CaKE: Circuit-aware Editing Enables Generalizable Knowledge Learners

Anonymous ACL submission

Abstract

Knowledge Editing (KE) enables the modification of outdated or incorrect information in large language models (LLMs). While existing KE methods can only update isolated facts, they struggle to generalize these updates to multi-hop reasoning tasks that depend on the modified knowledge. Through an analysis of reasoning circuits—the neural pathways LLMs use for knowledge-based inference, we identify that current layer-localized KE approaches fail to effectively integrate updated information into these reasoning pathways. To address this limitation, we propose **CaKE** (**C**ircuit-**a**ware **K**nowledge **E**diting), a novel method that facilitates more effective integration of updated knowledge in LLMs. CaKE leverages meticulously curated data that enforces the model to utilize the modified knowledge, guiding the model to develop appropriate reasoning circuits for newly integrated knowledge. Experimental results show that CaKE enables more accurate and consistent use of updated knowledge across related reasoning tasks, leading to an average 20% improvement in multi-hop accuracy on MQuAKE compared to existing KE methods.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in diverse tasks (Yang et al., 2024a; Dubey et al., 2024; OpenAI, 2024; Guo et al., 2025), achieving performance that rivals or even exceeds human experts. However, their practical deployment faces some critical limitations: parametric knowledge remains static after pretraining, making it challenging to track evolving real-world information, and their propensity for hallucinations undermines reliability. Knowledge editing (KE) has emerged as a promising solution to update models directly (Mitchell et al., 2021; Wang et al., 2024c; Jiang et al., 2025). Although existing KE methods achieve good results on simple factual updates (Yao et al., 2023; Zhang et al., 2024b), they also exhibit fundamental limitations: edits propagate inconsistently through related knowledge structures and downstream reasoning tasks (Cohen et al., 2024; Qin et al., 2024; Yao et al., 2023); excessive focus on surface-level pattern matching (Hoelscher-Obermaier et al., 2023), and locality issues for other unrelated knowledge and general ability (Gu et al., 2024; Gupta et al., 2024).

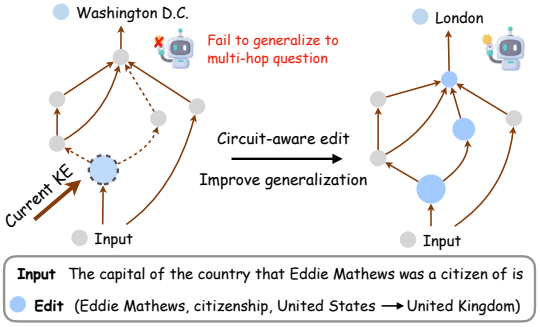
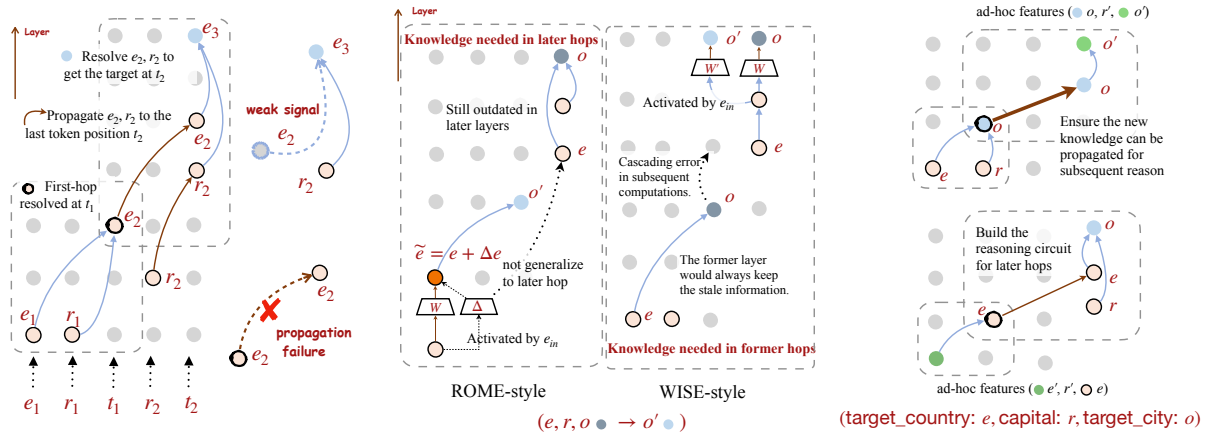


Figure 1: The current edit cannot propagate the new knowledge⇢multi-hop reasoning. We propose a circuit-aware edit to improve the model's multi-hop reasoning performance involving the updated knowledge.

Our work specifically addresses the poor performance of edited models in downstream reasoning tasks that involve the updated knowledge (Zhong et al., 2023; Zhang et al., 2024d). Consider a representative case in Figure 1: after editing '*Eddie Mathews, citizenship, United States → United Kingdom*', models correctly answer direct queries but fail multi-hop reasoning like '*The capital of the country that Eddie Mathews was a citizen of is?*' (still outputting '*Washington D.C.*'). Critically, this is not merely an editing artifact: vanilla LLMs often correctly answer single-hop questions while failing their multi-hop counterparts (Yang et al., 2024c; Biran et al., 2024), suggesting deeper architectural limitations in knowledge utilization.

We trace these limitations to a misalignment between KE strategies and the inherent reasoning architectures of LLMs. To understand



(*a*) Reason circuit and failure phenomena (*b*) Rethink KE in Circuit View (*c*) Circuit-aware Knowledge Editing

Figure 2: An overview of our work. (a) Analyze the multi-hop reasoning circuit in LLMs and identify the reasons behind their failure in reasoning: weak signals and propagation failure. (b) Rethink the limitations of the most popular editing paradigms from a circuit perspective. Both ROME-style (edits the early layers) and WISE-style (edits the later layers) fail to propagate their modifications through the reasoning circuit. (c) CaKE leverages ad-hoc features to curate reasoning tasks, guiding the LLM in constructing a reasoning circuit to process new knowledge.

this disconnect, we analyze how LLMs utilize knowledge in downstream reasoning tasks. Recent analysis suggests that knowledge is not merely statically stored but dynamically activated through specialized circuits (Yao et al., 2024; Biran et al., 2024). Our investigation (§2) reveals that multi-hop reasoning emerges from coordinated computing circuits: early layers gather entity and relation information, which is then routed to the last token position during the middle layers. Subsequently, later layers progressively complete reasoning steps using this information at the last token position (Figure 2 (a)). We then analyze the entity and relation information at the last token position in failed multi-hop reasoning cases. Our observations reveal that critical information either fails to be properly routed or exhibits a weak signal, preventing effective reasoning. This explains why current KE methods underperform (§3.1): they optimize for isolated parameter changes rather than circuit-level integration needed for compositional reasoning, as shown in Figure 2 (b).

To bridge this fundamental gap, we propose **C**ircuit-**a**ware **K**nowledge **E**diting (CaKE) in §3.2. By explicitly aligning edits with the LLM's native reasoning architecture, CaKE transforms static knowledge updates into generalizable knowledge learners—models that not only store static edited facts but also dynamically apply them in downstream reasoning tasks. Specifically, for the updated knowledge, we first design the circuit-aware tasks with ad-hoc features that require the LLM to utilize the new knowledge for latent reasoning (in Figure 2 (c)). We then guide the LLM in constructing the reasoning circuit for the updated knowledge by training with the curated data. Extensive experiments (§4) demonstrate the effectiveness of CaKE, outperforming existing knowledge editing methods on the multi-hop editing benchmarks MQuAKE on LLMs of different sizes, including LLAMA3-8B-Instruct (Dubey et al., 2024), Qwen2.5-7B-Instruct (Yang et al., 2024a), and LLAMA3-70B-Instruct.

