

LEAP: Learning Edit-Adaptable Parameters for Multi-Hop Knowledge Propagation

Taichi Shiba^{1,2} Max Müller-Eberstein^{1,3} Masaru Isonuma^{1,2,4}

Yohei Oseki^{1,2} Yusuke Miyao^{1,2}

¹The University of Tokyo ²NII LLMC

³IT University of Copenhagen ⁴Tohoku University

{shiba-taichi60, yusuke}@is.s.u-tokyo.ac.jp

{iijimax, oseki}@g.ecc.u-tokyo.ac.jp isonuma@nii.ac.jp

Abstract

Knowledge editing aims to update factual knowledge in language models without full retraining. However, edits should also propagate to downstream queries that require combining the edited fact with other knowledge, such as multi-hop queries. Existing approaches address this propagation problem with more elaborate edit-time procedures, through augmented reasoning examples or external editing modules. In this work, we take a different approach: we meta-learn model parameters from which simple edits are more likely to propagate. We propose LEAP (*Learning Edit-Adaptable Parameters*), a MAML-based framework that learns edit-adaptable parameters such that simple gradient-based fine-tuning on new single-hop facts improves answers to their multi-hop consequences. Experiments on MQuAKE show that LEAP consistently improves over standard LoRA editing and achieves a favorable accuracy–efficiency trade-off compared with representative knowledge-editing baselines. Further analysis suggests that LEAP makes bridge entities more accessible and strengthens the bridge-to-answer transition needed for multi-hop reasoning. These results suggest a new direction for improving edit propagation: preparing the model itself to be more amenable to propagation-friendly updates, rather than relying only on increasingly complex edit-time procedures.¹

1 Introduction

Large language models (LLMs) acquire substantial factual knowledge during large-scale pretraining (Petroni et al., 2019; Chang et al., 2024). However, such parameterized knowledge can become outdated or incorrect after deployment and is difficult to update selectively. Knowledge editing addresses this problem by efficiently modifying a model’s behavior with respect to specific facts

¹Code is available at <https://github.com/mynlp/LEAP>.

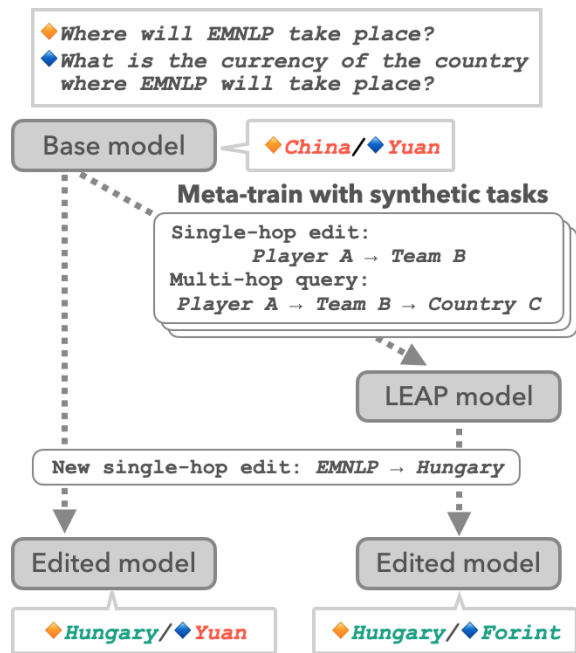


Figure 1: Overview of LEAP. Standard knowledge editing often fails to propagate an edit to related multi-hop questions. LEAP meta-learns edit-adaptable parameters so that, after the same single-hop edit, the edited model updates the downstream multi-hop answer as well.

without full retraining while preserving unrelated behavior (De Cao et al., 2021; Mitchell et al., 2022; Wang et al., 2024b). Yet successful editing requires more than recalling the edited fact itself: when a fact changes, its downstream consequences should change as well. For example, after editing where EMNLP will take place, the answer to “What is the currency of the country where EMNLP will take place?” should change accordingly (see Figure 1). This is challenging because the model must combine the edited fact with other relational knowledge. This ability, often referred to as portability or edit propagation, remains a major challenge (Yao et al., 2023; Zhong et al., 2023; Cohen et al., 2024).

Existing approaches typically improve propagation by making the edit-time procedure more

elaborate, e.g., by adding augmented reasoning examples (Yao et al., 2025b) or auxiliary editing modules (Liu et al., 2025). In contrast, we view edit propagation as a property of the model that can be improved before editing. Rather than making the editing algorithm more complex, our goal is to obtain a model that is intrinsically more edit-adaptable. After being edited using only single-hop examples, such a model should not only recall the edited fact, but also use it as part of downstream tasks, including multi-hop reasoning.

We propose LEAP (*Learning Edit-Adaptable Parameters*), a MAML (Finn et al., 2017)-based meta-learning framework to acquire such edit-adaptable parameters (Section 4). During meta-training, synthetic multi-hop data is used to train the model on single-hop edits in the inner loop, while the outer loop optimizes the pre-edit parameters to answer corresponding multi-hop queries correctly. Subsequent test-time editing relies solely on ordinary gradient-based updates of the target facts, making it lightweight overall.

We evaluate LEAP on MQuAKE (Zhong et al., 2023), a benchmark for multi-hop edit propagation. As MQuAKE does not provide meta-training tasks, we construct synthetic tasks whose entities and relations are disjoint from those in evaluation. This tests whether LEAP generalizes edit-propagation beyond the facts seen during meta-training.

In our experiments, we instantiate the editable parameters with LoRA (Hu et al., 2022) and evaluate on Llama3-8B-Instruct (Dubey et al., 2024) and Qwen2.5-7B-Instruct (Yang et al., 2025). Even when the meta-learned parameters are limited to lightweight LoRA adapters, LEAP substantially improves multi-hop propagation over ordinary LoRA under the same test-time editing procedure. It also performs competitively against representative knowledge editing methods while using only single-hop edit supervision at test time. These results show that edit propagation can be improved not only by designing more complex edit-time algorithms, but also by improving the model’s ability to adapt from simple edit updates (Section 5).

We further analyze whether LEAP improves the internal computation needed for multi-hop edit propagation, rather than merely increasing final-answer accuracy. Using activation patching (Zhang and Nanda, 2024) and Logit Lens (nostalgebraist, 2020), we find that LEAP makes bridge entities more accessible in middle layers and promotes final answers in later layers (Section 6).

Our contributions are as follows:

- We introduce a new direction for improving edit propagation: preparing models themselves to be more amenable to propagation-friendly updates. We instantiate this idea with LEAP, a MAML-based framework that learns edit-adaptable parameters.
- We construct synthetic meta-training tasks whose entities and relations are disjoint from those in MQuAKE evaluation, and show that LEAP achieves a favorable accuracy–efficiency trade-off against existing methods.
- We provide further analyses showing that LEAP’s improvements are accompanied by stronger bridge-to-answer dynamics in the edited model.

2 Related Work

Knowledge editing. Knowledge editing aims to update specific factual associations in language models while preserving unrelated knowledge. Representative parameter-editing methods include ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), and AlphaEdit (Fang et al., 2025), which directly modify model parameters to incorporate new facts. Although these methods can achieve strong single-hop edit success, this does not necessarily imply that their downstream consequences are propagated (Zhong et al., 2023; Cohen et al., 2024).

Edit propagation. MQuAKE (Zhong et al., 2023) introduced a benchmark for evaluating whether knowledge edits propagate across multi-hop questions. Subsequent work has proposed data-driven strategies for improving propagation, including retrieval-based editing (Shi et al., 2024), question decomposition (Gu et al., 2024; Lee et al., 2025; Simon and Ewetz, 2025), chain-of-thought prompting (Cohen et al., 2024; Wang et al., 2025), decoding-time correction (Sun et al., 2024), and circuit-aware data augmentation as in CaKE (Yao et al., 2025b).

