RIPPLECOT: Amplifying Ripple Effect of Knowledge Editing in Language Models via Chain-of-Thought In-Context Learning

Zihao Zhao[1], Yuchen Yang[1], Yijiang Li[2], Yinzhi Cao[1]
[1]Johns Hopkins University
[2]University of California San Diego
{zzhao71, yc.yang, yinzhi.cao}@jhu.edu, yijiangli@ucsd.edu

Abstract

The ripple effect poses a significant challenge in knowledge editing for large language models. Namely, when a single fact is edited, the model struggles to accurately update the related facts in a sequence, which is evaluated by multi-hop questions linked to a chain of related facts. Recent strategies have moved away from traditional parameter updates to more flexible, less computation-intensive methods, proven to be more effective in the ripple effect. In-context learning (ICL) editing uses a simple demonstration `Imagine that + new fact` to guide LLMs, but struggles with complex multi-hop questions as the new fact alone fails to specify the chain of facts involved in such scenarios. Besides, memory-based editing maintains additional storage for all edits and related facts, requiring continuous updates to stay effective. As a result of the design limitations, the challenge remains, with the highest accuracy being only 33.8% on the MQUAKE-CF benchmarks for Vicuna-7B. To address this, we propose RIPPLECOT, a novel ICL editing approach integrating Chain-of-Thought (COT) reasoning. RIPPLECOT structures demonstrations as {new fact, question, thought, answer}, incorporating a *thought* component to identify and decompose the multi-hop logic within questions. This approach effectively guides the model through complex multi-hop questions with chains of related facts. Comprehensive experiments demonstrate that RIPPLECOT significantly outperforms the state-of-the-art on the ripple effect, achieving accuracy gains ranging from 7.8% to 87.1%. RIPPLECOT is open-source and available at `https://github.com/zzhao71/RippleCOT`

1 Introduction

As large language models (LLMs) become more prevalent in various sectors, their limitations, such as storing inaccurate or sensitive knowledge, pose growing concerns (Dhingra et al., 2022; Carlini et al., 2021; Wolf et al., 2019). This has led to the development of knowledge editing methods aimed at updating the facts. The ripple effect represents a significant challenge in knowledge editing for LLMs that was not explored until very recently (Cohen et al., 2023). When one fact is edited in a model, the ripple effect refers to the chain of related facts that should be updated following the edited one, which is evaluated by the multi-hop questions (Zhong et al., 2023) linked to a chain of facts. Figure 1 illustrates an example of the multi-hop question: if we modify the author of Misery to Richard Dawkins, the related multi-hop facts, such as the citizenship of the author of Misery, should also be updated.

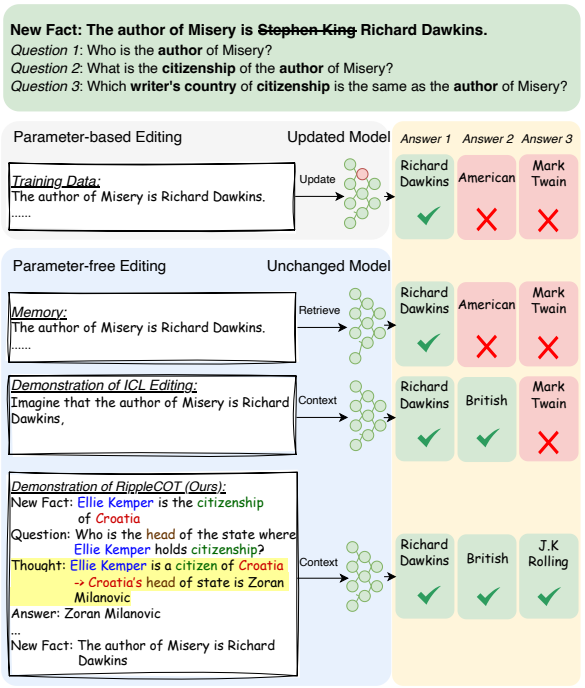

Figure 1: An illustration of RIPPLECOT and existing parameter-based and parameter-free knowledge editing methods addressing the ripple effect via multi-hop questions.

