STRUCTKE: MITIGATING STRUCTURAL DEGRADATION IN MULTI-HOP KNOWLEDGE EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge editing aims to update specific facts in large language models efficiently. Although non-parametric in-context editing is robust for single-fact correction, it degrades severely on complex reasoning tasks such as multi-hop question answering. A fine-grained failure analysis identifies structural degradation as a central bottleneck. Under parametric priors, a model often deviates from the required factual-transition structure by taking shortcuts or stopping early, preventing injected knowledge from propagating along a valid reasoning skeleton. To address this problem, we propose StructKE, a structure-aware in-context editing framework. StructKE reformulates multi-hop editing as a controllable four-stage search process with structure prediction, structure-constrained generation, conflict-aware filling, and SDPR selection. It first trains StructEst, a lightweight estimator, to predict the target hop count, then filters candidate reasoning paths under the target-hop constraint, performs conflict-aware filling, and finally selects the answer through a Structure-Diagnostic Pairwise Ranker (SDPR) over structural and consistency signals. Experiments on multiple benchmarks show that StructKE mitigates structural degradation in multi-hop knowledge editing and achieves strong gains over the compared baselines by preserving structural integrity.

1 INTRODUCTION

Large language models (LLMs) perform strongly on knowledge-intensive tasks(Team et al., 2023; Guo et al., 2025; Achiam et al., 2023), but their static parametric knowledge can become misaligned with evolving facts. This temporal mismatch reduces reliability when models face news, updated databases, or other changing facts. Continual pretraining can update such knowledge, yet its cost and forgetting risk make it impractical for frequent updates. Knowledge editing (KE)(Yao et al., 2023; Zhang et al., 2024) addresses this limitation by modifying specific factual associations without retraining the whole model, enabling targeted and low-cost updates.

Existing KE methods mainly follow parametric editing and non-parametric editing, also called in-context editing (ICE). Parametric methods such as ROME(Meng et al., 2022) and MEMIT(Meng et al., 2023) rewrite internal components so that new knowledge is stored in weights, but this intervention may cause forgetting, rollback difficulty, and reasoning damage. In contrast, IKE(Cheng et al., 2023), MeLLo(Zhong et al., 2023), and other context-based methods freeze parameters and inject knowledge through the input context. This non-invasive property avoids weight corruption and is especially useful for closed-source models or personalized updates.

Although ICE is robust for single-fact correction, it degrades sharply on multi-hop question answering. Chain-of-thought (CoT) prompting tries to expose reasoning, but generated chains often remain superficial and miss the question's logical structure, consistent with studies on CoT unfaithfulness(Turpin et al., 2023) and structural degradation(Lanham et al., 2023; Adlakha et al., 2024). In KE, conflicts between parametric memory and injected knowledge make the mismatch between required and generated reasoning structure especially severe. We refer to this failure as structural degradation: the generated trajectory no longer matches the factual transition structure required by the edited question.

Figure 1 illustrates this structural failure. In the ideal case, a four-hop question is answered by following the edited chain step by step: the edited director fact leads to the director's birth city, then to the country, and finally to the country's language. The edited fact can affect the final answer only because the intermediate transitions are preserved. In contrast, under strong parametric priors, the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

model may collapse the trajectory into a shortcut from the movie to a statistically likely language, skipping the intermediate nodes where the edited fact should propagate. More generally, for a question requiring N factual transitions, the effective generated chain may be shorter because of shortcuts or early stopping ($L < N$), or longer because of redundant expansion and loops ($L > N$). This hop-level mismatch is an observable symptom of structural degradation, not a complete definition of reasoning structure: even when edited facts are retrieved, they cannot determine the answer if generation has left the factual-transition skeleton where those facts are needed.

We propose StructKE, a structure-aware ICE framework built on a plan-then-fill principle. StructKE separates path planning from entity filling: it first constructs a structurally valid reasoning skeleton, then injects edited knowledge into that skeleton, and finally selects the most structurally and semantically consistent answer. Concretely, StructKE first predicts the target hop count and searches for masked reasoning paths under hop and type constraints. It then performs conflict-aware filling over the retained skeletons, forcing edited facts into the positions where they are structurally needed. Finally, a Structure-Diagnostic Pairwise Ranker (SDPR) selects the answer using structural, confidence, and consistency signals.

Our contributions are summarized as follows:

- First, we identify structural errors as a major source of failure in multi-hop KE. Through fine-grained failure decomposition, we show that many editing failures arise because the generated trajectory deviates from the required factual-transition structure, preventing edited facts from propagating to the final answer.
- Second, we design StructEst, a lightweight reasoning-hop estimator that generates stepwise analysis and a target hop count, providing a callable structural prior for candidate generation. Experiments show that StructEst substantially outperforms several much larger general-purpose models on target-hop prediction.
- Third, we introduce StructKE, a structure-aware in-context editing framework that uses StructEst to guide target-hop-constrained candidate search, conflict-aware filling, and SDPR selection. Experiments on multiple multi-hop editing benchmarks show strong gains across evaluated backbones and datasets, and ablations show that structural constraints and SDPR provide complementary robustness in long-chain reasoning.

2 Preliminary Study

2.1 Problem Definition and Notation

Following the common setup of multi-hop knowledge editing(Zhong et al., 2023), let f_θ be a pretrained LLM and let an edit request be $e = (s, r, o^*)$, where o^* denotes the updated target object for subject s under relation r. We define a k-hop question as Q. Solving Q requires a reasoning chain $\mathcal{C} = \{t_1, t_2, \ldots, t_k\}$, where each node $t_i = (s_i, r_i, o_i)$ is a factual triple, adjacent triples are connected by $o_i = s_{i+1}$, and at least one triple t_j is modified by the edit request e.

The task is to answer the question under the edited knowledge state. Formally, the objective is

$$\arg\max_A P(A \mid Q, e; \theta), \tag{1}$$

where A^* is the final entity o_k of the reasoning chain. The desired output is $A = A^*$ rather than the stale answer implied by outdated parametric knowledge.

2.2 Empirical Analysis: The Structural Bottleneck

Figure 1 illustrates a representative shortcut-reasoning pattern induced by parametric priors. To understand the main bottleneck of CoT-based methods in multi-hop knowledge editing, we conduct a fine-grained failure analysis on MQuAKE-CF-3k-v2(Zhong et al., 2023). This benchmark contains 3,000 complex samples involving counterfactual updates and covers 2-hop to 4-hop questions. A model must use injected knowledge and preserve a valid logical derivation.

We focus on multi-hop editing methods with interpretable trajectories: MeLLo(Zhong et al., 2023), PokeMQA(Gu et al., 2024a), RAE(Shi et al., 2024), EditCoT(Wang et al., 2025a), and DecKER(Wang

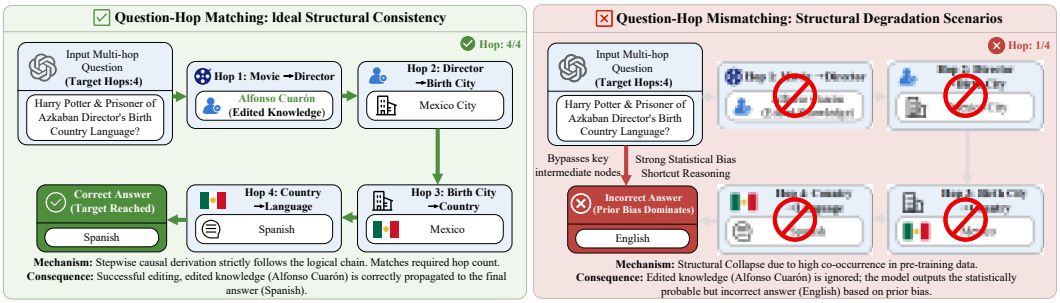

Figure 1: Shortcut reasoning induced by parametric priors

et al., 2025b). The target hop count follows the dataset annotation: the number of non-redundant factual transitions in the updated gold reasoning chain. This provides a consistent reference for evaluating whether a method preserves the intended structure under editing.