On the model side, learning-to-edit approaches like MEND (Mitchell et al., 2022) learn auxiliary networks that transform fine-tuning gradients, while LTE (Jiang et al., 2024) trains LLMs to apply updated knowledge and uses retrieval-based inference. Most closely related to our work, PropMEND (Liu et al., 2025) meta-learns a hypernet-

work that transforms gradients for knowledge propagation. However, generalization to unseen relations and entities remains challenging. Our method, LEAP, is similar in spirit, but differs in what is meta-learned: rather than learning an auxiliary editor, LEAP makes model parameters themselves edit-adaptable, such that single-hop edits propagate effectively. As a result, test-time editing remains a simple procedure, without propagation-specific supervision or additional reasoning mechanisms.

Internal mechanisms of multi-hop reasoning. Recent work suggests that multi-hop reasoning in language models often involves intermediate bridge entities, although models may fail to use them reliably or instead exploit factual shortcuts (Yang et al., 2024; Ju et al., 2024). Other analyses suggest that multi-hop failures can arise when models access bridge entities too late or fail in later relational extraction stages (Biran et al., 2024; Yu et al., 2025). Motivated by these findings, we examine whether LEAP makes bridge entities and final answers more accessible after editing.

3 Knowledge Edit Propagation

We first formalize the problem of knowledge edit propagation. Let p_θ denote a language model parameterized by θ , which defines a conditional distribution over output sequences given an input query q . For evaluation, the model’s answer is obtained by greedy decoding from $p_\theta(\cdot | q)$.

A knowledge edit specifies one or more factual associations that should hold after editing, regardless of whether they replace prior associations or introduce new ones. We represent a set of edits as

$$E = \{e_j\}_{j=1}^M, \quad e_j = (s_j, r_j, o_j^*), \quad (1)$$

where s_j is a subject entity, r_j is a relation, and o_j^* is the target object that should be associated with (s_j, r_j) after editing. If an old object o_j is already associated with (s_j, r_j) , the edit corresponds to updating o_j to o_j^* . If no previous association is assumed, the edit corresponds to adding a new fact.

In this work, we focus on parameter-updating knowledge editing methods. Such a method takes the original parameters θ and the edit set E , and returns edited parameters

$$\theta' = \text{Edit}(\theta, E). \quad (2)$$

The edited model $p_{\theta'}$ is expected to reflect the target facts $\{(s_j, r_j, o_j^*)\}_{j=1}^M$ while preserving unrelated knowledge.

One basic requirement for an edited model is single-hop edit success: the model should answer queries that directly ask about the edited facts. Let

$$Q_{\text{hop}=1}(E) = \{(q_i, a_i)\}_{i=1}^{N_{\text{hop}=1}} \quad (3)$$

be a set of single-hop queries whose answers are determined directly by the edited facts. For example, if an edit specifies that EMNLP will take place in Hungary, a single-hop query may ask, “Where will EMNLP take place?” For each $(q_i, a_i) \in Q_{\text{hop}=1}(E)$, the answer decoded from $p_{\theta'}(\cdot | q_i)$ is expected to match a_i , where a_i is typically the corresponding edited object.

However, edits often have consequences beyond single-hop recall. We define multi-hop edit propagation as the ability of the edited model to answer queries whose answers should change as a consequence of multi-hop reasoning. Let

$$Q_{\text{hop}\geq 2}(E) = \{(q_i, a_i)\}_{i=1}^{N_{\text{hop}\geq 2}} \quad (4)$$

be a set of multi-hop queries affected by the edits E . Each multi-hop query requires using one or more edited facts together with other knowledge. For example, if an edit specifies that EMNLP will take place in Hungary, a multi-hop query may ask, “What is the currency of the country where EMNLP will take place?” In a multi-edit setting, a query may require composing several edited facts along a reasoning chain. The expected answer is therefore not necessarily one of the edited objects o_j^* , but a consequence derived from the edited facts and additional knowledge. For each $(q_i, a_i) \in Q_{\text{hop}\geq 2}(E)$, the answer decoded from $p_{\theta'}(\cdot | q_i)$ is expected to match a_i , where a_i denotes the multi-hop answer implied by the edits.

This requirement is more challenging than single-hop edit success. To answer such queries, the edited model must not only recall the edited facts, but also use them to identify intermediate bridge entities, integrate these entities with existing relational knowledge, and compose the resulting information across multiple hops. Knowledge edit propagation therefore tests whether an edit can influence the model’s downstream use of related knowledge, rather than merely whether the edited fact can be recalled in isolation.

4 Learning Edit-Adaptable Parameters

4.1 Meta-learning Objective

We propose LEAP (*Learning Edit-Adaptable Parameters*), a meta-learning approach for learn-

ing parameters that are amenable to edit propagation. LEAP is designed so that ordinary gradient-based fine-tuning on new single-hop facts also improves the model’s answers to related multi-hop queries. At test time, given an edit set E , the editing procedure only has access to the single-hop queries $Q_{\text{hop}=1}(E)$, while the multi-hop queries $Q_{\text{hop}\geq 2}(E)$ are not available. Therefore, we formulate meta-training to mirror this setting: the model is updated using only single-hop queries, while the quality of the resulting edited model is evaluated on multi-hop queries.

Let θ denote the editable parameters of the language model p_θ . The editable parameters can be either the full parameters of a pretrained model or a parameter-efficient subset, such as lightweight adapter parameters. For each edit set E , we update the model using only $Q_{\text{hop}=1}(E)$. The single-hop loss is defined as

$$\mathcal{L}_{\text{hop}=1}(E; \theta) = - \sum_{(q,a) \in Q_{\text{hop}=1}(E)} \log p_\theta(a | q). \quad (5)$$

Starting from the current parameters θ , applying the editing procedure to this loss produces edited parameters θ'_E . Conceptually, $p_{\theta'_E}$ is the model obtained after fitting the direct single-hop queries associated with the edit set E .

The edited model is then evaluated on multi-hop queries. We define the multi-hop loss as

$$\mathcal{L}_{\text{hop}\geq 2}(E; \theta'_E) = - \sum_{(q,a) \in Q_{\text{hop}\geq 2}(E)} \log p_{\theta'_E}(a | q). \quad (6)$$

The meta-learning objective is to obtain edit-adaptable parameters whose single-hop editing updates lead to low multi-hop loss:

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{E \sim \mathcal{P}_{\text{edit}}} [\mathcal{L}_{\text{hop}\geq 2}(E; \theta'_E)]. \quad (7)$$

4.2 Optimization

We optimize the objective in Eq. (7) with a MAML-style inner and outer loop (Finn et al., 2017). The inner loop simulates test-time editing by updating the editable parameters using only the single-hop loss. The outer loop uses the multi-hop loss of the edited model as the meta-training signal, encouraging single-hop updates to produce the desired downstream consequences. In practice, the distribution $\mathcal{P}_{\text{edit}}$ is represented by the collection of edit sets in the meta-training data. Algorithm 1 summarizes the resulting meta-training procedure detailed below, together with the test-time editing procedure described in Section 4.3.