2 Analyzing Reasoning Circuit in LLM

In this part, we integrate previous findings on multi-hop reasoning and circuit analysis (Yao et al., 2024; Biran et al., 2024; Yang et al., 2024b; Wang et al., 2024f) and factual recall (Merullo et al., 2024) to illustrate how language models leverage knowledge to tackle the multi-hop reasoning task. Based on this, we can view the cause of the failure behind them. We employ the WikiData subset proposed by Biran et al. (2024) and name it *HoppingTooLate*, which contains 82,021 two-hop queries. We denote each fact as a triplet (e, r, e'), where e is the head entity, r is the relation, and e' is the tail entity. We view two-hop queries as (e_1, r_1, e_2) and (e_2, r_2, e_3), where e_1 is the source entity, e_2 is the bridge entity, and e_3 is the target entity. We focus on the latent reasoning framework to evaluate whether a model can output the expected answer e_2 directly given the composite query $(e_1, r_1, r_2, ?)$. In addition, we follow *HoppingTooLate* and define

Model	Entity Patch	Relation Patch
LLaMA3-8B-Ins.	85.35	56.20
Qwen2.5-7B-Ins.	97.29	55.40

Table 1: Activation Patching Success Rates (%).

t_1 as the last token of the first-hop prompt and t_2 as the last token of the whole two-hop prompt.

2.1 Multi-hop Reasoning Circuit

Building on the insights from prior work (Biran et al., 2024; Yao et al., 2024), we can define a structured circuit mechanism for multi-hop reasoning in transformer-based LLMs, as illustrated in Figure 2(a). Our analysis reveals three distinct computational phases: 1) The model processes the initial relation r_1 and entity e_1, encoding the bridge entity e_2 in the final token position of the first prompt segment (t_1). 2) Critical features, including e_2 and the second relation, r_2 are transferred to the last token position t_2, preparing for final resolution. 3) The model computes the target e_3 by resolving r_2 and e_2, giving the result in the final token position. Hence, based on the linearity theory (Hernandez et al., 2024), multi-hop reasoning in LLM can be formalized as:

$$F_n(F_{n-1}(e_{n-1}, r_{n-1}), r_n) \quad (1)$$

Each function F_{n-1} produces a bridge entity e_n for subsequent computation, demonstrating how intermediate results propagate vertically through network layers.

Evaluation To validate this circuit hypothesis, we conduct a causal analysis to determine whether modifying the variables in the function F leads to corresponding changes in the output. Our intervention strategy focuses on the critical last token position for the second hop, where intermediate variables are stored. Specifically, we consider: 1).Entity Patching: Replacing the representation of e_2 with an alternative entity e_{patch}. For example, given the prompt *'The official currency of the country where the Eiffel Tower is located is'*, we substitute the representation of the bridge entity *'France'* with *'China'*, expecting the output to change to *'Renminbi'*. 2).Relation Patching: Replacing the representation of r_2 with an alternative relation r_{patch}. For instance, given the same prompt, we substitute the representation of *'official currency'* with *'capital'*, expecting the output to change to *'Paris'*. A successful patch (producing $F_2(e_{patch}, r_2)$ or $F_2(e_2, r_{patch})$) would confirm

Model	Metric	Correct		Inconsistent		Incorrect	
		Cases	Layer	Cases	Layer	Cases	Layer
LLAMA3	e_2 from t_1	63.1%	6.3	75.2%	6.0	48.7%	8.2
	e_2 from t_2	67.8%	13.2	59.8%	9.8	17.7%	21.1
	r_2 from t_2	66.9%	14.0	49.0%	13.8	28.1%	13.7
	e_3 from t_2	56.5%	18.8	22.7%	20.7	18.3%	18.0
Qwen2.5	e_2 from t_1	71.2%	4.3	74.1%	4.7	46.7%	5.1
	e_2 from t_2	52.9%	7.9	63.7%	9.5	18.9%	13.5
	r_2 from t_2	75.8%	8.1	75.2 %	10.4	44.8%	9.7
	e_3 from t_2	71.2%	16.4	39.4%	17.4	25.2%	11.4

Table 2: The results of LLAMA3-8B-Instruct (32 layers) and Qwen2.5-7B-Instruct (28 layers). Cases are the percentage of data we can detect the information, and Layer is the mean of the first layer in which we detect the information.

the model's reliance on these specific representations for reasoning. We conduct experiments on LLAMA3-8B-Instruct and Qwen2.5-7B-Instruct and employ PatchScope (Ghandeharioun et al., 2024) for targeted activation patching (detailed in §B.1 and Figure 7). Table 1 shows that in both the LLAMA3-8B and Qwen2.5-7B models, substituting variable representations at the last token position leads to corresponding behavioral changes, particularly in entity patching, where accuracy exceeds 80%. These results provide mechanistic evidence for the reasoning circuit we identified before.

2.2 Circuit in Failure Phenomena

Then, we aim to understand why language models sometimes fail at multi-hop reasoning despite successfully answering individual single-hop questions. For instance, a model may correctly answer *'the capital of Russia'* with *'Moscow'* and *'the country of citizenship of Fyodor Dostoyevsky'* with *'Russia'*, yet fail to answer the multi-hop question *'the capital of the country of citizenship of Fyodor Dostoyevsky is'* correctly. To systematically analyze this issue, we focus on the second hop of reasoning, as the model typically performs well on the first hop. We categorize the data from the *HoppingTooLate* dataset[1] into three subsets based on the model's behavior: **Correct**: The model answers both single-hop questions (e_1, r_1, e_2) and (e_2, r_2, e_3) correctly, as well as the multi-hop question $(e_1, r_1, r_2, ?)$. **Inconsistent**: The model answers both single-hop questions correctly but fails on the multi-hop question. However, some cases in the *Correct* set contain the bridge entity e_2 but have a different subject e_1' and relation r_1', for which the model correctly answers $(e_1', r_1', r_2, ?)$. This suggests that while the model can leverage knowledge

[1]We filter out short-cut cases as done by Biran et al. (2024).

in some contexts, it fails to generalize, indicating reasoning gaps rather than missing knowledge. **Incorrect**: The model answers both single-hop questions correctly but fails on the multi-hop question in all contexts (e_1', r_1', e_2). This implies a complete failure to employ the knowledge for multi-hop reasoning. To investigate these failure modes, we check whether the models construct the reasoning circuit by monitoring key variables (e_1, e_2, and r_2) at critical positions (t_1 and t_2) across the model's layers. Our analysis reveals several interesting patterns, extending beyond the 'hopping too late' problem identified by Biran et al. (2024).