For methods with explicit subquestion decomposition, graph paths, or final reasoning trajectories, we recover the effective fact chain on which the final answer depends and compare its length with the target hop count. This comparison does not measure the number of written CoT lines or search iterations. It measures how many effective factual transitions the final answer actually relies on. For MeLLo, PokeMQA, and EditCoT, this chain can be recovered from the explicit subquestion chain or final CoT. For RAE, we use the number of relation-object transitions in the final graph path. For DecKER, we use its case-level diagnostic category and map hop-count mismatch to structural error. Structural errors are failed samples with a recovered chain-length mismatch, while other errors comprise the remaining failed samples.

Table 1: Failure Decomposition in Multi-Hop Editing

Method	Structural Error (%)	Other Error (%)
DecKER	50.85	49.15
MeLLo	46.59	53.41
PokeMQA	74.75	25.25
RAE	86.89	13.11
EditCoT	92.26	7.74

Table 1 shows that structural errors are a substantial source of failure across the compared methods, with particularly high rates for PokeMQA, RAE, and EditCoT. MeLLo has a more balanced profile, indicating that matching the target length is not sufficient when intermediate entities or answers remain incorrect. DecKER also exhibits both structural and non-structural failures, motivating an explicit focus on controlling the reasoning skeleton before knowledge filling.

These results suggest that the main weakness of current non-parametric multi-hop editing methods is not only content-level filling. Failure often occurs earlier at the structural level. One failure mode is fact-chain mismatch, where shortcuts, redundant expansion, or loops deviate from the required factual transitions. Another mode arises when explicit decomposition appears to match the target length, while intermediate entities still drift toward old knowledge. Without guiding the model into a semantically correct fact-transition skeleton, content prompts, retrieval augmentation, and local edit memories cannot reliably prevent structural degradation.

3 METHOD

Figure 2 overviews StructKE's structure-aware candidate search, conflict-aware filling, and SDPR selection.

3.1 STAGE 1: BUILDING AND TRAINING THE STRUCTEST HOP ESTIMATOR

StructEst predicts the target hop count before candidate generation, giving StructKE a shared structural prior for filtering, recovery, and SDPR selection. Given a question q, StructEst outputs $Y = (R, h)$,

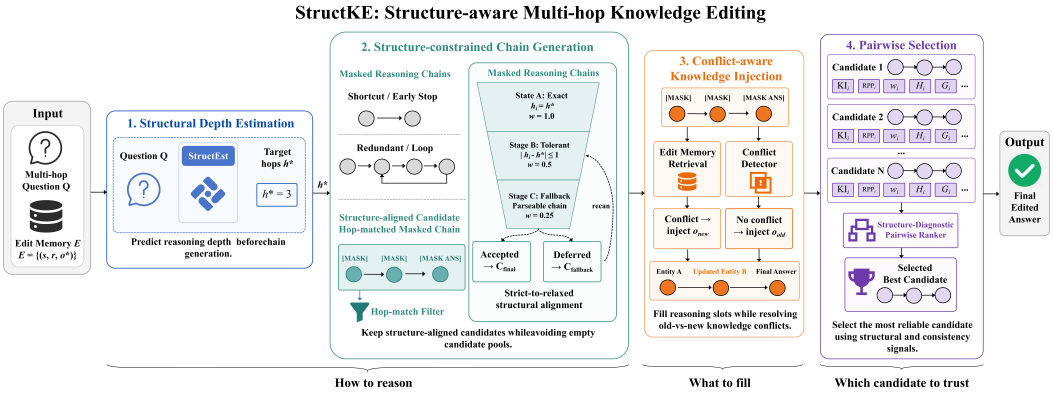

Figure 2: The Four-Stage Structure-Aware Editing Framework of StructKE

where R is a stepwise reasoning analysis and h is the predicted hop count. Since our setting covers $\mathcal{H} = \{2, 3, 4\}$, the final prediction is constrained to this set. The training objective is

$$\mathcal{L}_{est} = -\log P_\phi(R, h \mid q), \tag{2}$$

where ϕ denotes trainable parameters. Generating R before h encourages structural cues beyond surface semantics and makes the estimated hop count a callable prior rather than an implicit byproduct of free-form generation. This prior is used consistently by the later generator, controller, and SDPR.

StructEst uses Qwen2.5-7B-Instruct(Qwen et al., 2024) with LoRA(Hu et al., 2022). Training first learns general hop estimation from MuSiQueAns(Trivedi et al., 2022), then adapts on hop-balanced MQuAKE-CF structural samples with replay; to avoid leakage, we remove MQuAKE-CF cases overlapping with MQuAKE-CF-3k-v2, which is later used as a main test benchmark in our experiments. At inference, deterministic decoding restricts outputs to $\mathcal{H}$, parses JSON, uses explicit step markers when available, and otherwise falls back to the hops field. For the general-purpose model comparisons in Table 2, we use the publicly documented model versions from Google DeepMind, DeepSeek-AI, Z.AI, Moonshot AI, and Qwen(Google DeepMind, 2025; DeepSeek-AI et al., 2025; Z.AI, 2025; Kimi Team et al., 2025; Qwen et al., 2024).

Table 2: Target-Hop Prediction Accuracy

Model	Parameters	Acc (%)
Gemini-3-Flash	N/A	72.13
Qwen2.5-7B-Instruct	7B	59.89
DeepSeek-V3.2	671B	60.51
GLM-4.6	357B	71.84
Kimi-K2-Instruct	1000B	69.31
StructEst	7B	**93.34**

StructEst reaches 93.34% exact accuracy, outperforming its unfine-tuned backbone and several larger general-purpose models (Table 2). This result supports using explicit structural supervision for hop-count prediction rather than relying on general semantic ability alone. StructEst therefore supplies the later stages with a stable hop prior while remaining independent of the evaluated backbone LLMs. Keeping this estimator fixed also makes later performance changes easier to attribute to candidate generation, filling, and selection.

3.2 Stage 2: Structure-Constrained Candidate Reasoning-Chain Generation

Given q, target hop count h^*, and budget N, StructKE searches for masked reasoning chains rather than sampling one free-form chain. It first generates a direct answer a_d and estimates confidence. Forward generation derives an answer from q, while backward generation uses a_d as a weak anchor; backward sampling is disabled when a_d is empty, too short, or refusal-like, and otherwise bounded to avoid anchor-driven error propagation. This gate lets backward search recover useful intermediate nodes when the direct answer is plausible without forcing an unreliable answer into the candidate pool.

Each candidate must contain [STEP] markers, [MASK k] slots for intermediate entities, [MASK ANS] for the final answer, and a category block assigning a type to every mask. The number of [STEP] markers defines $\hat{h}_i$, while the mask slots decouple path planning from entity filling. Deterministic parsing records the step count, mask slots, type constraints, generation direction, and token trace, so candidate acceptance does not rely on an additional review model.

StructKE maintains $\mathcal{C}_{final}$ and $\mathcal{C}_{fallback}$ under a three-state controller. State A accepts exact-hop candidates with $w_i = 1.0$; State B recovers candidates with $|\hat{h}_i - h^*| \leq 1$ and $w_i = 0.5$; State C keeps remaining parseable candidates with [MASK ANS] and valid types with $w_i = 0.25$. When exact-hop candidates are insufficient, the controller relaxes from A to B to C and rescans fallback candidates, preserving a state-level confidence signal for SDPR. It only decides candidate availability, not final correctness. For every accepted c_i, StructKE records S_i, $g_i \in \{A, B, C\}$, $d_i \in \{forward, backward\}$, $\hat{h}_i$, and w_i for SDPR. The structural constraint is applied before entity filling. This ordering prevents edited facts from being injected into paths that have already skipped required transitions, and it gives SDPR explicit structural states rather than post-hoc guesses.

