraded consistency. To address this issue, a ripple-aware knowledge editing framework, namely *EchoEdit*, is proposed. *EchoEdit* introduces the RippleGraph to explicitly model potentially affected knowledge regions and employs a RippleRule generator to dynamically produce diffusion rules, precisely constraining knowledge propagation. Furthermore, we distill a Chain-of-Thought (CoT) planner from an external teacher model, which decouples complex reasoning chain planning into RippleGraph-guided reasoning, thereby alleviating the reasoning burden on low-resource LLMs in multi-hop tasks. Experimental results on the MQuAKE and RIPPLEEDITS multi-hop reasoning benchmarks demonstrate that *EchoEdit* significantly outperforms existing mainstream methods, effectively enhancing post-edit reasoning consistency and generalization capabilities.

**Code** — https://github.com/jinweiAi/EchoEdit.git

## Introduction

Large language models (LLMs) have demonstrated impressive performance across a broad spectrum of knowledge-intensive tasks (AlKhamissi et al. 2022). However, their dependence on static, parametric knowledge acquired during pre-training presents a significant challenge in maintaining alignment with continuously evolving real-world facts (Dhingra et al. 2022). To address this limitation, knowledge editing techniques have been proposed to efficiently modify specific factual associations within an LLM (Wang et al. 2024d), enabling low-cost, targeted updates while preserving the model's overall performance.

Modifying one fact in an LLM often has unintended side effects on related knowledge—a phenomenon known as the *ripple effect* (Cohen et al. 2024). These effects occur because knowledge representations in LLMs are highly interconnected: a single factual change may unintentionally affect semantically or structurally associated information (Li et al. 2025a,b). This problem becomes especially challenging in multi-hop reasoning tasks, where even minor inconsistencies can lead to a chain of errors and cause the model to generate incorrect or self-contradictory answers. For example, updating the fact "the president of the USA is Donald Trump" requires consistent updates to a range of related facts, such as "Trump's spouse" and "Trump's sibling," to maintain internal coherence. Additionally, entity aliases (e.g., "USA" and "America") and relation variants (e.g., "president," "leader," "head of state") must be correctly aligned to reflect the new knowledge. Without proper constraints on knowledge propagation, such changes can result in overgeneralization. For instance, if "UK" is mistakenly aliased to "USA," the model may incorrectly respond that "Trump" is the UK's prime minister. These subtle but impactful reasoning chain shifts, illustrated in Figure 1, highlight the need for precise and localized control over how edits propagate. Balancing the propagation of intended updates with the preservation of unrelated knowledge is essential to ensuring that the model retains both factual accuracy and reasoning consistency after an edit (Zhong et al. 2023).

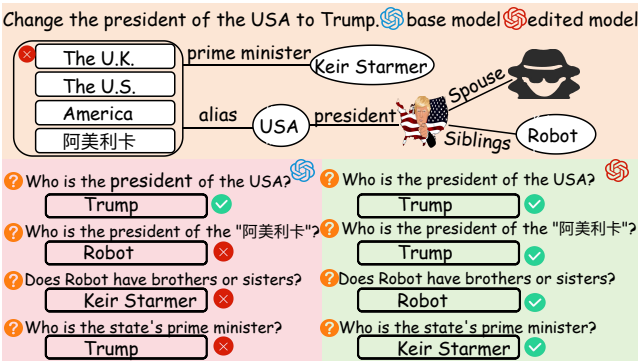


Figure 1: Ripple effects from factual edits cause reasoning drift and inconsistency. EchoEdit constrains propagation via RippleGraph and RippleRule to preserve relevant updates and unaffected knowledge.

However, current knowledge editing methods often fail to address this aspect. Parameterized editing methods (Meng

*Bin Chong and Tieke He are the corresponding authors.

et al. 2022; Fang et al. 2025) directly update model weights by modifying neurons associated with target knowledge. Parameterized editing approaches (Meng et al. 2022; Fang et al. 2025) directly modify model weights by altering neurons associated with target knowledge. While effective for local edits, these methods lack control over ripple effects, leading to internal knowledge perturbations, and diminished capabilities in complex reasoning (Chen et al. 2025). On the contrary, parameter-preserving methods (Qi et al. 2024; Zhao et al. 2024) avoid altering model weights but still struggle to fully mitigate ripple effects: when reasoning with newly injected facts, they may conflict with internal knowledge, leading to reasoning chain shifts or even a "forgetting" state (Wang et al. 2025a). To control ripple effects, DecKER (Wang et al. 2025a) reduces reasoning chain disruptions by extracting masked reasoning paths and injecting new knowledge in a hybrid manner. However, it relies on the edited model itself to generate reasoning chains, and low-resource LLMs (e.g., those with fewer than 7B parameters) often exhibit limited planning capabilities, particularly in complex multi-hop reasoning tasks. KEDKG (Lu et al. 2025) addresses knowledge conflicts caused by ripple effects by constructing dynamic knowledge graphs for edited facts and introducing a secondary conflict detection mechanism during editing. However, it constructs graphs only for directly edited facts and fails to explicitly model other knowledge potentially affected by ripple effects.