Algorithm 1 LEAP: Meta-training and Test-time Editing

Meta-training

- 1: **for** each edit set $E \sim \mathcal{P}_{\text{edit}}$ **do**
- 2: $\theta_E^{(0)} \leftarrow \theta$
- Inner loop:*
- 3: **for** $k = 0, \dots, K - 1$ **do**
- 4: $\theta_E^{(k+1)} \leftarrow \theta_E^{(k)} - \alpha \nabla_{\theta_E^{(k)}} \mathcal{L}_{\text{hop}=1}(E; \theta_E^{(k)})$
- 5: **end for**
- 6: $\theta'_E \leftarrow \theta_E^{(K)}$
- Outer loop:*
- 7: $\theta \leftarrow \theta - \beta \nabla_{\theta} \mathcal{L}_{\text{hop}\geq 2}(E; \theta'_E)$
- 8: **end for**
- 9: $\theta^* \leftarrow \theta$

Test-time editing

- 10: Given a held-out edit set E , starting from θ^* :
- 11: **for** $k = 0, \dots, K - 1$ **do**
- 12: $\theta_E^{(k+1)} \leftarrow \theta_E^{(k)} - \alpha \nabla_{\theta_E^{(k)}} \mathcal{L}_{\text{hop}=1}(E; \theta_E^{(k)})$
- 13: **end for**
- 14: $\theta'_E \leftarrow \theta_E^{(K)}$ $\triangleright Q_{\text{hop}\geq 2}(E)$ used only for evaluation

Inner loop. For each edit set E , the inner loop applies K gradient descent steps to the single-hop loss:

$$\begin{aligned} \theta_E^{(0)} &= \theta, \\ \theta_E^{(k+1)} &= \theta_E^{(k)} - \alpha \nabla_{\theta_E^{(k)}} \mathcal{L}_{\text{hop}=1}(E; \theta_E^{(k)}) \end{aligned} \quad (8)$$

for $k = 0, \dots, K - 1$. We denote the resulting edited parameters by

$$\theta'_E = \theta_E^{(K)}. \quad (9)$$

Here, α is the inner-loop learning rate and K is the number of edit steps.

Outer loop. After the inner-loop update, the outer loop updates the pre-edit parameters using the multi-hop loss of the edited model:

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \mathcal{L}_{\text{hop}\geq 2}(E; \theta'_E), \quad (10)$$

where β is the outer-loop learning rate.

Computing the exact meta-gradient requires differentiating through the inner-loop updates in Eq. (8), which introduces expensive second-order derivative terms. We therefore use the first-order MAML approximation (Finn et al., 2017):

$$\nabla_{\theta} \mathcal{L}_{\text{hop}\geq 2}(E; \theta'_E) \approx \nabla_{\theta'_E} \mathcal{L}_{\text{hop}\geq 2}(E; \theta'_E). \quad (11)$$

This approximation avoids backpropagating through the full inner-loop computation graph, while retaining the central training signal: the parameters should be learned so that single-hop updates improve multi-hop answers.

4.3 Test-time Editing

Through LEAP, we obtain the meta-learned parameters θ^* . Given a held-out edit set E , we apply standard fine-tuning, corresponding to Eq. (8), starting from θ^* and using only the single-hop loss $\mathcal{L}_{\text{hop}=1}(E; \theta)$. This yields edited parameters θ'_E and the edited model $p_{\theta'_E}$.

Thus, LEAP uses the same test-time supervision as ordinary gradient-based editing on single-hop facts. The only difference is that the pre-edit parameters have been meta-trained so that single-hop updates are encouraged to improve multi-hop answers. The multi-hop queries $Q_{\text{hop} \geq 2}(E)$ are used only for evaluation, not for test-time editing.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate our method on MQuAKE (Zhong et al., 2023), a benchmark for measuring whether knowledge edits propagate to multi-hop questions. MQuAKE contains two subsets, MQuAKE-CF and MQuAKE-T. MQuAKE-CF evaluates propagation under counterfactual edits, while MQuAKE-T evaluates propagation under temporal knowledge updates. For MQuAKE-CF, we use the v2 release. Each case contains one or more edited facts and a question whose answer should change as a consequence of those edits. MQuAKE includes 2-, 3-, and 4-hop reasoning chains and covers diverse edit patterns, including intermediate-hop edits and non-contiguous combinations of edited hops.

Because MQuAKE provides evaluation cases but not meta-training tasks, we construct a separate dataset of synthetic tasks for meta-training. We first sample relation chains from Wikidata (Vrandečić and Krötzsch, 2014) using SPARQL queries through the Wikidata Query Service². To avoid leakage, we exclude any chain containing a Wikidata property ID or entity ID that appears in either MQuAKE-CF or MQuAKE-T. These sampled chains serve as suffixes in the resulting multi-hop chains. We then construct edited prefixes that start from fictional entities, optionally pass through other fictional entities, and end at a real Wikidata entity. These prefix hops define the edit set E . For each edited prefix, we either append sampled suffixes to its final real entity or use the multi-hop prefix itself as the full chain. Using simple man-

Dataset	2-hop	3-hop	4-hop	Total
Meta-train	2,796	3,000	2,412	8,208
MQuAKE-CF	1,134	1,136	729	2,999
MQuAKE-T	1,420	445	2	1,867

Table 1: Hop distribution of the meta-training tasks and MQuAKE dataset.

ually written cloze-style templates, we instantiate single-hop queries from the edited prefix and multi-hop queries from the full chain. Thus, the resulting tasks require either composing multiple edited facts or combining edited facts with background relational knowledge, beyond direct single-hop recall. Details of the synthetic task construction are provided in Appendix A, while Table 1 summarizes the hop distributions of each dataset.

By construction, meta-training and evaluation do not share Wikidata entity or property IDs. This prevents direct reuse of the same entities or relations across meta-training and evaluation, ensuring that improvements reflect generalization beyond the facts and relations seen during meta-training.

Using fictional entities in the edited prefixes further encourages the model to rely on the supplied edits rather than memorized pretraining knowledge, while avoiding overwriting real-entity facts that could degrade the model’s prior knowledge.

Baselines. We compare our method with representative parameter-updating knowledge editing methods, including ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), AlphaEdit (Fang et al., 2025), and CaKE (Yao et al., 2025b).

To isolate the effect induced by LEAP’s meta-learning strategy, we further include a LoRA-only fine-tuning baseline, which updates the same LoRA modules for the same number of gradient steps during the test-time editing procedure, but uses standard-initialized LoRA parameters, instead of the meta-trained parameters from LEAP. We further test the effect of synthetic data exposure in an additional ablation study in Section 5.4.

Implementation details. We evaluate LEAP and the main baselines on Llama3-8B-Instruct (Dubey et al., 2024) and Qwen2.5-7B-Instruct (Yang et al., 2025). For LEAP, the editable parameters are implemented as LoRA adapters attached to all attention and MLP projection layers. These adapters are first meta-trained, and then updated during test-time editing. Further hyperparameters and optimization details are provided in Appendix B.

²<https://query.wikidata.org>.

Model	Method	MQuAKE-CF			MQuAKE-T		
		Instance-wise	Multi-hop		Instance-wise	Multi-hop	
			Cloze	QA		Cloze	QA
Llama3	ROME	87.7	13.0	14.0	88.7	4.4	1.0
	MEMIT	28.5	4.9	5.4	47.4	0.6	0.1
	AlphaEdit	78.7	7.9	7.4	84.6	1.8	0.7
	CaKE	90.5	58.7	53.5	88.8	70.9	55.0
	LoRA	92.4	57.3	47.0	94.0	85.1	40.5
	LEAP (ours)	89.5	80.3	82.0	93.6	87.4	90.8
Qwen2.5	CaKE	90.7	65.3	60.2	95.7	94.8	79.3
	LoRA	74.7	33.4	20.0	89.9	77.7	56.2
	LEAP (ours)	90.8	77.2	75.1	91.4	85.8	83.6

Table 2: Main results on MQuAKE-CF and MQuAKE-T with Llama3-8B-Instruct and Qwen2.5-7B-Instruct. We report exact-match accuracy (%) on instance-wise, multi-hop cloze, and multi-hop QA evaluations.