We list the results in Table 2 (more details in Figure 10 in Appendix). For the *correct* subset, we observe strong evidence of the reasoning circuit functioning as expected: a large portion of e_2 is detected at both t_1 (e_2 from t_1) and t_2 (e_2 from t_2) in both LLAMA3 and Qwen2.5 models. The model correctly uses the r_2 and e_2 information at t_2 to produce the final answer e_3. Contrastly, in the *Incosistent* subsets, we can find that despite detecting e_2 and r_2 at t_2, the model often fails to produce the correct e_3 answer (e_3 from t_2: only 22.7% in LLAMA3 and 39.4% in Qwen2.5 of cases we can detect at t_2). We hypothesize that the e_2 information, though present, may be insufficient to trigger the second-hop reasoning circuit, leading to the failure to execute the function $F(e_2, r_2)$ effectively. What's more, in the *Incorrect* subsets, we can find the needed e_2 information is rarely detected at the t_2 position (e_2 from t_2: Only 17.7% in LLAMA3 and 18.9% in Qwen2.5). Even when e_2 is detected, it typically emerges in much later layers (layer 21 in LLAMA3 and layer 13.5 in Qwen2.5), making it too late to be effectively utilized for the second-hop computation, aligned with Biran et al. (2024)'s findings. We conjecture the model fails to propagate e_2 to the t_2 position, resulting in the variable e_2 missing for conducting the $F(e_2, r_2)$ function.

Evaluation To test our hypothesis, we conduct interventions to enhance the information at the specific positions to see if we can improve the model's performance in these failure cases. We test three ways: back-patching the t_1 and $t2$ position as Biran et al. (2024) did, which would enhance the information at the position, and cross-position patching the information from t_1 to the t_2 position, which explicitly propagates the information from t_1 to t_2 (details in Figure 8 in Appendix). From the results in Figure 3, we can find a high success rate for

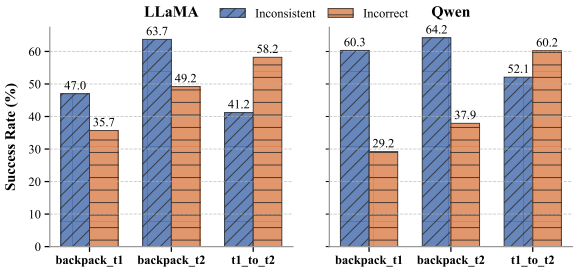


Figure 3: Results of the intervention on the failure cases in multi-hop reasoning of LLAMA3 and Qwen2.5.

all the inconsistent and incorrect cases, but they demonstrate different paradigms. For the inconsistent cases, back-patching would lead to better performance, while for the incorrect cases, patching knowledge from the t_1 to t_2 usually shows better outcomes. This proves our previous hypothesis that for the incorrect cases, due to the **propagation failure**, the model fails to move the e_2 to t_2 position, and manual routing via cross-patching can mitigate the issue. Meanwhile, for inconsistent cases, amplification via back-patching compensates the **weak signal** when valid e_2 representations reach t_2 but lack sufficient magnitude for subsequent reasoning.

3 Circuits-aware Knowledge Editing

3.1 Rethinking KE from the Circuit View

Despite the success of current KE on single facts benchmarks, from previous studies (Zhang et al., 2024d; Zhong et al., 2023), and our analysis in §4, we can see that the edited model's performance on multi-hop reasoning tasks is often unsatisfactory. Building on our previously identified circuit for multi-hop reasoning, we rethink the reason why current knowledge editing methods fail under multi-hop reasoning circumstances.

Unified Editing Details When updating a piece of knowledge $(e, r, o \rightarrow o')$, the most popular knowledge editing techniques would modify the parameters that are responsible for the knowledge. There are two kinds of paradigms: editing the Feed-Forward Networks (FFN) in the early layers, such as ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) or modifying the later layers' FFN output, like WISE (Wang et al., 2024c) and T-Patcher (Huang et al., 2023). In fact, some studies have queried the effectiveness of these localization settings (Chang et al., 2024; Hase et al., 2024) as the localization area is not correlated to the performance of the knowledge editing methods. Here, we

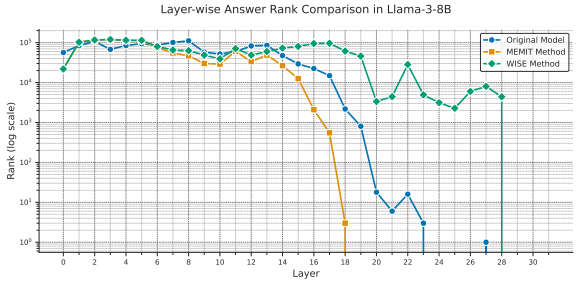


Figure 4: The target answer token's rank in the vocabulary of different editing methods when editing the fact '*The official language of Japan is Japanese → Korean.*'

propose a unified view of the mechanisms and limitations from the circuit perspective. From Figure 2 (c), these two editing paradigms can be achieved by a gated function $\mathcal{G}(\mathbf{x})$:

$$\text{FFN}_{\text{out}}(\mathbf{x}) = \underbrace{\boldsymbol{W}\mathbf{x}}_{\text{Original term}} + \mathcal{G}(\mathbf{x}) \cdot \underbrace{\boldsymbol{\delta}(\mathbf{x})}_{\text{Edit term}} \quad (2)$$

$$\mathcal{G}(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \boldsymbol{e}_{in} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

ROME-style would modify the weight $\boldsymbol{W}$ with a perturbation $\boldsymbol{\Delta}$ and obtain a new weight $\boldsymbol{W}' = \boldsymbol{W} + \boldsymbol{\Delta}$. Here, $\boldsymbol{\delta}(\mathbf{x}) = \boldsymbol{\Delta}\mathbf{x}$. When calculating the Δ, ROME-style methods, apply the *least squares estimation* and *null space constraint* to make sure the Δ is only activated by the corresponding entity representation $\boldsymbol{e}_{in}$ and keep the original output for other representations. In parallel, WISE-style editing methods would directly introduce the new weight $\boldsymbol{W}'$ that would be activated by the related representation $\boldsymbol{e}_{in}$, and $\boldsymbol{W}'$ would encode the updated knowledge. Here, $\boldsymbol{\delta}(\mathbf{x}) = (\boldsymbol{W}' - \boldsymbol{W})\mathbf{x}$.

Defect from circuit view To see the editing mechanism better, we first compare the rank of the target answer at the last token position via MEMIT, WISE, and the original model in Figure 4. We edit the fact: '*The official language of Japan is Japanese → Korean.*' From the figure, we can see that in the original model and MEMIT-edited model, the answer token is dealt with gradually through the mid-to-later layers, and MEMIT would make this happen in advance. The computation of the ROME-style method for the new knowledge recall is: $F(\tilde{e}, r) = o', \tilde{e} = e + \boldsymbol{\Delta}e$, which modifies the knowledge stored in the previous layer and gives us the new representation e' for further computing. On the contrary, the WISE method would directly alter the information at the edited layer as we can see the sharp drop at layer 29. In particular, the editing would take effect when the added or updated parameters are activated by query representation and work as $F(e, r) = o'$.

In single-hop knowledge editing, these kinds of methods would give us the correct information, but for the multi-hop cases, this would fail. As shown in Figure 2 (b), both these layer-specific editing methods **cannot propagate the updated knowledge to the reasoning circuit**, leading to unsatisfactory multi-hop reasoning performances. An essential requirement for these methods is that the gated function $\mathcal{G}(\mathbf{x})$ is activated by the specific representation $\boldsymbol{e}_{in}$. However, under the multi-hop reasoning scenario, the model would deal with different single-hop questions in different layers, like the two-hop reasoning circuit in §2: for ROME-style editing, if the new fact $(e, r, o \rightarrow o')$ is the second-hop question and the entity e appears after the edited layers, the gated function would $\mathcal{G}(\mathbf{x})$ not be activated and the model would still follow the previous stale knowledge $F(e, r)$ instead of $F(\tilde{e}, r)$ and give us the wrong answer. Likewise, the WISE-style editing would retain reliance on the original knowledge when the new fact is finished in the former layers as the first hop, bypassing the edit function in later layers and cascading the error in the subsequent reasoning. In conclusion, these layer-specific editing methods cannot learn the new knowledge generally to make the knowledge usable in downstream reasoning tasks.