Conventional parameter-based editing methods,

such as fine-tuning (Zhu et al., 2020) or matrix computation (Meng et al., 2022b), update the model's parameters to recall specific edited facts effectively but risk catastrophic forgetting (Zheng et al., 2023) and were proven failure on the ripple effect (Cohen et al., 2023). The edits are limited to the facts within the training data and struggle with related but untrained facts. Recent parameter-free editing approaches, like memory-based editing (Mitchell et al., 2022), also face challenges when related facts fall outside the maintained memory's scope. In-context learning (ICL) editing uses the simple prompt `Imagine that + new fact` to help the model recall the new fact. However, it struggles with complex, multi-hop questions because the new fact alone does not specify the chain of facts within such scenarios. Due to those limitations, the best result (Zhong et al., 2023) achieves only a 33.8% accuracy on the MQuAKE-CF benchmarks with Vicuna-7B model, with other methods performing around 15%.

To address this, we propose to integrate COT reasoning into the ICL framework, guiding LLMs to process multi-hop questions sequentially. While we observe the direct use of a "Think step by step" COT prompt improving performance, it falls short in open-source models with limited reasoning capacities. To arrive at a better solution, we develop RippleCOT, by structuring the demonstration to (`new fact`, `question`, `thought`, `answer`). RippleCOT operates in two stages: demonstration generation and refinement. During generation, RippleCOT identifies multiple relationships and missing items within the defined fact triplets (s, r, o) (Cohen et al., 2023), i.e., subject, relation, and object. For example, in Figure 1, the question involves multiple relations: `citizenship` and `head`. RippleCOT decomposes the questions into (`Ellie Kemper`, `citizenship`, ?) and (?, `head`, ?*). Given the new fact (`Ellie Kemper`, `citizenship`, `Croatia`), the ? is identified as `Croatia`. The `thought` then becomes *Ellie Kemper is a citizen of Croatia* $\rightarrow$ *Croatia's head of state is Zoran Milanovic*. This effectively triggers the COT reasoning ability of the LLMs, significantly improving the ripple effect for more complex questions. In the refinement stage, RippleCOT selects the top-k candidates among generated (`new fact, question, thought, answer`) pairs whose questions have the highest cosine similarity with the task question.

Beyond COT reasoning ability, RippleCOT offers several advantages. First, RippleCOT operates without altering model parameters, resulting in lower computational costs and enabling efficient adaptation to multi-hop questions for a single edit. It also supports multiple editing scenarios, which have been less explored by the literature, such as accurately updating the related facts if changing the President of the United States from Obama to Trump and then to Biden. Second, our COT demonstration generation is automatic and highly flexible, tailored to task-specific questions. We have designed and evaluated multiple methods for generating demonstrations, including human selection from benchmark datasets along with the existing approaches, few-shot generation using GPT-4o (Achiam et al., 2023) based on selected references, and zero-shot generation with GPT-4o, identifying the optimal approach for enhancing ripple effects. Additionally, RippleCOT can be integrated with existing knowledge editing methods to further improve ripple effect performance.

In summary, this paper has three main contributions:

- We propose a novel ICL knowledge editing framework with automatic COT demonstration generation and refinement, namely RippleCOT.
- We explore different ways for generating COT demonstration, namely full-shot selection, few-shot generation, and zero-shot generation
- RippleCOT significantly improves accuracy, ranging from 7.8% to 87.1%, in addressing ripple effects on multi-hop questions, as demonstrated on the RippleEdit (Cohen et al., 2023) and MQuAKE (Zhong et al., 2023) datasets.

2 Related Work

2.1 Knowledge Editing

Knowledge editing methods include parameter-based methods (Mitchell et al., 2021; Meng et al., 2022a; Dong et al., 2022) and parameter-free methods (Zheng et al., 2023; Zhong et al., 2023; Wang et al., 2024; Chen et al., 2024). In parameter-based editing methods, Fintuning (Zhu et al., 2020) uses gradient descent to update model parameters based on the edit; MEND (Mitchell et al., 2021) introduces hyper-networks that convert the gradients to model parameter changes; ROME (Meng et al., 2022a) introduces causal tracing that locates and updates the parameters responsible for factual associations. However, these methods can result in catastrophic forgetting of previously learned

knowledge, perform poorly at generalizing edits to related facts, and are computationally expensive. Parameter-free knowledge editing methods, mainly ICL editing (Zheng et al., 2023), utilize demonstration to prompt the model to generate outputs aligned with the injected knowledge. Later in (Cohen et al., 2023), simply prompting the model with the edited fact without demonstration can achieve better performance than parameter-based methods. Combined with retrieval-augmented methods, recent parameter-free methods such as EREN (Chen et al., 2024) and MeLLo (Zhong et al., 2023) achieve SOTA in multiple metrics(Chen et al., 2024; Zhong et al., 2023).