Method	MQuAKE-CF (sec.)	MQuAKE-T (sec.)
ROME	19.35	5.84
MEMIT	17.49	12.87
AlphaEdit	26.63	18.54
CaKE	82.24	46.06
LoRA	2.94	2.80
LEAP	2.91	2.74

Table 3: Average test-time editing time per case in seconds with Llama3-8B-Instruct.

	2-hop		3-hop			4-hop			
	1	2	1	2	3	1	2	3	4
#Cases	240	894	204	306	626	7	55	138	529
CaKE	20.8	69.4	2.7	8.5	77.4	14.3	10.0	30.8	84.4
LoRA	22.9	65.8	3.2	12.6	75.2	21.4	7.3	16.7	71.2
LEAP	56.9	96.6	26.7	49.0	97.8	35.7	8.2	61.6	99.1

Table 4: Accuracy breakdown on MQuAKE-CF with Llama3-8B-Instruct by reasoning-chain length and the position of the last edited hop.

Evaluation metrics. We use instance-wise accuracy and multi-hop accuracy as evaluation metrics, to enable comparisons with prior work (Zhong et al., 2023; Yao et al., 2025b).

Instance-wise accuracy measures performance on the single-hop queries associated with each multi-hop reasoning chain. For each multi-hop instance, we compute the fraction of associated single-hop queries answered correctly, including both edited and unedited hops, and then average this value across instances. This metric indicates whether the model can recover the individual facts needed for the corresponding multi-hop query.

Multi-hop accuracy measures whether the model correctly answers propagated multi-hop queries. We report this metric separately for cloze-style and QA-style queries.

For answer matching, we generate up to 10 tokens for each query and count the output as correct if it contains the expected answer exactly.

5.2 Main Results

Table 2 shows the main results on MQuAKE-CF and MQuAKE-T using Llama3-8B-Instruct and Qwen2.5-7B-Instruct. Across both base models and datasets, LEAP substantially improves over the LoRA baseline under the same test-time editing procedure. On Llama3 MQuAKE-CF, LEAP improves multi-hop QA accuracy from 47.0% to 82.0%, and on Qwen2.5 MQuAKE-CF from 20.0% to 75.1%. These improvements are notable because LEAP is meta-trained only with cloze-style queries, yet the gains transfer to QA-style evaluation. Together, these results indicate that the gains stem not merely from using LoRA adapters, but from these parameters having been meta-learned to be more amenable to edit propagation.

Consistent with prior observations (Zhong et al., 2023; Yao et al., 2025b), high instance-wise accuracy does not necessarily imply high multi-hop accuracy. Several baselines recall many of the individual single-hop facts but still perform poorly on multi-hop evaluations. LEAP narrows this gap by maintaining strong instance-wise accuracy while substantially improving multi-hop outputs.

LEAP is also competitive with CaKE, a strong baseline that relies on additional augmented edit-time training data for multi-hop reasoning. On Llama3, LEAP outperforms CaKE on 5 out of 6 reported metrics, with the exception of MQuAKE-CF instance-wise accuracy (89.5 vs. 90.5). On Qwen2.5, LEAP achieves the best results across all MQuAKE-CF metrics, although CaKE remains stronger on some MQuAKE-T metrics. Overall, these results suggest that meta-learned parameters provide an effective and lightweight alternative for improving knowledge edit propagation.

Comparing the test-time editing cost of Llama3 in Table 3, LEAP has an editing time comparable

Model	Method	MQuAKE-CF			MQuAKE-T		
		Instance-wise	Multi-hop		Instance-wise	Multi-hop	
			Cloze	QA		Cloze	QA
Llama3	LoRA	92.4	57.3	47.0	94.0	85.1	40.5
	LoRA+Synthetic	80.2	69.9	67.1	87.6	94.6	94.7
	LEAP (ours)	89.5	80.3	82.0	93.6	87.4	90.8
Qwen2.5	LoRA	74.7	33.4	20.0	89.9	77.7	56.2
	LoRA+Synthetic	81.4	71.0	67.8	85.4	98.1	93.7
	LEAP (ours)	90.8	77.2	75.1	91.4	85.8	83.6

Table 5: Ablation comparing LoRA, LoRA+Synthetic (ordinary supervised fine-tuning on LEAP’s synthetic single-hop and multi-hop queries, without the MAML inner/outer-loop split), and LEAP on MQuAKE-CF and MQuAKE-T.

to ordinary LoRA fine-tuning, while being much faster than CaKE. On MQuAKE-CF, LEAP takes 2.91 seconds per case on average, compared with 82.24 seconds for CaKE. Thus, LEAP improves multi-hop propagation while preserving the efficiency of simple gradient-based editing.

5.3 Breakdown by Reasoning Pattern

We further analyze performance by grouping MQuAKE-CF by the reasoning-chain length and the position of the last edited hop. The last edited hop indicates how far edited information must be propagated before reaching the answer. When the last edit appears early in the chain, the model must carry its effect through more subsequent reasoning steps, making such cases more challenging.

Table 4 reports multi-hop accuracy aggregated over both cloze-style and QA-style queries. It shows that accuracy generally decreases as the last edited hop moves earlier in the chain. For example, in 3-hop questions, LEAP achieves 97.8% accuracy when the last edited hop is the third hop, but drops to 49.0% and 26.7% when the last edited hop is the second and first hop, respectively. This trend is also consistent across LoRA and CaKE (e.g., dropping from 77.4% to 2.7%), indicating that long-range propagation remains difficult even when the edited facts can be incorporated at the single-hop level.

Despite deeper edit propagation posing a substantial challenge, LEAP improves over all methods across all settings with at least 100 cases. Compared to CaKE, LEAP improves accuracy from 20.8% to 56.9% for 2-hop queries with first-hop edits, and from 8.5% to 49.0% for 3-hop cases with second-hop edits. Even for the 3-hop cases with first-hop edits, which are the most challenging for CaKE and LoRA, LEAP achieves a roughly 10-fold accuracy increase from 2.7% to 26.7%.

5.4 Disentangling Data Exposure from Meta-Learning

To verify that LEAP’s improvements stem from the proposed MAML-style optimization, rather than from additional exposure to synthetic multi-hop data, we introduce an additional ablation baseline, **LoRA+Synthetic**, which fine-tunes the same LoRA adapters directly on the same synthetic single-hop and multi-hop queries used for LEAP’s meta-training, without the MAML inner/outer-loop split. At test time, LoRA+Synthetic is then edited using the same single-hop editing procedure as LoRA and LEAP.

Table 5 compares LoRA, LoRA+Synthetic, and LEAP on MQuAKE-CF and MQuAKE-T, using Llama3-8B-Instruct and Qwen2.5-7B-Instruct. LoRA+Synthetic improves multi-hop accuracy over plain LoRA in every setting, confirming that synthetic data itself is informative for propagation. However, this comes at the cost of instance-wise accuracy on Llama3, where LoRA+Synthetic drops sharply relative to plain LoRA (92.4 $\rightarrow$ 80.2 on MQuAKE-CF). LEAP retains instance-wise accuracy much closer to the LoRA baseline than LoRA+Synthetic does (89.5 vs. 80.2 on MQuAKE-CF; 93.6 vs. 87.6 on MQuAKE-T) while also outperforming LoRA+Synthetic on 8 of 12 metrics overall.