In response to these limitations, *EchoEdit* is proposed to better manage ripple effects. The framework integrates a RippleGraph module to explicitly capture knowledge regions potentially affected by edits and employs a RippleRule generator to dynamically derive diffusion rules for precise knowledge propagation. Additionally, *EchoEdit* distills a lightweight Chain-of-Thought (CoT) planner (Yu et al. 2024) from an external large-scale teacher model (e.g., GPT-4o), which decouples complex reasoning chain planning into RippleGraph-guided reasoning, reducing the reasoning burden on lightweight LLMs in multi-hop settings. We follow (Zhao et al. 2024; Qi et al. 2024; Wang et al. 2025a; Dong et al. 2025) to evaluate the effectiveness of *EchoEdit* using multi-hop question answering. Specifically, we use Neighborhood Multi-hop Chain Sampling (NMCS), combined with Directional Distance Encoding (DDE) and parallel triple scoring, to dynamically extract a localized subgraph $G_{init}$ from the external knowledge graph. This subgraph centers on the edited fact $f_{edit} = (h, r, t)$ within radius $k$. Next, the LLMs perform *Inner Knowledge Explicitation* on their memory to extract a set of relevant entities $E_{mem}$ in natural language (Huang et al. 2025). These entities and their associated relations are then integrated into $G_{init}$, resulting in a semantically enriched subgraph $G^{'}_{init}$. Finally, in RippleRule Generator, inspired by "rule mining and alignment"(Dong et al. 2025), the multi-hop path patterns within $G^{'}_{init}$ are extracted and "aligned" to extract logical rules $R_{opt}$. We then apply $R_{opt}$ to refine $G^{'}_{init}$, obtaining a streamlined and task-specific RippleGraph $G_{ripple}$. When performing MHQA, *EchoEdit* employs the distilled CoT planner (Yu et al. 2024) to decompose complex multi-hop questions into a sequence of simpler sub-questions. Each sub-question is then processed through guided reasoning over the structured RippleGraph $G_{ripple}$. This design separates the planning of reasoning chains from factual inference, enabling more reliable and efficient multi-hop reasoning in edited knowledge contexts.

In summary, the key contributions of this work are as follows:

- *EchoEdit* is proposed to alleviate the conflicts between old and new information that lead to shifts in reasoning chains and degraded consistency in knowledge editing.
- *EchoEdit* introduces RippleGraph to model affected knowledge regions and uses a RippleRule generator to dynamically create diffusion rules that constrain knowledge propagation. It also distills a Chain-of-Thought (CoT) planner from an external teacher model to decouple reasoning chain planning from factual inference, thereby reducing the reasoning burden on low-resource LLMs in multi-hop tasks.
- We conduct extensive experiments on various LLMs and datasets to validate the effectiveness and usability of *EchoEdit*, demonstrating its superior ability to mitigate ripple effects and maintain reasoning consistency in complex tasks compared to existing methods.

## Related Work

### Knowledge Editing

In this subsection, we provide an introduction to recent knowledge editing methods, which can be categorized into parameter-modifying and parameter-preserving approaches.

**Parameter-modifying methods** directly modifying internal model weights is a common strategy for factual knowledge editing. Full-parameter fine-tuning (Zhu et al. 2020) updates all weights via gradient descent, but is computationally expensive and prone to catastrophic forgetting. To improve efficiency and locality, meta-learning approaches such as MEND (Mitchell et al. 2021) and KGEditor (Cheng et al. 2024) employ hypernetworks to produce targeted, low-rank updates. De Cao et al. (De Cao, Aziz, and Titov 2021) further emphasize preserving prior knowledge to ensure fact-level precision. Locate-then-edit methods improve edit specificity by first identifying where a fact is stored: Dai et al. (Dai et al. 2021) identify factual neurons in MLP layers; ROME (Meng et al. 2022) and MEMIT (Meng et al. 2023) perform causal-tracing-guided edits on FFN weights; and PMET (Li et al. 2024) refines this idea by decoupling updates across MHSA and FFN components.

**Parameter-preserving methods** modify knowledge without changing existing weights, typically through auxiliary modules. CaliNet (Dong et al. 2022) and T-Patcher (Huang et al. 2023) inject trainable components into FFN layers, while GRACE (Hartvigsen et al. 2023) maintains a dynamic memory for continual fact correction. Beyond memory, context-aware strategies such as IKE (Zheng et al. 2023) guide the model to edit factual knowledge via constructed demonstration contexts. PokeMQA (Gu et al. 2023) employs a scope detector that dynamically modulates

the behavior of the LLM based on conflicts between new and existing knowledge. WISE (Wang et al. 2024c) adopts a dual-memory architecture to separate pretrained knowledge and edits, enabling reliable and conflict-free updates. Retrieval-based methods such as ReMaKE (Wang, Haddow, and Birch 2023), RAE (Shi et al. 2024), and KEDKG (Lu et al. 2025) enable dynamic, cross-lingual, and fine-grained editing through knowledge graph integration and prompt learning.

### Ripple Effects

Knowledge editing often triggers ripple effects, altering both explicit and implicit related knowledge, making global logical consistency difficult to maintain. Zhong et al. (2023) show that performance deteriorates with increasing reasoning hops, indicating weak propagation to indirect knowledge, while Cohen et al. (2024) highlight that existing evaluations focus on local correctness and overlook broader consistency, offering the first systematic taxonomy of ripple effects. Recent methods attempt to improve propagation control: RIPPLECOT (Zhao et al. 2024) leverages chain-of-thought prompting to reinforce logical links; Self-Edit (Liu et al. 2024) constrains edits using implicit reasoning anchors; AlphaEdit (Fang et al. 2025) stabilizes sequential edits by projecting updates into the null space of preserved knowledge; and GIE with SIR (Wang et al. 2024b) uses knowledge graphs and selective refinement to adjust the most affected triplets. Nonetheless, most approaches still treat edits as predominantly local and lack mechanisms for coherent, controllable diffusion. In Section 3, we introduce a ripple-aware framework to address these limitations.