3.3 Stage 3: Conflict-Aware Filling

This module fills the masked slots in each retained candidate with edited knowledge or type-constrained model predictions. For a mask in step t, StructKE builds query q_t by removing mask markers and appending the most recent filled entity if available. Retrieval is restricted to the edit memory $\mathcal{E}$, the union of all edit requests. Conflict detection first uses retrieval scores:

$$\begin{array}{ll} S(d_1) & > \alpha \\ S(d_1) - S(d_2) & > \beta. \end{array} \tag{3}$$

Here $S(\cdot)$ is dot-product similarity and d_i is the i-th retrieved item. If the score gate is not triggered, StructKE asks the LLM whether the discarded original fact supports the current step; support indicates reasoning from outdated parametric memory. This hybrid detector keeps retrieval restricted to $\mathcal{E}$ while handling edited facts that are semantically relevant but not sharply separated in embedding space. Under conflict, free generation is blocked and the object of the retrieved edited triple fills the mask. Otherwise, the slot is filled as a constrained cloze task under its type constraint. Filled entities are propagated to later occurrences of the same mask until [MASK ANS] is filled, keeping later steps consistent with earlier filling decisions.

3.4 Stage 4: SDPR Selection

SDPR selects the final answer through pairwise preference ranking, which is more expressive than a fixed weighted sum because it compares conflicting signals, such as a structurally exact but low-confidence path versus a high-confidence shortcut. Inspired by pairwise learning-to-rank and LambdaMART-style boosted ranking(Burges et al., 2005; Burges, 2010), SDPR lets structural state, hop match, generation direction, knowledge consistency, and answer-group statistics interact. SDPR is question-local: it learns relative preferences among candidates for the same question instead of a global relevance score shared across questions. It represents each candidate c_i by

$$\mathbf{x}(c_i) = [KI_i, RPP_i, H_i, G_i, D_i, w_i, F_i, \mathbf{r}_i, \mathbf{a}_i]. \tag{4}$$

Here KI_i is the knowledge-injection consistency score, RPP_i is the path-planning certainty score derived from token-level prediction entropy, $H_i = \mathbb{I}[\hat{h}_i = h^*]$ indicates whether the candidate matches the target hop count, G_i and D_i encode the structural state and generation direction, w_i is the state confidence prior, F_i contains candidate-pool context such as pool size and expected hop count, $\mathbf{r}_i$ contains the within-pool ranks of KI and RPP, and $\mathbf{a}_i$ contains answer-level aggregation features such as answer frequency, source diversity, and state diversity. These features combine local path quality with pool-level evidence, so repeated answers from diverse structural states can be favored over isolated but superficially confident candidates.

The KI score measures compatibility between a filled entity e and type constraint t using a yes/no judgment:

$$\begin{aligned} KI(e,t) = \mathbb{I}[&P_M(\text{"yes"} \mid e, t) > \\ &P_M(\text{"no"} \mid e, t)]. \end{aligned} \tag{5}$$

Candidate-level KI_i is the consistency ratio over checked entities, optionally discounted by w_i. RPP_i is the negative average prediction entropy during chain generation:

$$PE_t = -\sum_{\tilde{w} \in V} p_t(\tilde{w}) \log p_t(\tilde{w}), \tag{6}$$

$$RPP_i = -\frac{1}{|E_i|} \sum_{t \in E_i} PE_t. \tag{7}$$

A larger RPP_i indicates lower average uncertainty when the model generated the reasoning skeleton. Training details are reported in Appendix F.1. At inference time, SDPR compares all candidate pairs through feature differences $\mathbf{x}_{ij} = \mathbf{x}(c_i) - \mathbf{x}(c_j)$:

$$P(c_i \succ c_j \mid \mathbf{x}_{ij}) = f_{\mathrm{SDPR}}(\mathbf{x}_{ij}). \tag{8}$$

Feature differences keep comparisons invariant to question-specific scales and focus the model on within-pool relative preference. For $p_{ij} = P(c_i \succ c_j \mid \mathbf{x}_{ij})$, tournament scores are accumulated as

$$s(c_i) \leftarrow s(c_i) + p_{ij}, \quad s(c_j) \leftarrow s(c_j) + (1 - p_{ij}). \tag{9}$$

The final candidate is

$$c^* = \arg\max_{c_i \in \mathcal{C}} s(c_i). \tag{10}$$

If multiple candidates have the same tournament score, KI_i is used as the tie breaker. The answer associated with c^* is returned. This tournament form keeps all candidates comparable even when they come from different structural states or generation directions.

4 Experiments

4.1 Experimental Setup

4.1.1 Datasets and Metrics

We evaluate StructKE on four multi-hop question answering datasets. MQuAKE-CF-3k-v2 contains 3,000 questions and 2,764 edits, while MQuAKE-T contains 1,868 questions and 96 edits(Zhong et al., 2023); both cover 2-hop to 4-hop questions. RippleEdits-Popular contains 500 questions and 266 edits, and RippleEdits-Random contains 1,137 questions and 626 edits(Cohen et al., 2024); both are mainly 2-hop datasets. Following MeLLo(Zhong et al., 2023), we report multi-hop accuracy. For MQuAKE, each test case includes three rewritten questions, and a case is counted as correct if at least one rewritten version is answered correctly. RippleEdits contains one question per case.

4.1.2 Baselines and Retrieval Setting

We compare StructKE with three groups of knowledge editing methods: parametric methods, including MEMIT(Meng et al., 2023) and AlphaEdit(Fang et al., 2024); ICE methods, including MeLLo(Zhong et al., 2023), PokeMQA(Gu et al., 2024a), DeepEdit(Wang et al., 2024), EditCoT(Wang et al., 2025a), and DecKER(Wang et al., 2025b); and the knowledge-graph method RAE(Shi et al., 2024). For fair comparison, ICE methods use MeLLo-style contriever-msmarco(Izacard et al., 2021) retrieval over edited facts, while RAE uses graph-based retrieval.

4.1.3 Evaluation Models

We evaluate three instruction-tuned LLMs: Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Qwen2.5-14B-Instruct. All runs use each model's official chat template. Because MEMIT, AlphaEdit, and RAE are computationally expensive, we evaluate them only on Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct.

4.2 Accuracy Comparison

Table 3 shows that StructKE achieves the strongest overall performance on most backbone–dataset settings, with particularly consistent gains on MQuAKE-CF-3k-v2 and RippleEdits. On MQuAKE-CF-3k-v2, StructKE reaches 63.5%, 59.7%, and 62.8% with Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Qwen2.5-14B-Instruct, respectively. This benchmark requires edited facts to propagate

Table 3: Main Results on Multi-Hop Knowledge Editing. The best result under each backbone is in bold, and the second-best result is underlined.