3.2 Proposed Method: CaKE

Inspired by previous analysis, we propose a novel method, **C**ircuit-**a**ware **K**nowledge **E**diting (**CaKE**), which enhances the model's ability to update and effectively utilize knowledge. CaKE comprises two key components: (1) generating circuit-aware training data that explicitly requires reasoning with the updated knowledge, and (2) training the model to construct robust reasoning circuits that integrate the new knowledge.

Data Generation To ensure that the model builds effective reasoning circuits, we address two critical challenges: preventing *failure propagation* and mitigating *weak signals* (as identified in §2), while ensuring that updated knowledge is properly integrated across different layers (as described in §3.1). For each updated knowledge item, we construct the following contexts to mitigate these issues: (1) **Original Narrative**: We begin by gen-

Method	Model	MQUAKE-CF		MQUAKE-CF-v2		MQUAKE-T	
		H-Acc.↑	MAcc.↑	H-Acc.↑	MAcc.↑	H-Acc.↑	MAcc.↑
Pre-edited		79.0	27.0	78.4	28.6	71.0	5.3
LoRA	LLaMA3-8B-Ins	66.0	<u>27.6</u>	64.7	24.6	**92.3**	66.0
WISE	LLaMA3-8B-Ins	38.2	24.0	37.2	21.0	63.5	<u>62.9</u>
MeLLo	LLaMA3-8B-Ins	16.5	16.1	19.5	16.0	42.3	50.1
ROME	LLaMA3-8B-Ins	86.8	17.6	<u>86.4</u>	15.5	89.5	8.4
MEMIT	LLaMA3-8B-Ins	76.3	11.5	74.0	10.0	86.0	3.7
AlphaEdit	LLaMA3-8B-Ins	66.1	10.1	63.7	8.5	73.4	1.0
IFMET ♣	LLaMA3-8B-Ins	81.9	23.2	75.3	<u>36.5</u>	82.1	46.1
CaKE(ours)	LLaMA3-8B-Ins	**90.6**	**57.3**	**90.1**	**57.1**	<u>91.5</u>	**81.4**
Pre-edited		75.6	34.7	76.8	37.7	60.1	15.6
LoRA	L-70B	<u>93.1</u>	<u>53.2</u>	<u>90.5</u>	<u>50.2</u>	<u>90.1</u>	<u>90.6</u>
MeLLo	L-70B	8.0	6.4	8.6	9.9	11.6	32.9
CaKE(ours)	L-70B	**93.5**	**65.4**	**93.3**	**63.3**	**91.1**	**94.6**

Table 3: Comparison of CaKE with existing methods on MQuAKE for LLAMA3-8B-Instruct and LLAMA3-70B-Instruct. The best results are highlighted in bold, while the second-best results are underlined. ♣ means the results are based on our re-implementation since the original code is not open by the authors, and we will update it after the source code is open. Due to the computational limitations, we just run the LoRA and MeLLo in 70B model.**Results for Qwen2.5-7B-Ins can be found in Table 6.**

erating straightforward factual statements that explicitly convey the updated information. For example, when updating the fact k: (PersonX, citizen_country, Switzerland → Japan), we use the narrative representation: *'PersonX is a citizen of Japan'* and generate several paraphrases. These statements serve as the foundation for the model to learn the updated knowledge. (2) **Circuit-aware Tasks**: Next, we design specialized reasoning scenarios that address the identified circuit-level challenges, as illustrated in Figure 2(c). Moreover, to avoid introducing extraneous knowledge that could leak into downstream evaluations—and to test the generalization of our method (inspired by prior research (Zhang et al., 2024c))—we incorporate ad-hoc features into these scenarios. These tasks link the fact with intermediate attributes or reasoning steps and fall into two categories: **Late-layer Knowledge Integration:** These tasks ensure that the updated knowledge is effectively learned in the later layers, alleviating issues such as *weak signals and the limitations of ROME-style editing*. For the fact k, we construct prompts like: *'Suppose {random_entity_1} wears red clothes, {random_entity_2} wears blue clothes, and {PersonX} wears green clothes. The country of citizenship of the person in green is:'* Here, the model is expected to output 'Japan,' requiring it to employ the new fact k in later layers. **Reasoning Circuit Enhancement:** These tasks require the model to use the updated knowledge for subsequent reasoning, thereby mitigating *propagation failure*, *weak signal,* and *WISE-style's limitations*. Following the same fact k: *'In a book about countries, Japan is mentioned on page 6 of the book, while China is mentioned on page 72. On which page of the book is the country of citizenship of the {PersonX} shown?'* Here, the model must first recall the updated citizenship (Japan) and then use this information to determine the correct page number (6). Furthermore, for each knowledge type, we develop specific task templates and leverage GLM-4-plus (GLM et al., 2024) to generate data using randomly selected related entities (detailed in Appendix A).

Edit Training After obtaining the curated circuit-aware data, we fine-tune the LLM using LoRA across all layers, enabling the model to optimize its internal knowledge organization. We minimize the cross-entropy loss $\mathcal{L}$ between the model's outputs and the ground-truth tokens expressing the updated fact:

$$\mathcal{L} = \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\left[-\sum_{t=1}^{|\mathbf{y}|}\log p(y_t \mid \mathbf{x}, \theta_{\text{LoRA}})\right] \quad (4)$$

where θ_{LoRA} represents the LoRA parameters, $\mathbf{x}$ is the input prompt, and $\mathbf{y}$ is the desired updated output sequence.

	CSQA	BBH	MMLU	GSM8k
LLaMA3-8B-Ins	76.09	67.89	63.83	75.20
MEMIT	76.08	67.88	63.82	75.21
ROME	72.98	61.37	62.95	74.59
CAKE	75.10	67.20	62.98	76.04
Qwen2.5-7B-Ins	82.31	33.39	71.80	82.26
MEMIT	82.39	37.37	71.80	81.96
ROME	72.57	34.22	63.38	72.21
CAKE	82.64	37.44	71.76	82.79

Table 4: **Locality Performance** on several general benchmarks of CaKE and other editing methods.

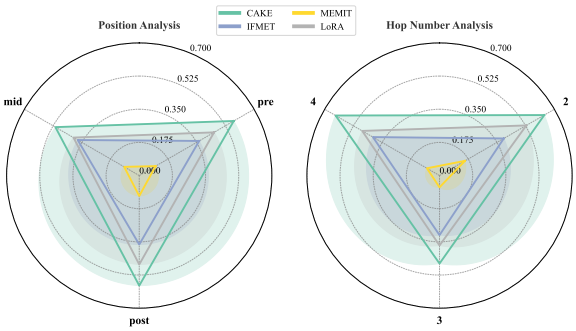


Figure 5: Accuracies of different number hops and edit-positions in MQuAKE-CF-3k-v2 on LLAMA3-8B-Instruct.