2.2 Ripple Effect

A common under-addressed problem of all knowledge editing methods is the propagation of knowledge updates to other logically connected facts, which is referred to as the ripple effect. Cohen et al. (Cohen et al., 2023) first define and categorize the ripple effects into logical generalization, compositionality I & II, subject aliasing, preservation, and relation specificity, each denoting a different logical pattern. Compositionality I & II involves two-hop questions, in which the models perform the worst. This observation is recapitulated by Zhong et al. (Zhong et al., 2023), where the performance of the edited model is evaluated by 2,3,4-hop questions. Overall, the knowledge editing performance decreases as the intermediate logical steps increase.

3 Problem Formulation

Knowledge editing aims to update the fact triplet from (s, r, o) to (s, r, o^*), where s is the subject, r the relation, o the original object and o^* the new object. These triplets are formulated (Petroni et al., 2018) as prompt templates $p(s, r, \emptyset)$—for example, with $s =$ `Stephen King`, $r =$ `Citizenship` and $\emptyset$ is a place holder for the object, the prompt is: *The citizenship of Stephen King is _*. Using a language model $f : \mathcal{X} \rightarrow \mathcal{Y}$, which processes input prompt $x \in \mathcal{X}$ to generate output $y \in \mathcal{Y}$, we probe the model with $p(s, r, \emptyset)$. The output $f(p(s, r, \emptyset))$ should match the original object o, such as *American*. After editing, the model f^* should return $f^*(p(s, r, \emptyset))$, matching the updated object o^*, such as *British*.

3.1 In-Context Knowledge Editing

Given a new fact triplet (s, r, o^*), In-context knowledge editing injects it via an input prompt starting with the prefix *"Imagine that"* (Cohen et al., 2023), following the template $p(s, r, o^*)$ We denoted this new fact injection as $e(s, r, o^*) =$ "`Imagine that`" $+ p(s, r, o^*)$. The edited model is then defined as $f* = f \circ (e(s, r, o^*))$. To verify the edit, we query the edited model with $f^*(p(s, r, \emptyset))$ and check if it successfully recalls o^*.

3.2 Ripple Effect with Multi-hop Questions

The ripple effect is assessed through multi-hop questions (Zhong et al., 2023). Imagine a chain of facts $\mathcal{Q} = \{(s_1, r_1, o_1), \ldots, (s_n, r_n, o_n)\}$, where each object o_i serves as the subject s_{i+1} in the subsequent fact. We refer to the set of relations as $\mathcal{R} = \{r_1, \ldots, r_n\}$ and the set of subjects as $\mathcal{S} = \{s_1, \ldots, s_n\}$. The multi-hop questions are formulated using $\mathcal{Q}$ that begins with the head entity s_1 till the r_n, and the answer is the tail entity o_n. For instance, consider the question *What is the citizenship of the author of Misery?* composed of the fact chains {`Misery`, `Author`, `Stephen King`} and {`Stephen King`, `Citizen`, `American`}. If we update the first fact to {`Misery`, `Author`, `Ellie Kemper`}, the edited model f^* is validated by checking if the response for the above question is `British` instead of `American`, reflecting the related fact {`Ellie Kemper`, `Citizen`, `British`}.

4 RippleCOT

As discussed above, knowledge editing faces the challenge of ripple effects where a sequence of related facts should also be updated to arrive to the correct answer to some particular question. This chain of related facts for knowledge editing resembles a chain of thoughts in reasoning which motivates us to integrate CoT reasoning into the ICL pipeline and propose RippleCOT as a unified solution.

4.1 RippleCOT Formulation

For each knowledge editing, we construct k demonstrations $\mathcal{D} = \{d_1, \ldots, d_k\}$, which $d_i \in \mathcal{D}$ consist of four main components: new fact, question, thought, and answer.

New facts. (s_1, r_1, o_1^*), which are examples of information designated for edits.

Questions. Multi-hop questions $p(\mathcal{S}, \mathcal{R}, \emptyset)$ with $s \in \mathcal{S}$ and $r \in \mathcal{R}$, which are formulated to probe the related facts following the new facts, thereby assessing the ripple effects of edits.