Models	Methods	MQuAKE-CF-3k-v2	MQuAKE-T	Ripple-Pop	Ripple-Rand
Llama-3.1-8B-Instruct	MEMIT	6.8	44.1	8.4	9.1
	AlphaEdit	7.0	42.0	8.6	9.6
	MeLLo	11.0	57.7	25.8	32.7
	PokeMQA	16.8	77.8	30.4	33.1
	DeepEdit	8.9	57.5	3.4	5.0
	RAE	54.3	50.9	20.6	17.0
	EditCoT	36.0	74.6	31.6	22.3
	DecKER	59.0	81.3	**47.4**	44.9
	StructKE	**63.5**	**81.8**	46.2	**45.9**
Qwen2.5-7B-Instruct	MEMIT	6.6	21.5	9.8	17.2
	AlphaEdit	7.1	21.1	11.0	17.0
	MeLLo	1.7	50.3	3.6	1.4
	PokeMQA	7.1	45.3	7.0	7.6
	DeepEdit	12.3	25.3	6.0	12.0
	RAE	38.3	28.7	24.0	18.3
	EditCoT	26.2	**74.5**	29.2	30.7
	DecKER	49.8	70.8	39.8	38.3
	StructKE	**59.7**	71.4	**42.4**	**44.5**
Qwen2.5-14B-Instruct	MeLLo	1.2	26.4	4.6	2.3
	PokeMQA	2.3	32.4	5.9	3.9
	DeepEdit	7.3	51.7	9.4	14.4
	EditCoT	37.0	82.7	35.0	33.5
	DecKER	56.6	83.6	44.0	46.1
	StructKE	**62.8**	**84.4**	**44.8**	**46.6**

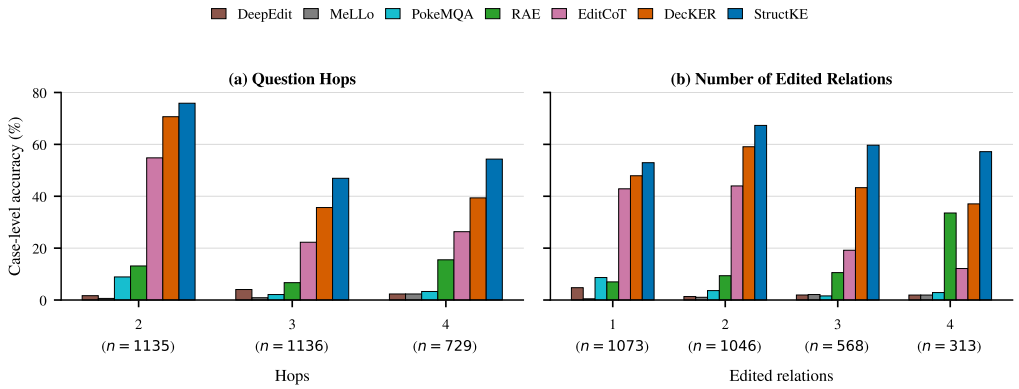

Figure 3: Case-level accuracy on MQuAKE-CF-3k-v2 with Qwen2.5-7B-Instruct, stratified by (a) reasoning depth and (b) the number of edited relations. StructKE leads every group, with larger margins in the more structurally demanding settings.

through 2-hop to 4-hop reasoning chains, making it a direct test of structural reliability in multi-hop editing.

The gains extend across the evaluated backbones and are also visible on the shorter-chain RippleEdits benchmarks, while the margins vary across datasets. This pattern suggests that explicit target-hop estimation, constrained candidate generation, conflict-aware filling, and structure-aware selection provide a consistent mechanism for improving edited-fact propagation, with the clearest effect when the task requires preserving longer dependency chains. Parametric editing and unconstrained in-context methods remain competitive in some settings, indicating that structural control is complementary to, rather than a replacement for, existing editing strategies.

4.3 Robustness to Reasoning Depth and Edit Scale

To test robustness as structural demands increase, we stratify the same 3,000 MQuAKE-CF-3k-v2 cases by dataset-annotated reasoning depth and number of edited relations. Figure 3 reports case-level accuracy along both dimensions.

StructKE achieves the highest accuracy in every group, and its lead becomes larger in the most structurally demanding settings. Its margin over the strongest competing method grows from 5.20 points on 2-hop cases to 14.95 points on 4-hop cases, while the corresponding margin increases from 5.04 points with one edited relation to 20.13 points with four. At four hops, StructKE reaches 54.32% accuracy; with four edited relations, it reaches 57.19%. These widening margins support the central design of StructKE: target-hop estimation and exact-hop candidate control preserve paths of the required depth, while conflict-aware filling and SDPR maintain reliable propagation as the edit set grows.

4.4 Final-Output Structural Consistency

We further evaluate whether the actual hop count $\hat{h}$ of the final output matches the dataset target hop count h. Outputs are grouped by final correctness, and we report hop-match rates within correct outputs, wrong outputs, and all outputs. For each MQuAKE case with three paraphrases, we use the hop count of the first correct selected paraphrase if the case is answered correctly; otherwise, we use the hop count of the first answered selected paraphrase.

Table 4: Hop-match rates conditioned on final correctness. Structural errors are failed cases with a recoverable fact chain whose hop count mismatches the target; other errors are errors besides hop match.

Method	Match | Correct (%)	Match | Wrong (%)	Overall Match (%)
MeLLo	44.12	41.54	41.57
EditCoT	6.77	7.74	7.63
PokeMQA	28.19	17.78	18.30
RAE	68.93	13.11	19.40
DecKER	83.67	45.82	64.67
StructKE	**91.73**	**72.81**	**84.10**

RAE and PokeMQA show that hop match is more frequent among correct outputs than wrong outputs. MeLLo has a relatively high overall match rate, but its accuracy remains very low, indicating semantic drift inside apparently matched chains. EditCoT shows similar match rates conditioned on correct and wrong outputs, so explicit CoT editing alone does not reliably convert hop match into correct answers. DecKER achieves high hop-match coverage, confirming that decoupled editing already improves structural alignment. StructKE further raises the overall match rate to 84.10% and reaches 91.73% among correct cases, indicating that target-hop-aware candidate generation and SDPR selection more consistently place the final answer on a structurally matched reasoning path. The remaining wrong-but-matched cases also show that hop alignment is not sufficient by itself; reliable editing still requires correct entity filling and answer selection within the matched structure.

4.5 Ablation Study

We run a component-level diagnostic ablation on MQuAKE-CF-300-v2 with Qwen2.5-7B-Instruct. This fixed 300-case subset preserves the data format, edit setting, and case-level evaluation protocol of MQuAKE-CF-3k-v2. The ablations remove SDPR or StructEst, restrict the search direction, replace hybrid conflict detection with a score-only variant, remove exact-hop candidates, or replace the predicted hop count with the annotated value as an oracle diagnostic.

In Table 5, Full StructKE is the complete deployable model. The complete model reaches 65.67% on the 300-case diagnostic subset.

Removing SDPR reduces accuracy by 5.34 points, while removing exact-hop candidates produces the largest deployable-model drop of 6.00 points. Removing StructEst reduces accuracy by 5.00 points, and the forward-only, backward-only, and score-only conflict variants also underperform the full system. These results indicate that the components address different failure points: StructEst and exact-hop filtering shape the candidate pool, conflict-aware filling controls edited-fact propagation, and SDPR selects among the resulting candidates. The oracle-hop result of 67.33% indicates remaining headroom from more accurate structural estimation. Because these stages interact, the individual drops should not be interpreted as additive contributions.

Table 5: Component-level diagnostic ablation on MQuAKE-CF-300-v2 with Qwen2.5-7B-Instruct. The full model accuracy is 65.67%; the Δ column is relative to this value.

Setting	Acc. (%)	Δ vs. Full
Full StructKE	**65.67**	0.00
w/o SDPR (Base output)	60.33	-5.34
w/o SDPR (Prompt reranking)	62.33	-3.34
w/o StructEst	60.67	-5.00
Forward only	62.67	-3.00
Backward only	62.33	-3.34
Score-only conflict detector	61.00	-4.67
No exact-hop candidates	59.67	-6.00
Oracle hop (upper bound)	67.33	+1.66

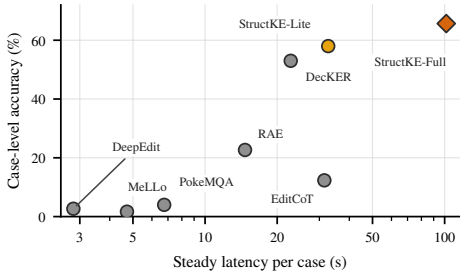

Figure 4: Accuracy–latency trade-off on MQuAKE-CF-300-v2 with Qwen2.5-7B-Instruct.