4 Experiments

4.1 Experiment Settings

We mainly utilize the multi-hop reasoning knowledge editing dataset MQuAKE (Zhong et al., 2023), which considers different number of hops (from 2 to 4) and different positions of the knowledge used in the multi-hop questions. We utilize three versions of the datasets: MQuAKE-CF-3k and MQuAKE-CF-3k-v2 are two subsets that contain different question types and editing hopping numbers, and MQuAKE-T is a time-aware knowledge editing benchmark.

Baselines and Models We consider several knowledge editing baselines, including: IFMET (Zhang et al., 2024d), AlphaEdit (Fang et al., 2024), ROME (Meng et al., 2022), MEMIT (Meng et al., 2023),WISE (Wang et al., 2024c) and MeLLo (Zhong et al., 2023). Here, AlphaEdit, ROME, and MEMIT are methods that edit the model's parameters at early layers; WISE adds additional parameters at later layers, and IFMET edits both the early and later layers's FFN to achieve better multi-hop reasoning performance. MeLLo is a prompt-based retrieval-augmented method. We conduct experiments on LLAMA-3-8B-Instruct, Qwen-2.5-7B-Instruct, and LLAMA-3-70B-Instruct.

Evalutation Metric Following Zhong et al. (2023), we evaluate model performance using Multi-hop Accuracy (MAcc) and Hop-wise Answering Accuracy (H-Acc). MAcc measures the accuracy of multi-hop question answering, while H-Acc assesses correctness at each reasoning step. Higher values indicate better performance. For KE, we also need to consider locality, which ensures edits do not affect unrelated knowledge and abilities. To assess this, we evaluate the model on general benchmarks, including CommonsenseQA (Talmor et al., 2019), BigBenchHard (Suzgun et al., 2023), MMLU (Hendrycks et al., 2021), and GSM8k (Cobbe et al., 2021).

4.2 Experiments Results

Main Results Table 3 summarizes our results. Although current KE methods achieve high hop-wise accuracy (H-Acc.), their performance on the three versions of MQuAKE is quite low (with an average accuracy of less than 20%). For example, MEMIT and ROME achieve over 80% accuracy on single-hop questions in MQuAKE-v2; however, their accuracy on multi-hop reasoning drops to only around 10%, indicating that the LLM fails to effectively utilize the updated knowledge during reasoning. In contrast, CaKE demonstrates significant improvements in multi-hop reasoning. On the LLAMA3-8B-Instruct model, CaKE achieves accuracies of 57.3, 57.2, and 81.5 on MQuAKE-CF, MQuAKE-CF-v2, and MQuAKE-T, respectively—outperforming all compared methods. Additionally, IFMET, which also considers different layers for multi-hop reasoning but neglects the information flow within the circuit, performs not as well as CaKE . Moreover, when compared with RAG-based methods such as MeLLo, CaKE also yields better results. Furthermore, compared to the baseline LoRA tuning methods that simply incorporate the raw knowledge, the improvements observed with CaKE underscore the effectiveness of our approach. Interestingly, while LoRA demonstrates strong performance on the 70B model, reflecting enhanced learning capabilities in larger models, CaKE still achieves an additional improvement of approximately 10%.

Locality Performance In this section, we evaluate the model's performance on general ability benchmarks to ensure that acquiring new knowl-

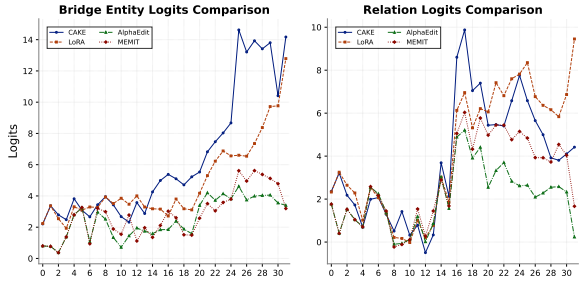


Figure 6: e_2 and r_2's logits at t_2 in models after different knowledge editing methods.

edge does not compromise its overall capabilities. As shown in Table 4, CaKE achieves performance comparable to the original model on both the LLAMA3-8B and Qwen2.5-7B models.

5 Analysis

5.1 Position and Number of Hop

We then examine the effects of the number of edits and the position of the updated knowledge in multi-hop scenarios, with results shown in Figure 5. Notably, even when the model is trained solely on two-hop questions, CaKE yields improvements across varying numbers of editing hops. The benefits are particularly pronounced for four-hop questions, where methods like IFMET (designed only for two-hop scenarios) struggle. Besides, CaKE enhances performance regardless of the position of the edited knowledge within the multi-hop questions, demonstrating the generalizability of CaKE .

5.2 Case Analysis

In this part, we show the cases in which the **CaKE helps the model learn the multi-hop reasoning circuit and other methods fail**. For illustration, we consider the two-hop question: 'The capital city of the country that Eddie Mathews was a citizen of is'. Here, the editing case is *(Eddie Mathews, citizenship, United States → United Kingdom)* and the updated model is expected to output 'London'. However, CaKE gives the correct answer, while other methods fail: MEMIT gives us the 'Moscow', AlphaEdit gives us 'Birmingham', and LoRA gives us 'not known'. To further understand these differences, we analyze the computing circuit of each method to determine whether the updated model successfully propagates the bridge entity e_2 and relation r_2 to the last token t_2 position. Figure 6 displays the logits of e_2 and r_2 at t_2 for models edited by different methods. As shown, the bridge entity e_2 in CaKE exhibits significantly stronger logits compared to those of AlphaEdit and MEMIT, indicating that CaKE effectively constructs the reasoning circuit and propagates the necessary information to the target position. Similarly, the r_2 information is more prominent in CaKE , further demonstrating its superiority in circuit construction and information flow.

6 Related Work

How the knowledge in LLM is acquired and stored has been a keep-going research topic recently (Wang et al., 2024b). Current research (Zhou et al., 2023) demonstrates most of the knowledge is learned during the *pretraining stage*. After pretraining, LLMs are anticipated to refresh their internal knowledge to keep pace with the evolving world, and knowledge editing (Zhang et al., 2024b; Jiang et al., 2024a; Sun et al., 2024; Hsueh et al., 2024; Powell et al., 2024; Wang et al., 2024a; Rozner et al., 2024; Zhang et al., 2024a; Wang et al., 2024g; Shi et al., 2024; Wang et al., 2025) is a promising way to do this. Current knowledge editing methods contain several ways: editing the former layers' MLP (Meng et al., 2022, 2023; Fang et al., 2024), enhancing later layers' MLP (Wang et al., 2024e; Yao et al., 2022; Hartvigsen et al., 2023) and retrieving the fact as prompt (Jiang et al., 2024b; Zhong et al., 2023). These works are mainly based on the previous knowledge mechanism of the "black box" of neural models through (Ferrando et al., 2024). However, these knowledge editing methods always focus on the simple facts and often fail on the downstream tasks, like the multi-hop reasoning scenario. Our work focuses on the mechanism of the reasoning in LLM and improves the generalization of the editing knowledge.

7 Conclusion

We present CaKE, a framework designed to align knowledge editing with the inherent reasoning architectures of LLMs. By examining the multi-hop reasoning circuits within LLMs, we identify that existing knowledge editing methods fall short due to their isolated parameter adjustments, which fail to adequately propagate updated knowledge through the model's reasoning circuit. CaKE addresses this gap by incorporating circuit-aware tasks that compel the model to dynamically integrate and utilize new knowledge during reasoning. Experimental results demonstrate that CaKE achieves generalizable multi-hop knowledge editing.