While LoRA+Synthetic obtains the highest performance on four MQuAKE-T settings, it is important to note that 99.63% of MQuAKE-T cases have their last edited hop coinciding with the final hop of the reasoning chain, such that high multi-hop accuracy does not necessarily require propagation through background knowledge. MQuAKE-CF’s last-edited-hop positions, in contrast, are well distributed across chain lengths (Table 4), making it a more informative testbed for propagation. We therefore view the MQuAKE-CF ablation results

as clear evidence that LEAP’s MAML-style formulation induces parameter-level changes that drive genuine multi-hop propagation, rather than being merely an effect of exposure to additional multi-hop data as in LoRA+Synthetic.

6 Analysis

Our previous results show that LEAP improves the propagation of single-hop edits to multi-hop queries. To better understand the mechanisms underlying these improvements, we next investigate how LEAP changes the model-internal computation during multi-hop propagation by comparing it with the LoRA baseline, which uses the same LoRA modules and the same single-hop editing procedure but without meta-training. Because both models are edited on the same single-hop queries, their differences isolate the effect of the meta-learned pre-edit parameters.

Setup. All analyses in this section focus on Llama3-8B-Instruct, using the 240 MQuAKE-CF cases that consist of cloze-style 2-hop queries where only the first hop is edited. This subset has a simple bridge structure: the model must propagate the edited first-hop fact through a single bridge entity to reach the correct final answer. This subset also shows large improvements for LEAP over LoRA, making it useful for analyzing how meta-learning changes model-internal edit propagation.

Differences concentrate at the final input token.

We first localize where LEAP differs from the LoRA baseline during multi-hop answering. After applying the same single-hop edits E to both models, we run each corresponding multi-hop query and perform activation patching (Zhang and Nanda, 2024) in both directions: i.e., we replace hidden states in the LoRA run with the corresponding hidden states from the LEAP run, and vice versa. For each layer and token position, we measure the change in the log probability of the correct final answer and plot the effects in Figure 2. To visualize effects across prompts with different lengths, we align token positions relative to the end of the prompt, showing the last 16 prompt tokens, with position 0 denoting the final input token before answer generation.

In both patching directions, the largest effects are concentrated at the final input token. Patching LEAP activations into the LoRA run increases the log probability of the correct final answer, whereas

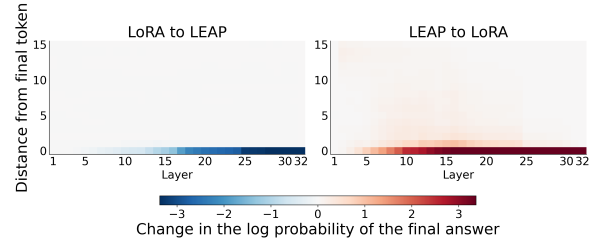


Figure 2: Activation patching results. We measure the change in the log probability of the correct final answer. Token positions are aligned relative to the end of the prompt, where position 0 denotes the final input token before answer generation.

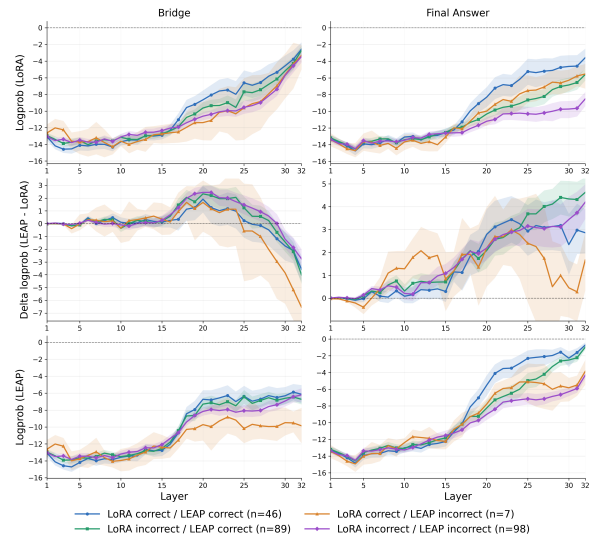


Figure 3: Logit Lens analysis at the final input token. Lines denote group means by LoRA/LEAP correctness, and shaded regions show 95% bootstrap confidence intervals.

patching LoRA activations into the LEAP run decreases it. This suggests that meta-learning induces an important representational difference immediately before answer generation. We therefore focus next on how the information represented at the final input token differs across methods.

LEAP induces a shift from bridge to final-answer decodability.

We next use Logit Lens (nostalgebraist, 2020) to examine what information is represented at the final input token while the edited model is answering the multi-hop query. For each layer l , we project the hidden state at the final input token to the vocabulary space and compute both the raw log probability of an entity token e as well as the probability difference

between LEAP and the LoRA baseline:

$$\Delta \log p_l(e) = \log p_{\text{LEAP},l}(e) - \log p_{\text{LoRA},l}(e), \quad (12)$$

where e denotes the first token of either the bridge entity or the final-answer entity. A positive value indicates that LEAP assigns higher probability to the entity token than LoRA at that layer. We divide the cases into four groups by LoRA/LEAP correctness and show the layer-wise trajectories for bridge and final-answer entities under LoRA, under LEAP, and for the LEAP–LoRA difference in Figure 3.

The LEAP–LoRA deltas show a layer-wise shift in entity decodability: LEAP increases log probabilities of the bridge entity around the middle layers across all groups, before decreasing its decodability in the later layers.

The later layers show a complementary pattern for the final-answer entity. While bridge-entity decodability diminishes toward later layers, LEAP increases final-answer decodability. This increase in final-answer decodability is observed not only in the *LoRA incorrect / LEAP correct* group, but also in other groups. This indicates that LEAP often moves probability mass toward the correct answer even when the final generated output remains incorrect. Overall, these trends suggest that LEAP first makes the bridge entity more accessible in intermediate layers and then promotes the final answer near the output.

The absolute LEAP trajectories further clarify what distinguishes successful from unsuccessful propagation. Although LEAP raises bridge-entity decodability in the middle layers even for unsuccessful cases, LEAP-correct cases (*LoRA correct / LEAP correct* and *LoRA incorrect / LEAP correct*) reach higher absolute bridge log probabilities in these layers than LEAP-incorrect cases (*LoRA correct / LEAP incorrect* and *LoRA incorrect / LEAP incorrect*). This higher bridge decodability in the middle layers is accompanied by a stronger rise of the final-answer entity from the middle layers toward the output. In LEAP-incorrect cases, LEAP still promotes the final-answer entity, indicating that it often moves the model toward the correct answer; however, the resulting absolute log probability remains too low to change the generated output. Thus, successful propagation depends not merely on increasing the probability of the correct final entity, but on making the bridge-to-answer transition strong enough to complete the later hop.

This interpretation is consistent with the “hop-

ping too late” theory of multi-hop failures (Biran et al., 2024). Concretely, if the bridge entity is not sufficiently available before the later answer-extraction stage, the model may fail to complete the second hop. Although our analysis is correlational, the results suggest that LEAP’s gains are associated with a stronger bridge-to-answer transition, rather than stemming solely from the improved recall of the edited first-hop fact, or an exclusive reliance on arriving directly at the final answer.

7 Conclusion

We presented LEAP, a meta-learning framework for improving knowledge edit propagation in language models (Section 4). LEAP learns edit-adaptable parameters from which simple gradient-based editing on single-hop facts is more likely to improve answers to their multi-hop consequences.