Thoughts. Break down the questions according to each relation $r_1, \ldots, r_n \in R$ as follows:

$$\begin{cases} \{s_1, r_1, o_1^*\} \\ \{s_2^* = o_1^*, r_2, o_2^*\} \\ \vdots \\ \{s_{n-1}^* = o_{n-1}^*, r_{n-1}, o_n^*\} \end{cases}$$

Answers. o_n^*, which provides the exact answer to the question, derived from the logical reasoning in the thoughts section.

The goal of the demonstration is to allow the model to generate the correct answer through its COT reasoning ability, establishing clear connections between the new facts and their related facts to the final answers.

4.2 Demonstration Generation

To generate k demonstrations $\mathcal{D} = \{d_1, \ldots, d_k\}$ with the COT formulation, we explore three approaches:

Full-shot Selection. $\forall d_i \in \mathcal{D}$, the {new fact, question, thought, answer} set is randomly selected from the MQuAKE benchmark. This ensures high-quality, logically coherent contexts that serve as reliable examples for the model.

Few-shot Generation. To generalize the demonstration beyond the scope of MQuAKE, RippleCOT first creates a reference set $\mathcal{D}_{\texttt{refer}}$ containing a few demonstrations using the human selection. This reference set is then used to guide LLMs, which have shown remarkable reasoning and instruction-following ability, in generating demonstrations with a similar format. We evaluate both GPT-4o and GPT-J generated demonstrations, the prompt is as follows:

```
Your task is to genereate knowledge editing examples
      for in context learning.
You need to first generate the knowledge being
      edited (fact being changed) and then ask a
      question that requires multi-hop (multi-step)
      reasoning. Finally you need to provide a
      answer with step-by-step reasoning in concise
      format.

Example: D_refer

Please respond in the following format without any
      markdown.
New Fact: <knowledge being editted>
Question: <question that requires multi-step
      reasoning>
Thought: <step-by-step reasoning in concise format>
Answer: <answer with step-by-step reasoning in
      concise format>

Please generate {k} knowledge editing examples.
      Please respond only the generated examples in
      the above format without any markdown or
      additional text.
```

Zero-shot Generation RippleCOT explore the Zero-shot Generation ability (Ramesh et al., 2021) of LLMs to directly generate examples following specific formats. Specifically, we remove the reference set $\mathcal{D}_{\texttt{refer}}$, using only the COT format introduced in Section 4.1. We use both GPT-4o and GPT-J to generate the demonstrations using the above prompt, omitting the line "`Example:` $\mathcal{D}_{\texttt{refer}}$".

4.3 Demonstration Refinement

After demonstration generation, we refine the demonstration by ordering it by the similarity (Lu et al., 2021) between the `question` components in the $\mathcal{D}$, denoted as $\{q_{\texttt{demo}}\}_{i=1}^k$, and the question that we want the model to answer, denoted by $\{q_{\texttt{target}}\}_{i=1}^k$. We follow Liu et al. (Liu et al., 2021) to use the all-MiniLM-L6-v2 (Wang et al., 2020) to get embeddings (Reimers and Gurevych, 2019) denoted as $\{\mathcal{E}(q_{\texttt{demo}})\}_{i=1}^k$ and $\{\mathcal{E}(q_{\texttt{target}})\}_{i=1}^k$ respectively. The similarity $\{m\}_i^k$ is calculated by the cosine similarity (Huang et al., 2008) with each m_i:

$$m_i = \frac{\mathcal{E}(q_{\texttt{demo}})_i \cdot \mathcal{E}(q_{\texttt{target}})_i}{\sqrt{||\mathcal{E}(q_{\texttt{demo}})_i||_2 \cdot ||\mathcal{E}(q_{\texttt{target}})_i||_2}} \quad (1)$$

Then, RippleCOT select the top-t demonstrations from $\{q_{\texttt{demo}}\}_{i=1}^k$ with with the highest $\{m\}_i^k$. This approach ensures that the most relevant demonstrations are selected, thereby improving the overall performance of the model.

5 Experiments

5.1 Experiment Setup

We primarily assess our method using the MQuAKE (Zhong et al., 2023) and RippleEdit (Cohen et al., 2023) dataset with the models GPT-J (6B) (Wang and Komatsuzaki, 2021), Vicuna-7B (Zheng et al., 2024), and GPT-3 (Brown et al., 2020; Ouyang et al., 2022). We adopt the accuracy metric from previous work (Zhong et al., 2023; Cohen et al., 2023), where an answer is deemed correct if the model's output contains the expected answer. We set our *default* setting as the full-shot selection with $k = 5$ demonstrations.