## Preliminaries

This work focuses on knowledge editing in LLMs, aiming to update specific factual knowledge without retraining while preserving logical consistency and reasoning capabilities. Let $f_\theta$ denote the original language model, where knowledge is represented as triples $(h, r, t)$ with head $h$, relation $r$, and tail $t$. A knowledge edit is defined as $e = (h, r, t \rightarrow t^*)$, where $t$ is replaced with $t^*$. Given an edit set $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$, the goal is to produce an updated model $f_\theta^*$ that meets two essential requirements:

**(1) Multi-hop QA Accuracy:** For each edited triple, the updated model should correctly answer multi-hop questions that rely on the updated fact. We adopt the standard metrics M-Acc (Modified Accuracy) and H-Acc (Hop-level Accuracy) to evaluate the correctness and completeness of reasoning over updated knowledge.

**(2) Ripple Consistency and Stability:** To evaluate the integrity of post-edit reasoning, we adopt the fine-grained criteria defined in RIPPLEEDITS (Cohen et al. 2024). We define a ripple consistency score $\mathcal{R}(f_\theta^*, \delta)$, computed as a weighted average across multiple evaluation dimensions, to quantify how well the edited model preserves logical structure and compositionality under factual updates. This includes:

- logical closure over symmetric and transitive relations,
- consistent composition of multi-hop paths involving the edited fact (both forward and backward),
- propagation of edits to aliased subjects,
- and preservation of unaffected facts under one-to-many relations.

To meet these requirements, we cast ripple-aware knowledge editing as a structured reasoning problem under local constraints. Given an atomic edit $f_{edit} = (h, r, t \rightarrow t^*)$, the goal is to update the target fact while ensuring consistent propagation along relevant reasoning paths, and avoiding changes to unrelated knowledge.We introduce three key components: a local RippleGraph $G_{\text{ripple}}$ to capture knowledge regions influenced by the edit, a RippleRule Generator that mines and applies diffusion rules to optimize $G_{\text{ripple}}$, and a Chain-of-Thought (CoT) Planner distilled from an external teacher model to help lightweight LLMs decouple reasoning from factual retrieval. These components will be detailed in the next section.

## Method

Figure 2 presents the framework of *EchoEdit*, which consists of three core components: RippleGraph construction, a dynamic RippleRule Generator, and a CoT planner that guides lightweight LLMs through structured multi-hop reasoning over the RippleGraph.

### RippleGraph Construction

To model the ripple effects triggered by editing a factual triple $f_{edit} = (h, r, t)$, we construct RippleGraph, a structured representation that captures the propagation of factual changes. Generally, *EchoEdit* constructs RippleGraph by first extracting a semantically grounded local subgraph centered around $f_{edit}$, and then enriching it with additional relational knowledge drawn from the model's internal memory. The detailed process is as follows:

**Local Subgraph Extraction.** We initiate the construction of RippleGraph by identifying an initial subgraph $G_{init}$ from an external knowledge graph using the NMCS strategy (Chen et al. 2025). Formally, we define:

$$G_{init} = \{(h_i, r_i, t_i) \mid d(h_i, h) \leq k \vee d(t_i, t) \leq k\}, \quad (1)$$

where $d(x, y)$ denotes the shortest path distance between entities $x$ and $y$ in the knowledge graph, and $k$ is a hyperparameter controlling the neighborhood radius. This step ensures the subgraph captures relevant entities and relations. It provides a semantically coherent local context and forms the basis for modeling ripple effects.

**Inner Knowledge Explicitation.** While external graphs offer structured context, LLMs also retain pertinent factual knowledge within their parametric memory $\mathcal{M}$. To exploit this internal resource, we adopt Inner Knowledge Explicitation (Huang et al. 2025), which extracts textual knowledge relevant to $f_{edit}$ directly from the model. We define the relevant entity set extracted from $\mathcal{M}$ as