We instantiate LEAP with LoRA adapters, while keeping the objective independent of this particular parameterization. Experiments on MQuAKE show that LEAP consistently improves over standard LoRA and achieves a favorable accuracy–efficiency trade-off against representative knowledge editing baselines across two model families (Section 5). These results show that multi-hop edit propagation can be improved not only by adding more complex edit-time mechanisms, but also by preparing the model in advance so that simple single-hop edit updates propagate more effectively.

Our analysis further suggests that LEAP strengthens bridge-to-answer dynamics, making bridge entities more accessible in middle layers and promoting correct final answers in later layers (Section 6). LEAP may thus mitigate a model-internal bottleneck identified in prior work (Biran et al., 2024), where multi-hop reasoning can fail when bridge entities are not available early enough for later answer extraction.

Overall, LEAP highlights a model-centric approach to knowledge editing: rather than relying only on propagation-specific supervision or external mechanisms during editing, models can be meta-trained so that ordinary single-hop updates propagate more effectively. This new parameter-driven pathway opens up broader opportunities for future work, including the extension of meta-training to more diverse edit patterns and different forms of knowledge updates.

8 Limitations

Model scale. Our experiments are limited to 7–8B scale instruction-tuned models. While Llama3-8B-Instruct and Qwen2.5-7B-Instruct were chosen to balance multi-hop reasoning performance with computational efficiency, evaluating larger models is important for understanding how the benefits of meta-learned edit-adaptable parameters scale. Because LEAP operates through lightweight LoRA adapters, it may be practically applicable to larger models. Our results across two model families support this possibility, although further evaluation is needed to determine whether the same benefits persist at larger scales.

Long-range propagation. Long-range propagation remains challenging. Our breakdown by reasoning pattern (Section 5.3) shows that performance decreases when the last edited hop appears earlier in the reasoning chain, meaning that the effect of an edit must be carried through more subsequent hops. Prior work suggests that multi-hop reasoning may require sufficient model depth as the number of hops increases (Yao et al., 2025a). This provides one possible perspective on why long-range propagation remains difficult in our setting. While LEAP substantially improves multi-hop edit propagation, it does not fully solve cases that require carrying the effect of an edit through many subsequent reasoning steps.

Locality. We additionally find that LEAP’s improved propagation comes with reduced locality on unrelated facts relative to plain LoRA editing, particularly on multi-hop queries – a trade-off also observed for CaKE (Appendix C.2). We see incorporating a locality-aware term into LEAP’s outer-loop objective as a promising direction for mitigating this trade-off in future work.

Scope of generalization. To isolate edit propagation in a controlled setting, our evaluation focuses on multi-hop reasoning over relation chains. We do not study other forms of edit generalization, such as inverse relations from (s, r, o) to (o, r^{-1}, s) , cross-lingual transfer, or more complex and paraphrased natural-language phrasings beyond the cloze- and QA-style prompts used here. Extending LEAP to these settings in future work could provide a more comprehensive understanding of how meta-learning affects the broader generalization capabilities of edited models.

9 Ethical Considerations

Knowledge editing can help correct outdated or incorrect facts in language models, but it also carries risks. If an edit introduces false, biased, private, or harmful information, the edited model may reproduce that information. Because LEAP aims to improve edit propagation, such effects may extend beyond direct queries about the edited fact to downstream queries that combine the edit with other knowledge. Thus, propagation-friendly editing may increase the impact of harmful or erroneous edits.

In deployment, edit requests should be validated before being applied, especially for sensitive domains or claims about real individuals and organizations. Edited models should also be audited for both direct edit behavior and downstream propagated effects. We view LEAP as a research contribution toward understanding edit propagation, not as a complete deployment-ready editing system.

Acknowledgments

This work was supported by JST BOOST Grant Number JPMJBY24A6 and JPMJBY24B2. In this research work, we used the “mdx: a platform for building data-empowered society” (Suzumura et al., 2022). MME is supported by the Carlsberg Foundation, grant CF-25-0624.

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.
- Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. 2024. [How do large language models acquire factual knowledge during pretraining?](#) In *Advances in Neural Information Processing Systems*, volume 37, pages 60626–60668. Curran Associates, Inc.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *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*, 12:283–298.
- Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. [Editing factual knowledge in language models](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The Llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.
- Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. [AlphaEdit: Null-space constrained model editing for language models](#). In *The Thirteenth International Conference on Learning Representations*.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. [Model-agnostic meta-learning for fast adaptation of deep networks](#). In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.
- Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. 2024. [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)*, pages 8069–8083, Bangkok, Thailand. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- 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*.
- Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjuan Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, Qun Liu, and Wei Wang. 2024. [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.
- Jaewook Lee, Dahyun Jung, and Heuiseok Lim. 2025. [StepKE: Stepwise knowledge editing for multi-hop question answering](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7752–7765, Suzhou, China. Association for Computational Linguistics.
- Zeyu Leo Liu, Greg Durrett, and Eunsol Choi. 2025. [PropMEND: Hypernetworks for knowledge propagation in LLMs](#). *Preprint*, arXiv:2506.08920.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations*.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. [Locating and editing factual associations in GPT](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 17359–17372. Curran Associates, Inc.
- 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*.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. [Fast model editing at scale](#). In *International Conference on Learning Representations*.
- nostalgebraist. 2020. [interpreting GPT: The logit lens](#).
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. [PyTorch: An imperative style, high-performance deep learning library](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. [Language models as knowledge bases?](#) In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. 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*

- '24, page 2056–2066, New York, NY, USA. Association for Computing Machinery.
- Dominic Simon and Rickard Ewetz. 2025. [Knowledge editing for multi-hop question answering using semantic analysis](#). In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI '25*.
- 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*, pages 9282–9293, Bangkok, Thailand. 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, and Jason Wei. 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, Toronto, Canada. Association for Computational Linguistics.
- Toyotaro Suzumura, Akiyoshi Sugiki, Hiroyuki Takizawa, Akira Imakura, Hiroshi Nakamura, Kenjiro Taura, Tomohiro Kudoh, Toshihiro Hanawa, Yuji Sekiya, Hiroki Kobayashi, Yohei Kuga, Ryo Nakamura, Renhe Jiang, Junya Kawase, Masatoshi Hanai, Hiroshi Miyazaki, Tsutomu Ishizaki, Daisuke Shimotoku, Daisuke Miyamoto, and 13 others. 2022. [mdx: A cloud platform for supporting data science and cross-disciplinary research collaborations](#). In *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCoM/CyberSciTech)*, pages 1–7.
- 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, Minneapolis, Minnesota. Association for Computational Linguistics.
- Denny Vrandečić and Markus Krötzsch. 2014. [Wiki-data: a free collaborative knowledgebase](#). *Commun. ACM*, 57(10):78–85.
- Changyue Wang, Weihang Su, Qingyao Ai, Yichen Tang, and Yiqun Liu. 2025. [Knowledge editing through chain-of-thought](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10673–10693, Suzhou, China. Association for Computational Linguistics.
- 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. 2024a. [EasyEdit: An easy-to-use knowledge editing framework for large language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93, Bangkok, Thailand. Association for Computational Linguistics.
- Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024b. [Knowledge editing for large language models: A survey](#). *ACM Comput. Surv.*, 57(3).
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. 2024. [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.
- Yuekun Yao, Yupei Du, Dawei Zhu, Michael Hahn, and Alexander Koller. 2025a. [Language models can learn implicit multi-hop reasoning, but only if they have lots of training data](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9684–9702, Suzhou, China. Association for Computational Linguistics.
- Yunzhi Yao, Jizhan Fang, Jia-Chen Gu, Ningyu Zhang, Shumin Deng, Huajun Chen, and Nanyun Peng. 2025b. [CaKE: Circuit-aware editing enables generalizable knowledge learners](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11366–11382, Suzhou, China. Association for Computational Linguistics.
- 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, Singapore. Association for Computational Linguistics.