5.1.1 Dataset

RippleEdit. This dataset contains counterfactual

knowledge editing examples (Meng et al., 2022a). It is divided into three different subsets. The popular subset contains edits on popular entities in wiki data (Vrandečić and Krötzsch, 2014); the random subset contains random entities; the recent subset contains recently added entities. We primarily focus on the popular subset.

MQuAKE. This dataset is used to test the edited model's multi-hop question-answering ability, which contains 2,3,4-hop questions.

5.1.2 Baseline

We conduct a comparative analysis of RippleCOT against several established techniques: Fine-tuning (FT) (Zhu et al., 2020), MEND (Mitchell et al., 2021), ROME (Meng et al., 2022b), DeepEdit (Wang et al., 2024), IKE (Cohen et al., 2023) and MEMIT (Meng et al., 2022a), as well as our proposed BaseCOT, which adds a "Think step by step" prompt (Kojima et al., 2022) after the question, as the baseline method for the RippleCOT approach.

FT: FT employs gradient descent to update model parameters based on the edits, directly modifying the weights to reflect the new information.

MEND: MEND trains a hypernetwork to transform raw fine-tuning gradients based on an edited fact, creating targeted weight updates to integrate new factual content.

ROME: ROME identifies and localizes factual knowledge within specific Transformer layers, then updates the feedforward networks in those layers to incorporate new facts.

MeLLo: MeLLo stores edited facts externally. During runtime, related facts are retrieved, and conflict detection ensures appropriate edited outputs.

MEMIT: MEMIT extends ROME by enabling simultaneous editing of a large set of facts. It updates feedforward networks across multiple layers, effectively encoding a broader range of factual information.

DeepEdit: This method views knowledge editing as a constrained decoding problem, ensuring outputs meet the proposed semantic constraints. DeepEdit uses a depth-first search-based progressive decoding technique for efficient updates without retraining.

IKE: The ICL editing approach with a demonstration as *"Imagine that" + new fact*.

5.2 Comparison with Baselines

To mimic the human-written chain-of-thought context, we extract new facts, questions, thoughts, and answers from multi-hop questions in the MQuAKE dataset. Each question in this dataset is a 2-hop, 3-hop, or 4-hop question. We combine each sub-question and its corresponding answer into a single sentence to form the thought process. Following the approach of Zhong et al. (Zhong et al., 2023), we post three similar questions. If one of these questions yields an accurate answer, we consider the model to have successfully edited the new facts. The default number of contexts used is five. The results are presented in Table 1.

Our model shows much better performance compared to previously proposed models in all three datasets. Furthermore, compared to BaseCOT, our method still shows better performance, which reinforces that our method helps improve the model's reasoning ability and amplifies the ripple effects.

5.3 Performance on One-time Edit

The number of edited instances. Following the methodology of (Zhong et al., 2023), we split the dataset into groups of g instances, where g values are 1, 100, 1000, and 3000. For a higher number of edited facts, RippleCOT introduces a dynamic retrieval method based on similarity measures between the thought and the stored new facts. The evaluation prompt becomes:

```
[5-shot demonstrations]
[New facts: m facts line by line
retrieved from the given 3000 facts]
[Question]
```

The m new facts are selected based on their similarity to the generated thoughts.

The details of the dynamic retrieval process are as follows. Note that m is not fixed, because a single question may relate to multiple edited facts. For example, for the question, "What is the capital of the country to which Lou Pearlman belonged?", the relevant facts might be "Lou Pearlman is a citizen of India" and "The capital of India is Taloga." To address this, RippleCOT retrieves up to m rounds with one fact per round and employs an early stopping criterion if no contradiction is detected during a self-check. The setting of self-check follows (Zhong et al., 2023) and (Wang et al., 2024) for identifying contradictions between the retrieved facts and the answer. For each retrieval