$$E_{mem} = e \mid P(e \mid f_{edit}, \mathcal{M}) \geq \tau, \quad (2)$$

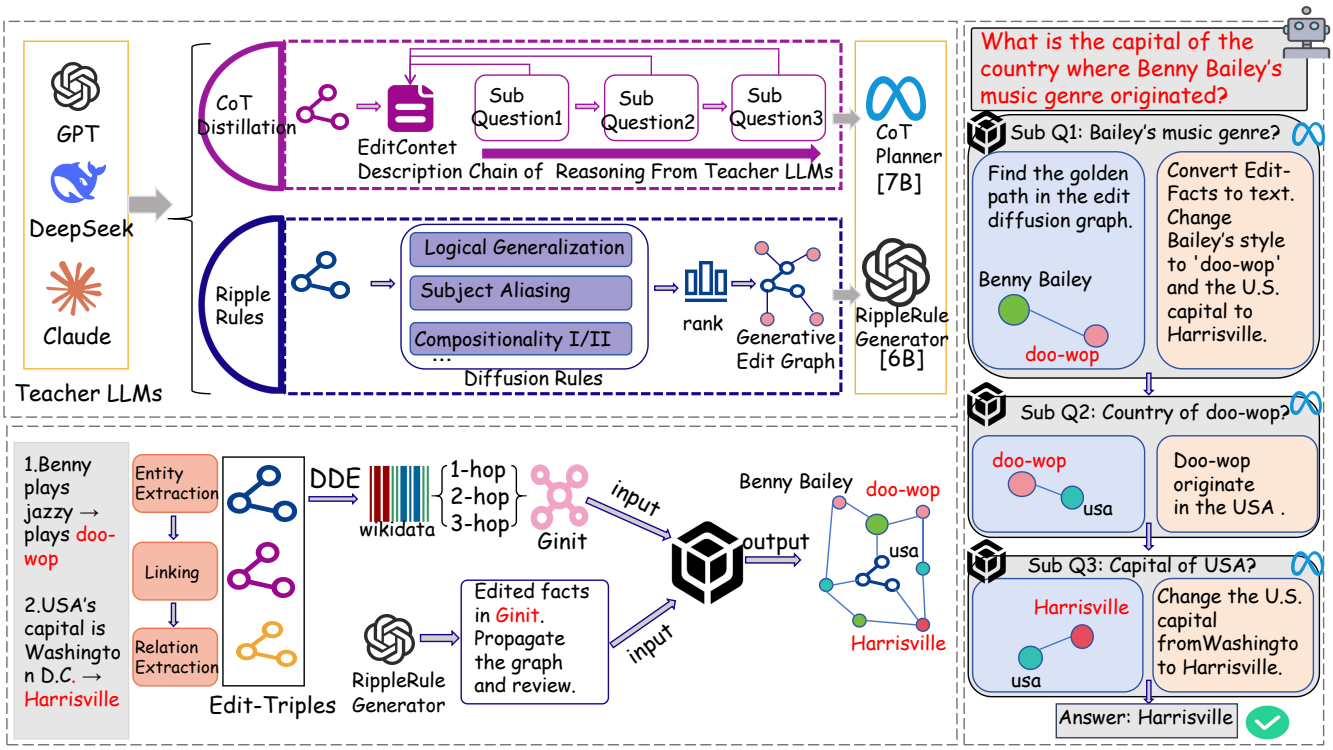


Figure 2: The framework of *EchoEdit*, which integrates a RippleGraph module to explicitly capture knowledge regions potentially affected by edits and employs a RippleRule generator to dynamically derive diffusion rules for precise knowledge propagation, and leverages a Chain-of-Thought (CoT) planner to transform complex multi-hop reasoning into structured queries executed on the RippleGraph.

where $P(e \mid f_{edit}, \mathcal{M})$ quantifies the relevance strength between entity $e$ and the edited triple, and $\tau$ is a threshold. These entities are then used to augment the initial subgraph, incorporating additional model-internal relational knowledge:

$$G'_{init} = G_{init} \cup \{(e, r', e') \mid e \in E_{mem}\}. \quad (3)$$

Step 1

**RippleGraph Construction Example:**
Given the edit *"Barack Obama was born in Hawaii"*, the Neighborhood Multi-hop Chain Sampling (NMCS) extracts an initial subgraph $G_{init}$ including:

- Barack Obama → PlaceOfBirth → Kenya
- Barack Obama → Nationality → Kenyan
- Barack Obama → Citizenship → Kenyan
- Barack Obama → Father → Barack Obama Sr.
- Barack Obama → Spouse → Michelle Obama
- Kenya → LocatedIn → Africa
- Hawaii → LocatedIn → USA

This $G_{init}$ provides a semantically rich and compact local view for further refinement.

The resulting augmented graph integrates both external and internal knowledge sources, yielding a more comprehensive and contextually enriched RippleGraph for modeling the broader implications of factual edits.

## Dynamic RippleRule Generator

The augmented subgraph $G'_{init}$ identifies a candidate region of knowledge affected by an edit. However, it may still include outdated entities, conflicting relations, or irrelevant noise. Using $G'_{init}$ directly for reasoning risks propagating these inconsistencies and degrading performance. In response, we propose *RippleRule Generator*, a lightweight module that optimizes and expands $G'_{init}$. It produces a refined RippleGraph $G_{ripple}$ for downstream reasoning. RippleRule generator first analyzes multi-hop relational paths in $G'_{init}$, guided by the edited fact $f_{edit}$. It extracts highly relevant path patterns as candidate rules. These rules are converted into natural language prompts and fed into the LLM. The LLM then evaluates which entities and relations should be retained, updated, removed, or augmented. This process removes conflicts and filters noise. It also expands the ripple effect by adding logical triples related to $f_{edit}$. The final RippleGraph $G_{ripple}$ provides a semantically coherent and structurally compact context to support reasoning. The rip-

ple rule is defined as

$$R_{opt} = f_{rule}(f_{edit}, q, hops(q), T), \tag{4}$$

where $f_{rule}$ represents the LLM-driven reasoning process. It analyzes multi-hop relational paths in $G^{'}_{init}$, guided by the edited fact $f_{edit}$. Based on structural proximity, semantic relevance to $q$, and type compatibility with $T$, it generates a natural language optimization rule. Applying $R_{opt}$ refines $G^{'}_{init}$ into the final RippleGraph $G_{ripple}$:

$$G_{ripple} = \mathcal{A}(G^{'}_{init}, R_{opt}). \tag{5}$$

Here, $\mathcal{A}$ removes conflicting and noisy entities, substitutes outdated ones, and supplements logical triples related to $f_{edit}$. This produces a semantically coherent and structurally compact graph for downstream reasoning.

Step 2

**Dynamic RippleRule Example:**
RippleRule generator prompts the LLM to analyze the editing context and $G^{'}_{init}$. The LLM produces the following ripple rule:

> *"Update all facts dependent on Barack Obama's place of birth. Replace (Barack Obama, PlaceOfBirth, Kenya) with (Barack Obama, PlaceOfBirth, Hawaii). Update location hierarchy accordingly: add (Hawaii, LocatedIn, USA) and (USA, LocatedIn, North America). Modify nationality and citizenship facts to reflect the change. Prune all unrelated entities and relations."*

**Updated $G_{ripple}$ after applying this rule:**

- Barack Obama → PlaceOfBirth → Hawaii
- Hawaii → LocatedIn → USA
- USA → LocatedIn → North America
- Barack Obama → Nationality → American
- Barack Obama → Citizenship → American

Unrelated facts such as Kenya's location and Barack Obama's family relations are pruned automatically, ensuring the resulting RippleGraph focuses only on knowledge essential for answering the current multi-hop question and mitigating ripple-induced noise.