Zeping Yu, Yonatan Belinkov, and Sophia Ananiadou. 2025. [Back attention: Understanding and enhancing multi-hop reasoning in large language models](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11257–11272, Suzhou, China. Association for Computational Linguistics.

Fred Zhang and Neel Nanda. 2024. [Towards best practices of activation patching in language models: Metrics and methods](#). In *The Twelfth International Conference on Learning Representations*.

Zexuan Zhong, Zhengxuan Wu, Christopher 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, Singapore. Association for Computational Linguistics.

A Synthetic Meta-training Data Construction

We construct synthetic meta-training tasks for LEAP using Wikidata relation chains. The goal of this construction is to create tasks in which the model must learn from edited facts and then either compose multiple edited facts or combine them with background relational knowledge. Each task consists of an edited prefix, whose facts are used as the edit set, and optionally a Wikidata-derived suffix, which provides background knowledge for tasks whose answers require knowledge beyond the edited prefix.

Relation and entity separation from MQuAKE.

To avoid overlap with the MQuAKE evaluation sets, we first collect the Wikidata entity IDs and property IDs appearing in MQuAKE-CF and MQuAKE-T. All properties used in our synthetic meta-training tasks are manually selected from a predefined set of Wikidata property IDs that do not appear in MQuAKE. We also exclude any Wikidata triples (s, r, o) whose subject or object entity appears in MQuAKE. As a result, the synthetic meta-training tasks and the MQuAKE evaluation sets do not share Wikidata entities or relations.

Constructing Wikidata-derived suffixes. For tasks with a Wikidata-derived suffix, we construct the suffix part of the chain from real Wikidata triples. The suffix is intended to represent background knowledge that the model can combine with the edited prefix. To increase the likelihood that this background knowledge is already available to the model, we restrict candidate triples in two

ways. First, we keep only triples whose entities are among the top one million Wikidata entities ranked by the number of sitelinks. Second, we filter these triples using Llama3.2-1B-Instruct (Dubey et al., 2024) and retain only triples that the model answers correctly, treating them as likely known facts.

From the resulting pool of known Wikidata triples, we compose relation chains. These chains serve as suffixes for the synthetic examples that require Wikidata-derived background knowledge. No fictional entities are used in the suffix; it consists only of real Wikidata entities and relations that are disjoint from MQuAKE.

Adding fictional edited prefixes. We then construct one or more edited prefix hops for each synthetic chain. For tasks with a suffix, the prefix is constructed so that its final object connects to the first entity of the Wikidata-derived suffix. For tasks without a suffix, the edited prefix itself forms the full multi-hop chain. For each prefix hop, we choose a property from the predefined non-overlapping property list whose expected object type is compatible with the type of the next entity in the chain. The subject of each prefix hop is a fictional entity generated by randomly concatenating pronounceable syllables. For multi-hop prefixes, intermediate entities are also fictional; in tasks with a suffix, the final prefix hop links a fictional entity to the first real entity of the suffix.

All facts in the prefix are treated as edited facts and constitute the edit set E . For example, a synthetic chain may have the form

$$(\tilde{s}_1, r_1, \tilde{s}_2), (\tilde{s}_2, r_2, v_1), (v_1, r_3, v_2), \dots,$$

where $\tilde{s}_1$ and $\tilde{s}_2$ are fictional, and $v_1, v_2, \dots$ are real Wikidata entities. The prefix facts $(\tilde{s}_1, r_1, \tilde{s}_2)$ and $(\tilde{s}_2, r_2, v_1)$ are used as edited facts, while the remaining real-entity chain forms the suffix.

Generating single-hop and multi-hop queries.

We convert triples and relation chains into natural-language queries using manually written templates for each Wikidata property ID. For each edited prefix fact, we generate a direct single-hop query whose answer is the corresponding object. These queries form $Q_{\text{hop}=1}(E)$ and are used for the editing update during meta-training.

We generate multi-hop queries $Q_{\text{hop} \geq 2}(E)$ from the full chain, which consists of the edited prefix followed by the Wikidata-derived suffix when such a suffix is present. Thus, answering these

Chain length	Edited prefix	#Tasks
2-hop	1-hop	1,296
2-hop	2-hop	1,500
3-hop	1-hop	1,000
3-hop	2-hop	1,000
3-hop	3-hop	1,000
4-hop	1-hop	412
4-hop	2-hop	1,000
4-hop	3-hop	1,000

Table 6: Statistics of the synthetic meta-training tasks. Edited prefix denotes how many initial hops in the reasoning chain are treated as edited facts.

queries requires the model to either compose multiple edited facts or combine the supplied edits with background relational knowledge, rather than relying on direct single-hop recall alone. We intentionally use fixed templates rather than diverse paraphrases, keeping the surface form simple. This allows us to test whether meta-learning can improve edit propagation without relying on large-scale prompt or paraphrase diversity.

For example, a 2-hop chain involving the fictional entity *Gouthdaingal* yields the meta-training cloze query

*The currency used in the country that
Gouthdaingal represents in sport is ____*

MQuAKE’s evaluation questions, by comparison, are not built from a fixed template at all: they are generated by prompting GPT-3.5-turbo with the underlying relation chain, yielding freely-phrased questions such as

*The occupation of the creator of Nick
Bottom is ____
What is the occupation of the creator of
Nick Bottom?*

for its cloze-style and QA-style formats, respectively.

Statistics of synthetic meta-training tasks. Table 6 summarizes the number of constructed meta-training tasks by total reasoning-chain length and edited prefix length. Here, edited prefix length denotes how many initial hops in the chain are treated as edited facts.

B Implementation Details

Editing baselines. For ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), and AlphaEdit (Fang et al., 2025), we use the default configurations provided by EasyEdit (Wang et al.,

Hyperparameter	Value
LoRA target modules	Attn. and MLP projections
LoRA rank r	16
LoRA scaling α_{LoRA}	32
LoRA dropout	0.1
Edit steps K	10
Inner LR α	1×10^{-4}
Outer LR β , LEAP	1×10^{-5}
Outer epochs, LEAP	1

Table 7: Hyperparameters for LoRA and LEAP. Both methods use the same LoRA architecture and test-time editing procedure.

2024a). These methods require model-specific hyperparameters and pre-computed covariance statistics; at the time of our experiments, we were unable to locate these pre-computed statistics for Qwen2.5-7B-Instruct via EasyEdit. We therefore omit ROME, MEMIT, and AlphaEdit for Qwen2.5-7B-Instruct in Table 2. For CaKE (Yao et al., 2025b), we use the hyperparameter settings from the released implementation.

LoRA and LEAP. For both the LoRA baseline and LEAP, we use the same LoRA architecture and the same test-time editing procedure. LoRA adapters are attached to all attention and MLP projection layers: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, `down_proj`, and `gate_proj`. The pre-trained backbone parameters are kept fixed, and only the LoRA parameters are updated. For LEAP inner-loop adaptation and for test-time editing, we use AdamW (Loshchilov and Hutter, 2019) to update the LoRA parameters. The LEAP outer-loop meta-update is also optimized with AdamW. Table 7 summarizes the hyperparameters used for both Llama3-8B-Instruct and Qwen2.5-7B-Instruct.

Experimental setup and computational budget. We run all experiments on NVIDIA A100 40GB GPUs. Our implementation is based on PyTorch (Paszke et al., 2019) version 2.9.0 and Hugging Face Transformers (Wolf et al., 2020) version 4.55.4. For LEAP, we perform meta-training once for each base model with a fixed random seed. All reported results, including baselines, are obtained from a single run per configuration. During evaluation, we use deterministic greedy decoding by setting `do_sample=False`.