Limitation

Dataset Our work primarily focuses on the factual knowledge embedded in large language models (LLMs) and their capacity for multi-hop reasoning over these facts. We recognize that LLM reasoning also encompasses other domains—such as long-form mathematics and reverse-curse reasoning—that merit further investigation.

Reasoning Pattern As discussed in the previous analysis, we concentrate on direct reasoning phenomena. Current LLMs have shown impressive capabilities in slow-thinking paradigms, including chain-of-thought and reflective reasoning. Beyond direct reasoning, enhancing the utilization of knowledge within these paradigms represents an important avenue for future research.

Fine-grained Circuit Components Our analysis revealed relational information within the circuits; however, CaKE currently does not delve deeply into these relationships. We believe that a more focused investigation into these components is necessary. Additionally, while our study emphasizes general circuit behavior, developing a more concise and effective method for knowledge editing remains an exciting challenge for future work.

Data Attribution Although we demonstrate the ability to construct reasoning circuits using curated data, the connection between a model's acquired abilities in its parameters and its training data is still underexplored. A deeper understanding of this relationship could lead to more efficient training processes and the generation of higher-quality synthetic data.

References

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. 2024. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14113–14130, Miami, Florida, USA. Association for Computational Linguistics.

Ting-Yun Chang, Jesse Thomason, and Robin Jia. 2024. Do localization methods actually localize memorized data in llms? a tale of two benchmarks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3190–3211.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 11:283–298.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R Costa-jussà. 2024. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024.

Model editing harms general abilities of large language models: Regularization to the rescue. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16801–16819.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. 2024. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15202–15232, Bangkok, Thailand. Association for Computational Linguistics.

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

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2024. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2024. Linearity of relation decoding in transformer language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. 2023. Detecting edit failures in large language models: An improved specificity benchmark. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11548–11559.

Yifan Hou, Jiaoda Li, Yu Fei, Alessandro Stolfo, Wangchunshu Zhou, Guangtao Zeng, Antoine Bosselut, and Mrinmaya Sachan. 2023. Towards a mechanistic interpretation of multi-step reasoning capabilities of language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4919.

Cheng-Hsun Hsueh, Paul Kuo-Ming Huang, Tzu-Han Lin, Che-Wei Liao, Hung-Chieh Fang, Chao-Wei Huang, and Yun-Nung Chen. 2024. Editing the mind of giants: An in-depth exploration of pitfalls of knowledge editing in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 9417–9429. Association for Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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

Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, Qun Liu, and Wei Wang. 2024a. Learning to edit: Aligning llms with knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 4689–4705. Association for Computational Linguistics.

Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, Qun Liu, and Wei Wang. 2024b. Learning to edit: Aligning LLMs with knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4689–4705, Bangkok, Thailand. Association for Computational Linguistics.

Tianjie Ju, Yijin Chen, Xinwei Yuan, Zhuosheng Zhang, Wei Du, Yubin Zheng, and Gongshen Liu. 2024. 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)*, pages 8987–9001, Bangkok, Thailand. Association for Computational Linguistics.

Ming Li, Yanhong Li, and Tianyi Zhou. 2024. What happened in llms layers when trained for fast vs. slow thinking: A gradient perspective. *arXiv preprint arXiv:2410.23743*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Language models implement simple word2vec-style vector arithmetic. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5030–5047.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

OpenAI. 2024. Introducing OpenAI O1 preview.

Derek Powell, Walter Gerych, and Thomas Hartvigsen. 2024. TAXI: evaluating categorical knowledge editing for language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 15343–15352. Association for Computational Linguistics.

Jiaxin Qin, Zixuan Zhang, Chi Han, Pengfei Yu, Manling Li, and Heng Ji. 2024. Why does new knowledge create messy ripple effects in llms? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12602–12609.

Amit Rozner, Barak Battash, Lior Wolf, and Ofir Lindenbaum. 2024. Knowledge editing in language models via adapted direct preference optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 4761–4774. Association for Computational Linguistics.

Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 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, CIKM 2024, Boise, ID, USA, October 21-25, 2024*, pages 2056–2066. ACM.

Zengkui Sun, Yijin Liu, Jiaan Wang, Fandong Meng, Jinan Xu, Yufeng Chen, and Jie Zhou. 2024. Outdated issue aware decoding for factual knowledge editing. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 9282–9293. Association for Computational Linguistics.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Haoyu Wang, Tianci Liu, Ruirui Li, Monica Xiao Cheng, Tuo Zhao, and Jing Gao. 2024a. Roselora: Row and column-wise sparse low-rank adaptation of pre-trained language model for knowledge editing and fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 996–1008. Association for Computational Linguistics.

Mengru Wang, Yunzhi Yao, Ziwen Xu, Shuofei Qiao, Shumin Deng, Peng Wang, Xiang Chen, Jia-Chen Gu, Yong Jiang, Pengjun Xie, et al. 2024b. Knowledge mechanisms in large language models: A survey and perspective. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7097–7135.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024c. WISE: Rethinking the knowledge memory for lifelong model editing of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. 2024d. EasyEdit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93, Bangkok, Thailand. Association for Computational Linguistics.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2025. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3):59:1–59:37.

Xiaohan Wang, Shengyu Mao, Shumin Deng, Yunzhi Yao, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. 2024e. Editing conceptual knowledge for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 706–724, Miami, Florida, USA. Association for Computational Linguistics.

Yifei Wang, Yuheng Chen, Wanting Wen, Yu Sheng, Linjing Li, and Daniel Dajun Zeng. 2024f. Unveiling factual recall behaviors of large language models through knowledge neurons. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7388–7402, Miami, Florida, USA. Association for Computational Linguistics.

Yiwei Wang, Muhao Chen, Nanyun Peng, and Kai-Wei Chang. 2024g. Deepedit: Knowledge editing as decoding with constraints. *CoRR*, abs/2401.10471.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. 2024b. Do large language models latently perform multi-hop reasoning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10210–10229, Bangkok, Thailand. Association for Computational Linguistics.

Sohee Yang, Nora Kassner, Elena Gribovskaya, Sebastian Riedel, and Mor Geva. 2024c. Do large language models perform latent multi-hop reasoning without exploiting shortcuts? *Preprint*, arXiv:2411.16679.

Yunzhi Yao, Shaohan Huang, Li Dong, Furu Wei, Huajun Chen, and Ningyu Zhang. 2022. Kformer: Knowledge injection in transformer feed-forward layers. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 131–143. Springer.

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

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2024. Knowledge circuits in pretrained transformers. *Advances in Neural Information Processing Systems*.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024a. Knowledge graph enhanced large language model editing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 22647–22662. Association for Computational Linguistics.

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

Xiao Zhang, Miao Li, and Ji Wu. 2024c. Co-occurrence is not factual association in language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

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

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: less is more for alignment. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Appendix

A Setting Detail

Dataset We list the details of the dataset in Table 5.

Model	Correct	Inconsistent	Incorrect
LLaMA3-8B-Ins.	1,005	1,032	1,240
Qwen2.5-7B-Ins.	241	252	275

Table 5: The dataset we used in the analysis/.

Environment Setting We run our experiments on 2 NVIDIA-A800 GPUs. For data generation, we utilize glm-4-plus and glm-4-air and a total of 10,000,000 tokens (about 20 dollars) to generate all synthetic data for the whole dataset. We use LLM-Eval (Gao et al., 2024) to test the model's general performance.