The ripple rules generated by RippleRule generator ensure that RippleGraph maintains a compact and semantically relevant subgraph. This optimization reduces irrelevant entities and relations while preserving triples logically associated with $f_{edit}$. As a result, even resource-constrained LLMs perform reliably in complex multi-hop reasoning tasks, benefiting from a reduced search space and improved semantic coherence.

## CoT Planner

We introduce an iterative CoT planner to improve reasoning consistency after knowledge edits. Unlike static decomposition methods, our planner dynamically generates and verifies each sub-question step-by-step, adapting to edited knowledge and mitigating reasoning drift. Given a multi-hop question $Q$, the planner first instantiates a decomposition template $P_{\text{decomp}}$ to produce the initial sub-question $q_1$. At each reasoning step $i$, it conditions on the prior context $[q_1, a_1, ..., q_{i-1}, a_{i-1}]$ to generate the next sub-question $q_i$, ensuring coherence across the chain. For each $q_i$, *EchoEdit* employs a dual-path execution strategy. A structured path queries the RippleGraph $G_{\text{ripple}}$ to retrieve $a_i^{graph}$, while a semantic path uses the edited fact as context to generate $a_i^{edit}$ via a language model. If the two answers agree, the result is accepted. Otherwise, a conflict detector—distilled from a stronger verifier—selects the more faithful answer based on consistency with the updated knowledge(Wang et al. 2025b). If no valid answer is found, the planner triggers rollback. It discards $q_i$ and re-generates it under the same context, allowing correction of faulty decomposition. Verified answers are inserted as placeholders into the next query template, guiding subsequent steps. The process repeats until all $H$ hops are completed. The final answer $a_H$ is then returned.

We train the planner using a conditional language modeling objective:

$$\mathcal{L}_{\text{plan}}(\theta) = -\sum_{i=1}^{H} \log p_\theta(q_i \mid q_{<i}, a_{<i}, Q, P_{\text{decomp}}). \tag{6}$$

Here, $\theta$ are model parameters, and $q_i$ is predicted given prior sub-questions, answers, the original query, and the decomposition prompt. The planner is distilled from a teacher model (e.g., GPT-4o), transferring structured reasoning behaviors to a lightweight student. This design ensures edit-aware reasoning, enabling recovery from intermediate inconsistencies. The combined use of dual-path answering and verifier-guided validation supports more robust, controllable multi-hop reasoning under knowledge updates.

# Experiments

We evaluate *EchoEdit* on two standard benchmarks for knowledge editing: RIPPLEEDITS and MQuAKE. These datasets test various update scenarios, including common facts, random edits, and time-sensitive knowledge. All experiments are performed on Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct.

## Experimental Settings

**RIPPLEEDITS Dataset.** RIPPLEEDITS evaluates ripple effects in large language models. We focus on two subsets: POPULAR, which contains widely known, high-impact facts, and RANDOM, which samples triples uniformly to test generalization. Each edit is a (`subject, relation, object`) triple with paired pre- and post-edit QA tasks and reasoning paths. This design enables detailed analysis of how updates propagate through reasoning chains and affect related knowledge.

**MQuAKE Dataset.** MQuAKE focuses on multi-hop reasoning following knowledge edits. It includes two subsets:

| Models | Methods | MQuAKE-CF-3k-v2 | MQuAKE-T | Ripple-Pop | Ripple-Rand |
|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct | MEMIT | 7.1 | 45.3 | 8.0 | 9.4 |
| | AlphaEdit | 7.4 | 43.6 | 8.2 | 9.1 |
| | Mello | 11.6 | 58.9 | 26.2 | 33.5 |
| | PokeMQA | 17.1 | 76.2 | 31.0 | 32.5 |
| | DeepEdit | 9.4 | 56.7 | 3.8 | 5.2 |
| | RAE | 53.5 | 52.2 | 21.1 | 17.5 |
| | EditCoT | 35.2 | 75.1 | 32.0 | 21.9 |
| | KEDKG | 55.7 | 77.4 | 44.0 | 43.0 |
| | DecKER | 58.2 | 81.7 | 46.8 | 45.9 |
| | *EchoEdit* | **68.6** | **89.4** | **53.1** | **53.7** |
| Qwen2.5-7B-Instruct | MEMIT | 6.8 | 22.0 | 10.1 | 16.9 |
| | AlphaEdit | 7.3 | 22.7 | 11.2 | 16.6 |
| | Mello | 1.9 | 49.5 | 3.8 | 1.5 |
| | PokeMQA | 7.4 | 44.1 | 7.3 | 7.1 |
| | DeepEdit | 12.8 | 26.1 | 6.3 | 12.3 |
| | RAE | 37.9 | 29.4 | 23.5 | 19.0 |
| | EditCoT | 27.1 | 75.3 | 29.8 | 31.0 |
| | KEDKG | 41.0 | 64.2 | 34.0 | 30.1 |
| | DecKER | 43.0 | 68.3 | 36.5 | 31.5 |
| | *EchoEdit* | **52.1** | 74.8 | **44.2** | **38.9** |

Table 1: Performance of editing methods on MQuAKE and RIPPLEEDITS. MQuAKE-CF-3k-v2 and MQuAKE-T evaluate counterfactual and temporal multi-hop edits, while RIPPLEEDITS-Popular and -Random assess ripple effects on high-impact and diverse factual updates.