4.6 Inference Efficiency

Figure 4 compares case-level accuracy and per-case latency on the same diagnostic subset. StructKE-Lite uses three candidate samples with no additional resampling, while StructKE-Full uses the complete candidate budget and resampling configuration.

StructKE-Lite provides the lower-latency operating point, whereas StructKE-Full prioritizes accuracy at a higher latency. Both are configurations of the same structure-aware framework.

5 Conclusion

Our analysis identifies structural degradation as a key bottleneck in multi-hop knowledge editing. Shortcuts, early stopping, redundancy, and fact-chain mismatch can prevent edited facts from propagating through valid reasoning chains. We propose StructKE, a structure-aware in-context editing framework in which StructEst provides an explicit target-hop prior for constrained candidate generation, conflict-aware filling, and SDPR selection. Experiments across multiple benchmarks and backbones show that StructKE improves multi-hop editing robustness by preserving structurally matched reasoning paths. These results suggest that explicit structural estimation and control are practical principles for reliable long-chain knowledge editing.

Ethical Considerations

This work studies controlled factual updates in language models. While the goal is to improve reliability under updated knowledge, editing mechanisms can be misused if the edit memory contains incorrect or biased facts. We therefore view StructKE as a research framework for analyzing and improving multi-hop editing robustness. All experiments use existing public benchmarks and pretrained models; we do not collect personal data or involve human participants.

Reproducibility Statement

We support reproducibility through the detailed dataset and evaluation protocol in Section 4.1, the StructEst training configuration and runtime hyperparameters, the complete algorithmic workflow, and the prompt templates. We plan to release the implementation, configuration files, prompts, and processed candidate-generation logs upon publication, subject to the licenses and terms of the original datasets and models.

AI Use Statement

Generative AI tools were used for language polishing, LaTeX editing assistance, checklist drafting, and coding and debugging support. They were not used to generate experimental results or formulate the central method. The authors reviewed and verified all AI-assisted text, code, technical claims, citations, and experimental content, and take responsibility for the final content of this work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. doi: 10.48550/arXiv.2303.08774. URL https://arxiv.org/abs/2303.08774.

Vaibhav Adlakha, Parishad BehnamGhader, Xing Han Lu, Nicholas Meade, and Siva Reddy. Evaluating correctness and faithfulness of instruction-following models for question answering. *Transactions of the Association for Computational Linguistics*, 12:681–699, 2024. doi: 10.1162/tacl_a_00667. URL https://aclanthology.org/2024.tacl-1.38/.

Christopher J. C. Burges. From ranknet to lambdarank to lambdamart: An overview. *Microsoft Research Technical Report MSR-TR-2010-82*, 2010. URL https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/.

Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 89–96, New York, NY, USA, 2005. Association for Computing Machinery. doi: 10.1145/1102351.1102363. URL https://doi.org/10.1145/1102351.1102363.

Siyuan Cheng, Bozhong Tian, Qingbin Liu, Xi Chen, Yongheng Wang, Huajun Chen, and Ningyu Zhang. Can we edit multimodal large language models? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13877–13888, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.856. URL https://aclanthology.org/2023.emnlp-main.856/.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298, 2024. doi: 10.1162/tacl_a_00644. URL https://aclanthology.org/2024.tacl-1.16/.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.581. URL https://aclanthology.org/2022.acl-long.581/.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, et al. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025. doi: 10.48550/arXiv.2512.02556. URL https://arxiv.org/abs/2512.02556.

Mengru Ding, Hanmeng Liu, Zhizhang Fu, Jian Song, Wenbo Xie, and Yue Zhang. Break the chain: Large language models can be shortcut reasoners. *arXiv preprint arXiv:2406.06580*, 2024. doi: 10.48550/arXiv.2406.06580. URL https://arxiv.org/abs/2406.06580.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AlphaEdit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*, 2024. doi: 10.48550/arXiv.2410.02355. URL https://arxiv.org/abs/2410.02355.

Google DeepMind. Gemini 3 Flash Model Card. Model card, 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf.

Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. PokeMQA: Programmable knowledge editing for multi-hop question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8069–8083, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.438. URL https://aclanthology.org/2024.acl-long.438/.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 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*, pp. 16801–16819, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.934. URL https://aclanthology.org/2024.emnlp-main.934/.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. doi: 10.48550/arXiv.2501.12948. URL https://arxiv.org/abs/2501.12948.

Zhiyuan He, Binghan Chen, Tianxiang Xiong, Ziyang Sun, Mozhao Zhu, and Xi Chen. On the limitations of rank-one model editing in answering multi-hop questions. *arXiv preprint arXiv:2601.04600*, 2026. doi: 10.48550/arXiv.2601.04600. URL https://arxiv.org/abs/2601.04600.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021. doi: 10.48550/arXiv.2112.09118. URL https://arxiv.org/abs/2112.09118.

Tianjie Ju, Yijin Chen, Xinwei Yuan, Zhuosheng Zhang, Wei Du, Yubin Zheng, and Gongshen Liu. Investigating multi-hop factual shortcuts in knowledge editing of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8987–9001, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.486. URL https://aclanthology.org/2024.acl-long.486/.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025. doi: 10.48550/arXiv.2507.20534. URL https://arxiv.org/abs/2507.20534.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023. doi: 10.48550/arXiv.2307.13702. URL https://arxiv.org/abs/2307.13702.

Gengyang Li, Yifeng Gao, Yuming Li, and Yunfang Wu. Thinkless: A training-free inference-efficient method for reducing reasoning redundancy. *arXiv preprint arXiv:2505.15684*, 2025. doi: 10.48550/arXiv.2505.15684. URL https://arxiv.org/abs/2505.15684.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15817–15831. PMLR, 2022. URL https://proceedings.mlr.press/v162/mitchell22a.html.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, and Fei Huang. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. doi: 10.48550/arXiv.2412.15115. URL https://arxiv.org/abs/2412.15115.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022. doi: 10.48550/arXiv.2210.01240. URL https://arxiv.org/abs/2210.01240.

Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 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*, pp. 2056–2066, New York, NY, USA, 2024. Association for Computing Machinery. doi: 10.1145/3627673.3679722. URL https://doi.org/10.1145/3627673.3679722.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. doi: 10.48550/arXiv.2312.11805. URL https://arxiv.org/abs/2312.11805.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ed3fea9033a80fea1376299fa7863f4a-Abstract-Conference.html.

Martin Tutek, Fateme Hashemi Chaleshtori, Ana Marasović, and Yonatan Belinkov. Measuring chain of thought faithfulness by unlearning reasoning steps. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9935–9960, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.504. URL https://aclanthology.org/2025.emnlp-main.504/.

Shunsuke Ubukata. D-cot: Disciplined chain-of-thought learning for efficient reasoning in small language models. *arXiv preprint arXiv:2602.21786*, 2026. doi: 10.48550/arXiv.2602.21786. URL https://arxiv.org/abs/2602.21786.

Changyue Wang, Weihang Su, Qingyao Ai, Yichen Tang, and Yiqun Liu. Knowledge editing through chain-of-thought. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10673–10693, Suzhou, China, November 2025a. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.540. URL https://aclanthology.org/2025.emnlp-main.540/.

Changyue Wang, Weihang Su, Qingyao Ai, Yujia Zhou, and Yiqun Liu. Decoupling reasoning and knowledge injection for in-context knowledge editing. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24543–24562, Vienna, Austria, 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.1260. URL https://aclanthology.org/2025.findings-acl.1260/.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021. doi: 10.1162/tacl_a_00360. URL https://aclanthology.org/2021.tacl-1.11/.