Model	Data	Parameter-based Methods				Parameter-free Methods			
		MEND	FT	ROME	MENIT	MELLO	IKE	BaseCOT	RIPPLECOT
GPT-J	MQUAKE-CF	11.5	1.9	18.1	12.3	41.2	67.2	59.3	**81.7**
	MQUAKE-T	38.2	0.2	11.3	4.8	46.8	71.8	66.9	**83.2**
	RIPPLEEDIT	-	-	48.4	50.2	-	26.1	30.2	**47.5**
Vicuna-7B	MQUAKE-CF	8.4	0.2	12.2	9.0	33.8	20.6	20.0	**87.3**
	MQUAKE-T	33.9	8.2	9.3	5.8	51.3	30.7	39.7	**78.9**
	RIPPLEEDIT	-	-	68.7	59.3	-	82.0	43.2	**89.8**

Table 1: Comparison between existing knowledge editing methods with RIPPLECOT and in MQUAKE and RIPPLEEDIT dataset. The number represents the accuracy (%) in answering the questions after the model is edited.

Model	Method	# Edit Instances			
		1	100	1000	3000
GPT-J	MEMIT	12.3	9.8	8.1	1.8
	MEND	11.5	9.1	4.3	3.5
	MeLLo	20.3	12.5	10.4	9.8
	RIPPLECOT	**80.1**	**79.3**	**78.1**	**79.9**
Vicuna-7B	MeLLo	20.3	11.9	11.0	10.2
	RIPPLECOT	**87.3**	**82.8**	**80.8**	**85.7**
GPT-3	MeLLo	68.7	50.5	43.6	41.2
	RIPPLECOT	**89.7**	**88.6**	**82.7**	**80.9**

Table 2: Performance of RIPPLECOT and baselines with GPT-J, Vicuna-7B. We evaluate the number of edited instances once as $1, 100, 1000, 3000$ on MQUAKE-CF. We include the best results reported by the baselines for comparison.

Method	2-hop	3-hop	4-hop	All
FT	3.7	1.4	0.5	1.9
MEND	13.9	11.3	9.5	11.5
ROME	33.8	9.1	11.4	18.1
MEMIT	22.5	6.0	8.4	12.3
MeLLo	47.5	27.2	45.3	42.1
RIPPLECOT	**80.2**	**78.8**	**79.2**	**80.1**

Table 3: Performance of RIPPLECOT and baselines with GPT-J. We evaluate the number of hop-in questions from MQUAKE-CF as $2, 3, 4$, and all are referred to as "All". We include the best results reported by the baselines for comparison.

round, RIPPLECOT selects one fact from the remaining stored new facts that are most similar to the generated thought, and append it to the [New facts] prompt. This novel dynamic retrieval mechanism is also a key contribution of RIPPLECOT in enhancing the retrieval process. The results are presented in Table 2.

Parameter-based methods face a large decline when the edit instances increase because it is very hard to update parameters for editing numerous instances. Additionally, retrieval accuracy becomes low when the number of edited instances increases, causing the accuracy of related edited questions to drop. However, our method focuses on teaching the model to think with the provided logic, so our method does not decline when the number of edited instances increases, demonstrating its potential for handling a large number of edits.

The number of hops. The dataset contains examples for 2-hop, 3-hop, and 4-hop questions. For multi-hop questions, we adhere to the previous prompt standard and concatenate these sentences to form the thoughts. This method assesses the model's ability to apply learned facts. As the number of hops increases, the model must utilize all learned new facts and apply logical reasoning to generalize the question. Table 3 demonstrates that typically, as the number of hops increases from 2 to 4, accuracy decreases. However, in RIPPLECOT, this decline is minimized compared to other methods, indicating our method's superior capability in enabling the model to apply new facts effectively.

5.4 Performance in medical applications

We conducted experiments on the MedCF dataset (Xu et al., 2024), a benchmark for medical question-answer tasks. We follow Xu et al. (Xu et al., 2024) to evaluate the Meditron-7B model. The Table 4 shows the applicability of RIPPLECOT to knowledge editing in the medical domain. Unlike BaseCOT which relies heavily on the model's reasoning ability, RIPPLECOT tailors the thought process for knowledge editing, effectively decomposing multi-hop logic in questions.

5.5 Performance on Multi-time Edit

Previously, to our knowledge, all methods have evaluated knowledge editing using multi-hop ques-

	BaseCOT	RIPPLECOT
Meditron (7B)	65.3	99.9

Table 4: We compare RIPPLECOT with BaseCOT on the MedCF dataset using the Meditron-7B model.