LEAP meta-training takes about 3.5–4 hours per model on a single A100 40GB GPU.

Manual inspection of the exact-match criterion. Our exact-match criterion follows the same substring-containment approach used in prior work (Yao et al., 2025b). To check for potential inflation, we manually inspected LEAP’s generations on 100 randomly sampled MQuAKE-CF cases (200 generations: cloze and QA each) on Llama3-8B-Instruct. We found 1/200 cases where the model explicitly rejects the gold answer before proposing another (e.g., “X is not correct, the correct answer is...”, where X was in fact gold), which are miscounted as correct under substring matching. We also found 2/200 cases listing multiple candidates without committing to one, which remain valid matches under this criterion. Additionally, we found 4/200 cases where a short gold answer incidentally appeared as a substring within an otherwise unrelated or non-committal generation. This suggests a modest degree of inflation in absolute accuracy; since all methods are evaluated under the same criterion, we expect relative comparisons to be largely unaffected.

C Further Evaluation

C.1 General Capability Evaluation

We evaluate whether the meta-trained LEAP adapters degrade the general capabilities of the base models. Specifically, we compare each base model with the same model equipped with LEAP adapters on GSM8K (Cobbe et al., 2021), MMLU (Hendrycks et al., 2021), CommonsenseQA (Talmor et al., 2019), and BBH (Suzgun et al., 2023). We conduct this evaluation for both Llama3-8B-Instruct and Qwen2.5-7B-Instruct. We use the same evaluation protocol for each base model and its LEAP-equipped counterpart.

Table 8 shows the results. Overall, adding the LEAP adapters does not cause a systematic degradation in general benchmark performance. Scores fluctuate slightly across tasks, with some metrics decreasing modestly and others remaining comparable or improving. This suggests that LEAP does not systematically harm general benchmark performance while improving edit propagation.

C.2 Locality of Edited Models

We evaluate whether editing preserves model behavior on unrelated facts. For each of the 2,999 edited cases in MQuAKE-CF, we sample 3 unrelated held-out cases (8,997 instances in total) and measure whether the edited model still produces

Model	GSM8K	MMLU	CSQA	BBH
Llama	77.94	68.02	77.31	71.76
Llama + LEAP	76.35	67.70	77.40	70.34
Qwen	82.03	71.85	82.56	46.37
Qwen + LEAP	82.71	71.65	82.80	51.25

Table 8: General capability evaluation of the base models and the models equipped with LEAP adapters. Scores are reported on GSM8K, MMLU, CommonsenseQA (CSQA), and BBH.

Method	Instance-wise	Multi-hop
Llama3 (no edit)	77.2	28.1
LoRA	70.9	24.8
CaKE	47.4	18.8
LEAP (ours)	48.4	12.0

Table 9: Locality on MQuAKE-CF: accuracy on unrelated held-out cases after editing, measured on instance-wise and multi-hop (Cloze+QA) queries.

the original, pre-edit answers to their instance-wise and multi-hop queries.

Table 9 reports locality for LoRA, CaKE, and LEAP alongside the unedited model on Llama3-8B-Instruct. All editing methods reduce locality relative to the unedited model, and the drop is most pronounced for methods that explicitly optimize for propagation. While LoRA mostly retains unrelated facts but does not substantially improve multi-hop propagation of new knowledge (Table 2), CaKE and LEAP both show lower retention of unrelated facts alongside stronger multi-hop propagation, suggesting a broader trade-off between propagation and locality. Within this group, LEAP retains instance-wise locality comparable to CaKE, but shows the lowest multi-hop locality among all methods.

We view this trade-off as a genuine limitation of the current formulation of LEAP: the same edit-adaptable parameters that help propagate edits to their intended consequences also appear to make the model more prone to letting edits leak into unrelated multi-hop reasoning. A natural mitigation is to incorporate a locality-aware term into LEAP’s outer-loop objective, e.g., penalizing multi-hop loss increases on unrelated held-out queries during meta-training, which we leave to future work.

C.3 Sequential Editing

We additionally evaluate LEAP under sequential editing on Llama3-8B-Instruct, where multiple edit sets are applied consecutively to the same model. The purpose of this experiment is to test whether the effect of meta-learning remains useful after

Edit Sets	Method	Instance-wise	Multi-hop	
			Cloze	QA
10	CaKE	28.9	18.8	17.8
10	LoRA	41.2	18.6	17.0
10	LEAP	38.7	28.5	26.8
100	CaKE	19.6	9.0	9.4
100	LoRA	27.9	9.7	9.7
100	LEAP	27.4	10.8	10.8

Table 10: Sequential editing results on MQuAKE-CF with Llama3-8B-Instruct. We apply n edit sets consecutively to the same model and evaluate on the corresponding n cases after all edits have been applied.

repeated edit updates, rather than disappearing after a single edit.

We randomly shuffle the MQuAKE-CF cases and split them into groups of n cases. For each group, we apply edits sequentially to the same model for all n cases. After applying n edit sets consecutively, we evaluate the edited model on the same n cases using instance-wise accuracy, multi-hop cloze accuracy, and multi-hop QA accuracy. We report results for $n = 10$ and $n = 100$.

Table 10 shows that performance decreases for all methods as the number of consecutive edits increases from 10 to 100, indicating that long sequences of edits remain challenging. Nevertheless, LEAP achieves the best multi-hop cloze and QA accuracy in both settings, though its instance-wise accuracy trails LoRA at both $n = 10$ and $n = 100$. For 10 consecutive edits, LEAP substantially improves multi-hop QA accuracy over LoRA and CaKE. For 100 consecutive edits, LEAP still maintains higher multi-hop accuracy than the baselines. These results suggest that the meta-learned parameters remain beneficial under repeated editing and that the effect of meta-learning does not disappear after a single edit.

D Responsible Use and Artifact Documentation

Artifact use and intended use. We use existing models and datasets only for research purposes and in accordance with their licenses, terms of use, and intended access conditions. Llama3-8B-Instruct and Qwen2.5-7B-Instruct are used as base models for controlled knowledge-editing experiments. MQuAKE is used both for benchmarking edit propagation and for identifying Wikidata entity and property IDs to exclude when constructing synthetic meta-training tasks. GSM8K, MMLU, CommonsenseQA, and BBH are used only for benchmarking model behavior. Our constructed

meta-training data are intended for research on knowledge edit propagation and should not be used to make claims about real individuals or real-world factual updates. We release our code and, where permitted, the constructed data at <https://github.com/mynlp/LEAP> for research use, subject to the licenses and terms of the underlying resources.

Personal information and offensive content.

Our synthetic meta-training tasks are constructed from Wikidata-derived relation chains with fictional edited prefixes. The edited subject entities are generated as random pronounceable nonce names rather than sampled from real people or existing entity lists. These names are not intended to refer to real individuals and contain no deliberately collected personal identifiers. The Wikidata-derived suffixes use public knowledge graph triples whose properties are manually selected for factual relation-chain construction. We do not collect private personal data or user-generated content when constructing these tasks, and we do not intentionally include offensive content in the constructed data.

Artifact documentation. We document the models, datasets, data construction procedure, source resources, and evaluation settings in Section 5 and Appendix A. All experiments are conducted in English. Our constructed meta-training data consist of synthetic multi-hop editing tasks with fictional edited prefixes and, when present, Wikidata-derived suffixes that are disjoint from MQuAKE. The artifacts are intended for research on knowledge edit propagation and focus on factual relation-chain reasoning, not on personal profiling or demographic inference.