Yiwei Wang, Muhao Chen, Nanyun Peng, and Kai-Wei Chang. DeepEdit: Knowledge editing as decoding with constraints. *arXiv preprint arXiv:2401.10471*, 2024. doi: 10.48550/arXiv.2401.10471. URL https://arxiv.org/abs/2401.10471.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022. URL https://proceedings.neurips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Yuchen Wu, Liang Ding, Li Shen, and Dacheng Tao. Reason-ke++: Aligning the process, not just the outcome, for faithful llm knowledge editing. *arXiv preprint arXiv:2511.12661*, 2025. doi: 10.48550/arXiv.2511.12661. URL https://arxiv.org/abs/2511.12661.

Sohee Yang, Nora Kassner, Elena Gribovskaya, Sebastian Riedel, and Mor Geva. Do large language models perform latent multi-hop reasoning without exploiting shortcuts? In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 3971–3992, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.205. URL https://aclanthology.org/2025.findings-acl.205/.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10222–10240, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.632. URL https://aclanthology.org/2023.emnlp-main.632/.

Z.AI. GLM-4.6. Hugging Face model card, 2025. URL https://huggingface.co/zai-org/GLM-4.6.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. How language model hallucinations can snowball. *arXiv preprint arXiv:2305.13534*, 2023. doi: 10.48550/arXiv.2305.13534. URL https://arxiv.org/abs/2305.13534.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*, 2024. doi: 10.48550/arXiv.2401.01286. URL https://arxiv.org/abs/2401.01286.

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15686–15702, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.971. URL https://aclanthology.org/2023.emnlp-main.971/.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. SELF-DISCOVER: Large language models self-compose reasoning structures. *Advances in Neural Information Processing Systems*, 37:126032–126058, 2024. doi: 10.52202/079017-4004. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e41efb03e20ca3c231940a3c6917ef6f-Abstract-Conference.html.

A Data and Code Availability

Our experiments use publicly available benchmarks, including MQuAKE-CF-3k-v2, MQuAKE-T, and RippleEdits, together with publicly available pretrained backbones. We use all datasets, pretrained models, and retrieval resources under their original licenses and terms of use. We plan to release the implementation, prompts, configuration files, and processed candidate-generation logs upon publication, subject to the licenses and terms of the original datasets and models. During anonymous review, any supplementary materials will be provided through anonymized links.

B Limitations

StructKE focuses on multi-hop knowledge editing benchmarks with relatively explicit entity-relation transitions. Its structural signals may be less informative for questions whose reasoning paths are highly ambiguous, underspecified, or depend on implicit commonsense assumptions rather than identifiable factual transitions. The framework also relies on the quality of the edit memory and candidate parsing; noisy edit records or malformed generated skeletons may weaken the downstream filling and selection stages. Finally, our structural diagnostics mainly use hop-level consistency as a proxy for factual-transition structure, which may not capture all semantic errors in intermediate states.

C StructEst Details and Performance

This appendix reports additional diagnostics, implementation details, prompt templates, structural-consistency analyses, ablations, and a case study.

This section reports additional StructEst metrics and the training configuration.

C.1 Additional StructEst Diagnostics

The main text reports exact hop-prediction accuracy in Table 2. Table 6 provides complementary diagnostics on tolerance, average error, and automatic parsing stability.

Table 6: Additional StructEst Diagnostics

Metric	Value
Tol-1 Accuracy (%)	99.70
MAE	0.0696
JSON Parse Rate (%)	99.97

Tol-1 Accuracy measures whether the predicted hop count is within one hop of the annotated target, and MAE measures the average hop-count deviation. The high Tol-1 Accuracy and low MAE indicate that most StructEst errors are small, matching the tolerance-alignment mechanism in Stage 2. The JSON parse rate measures whether the estimator output can be automatically parsed, and its high value supports stable downstream caching and candidate generation.

C.2 StructEst Training Configuration

The LoRA adapters are applied to the main attention and feed-forward projections. Table 7 reports the exact target modules together with the two-stage training configuration.

C.3 Core Runtime Configuration

This section reports the fixed inference-time configuration shared across all StructKE evaluations. These hyperparameters control candidate-pool size, allowed hop range, fallback behavior, and conflict-detection sensitivity, and are kept unchanged across backbones and datasets unless otherwise stated.

Table 7: Two-Stage LoRA Training Configuration for StructEst

Parameter	Value
Backbone	Qwen2.5-7B-Instruct
Adaptation method	LoRA
Target modules	q/k/v/o and gate/up/down projections
Stage I training data	Augmented MuSiQueAns (4,500 train / 600 dev)
Stage II training data	MQuAKE-CF structural adaptation with replay (12,987 train / 933 dev)
LoRA rank / alpha / dropout	64 / 128 / 0.1
Learning rate (Stage I / II)	5×10^{-5} / 2.5×10^{-5}
Effective batch size (Stage I / II)	16 / 16
Epochs (Stage I / II)	1 / 1
Maximum sequence length	1024
Precision	bf16
Model selection criterion	Validation loss

Table 8: Core Runtime Configuration of StructKE

Parameter	Value
Candidate budget N	6
Target-hop range	$\{2, 3, 4\}$
Maximum resampling rounds	2
State priors $w_A/w_B/w_C$	1.0 / 0.5 / 0.25
Retrieval thresholds α/β	1.5 / 0.1

D Related Work

D.1 Knowledge Editing Paradigms

Knowledge editing modifies factual knowledge in LLMs while limiting side effects on unrelated knowledge. Parametric methods update model weights directly: KN(Dai et al., 2022) fine-tunes selected layers, while ROME(Meng et al., 2022) and MEMIT(Meng et al., 2023) locate factual associations and apply rank-one updates. These methods are effective for single facts but can cause forgetting, harm reasoning ability(Gu et al., 2024b), and are hard to roll back. Non-parametric, in-context editing freezes the model and uses external memory or contextual guidance. IKE(Cheng et al., 2023) uses demonstrations, MeLLo(Zhong et al., 2023) retrieves edited facts during reasoning, and SERAC(Mitchell et al., 2022) separates edit-scope detection from answer generation. This black-box design fits closed-source models and frequent updates, but depends on whether the model can use injected facts in reasoning.

D.2 Multi-Hop Question Editing

Multi-hop editing requires edited facts to propagate across linked transitions rather than isolated associations. MeLLo(Zhong et al., 2023) decomposes a question into subquestions and retrieves edited knowledge for each step. PokeMQA(Gu et al., 2024a) adds programmable edit-scope detection, and EditCoT(Wang et al., 2025a) iteratively detects and rewrites chains of thought. These methods improve decomposition, retrieval, and chain rewriting, but still largely let the model organize the reasoning topology. StructKE targets this missing structural control explicitly.

D.3 Structural Failures in Reasoning Chains

Chain-of-thought reasoning improves LLM performance(Wei et al., 2022), but generated trajectories can be unfaithful(Tutek et al., 2025) or structurally unstable, including redundancy from irrelevant noise(Zhang et al., 2023), logical loops under low confidence(Li et al., 2025), premature termination(Saparov & He, 2022), and shortcut reasoning induced by pretraining priors(Yang et al., 2025). These failures are amplified in multi-hop editing, where injected facts conflict with parametric memory and can be bypassed by context drift, loops, or early stopping(Ju et al., 2024; Wu et al., 2025; He et al., 2026). General trajectory-control methods such as Self-Discover(Zhou et al., 2024), D-CoT(Ubukata, 2026), and shortcut analysis by Ding et al.(Ding et al., 2024) are not tailored to this edited-knowledge setting. StructKE instead applies explicit structural constraints to counter shortcuts, redundancy, and loops in multi-hop editing.

E Additional Structural Consistency Analysis

Using the same protocol as the preliminary study, we test whether hop-matched outputs are more likely to be correct. The target hop count comes from the annotated updated gold reasoning chain, while the method-side hop count is the recovered effective fact-chain length from the original trajectory.