Data Generation We first construct the question template $\mathcal{T}$ for each relation type, and we list some of them in Table 7. We then generate the data using the following prompt:

Prompt for Constructing the circuit-aware data

Here are some question templates for the specific relation. As you can see, the question use the knowledge in the input to conduct reasoning in different hops for multi-hop reasoning. Please generate 3 different questions based on the template. Please return a python json file. $\{\mathcal{T}\}$ Here is the input question:

B Implementation Detail

B.1 Analyzing Method

Patch Scope The process is carried out as follows. First, a source prompt, a source token, and a source layer are provided. The prompt is processed through the model's forward computation, and the hidden representation v of the source token at the specified layer is extracted and stored. This representation v is the focus of our investigation, as we seek to determine whether it encodes a specific entity. Next, we employ the same prompt used by Ghandeharioun et al. (2024): "Syria: Syria is a country in the Middle East. Leonardo DiCaprio: Leonardo DiCaprio is an American actor. Samsung: Samsung is a South Korean multinational corporation. x" This prompt is passed through the model, but the hidden representation of 'x' is replaced with v at a chosen target layer. The forward computation then proceeds, and the resulting generated text is analyzed to evaluate the effects of this substitution. We conduct different patch analyses and show them in Figure 7 and Figure 8. When we conduct back-patch and cross-patch, the source prompt and target prompt are the same.

B.2 Editing Method

We utilize EasyEdit (Wang et al., 2024d) to conduct our editing experiments. For ROME, MEMIT, WISE, AlphaEdit, and MeLLo, we directly employ the original parameters provided by their respective papers. Below, we introduce these methods in detail and describe our implementation.

ROME and MEMIT ROME leverages causal analysis to identify knowledge within specific MLP layers and modifies the corresponding weight matrix using least squares approximation. It operates under the strong assumption that the MLP layers primarily store knowledge and injects new information into these layers iteratively using a Lagrangian remainder. In our experiments, we edit the 5th layer of both LLAMA3-8B-Instruct and Qwen2.5-7B-Instruct.

Similarly, MEMIT assumes that the FFN layers function as a knowledge key-value store. It directly modifies the parameters of selected layers through least squares approximation. Unlike ROME, which updates a single layer, MEMIT is a multi-layer editing algorithm capable of simultaneously updating hundreds or thousands of facts

IFMET IFMET builds upon MEMIT by not only modifying earlier MLP layers in transformers but also adjusting later layers to enhance multi-hop reasoning for the edited knowledge. To ensure the updated knowledge propagates effectively, IFMET constructs an additional support set that reinforces learning in later layers. Based on our analysis in §2, we edit layers [17,18,19,20] for LLAMA3-8B-Instruct and layers [15,16,17,18] for Qwen2.5-7B-Instruct.

WISE WISE represents a different approach to model editing, focusing on later layers instead of earlier ones. It modifies the model's FFN output using a gating mechanism:

$$\text{FFN}_{\text{out}}(\mathbf{x}) = \begin{cases} \mathcal{G}(\mathbf{x}) \cdot \mathbf{W}_{v'} & \text{if } \mathcal{G}(\mathbf{x}) > \epsilon, \\ \mathcal{G}(\mathbf{x}) \cdot \mathbf{W}_{v} & \text{otherwise.} \end{cases} \quad (5)$$

Method	Model	MQUAKE-CF		MQUAKE-CF-v2		MQUAKE-T	
		H-Acc.↑	MAcc.↑	H-Acc.↑	MAcc.↑	Hop-wise.↑	MAcc.↑
Pre-edited		73.4	40.7	72.8	39.5	56.1	15.6
LoRA	Qwen2.5-7B-Ins	35.1	24.9	36.5	<u>25.9</u>	25.0	28.6
WISE		41.2	9.8	26.5	8.0	50.2	36.5
MeLLo		35.5	7.8	34.5	7.6	52.7	<u>56.5</u>
ROME		75.4	10.7	73.4	8.8	86.7	17.7
MEMIT		82.6	11.1	83.4	9.6	88.9	18.5
AlphaEdit		73.8	12.6	75.1	10.5	82.2	17.2
IFMET ♣		<u>83.7</u>	<u>25.7</u>	<u>84.6</u>	24.5	<u>90.0</u>	52.8
CaKE(ours)		**90.6**	**61.4**	**90.3**	**63.05**	**95.5**	**87.8**

Table 6: Comparison of CaKE with existing methods on MQuAKE on Qwen2.5-7B-Instruct. The best results are highlighted in bold, while the second-best results are underlined. ♣ means the results are based on our own implementation since the original code is not open by the authors, and we will update it after the source code is open.

Knowledge Type	Template	Answer
{target_person} works in the field of {target_field}	In a book related to different fields, Section A discusses {random_field}, Section B discusses {random_field}, and Section C discusses {target_field}. If you want to learn about {target_person}'s field, which section should you read?	The working field of {target_person} is discussed in Section C.
	In a biography book, Section A discusses the life of {random_person}, Section B discusses the life of {random_person}, and Section C discusses the life of {target_person}. The field of the person in Section C is?	The person in Section C works in the field of {target_field}.
{target_person} speaks the language of {target_language}.	The following facts are known: 1. {target_person} wears red clothes. 2. {random_person} wears blue clothes. 3. {random_person} wears green clothes. The language that the person in red clothes speaks is?	The language that the person in red clothes speaks is {target_language}.
	At a global company: {target_language}-speaking employees work in Team A. {random_language}-speaking employees work in Team B. In which team would {target_person} work when he/she is at work?	{target_person} would work in Team A when he/she is at work.

Table 7: Sample templates for generating the circuit-aware data.

Here, $\mathcal{G}(\mathbf{x})$ is a gate function that computes the activation score of the hidden reprsentation: $\|\mathcal{A}(\mathbf{x}) \cdot (\mathbf{W}_{v'} - \mathbf{W}_v)\|_2$. If the gate is activated, the model uses the updated knowledge to generate responses; otherwise, it relies on the original knowledge. Different methods define the gate function differently, but the core idea is to ensure that the updated memory aligns with relevant question representations.

MeLLo MeLLo is a non-parametric editing method that modifies a model's knowledge through prompting rather than weight updates. It maintains a memory of newly introduced facts and guides the model to decompose multi-hop queries into sub-questions. At each step, the model checks this memory to verify whether its existing knowledge contradicts the new facts. We follow the prompt structure provided in the original MeLLo method. However, in our experiments, we observe that the model struggles to consistently adhere to the intended reasoning pattern.

CaKE We utilize the original LoRA (Hu et al., 2022) and add parameters in both the FFN and attention module in the model. The hyperparameters are as follows:

- epoch: [40, 50, 60]
- batch size: [4]
- learning rate: [1e-4]
- rank: [8]
- lora_alpha: [32]

C More Analysis

C.1 Concurrence or Reasoning?

Studies such as Yang et al. (2024c); Ju et al. (2024); Hou et al. (2023); Zhang et al. (2024c) those that have discovered shortcuts in multi-hop reasoning. In the case of $((e_1,'', e_2), (e_2, r_2, e_3))$ (i.e., the query without r1), the model predicts correctly

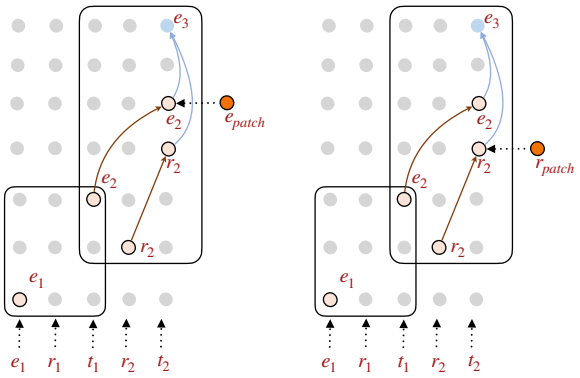


Figure 7: The way we test the function of the second hop. If the model conducts the function at the later layers, changing the representation would change the output of the model.