MQuAKE-CF, which targets counterfactual edits and requires the model to adjust reasoning based on altered facts; and MQuAKE-T, which addresses temporal updates to test consistency in time-sensitive contexts. Each sample features a $k$-hop question ($k \in \{2, 3, 4\}$), challenging the model to propagate updates across multiple related triples. This makes MQuAKE a strong benchmark for evaluating ripple-aware multi-hop reasoning.

**Baselines.** We compare *EchoEdit* with both parameter-editing and parameter-preserving methods. Parameter-editing approaches include MEMIT (Meng et al. 2023) and AlphaEdit (Fang et al. 2025). These methods directly modify model weights to inject target knowledge. AlphaEdit further uses matrix projection to limit disruptions. Parameter-preserving methods do not alter model parameters. Instead, they rely on prompts or external memory. This group includes Mello (Zhong et al. 2023) and PokeMQA (Gu et al. 2024), which decompose multi-hop questions into subproblems. DeepEdit (Wang et al. 2024e) applies depth-first search for editing. EditCoT (Wang et al. 2024a) integrates Chain-of-Thought reasoning. KEDKG (Lu et al. 2025) and RAE (Shi et al. 2024) use dynamic knowledge graphs to support updates without changing internal weights. We also compare with DecKER (Wang et al. 2025a), which mitigates conflicts between new and existing knowledge during reasoning.

## Main Results

*EchoEdit* shows consistent performance advantages across all evaluated tasks. Table 1 reports the main results. On

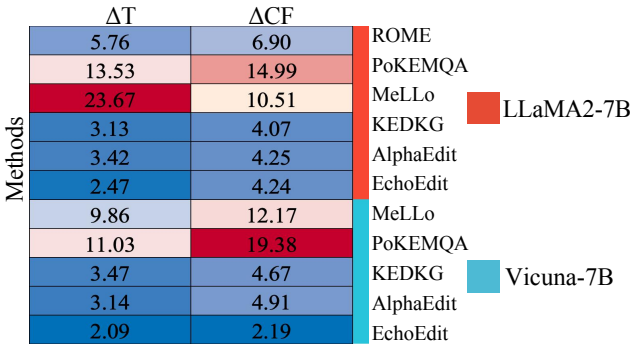


Figure 3: Comparison of M-Acc and H-Acc differences on MQuAKE-CF and MQuAKE-T. Smaller gaps reflect more consistent reasoning over edited knowledge.

MQuAKE-CF-3k-v2 and MQuAKE-T, *EchoEdit* achieves accuracies of 68.6 and 89.4, respectively. This outperforms DecKER (58.2 and 81.7) and KEDKG (55.7 and 77.4) by a large margin. These results confirm that *EchoEdit* handles knowledge editing and ripple effects more effectively in multi-hop reasoning. It is particularly strong on temporal reasoning tasks, where maintaining consistency is challenging. The RippleRule Generator contributes to this performance by dynamically controlling reasoning paths. In contrast, methods like PokeMQA and EditCoT achieve similar performance to DecKER in some cases but show larger fluctuations on complex tasks, especially in the RANDOM subset. In the RIPPLEEDITS POPULAR and RANDOM sub-

sets, *EchoEdit* also demonstrates strong generalization. On LLaMA-3.1-8B, it achieves 53.1 and 53.7 accuracy, surpassing DecKER's 46.8 and 45.9. On Qwen2.5-7B, *EchoEdit* achieves 44.2 and 38.9, while DecKER scores 36.5 and 31.5. These gains suggest that RippleGraph improves reasoning stability by filtering unrelated information and focusing on knowledge relevant to the edits.

Moreover, *EchoEdit* shows strong adaptability across different model scales. On Qwen2.5-7B, a smaller model, it uses the external CoT planner to overcome the limitations of implicit memory reasoning. This design improves the stability of multi-hop reasoning chains. In comparison, graph retrieval-based methods like RAE and KEDKG maintain local consistency using fixed graph structures. However, they lack the dynamic regulation of RippleRule. As a result, they perform worse when reasoning over low-frequency or long-tail facts. Parameter-editing methods such as MEMIT and AlphaEdit perform even worse. Their direct weight modifications introduce instability, which affects reasoning across all subsets. These results highlight the advantage of *EchoEdit*'s design. By combining structured query execution with reasoning decoupling, it preserves consistency and boosts multi-hop accuracy for lightweight LLMs. Overall, *EchoEdit* achieves stable performance across diverse models, tasks, and editing scenarios, demonstrating its universality and practical value.

Figure 3 presents the reasoning stability analysis on the MQuAKE dataset. We compare the differences between M-Acc and H-Acc, denoted as $\Delta_{CF}$ and $\Delta_T$. These metrics measure how closely intermediate reasoning steps align with the final answer. Smaller $\Delta_{CF}$ and $\Delta_T$ indicate more coherent multi-hop reasoning, where intermediate steps reliably support the final prediction. Larger values suggest deviations in reasoning paths, even when the final response is correct. As shown in Figure 3, *EchoEdit* consistently achieves lower $\Delta_{CF}$ and $\Delta_T$ across both model groups. This highlights its ability to maintain reasoning coherence and reduce disruptions after knowledge editing.

| Ablation Variant | MQuAKE-CF | MQuAKE-T |
|---|---|---|
| Full Model | **68.6** | **89.4** |
| w/o RippleGraph | 32.8 | 41.9 |
| w/o RippleRule Generation | 51.3 | 74.1 |
| w/o CoT Planner | 56.5 | 78.9 |

Table 2: Ablation study on *EchoEdit*'s components on MQuAKE.