Table 9: Accuracy Conditioned on Hop Match

Method	Match Rate (%)	Acc. | Match (%)	Acc. | Mismatch (%)	Δ Acc. (%)
MeLLo	41.57	1.20	1.08	+0.12
PokeMQA	18.30	7.65	4.37	+3.28
RAE	19.40	40.03	4.34	+35.69
EditCoT	7.63	9.61	10.93	-1.33
DecKER	64.67	64.43	23.02	+41.41

Table 9 shows positive conditional gains for RAE, PokeMQA, and DecKER, indicating that hop match increases their success probability. DecKER obtains the largest conditional gap, which confirms that structurally matched outputs form a much more reliable decision region for decoupled editing. MeLLo and EditCoT show that length alignment alone is insufficient: their hop-matched paths can still propagate old or ambiguous intermediate facts. This supports our focus on jointly constraining structural length and structural content while using SDPR to select among structurally plausible candidates.

F Ranking Supervision and Ablation Details

We describe the source of SDPR supervision and analyze how structural constraints affect the candidate pool. The supervision uses logged candidates rather than manual ranking labels.

F.1 SDPR Preference Supervision

SDPR obtains supervision from the candidate generation process rather than manual rankings or an external reranker dataset. To avoid test leakage, we remove cases that overlap with MQuAKE-CF-3k-v2 from MQuAKE-CF and sample 500 cases from the remaining training pool. These cases are expanded into 1,500 question-level samples, denoted MQuAKE-CF-train500. We run the first three StructKE stages on this set and use the logged candidate answers, candidate pools, and diagnostic features to construct preference supervision.

For each training question q, only candidates visible to the final SDPR module are kept. The correctness label $r_i \in \{0, 1\}$ is determined by normalized answer matching against the gold answer. For a pair (c_i, c_j), cases with $r_i = r_j$ are skipped; cases with $r_i \neq r_j$ form valid preferences. In practice, 1,499 of the 1,500 training questions have non-empty candidate pools, and questions containing both correct and incorrect candidates yield 748 pairwise training samples, with 374 samples in each direction.

To avoid inconsistent scales across questions, SDPR uses feature differences:

$$\mathbf{x}_{ij} = \mathbf{x}(c_i) - \mathbf{x}(c_j). \tag{11}$$

The label is

$$y_{ij} = \mathbb{I}[r_i > r_j]. \tag{12}$$

Each valid pair is also added in reverse, with $\mathbf{x}_{ji} = -\mathbf{x}_{ij}$ and $y_{ji} = 1 - y_{ij}$. The binary preference model is trained with cross-entropy over all valid pairs. We instantiate f_{SDPR} with a shallow gradient boosting tree to reduce overfitting on small preference data and capture nonlinear interactions among KI, RPP, hop match, state prior, and answer-group consistency.

G Method Details

We provide the complete StructKE workflow and implementation details.

Table 10: StructKE Algorithmic Workflow

Algorithm 1 StructKE Algorithmic Workflow

Input: Question q, base LLM M, StructEst E_ϕ, edit memory $\mathcal{E}$, candidate budget N, sampling budget B, hop set $\mathcal{H}$, retrieval thresholds α, β, state priors w_A, w_B, w_C, ranker f_{SDPR}.
Output: Final answer A.

Stage 1: Estimate target structure
$h^* \leftarrow \mathrm{Clamp}_{\mathcal{H}}(E_\phi(q))$
Stage 2: Generate and filter masked reasoning chains
$a_d \leftarrow M(q)$; estimate direct-answer confidence ρ and allocate forward/backward sampling from ρ
If the confidence gate fails, disable backward sampling; otherwise use a_d as the backward-chain anchor
$\mathcal{C}_{final} \leftarrow \emptyset, \mathcal{C}_{fallback} \leftarrow \emptyset$
Generate B masked chains with [STEP], [MASK], [MASK ANS], and type annotations
for each parsed chain c_i **do**
  Extract direction d_i, hop count $\hat{h}_i$, mask slots, type map, and token trace E_i
  If c_i is parseable and $\hat{h}_i = h^*$, add it to $\mathcal{C}_{final}$ with $g_i = \mathrm{A}, w_i = w_A$
  Otherwise add c_i to $\mathcal{C}_{fallback}$ for relaxed recovery
end for
If $|\mathcal{C}_{final}| < N$, recover candidates in $\mathcal{C}_{fallback}$ with $|\hat{h}_i - h^*| \leq 1$ and set $g_i = \mathrm{B}, w_i = w_B$
If $|\mathcal{C}_{final}| < N$, keep remaining parseable candidates with [MASK ANS] and valid types, and set $g_i = \mathrm{C}, w_i = w_C$
Stage 3: Conflict-aware filling
for each candidate $c_i \in \mathcal{C}_{final}$ **do**
  for each mask slot m_{ik} in step s_{ik} **do**
    $q_{ik} \leftarrow$ previous filled entity + $\textsc{RemoveMask}(s_{ik})$
    $\{(d_1, score_1), (d_2, score_2), \ldots\} \leftarrow \textsc{Retrieve}(\mathcal{E}, q_{ik})$
    if $score_1 > \alpha$ and $score_1 - score_2 > \beta$ **then** $IsEdited \leftarrow$ true
    else $IsEdited \leftarrow \textsc{ConflictCheck}(M, s_{ik}, \textsc{OriginalFact}(d_1))$
    if $IsEdited$ **then** $e_{ik} \leftarrow \textsc{Object}(d_1)$ **else** $e_{ik} \leftarrow M(q_{ik}, \mathrm{type}(m_{ik}))$
    Replace all occurrences of m_{ik} in c_i with e_{ik}; if $m_{ik} =$ [MASK ANS], store $A_i = e_{ik}$
  end for
end for
Stage 4: Select the final answer with SDPR
for each filled candidate c_i **do**
  Compute KI_i, RPP_i, $H_i = \mathbb{I}[\hat{h}_i = h^*]$, and encode $G_i, D_i, F_i, \mathbf{r}_i, \mathbf{a}_i$
  $\mathbf{x}(c_i) = [KI_i, RPP_i, H_i, G_i, D_i, w_i, F_i, \mathbf{r}_i, \mathbf{a}_i]$; initialize $s(c_i) \leftarrow 0$
end for
for each pair (c_i, c_j) in $\mathcal{C}_{final}$ **do**
  $p_{ij} \leftarrow f_{\mathrm{SDPR}}(\mathbf{x}(c_i) - \mathbf{x}(c_j))$
  $s(c_i) \leftarrow s(c_i) + p_{ij}$; $s(c_j) \leftarrow s(c_j) + (1 - p_{ij})$
end for
$c^* \leftarrow \arg\max_{c_i \in \mathcal{C}_{final}} s(c_i)$; break ties by KI_i
return $A(c^*)$

G.1 StructKE Algorithmic Workflow

Table 10 gives the complete inference workflow of StructKE.

Table 10 summarizes the four-stage editing workflow of StructKE. The diagnostic feature vector in the final SDPR stage follows the notation in the main method section.

H Baseline Implementation Details

All ICE baselines except RAE follow the MeLLo retrieval setting and use contriever-msmarco(Izacard et al., 2021) over the edited-fact knowledge base. For RAE, we adapt its graph retrieval implementation to keep the knowledge boundary consistent with the other methods. If the current subject and retrieved relation refer to an unedited triple, RAE does not directly read the object from Wikidata(Wang et al., 2021). Instead, the object is generated by the LLM from its own parameters. This prevents RAE from using external unedited graph facts unavailable to the other baselines.

I Key Prompt Templates

Tables 11, 12, and 13 present the main prompt templates used by StructKE for hop prediction and masked reasoning-chain generation.