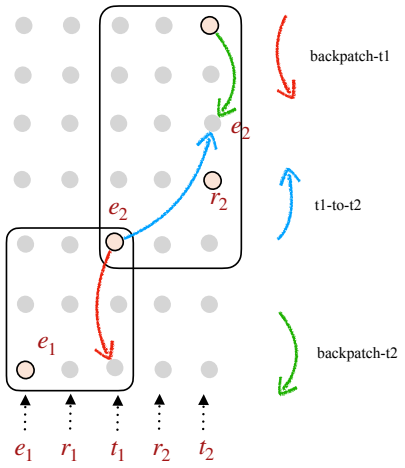


Figure 8: The way we conduct the backpatch and e_1 to e_2. We substitute the hidden representations from the source position to the target position.

due to a high correlation between e_1 and e_3. For instance, given the query: "The capital city of the country where the Eiffel Tower is located is..." LLMs can sometimes provide the correct answer even without the intermediate context ('the country where the Eiffel Tower is located'). In our analysis, we find that apart from the occurrence, the LLM would also sometimes conduct latent reasoning, such as 'latently conducting the r1 completion'. If the model gives the correct e3 for $((e_1,'', e_2), (e_2, r_2, ?))$ due to the occurrence, once we edit the $(e_1, r_1, e_2 \rightarrow e_2')$, the model would fail to give us the new answer. We select the shortcut data and conduct the editing in the first hop $(e_1, r_1, e_2 \rightarrow e_2')$ and then evaluate the model to see whether the edited model would output updated knowledge (e_1, r_1, r_2, e_3'). We conduct experiments on LLAMA3-8B-Instruct with the AlphaEdit method and demonstrate that about 65% percent of cases would give us the updated knowledge for the multi-hop questions, showing that edits to intermediate hops (e.g., updating the country) can disrupt reasoning when relying on pre-existing-shortcuts and correctly give us the newly updated reasoning results. This means that the LLM itself does not simply answer the questions due to the high correlation between e1 and e3, **but actually conducts the latent reasoning**.

C.2 Circuit Analysis

We present the model's critical information detection results in Figure 10. The results indicate that knowledge is distributed across different layers, with incorrect cases appearing in later layers compared to correct and inconsistent cases. Determining the optimal layer for editing remains challenging, so we choose to adjust the model across all layers. In the future, we aim to refine our approach by performing more targeted edits.

C.3 Failure Phenomenon

In the multi-hop reasoning, we view several failure cases to see how the language model made mistakes for reasoning and we see it as the **circuit competition**. Here, we find the LLM tends to give us a wrong answer for the middle cases of the different entities that appeared in the middle steps. Take 'The country that the creator of Hamlet was a citizen of' as an example; the bridge entity here is 'William Shakespeare'. We view 'Hamlet' as an entity that would influence the model to give us the results 'Denmark', which means the model has been distracted by other entities' information. As shown in Figure 9, the model gives us the correct answer 'England' around layer 27 but output the wrong answer 'Denmark', which is actually the country of the 'Hamlet'.

C.4 Comparison with Chain-of-Thought Reasoning and Prompt Learning

Instead of directly providing an answer, chain-of-thought (CoT) reasoning generates intermediate steps sequentially. As proposed by Yang et al. (2024c), CoT not only facilitates knowledge activation in large language models but also transforms them into effective in-context reasoners. The CoT process builds a chain of relevant facts within the prompt context, where each step's output serves as an *in-context memory* that subsequent steps can reference. This approach reduces the risk of losing

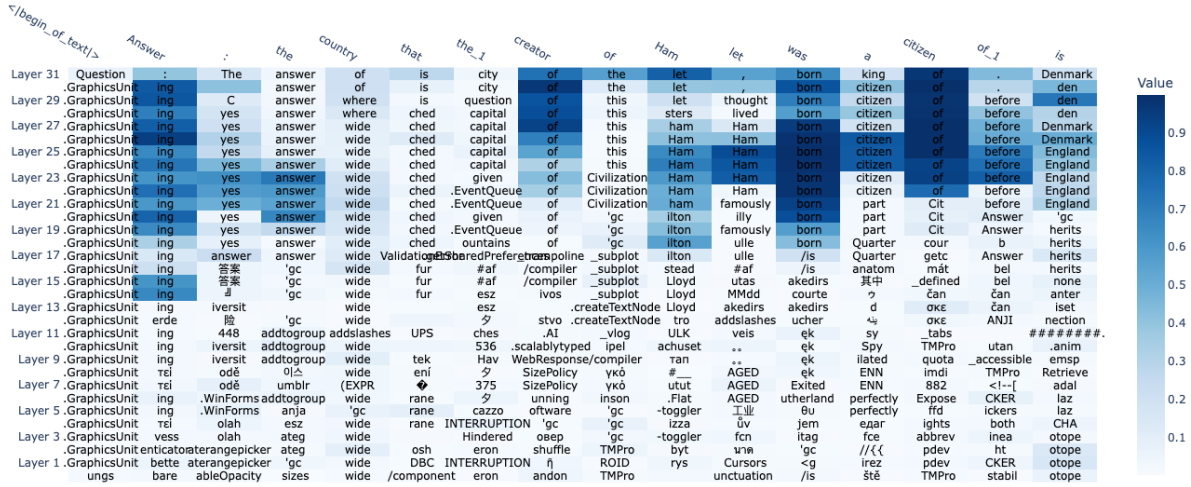


Figure 9: The failure case of the multi-hop reasoning.

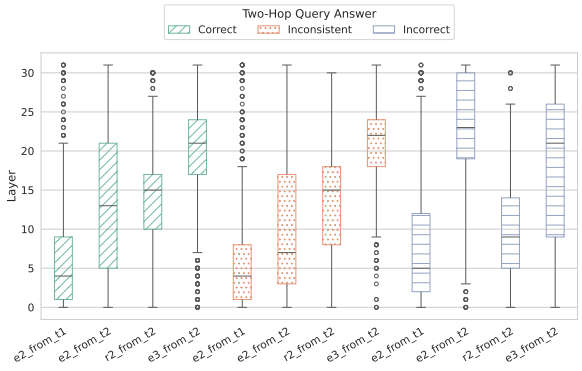


Figure 10: The distribution of the layers allows us to detect the information from critical positions in the model via patch_scope.

track of intermediate facts as the sequence length increases, thereby promoting more coherent multi-hop reasoning. Moreover, because a significant portion of the model's knowledge is stored in earlier layers, CoT can better leverage these neurons by decomposing complex questions into simpler sub-questions. Consequently, the reasoning circuit required for a single-hop inference is much simpler than that for multi-hop reasoning. This observation aligns with recent findings (Li et al., 2024), which demonstrate that fast thinking without CoT leads to larger gradients and greater gradient disparities across layers compared to CoT. Nonetheless, inconsistencies in the intermediate reasoning steps still occur, highlighting potential areas for improvement. We believe that further analysis is needed to address these issues, and we leave this exploration for future work.