## Ablation Study

We perform ablation experiments on Llama3.1-8B-Instruct using MQuAKE-CF-3k and MQuAKE-T. The goal is to evaluate the individual contributions of RippleGraph, RippleRule Generator, and CoT Planner in *EchoEdit*. Table 2 shows the results. Removing RippleGraph causes the largest performance drop. Accuracy drops from 68.6 to 32.8 on MQuAKE-CF and from 89.4 to 41.9 on MQuAKE-T. This result highlights the critical role of explicitly modeling affected knowledge regions. RippleGraph effectively limits unintended ripple effects by defining a localized reasoning context. Excluding RippleRule Generator results in moderate degradation. Accuracy drops to 51.3 on MQuAKE-CF and 74.1 on MQuAKE-T. This component filters irrelevant entities and relations, preventing noise from propagating in multi-hop reasoning. Removing the CoT Planner causes a smaller drop. Accuracy decreases to 56.5 and 78.9 on the two datasets. Although RippleGraph and RippleRule provide strong structural control, the CoT Planner further stabilizes reasoning chains. It is especially beneficial for complex multi-hop tasks where query decomposition is critical. These results confirm the complementary roles of all three components. RippleGraph defines the scope of edits. RippleRule Generator constrains knowledge propagation. CoT Planner enables reliable reasoning. Together, they ensure EchoEdit's effectiveness in complex knowledge editing scenarios.

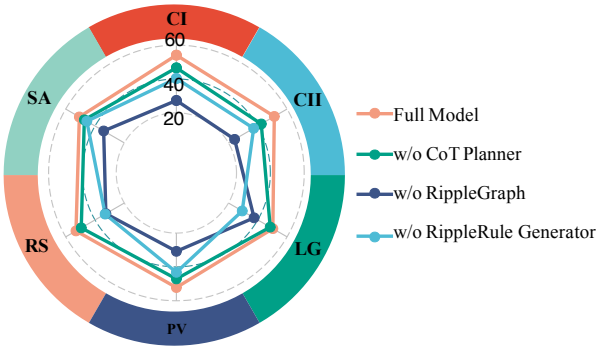


Figure 4: Ablation results across six RIPPLEEDITS dimensions, assessing the contribution of each component.

Figure 4 shows that removing RippleGraph leads to the largest drops in Logical Generalization (LG) and Compositionality II (CII), underscoring its role in multi-hop structural reasoning. Disabling the RippleRule Generator markedly reduces Relation Specificity (RS) and Preservation (PV), highlighting the need for dynamic diffusion control. Excluding the CoT Planner yields smaller but consistent declines, indicating its importance for reasoning coherence. Overall, the three modules address complementary facets of ripple effects and jointly support stable post-edit reasoning.

## Conclusion

We propose *EchoEdit*, a ripple-aware knowledge editing framework that reformulates multi-hop reasoning as structured traversal over an optimized *RippleGraph*. By modeling semantic scopes and constraining propagation via dynamic *RippleRules*, *EchoEdit* ensures local updates with global consistency. A lightweight CoT Planner decomposes complex reasoning into verifiable steps. Experimental results confirm improved reasoning stability and factual consistency across diverse editing scenarios.

## Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (7240041225, 62306137).

## References

AlKhamissi, B.; Li, M.; Celikyilmaz, A.; Diab, M.; and Ghazvininejad, M. 2022. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*.

Chen, Q.; Wang, D.; Zhang, T.; Yan, Z.; You, C.; Wang, C.; and He, X. 2025. UniEdit: A Unified Knowledge Editing Benchmark for Large Language Models. *arXiv preprint arXiv:2505.12345*.

Cheng, S.; Zhang, N.; Tian, B.; Chen, X.; Liu, Q.; and Chen, H. 2024. Editing language model-based knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 17835–17843.

Cohen, R.; Biran, E.; Yoran, O.; Globerson, A.; and Geva, M. 2024. Evaluating the Ripple Effects of Knowledge Editing in Language Models. *Transactions of the Association for Computational Linguistics*, 12: 283–298.

Dai, D.; Dong, L.; Hao, Y.; Sui, Z.; Chang, B.; and Wei, F. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

De Cao, N.; Aziz, W.; and Titov, I. 2021. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*.

Dhingra, B.; Cole, J. R.; Eisenschlos, J. M.; Gillick, D.; Eisenstein, J.; and Cohen, W. W. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10: 257–273.

Dong, Q.; Dai, D.; Song, Y.; Xu, J.; Sui, Z.; and Li, L. 2022. Calibrating factual knowledge in pretrained language models. *arXiv preprint arXiv:2210.03329*.

Dong, Z.; Shen, X.; Yang, Z.; and Xia, R. 2025. ChainEdit: Propagating Ripple Effects in LLM Knowledge Editing through Logical Rule-Guided Chains. *arXiv preprint arXiv:2507.08427*.

Fang, J.; Jiang, H.; Wang, K.; Ma, Y.; Shi, J.; Wang, X.; He, X.; and Chua, T.-S. 2025. AlphaEdit: Null-Space Constrained Model Editing for Language Models. In *The Thirteenth International Conference on Learning Representations*.

Gu, H.; Zhou, K.; Han, X.; Liu, N.; Wang, R.; and Wang, X. 2023. Pokemqa: Programmable knowledge editing for multi-hop question answering. *arXiv preprint arXiv:2312.15194*.