Table 11: StructEst Hop-Prediction Prompt

System:
Role
You are an expert in multi-hop reasoning that must output structured JSON.
Task
Given the user's question, decide how many reasoning hops (steps) are required to reach the answer.
Guidelines for Hop Counting
- 2-hop: Start ->Bridge ->Answer (1 intermediate entity).
- 3-hop: Start ->Bridge A ->Bridge B ->Answer (2 intermediates).
- 4-hop: Start ->Bridge A ->Bridge B ->Bridge C ->Answer (3 intermediates).
Count hops only when the entity is necessary to continue the chain.
Mandatory Output Format
Respond with EXACTLY the following JSON object (no prose, no markdown):
{
"hops": <INTEGER>,
"path": ["Start", "...", "Answer"],
"reasoning": "Short explanation"
}
- The value of "hops" MUST be one of [2, 3, 4].
- "path" must list every entity in order; length = hops + 1.
- Do not add extra keys or comments. Output nothing before or after the JSON.
Current Question
The user question will be provided next. Output ONLY the JSON object.

Table 12: Forward Masked Reasoning-Chain Prompt

System:
Produce a masked forward chain of thought that answers the given question. Requirements: (1) output exactly one reasoning step per line, and every reasoning line MUST start with [STEP]; (2) do NOT skip any intermediate reasoning hop; (3) replace specific knowledge, entities, dates, or locations that are NOT in the text of the question with [MASK], numbered as [MASK 1], [MASK 2], ... in order of first appearance; (4) the final-answer entity MUST be replaced with [MASK ANS]; (5) if the same entity appears more than once, reuse the same [MASK i]; (6) after all [STEP] lines, append a single [CATEGORY] line and then one line per mask of the form '[MASK i] [type]', using '[MASK ANS] [type]' for the final answer; (7) do NOT output any text before the first [STEP] or after the [CATEGORY] block. Reason forward: start from entities in the question and derive toward the final answer.

User: Question: What is the capital of the country where Plainfield Town Hall is located?
Assistant: [STEP] Plainfield Town Hall is located in the country of [MASK 1].
[STEP] The capital of [MASK 1] is [MASK ANS].
[CATEGORY]
[MASK 1] [country]
[MASK ANS] [city]

User: Question: {A question}
Assistant:

J Additional Experiments

J.1 Paraphrase-Level Robustness

We evaluate whether the editing methods remain reliable across the three paraphrased questions associated with each MQuAKE-CF-3k-v2 case. "Case Any" counts a case as correct if at least one of its three paraphrases is answered correctly. "Question" computes accuracy independently for each paraphrased question, while "All-3 Acc." requires all three paraphrases from the same case to be answered correctly.

As shown in Table 14, StructKE achieves the best performance under all three evaluation protocols, reaching 59.67% Case Any accuracy, 41.17% Question accuracy, and 22.90% All-3 accuracy. Compared with DecKER, StructKE improves these metrics by 9.87, 10.01, and 8.70 percentage points, respectively. The lower All-3 accuracy compared with Case Any accuracy shows that consistently answering all paraphrases remains more difficult than answering at least one formulation, but StructKE retains the strongest performance under this stricter criterion.

Table 13: Backward Masked Reasoning-Chain Prompt

System:
You are given a question and a tentative final answer. Produce a masked backward chain of thought that links the answer back to the question. Requirements: (1) output exactly one reasoning step per line, and every reasoning line MUST start with [STEP]; (2) do NOT skip intermediate hops; (3) replace specific knowledge, entities, dates, or locations that are NOT in the text of the question with [MASK], numbered as [MASK 1], [MASK 2], ... in order of first appearance; (4) the tentative final answer MUST be represented as [MASK ANS]; (5) if the same entity appears more than once, reuse the same [MASK i]; (6) after all [STEP] lines, append a single [CATEGORY] line and then one line per mask of the form '[MASK i] [type]', using '[MASK ANS] [type]' for the final answer; (7) do NOT output any text before the first [STEP] or after the [CATEGORY] block; (8) if the tentative answer looks inconsistent with the question, still try to construct the most plausible chain that ends at [MASK ANS].

User: Question: What is the capital of the country where Plainfield Town Hall is located?
Tentative answer: Washington, D.C.
Assistant: [STEP] Plainfield Town Hall is located in the country of [MASK 1].
[STEP] The capital of [MASK 1] is [MASK ANS].
[CATEGORY]
[MASK 1] [country]
[MASK ANS] [city]

User: Question: {A question}
Tentative answer: {A tentative answer}
Assistant:

Table 14: Paraphrase-level robustness on MQuAKE-CF-3k-v2 with Qwen2.5-7B-Instruct. "Case Any" counts a case as correct if at least one of its three paraphrases is answered correctly. "Question" evaluates paraphrases independently, and "All-3 Acc." requires all three paraphrases to be correct.

Method	Case Any (%)	Question (%)	All-3 Acc. (%)
MeLLo	1.13	0.68	0.30
PokeMQA	4.97	2.37	0.47
DeepEdit	2.73	1.48	0.57
RAE	11.27	7.50	3.97
EditCoT	10.83	5.22	1.17
DecKER	49.80	31.16	14.20
StructKE	**59.67**	**41.17**	**22.90**

J.2 Feature and Selector Ablations

These ablations isolate the information used by SDPR and compare alternative candidate-selection rules. In the feature table, 'w/o group' removes one feature group from the full selector, while 'Group only' retains that group alone. The prompt-reranking variant reaches 62.33% case accuracy, while the base output variant reaches 60.33%, showing that candidate reranking provides a measurable but incomplete improvement over direct base-model output.

Table 15: Feature-group ablation for SDPR.

Group	w/o group	Group only
KI_i	58.33	59.57
RPP_i	59.90	58.60
H_i	59.67	58.73
G_i	59.67	58.23
D_i	59.67	57.83
w_i	59.67	58.23
F_i	59.67	58.17
r_i	59.67	58.33
a_i	59.60	58.53

K Case Study

Table 16 presents a representative MQuAKE-CF-3k-v2 test case in which MeLLo, EditCoT, PokeMQA, and RAE all fail, while StructKE returns the correct answer. We report the question, relevant edits, gold answer, gold reasoning path, and method outputs.

Table 16: Case Study of Multi-Hop Editing

Question: Who is the head of state of the country where Fernando Santos hold a citizenship?
Related Edits: Fernando Santos is a citizen of Portugal → United Kingdom; The name of the current head of state in United Kingdom is Elizabeth II → Emmerson Mnangagwa.
Answer: Emmerson Mnangagwa.
Gold Reasoning Path: Fernando Santos is a citizen of United Kingdom. The head of state of United Kingdom is Emmerson Mnangagwa.

Method	Result	Status
PokeMQA	Entity extraction identifies Fernando Santos, determines his citizenship, and asks for the head of state of that country. The subquestion chain answers Portugal and then Marcelo Rebelo de Sousa.	Wrong
RAE	The retrieved edge maps Fernando Santos to the United Kingdom, but the final check falls back to the British monarch and returns Elizabeth II or a current British monarch.	Wrong
EditCoT	The original CoT links Fernando Santos to Portugal. After injecting the edited citizenship fact, the chain briefly moves to the United Kingdom but then drifts back to Portugal and concludes with President Marcelo Rebelo de Sousa.	Wrong
MeLLo	The planner asks for Fernando Santos's citizenship and answers Portugal, then asks for the head of state of Portugal and answers Marcelo Rebelo de Sousa.	Wrong
StructKE	Masked reasoning path: Fernando Santos holds a citizenship of [MASK 1]. The head of state of [MASK 1] is [MASK ANS]. Filled reasoning path: Fernando Santos holds a citizenship of United Kingdom. The head of state of United Kingdom is Emmerson Mnangagwa.	Right