Gu, H.; Zhou, K.; Han, X.; Liu, N.; Wang, R.; and Wang, X. 2024. PokeMQA: Programmable knowledge editing for Multi-hop Question Answering. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 8069–8083. Bangkok, Thailand: Association for Computational Linguistics.

Hartvigsen, T.; Sankaranarayanan, S.; Palangi, H.; Kim, Y.; and Ghassemi, M. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36: 47934–47959.

Huang, S.; Zhong, W.; Cai, D.; Wan, F.; Wang, C.; Wang, M.; Qiao, M.; and Xu, R. 2025. Empowering Self-Learning of LLMs: Inner Knowledge Explicitation as a Catalyst. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 24150–24158.

Huang, Z.; Shen, Y.; Zhang, X.; Zhou, J.; Rong, W.; and Xiong, Z. 2023. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*.

Li, H.; Zheng, W.; Hu, J.; Wang, Q.; Zhang, H.; Wang, Z.; Xuyang, S.; Fan, Y.; Zhou, S.; Zhang, X.; et al. 2025a. Predictable Scale: Part I–Optimal Hyperparameter Scaling Law in Large Language Model Pretraining. *arXiv e-prints*, arXiv–2503.

Li, H.; Zheng, W.; Wang, Q.; Ding, Z.; Wang, H.; Wang, Z.; Xuyang, S.; Ding, N.; Zhou, S.; Zhang, X.; et al. 2025b. Farseer: A Refined Scaling Law in Large Language Models. *arXiv preprint arXiv:2506.10972*.

Li, X.; Li, S.; Song, S.; Yang, J.; Ma, J.; and Yu, J. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 18564–18572.

Liu, J.; Yu, P.; Zhang, Y.; Li, S.; Zhang, Z.; and Ji, H. 2024. Evedit: Event-based knowledge editing with deductive editing boundaries. *arXiv preprint arXiv:2402.11324*.

Lu, Y.; Zhou, Y.; Li, J.; Wang, Y.; Liu, X.; He, D.; Liu, F.; and Zhang, M. 2025. Knowledge editing with dynamic knowledge graphs for multi-hop question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 24741–24749.

Meng, K.; Bau, D.; Andonian, A.; and Belinkov, Y. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35: 17359–17372.

Meng, K.; Sharma, A. S.; Andonian, A. J.; Belinkov, Y.; and Bau, D. 2023. Mass-Editing Memory in a Transformer. In *The Eleventh International Conference on Learning Representations*.

Mitchell, E.; Lin, C.; Bosselut, A.; Finn, C.; and Manning, C. D. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Qi, S.; Yang, B.; Jiang, K.; Wang, X.; Li, J.; Zhong, Y.; Yang, Y.; and Zheng, Z. 2024. In-context editing: Learning knowledge from self-induced distributions. *arXiv preprint arXiv:2406.11194*.

Shi, Y.; Tan, Q.; Wu, X.; Zhong, S.; Zhou, K.; and Liu, N. 2024. Retrieval-enhanced knowledge editing in language models for multi-hop question answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2056–2066.

Wang, C.; Su, W.; Ai, Q.; and Liu, Y. 2024a. Knowledge Editing through Chain-of-Thought. *arXiv preprint arXiv:2412.17727*.

Wang, C.; Su, W.; Ai, Q.; Zhou, Y.; and Liu, Y. 2025a. Decoupling Reasoning and Knowledge Injection for In-Context Knowledge Editing. *arXiv preprint arXiv:2506.00536*.

Wang, J.; Gu, Z.; Zhu, X.; Zhang, L.; Ye, H.; Xiong, Z.; Feng, H.; and Xiao, Y. 2024b. Efficiently Quantifying and Mitigating Ripple Effects in Model Editing. *arXiv preprint arXiv:2403.07825*.

Wang, P.; Li, Z.; Zhang, N.; Xu, Z.; Yao, Y.; Jiang, Y.; Xie, P.; Huang, F.; and Chen, H. 2024c. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37: 53764–53797.

Wang, S.; Zhu, Y.; Liu, H.; Zheng, Z.; Chen, C.; and Li, J. 2024d. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 1–37.

Wang, W.; Haddow, B.; and Birch, A. 2023. Retrieval-augmented multilingual knowledge editing. *arXiv preprint arXiv:2312.13040*.

Wang, Y.; Chen, M.; Peng, N.; and Chang, K.-W. 2024e. Deepedit: Knowledge editing as decoding with constraints. *arXiv preprint arXiv:2401.10471*.

Wang, Y.; Song, Y.; Zhu, T.; Zhang, X.; Yu, Z.; Chen, H.; Song, C.; Wang, Q.; Wang, C.; Wu, Z.; et al. 2025b. TrustJudge: Inconsistencies of LLM-as-a-Judge and How to Alleviate Them. *arXiv preprint arXiv:2509.21117*.

Yu, P.; Xu, J.; Weston, J.; and Kulikov, I. 2024. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*.

Zhao, Z.; Yang, Y.; Li, Y.; and Cao, Y. 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*, 6337–6347.

Zheng, C.; Li, L.; Dong, Q.; Fan, Y.; Wu, Z.; Xu, J.; and Chang, B. 2023. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*.

Zhong, Z.; Wu, Z.; Manning, C.; Potts, C.; and Chen, D. 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*, 15686–15702.

Zhu, C.; Rawat, A. S.; Zaheer, M.; Bhojanapalli, S.; Li, D.; Yu, F.; and Kumar, S. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.