	Mello	Mello after M.E.	RIPPLECOT	RIPPLECOT after M.E.
GPT-J	41.2	8.9	81.7	79.9

Table 5: We compare the performance across different numbers of demonstrations using the GPT-J and Vicuna models, evaluated on the MQUAKE-CF dataset. M.E. stands for Multiple Editing.

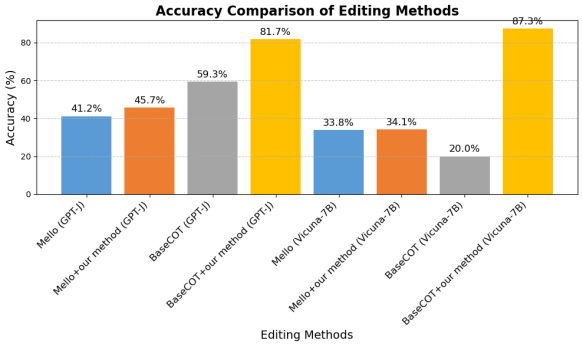

Figure 2: Comparative analysis of the performance enhancements achieved by our proposed method when applied to two baseline models, Mello and BaseCOT, in both the GPT-J and Vicuna-7B architectures.

tions or simple question-answering (Wang, 2022) under one-time editing. However, in real life, it is common to update knowledge multiple times. For instance, in the context of presidential elections, the president of the United States changes every 4 or 8 years, necessitating repeated updates to this knowledge. Due to the lack of datasets evaluating this aspect, we decided to modify the answers twice. For example, as shown in Figure 1, we change the author of Misery from Stephen King to Richard Dawkins, and then to a new author, Ernest Hemingway. We applied similar changes to a total of randomly selected 200 datasets by altering the answers to multi-hop questions to mimic multiple edits in real life.

Table 5 shows the number of edits significantly impacts retrieval-based methods, which struggle with these kinds of problems. When conflicting knowledge is injected into the knowledge base, retrieval accuracy decreases, leading to lower accuracy for multi-hop questions. However, our method, RIPPLECOT, is less affected by multiple edits. The performance of RIPPLECOT remains relatively stable even with multiple edits, provided the demonstrations do not change.

5.6 Combination with Baselines

In this study, we integrate our proposed method into existing approaches and baselines to evaluate its effectiveness in improving performance. The performance enhancements observed, as illustrated in Figure 2 , demonstrate the significant impact of our method in amplifying ripple effects.

5.7 Ablation Study

In this section, we ablate on different components of RIPPLECOT, which are generation strategy, number of referenced demonstrations used for the few-shot generation, and number of demonstrations selected for the full-shot selection.

5.7.1 Demonstration Generation

Generation Strategy. Table 6 indicates that few-shot generation may significantly amplify ripple effects. The comparable performance between few-shot generation and human selection, particularly with GPT-4o's few-shot performance surpassing that of human selection (human-written chain-of-thought content), suggests a viable alternative for automatic context generation. We utilize GPT-4o and GPT-J (6B) models to generate demonstrations using both few-shot and zero-shot generation (Levy et al., 2017). For zero-shot learning, we input the prompt directly, allowing the model to autonomously generate the result. For few-shot learning, we extract several contexts from the MQUAKE dataset, which includes new facts, questions, thoughts, and answers, and prompt the model to generate the context in a similar format. The generated content is then used as context for RIPPLECOT. We impose several criteria on the generated context: it must comprehensively include all four sections (facts, questions, thoughts, and answers), and the answer should be concise, ideally a term or a few words. The similarity between GPT-J and GPT-4o generation suggests that performance is optimal when provided with several high-quality examples. However, in the zero-shot scenario, GPT-4o may still outperform due to its superior capability in generating reasonable chain-of-thought prompts.

How many referenced demonstrations are needed for the few-shot generation? We try to

	Zero-shot Generation	Few-shot Generation	Full-shot Selection
GPT-J	69.5	80.8	**81.7**
ChatGPT-3.5	74.2	81.2	
ChatGPT-4	80.8	81.9	
GPT-4o	72.7	82.3	

Table 6: Comparison between different demonstration generation. We use the demonstration generated by GPT-J and GPT-4o for zero-shot and few-shot generation. The result is on MQuAKE-CF dataset.

answer this question in Table 7. By alternating the different number of references given to the LLMs to few-shot generate demonstrations, we observe that with more references, performance steadily grows to the best. We also want to emphasize that a competitive performance can be achieved using as little as 1 reference, which illustrates that RIPPLECOT can work well even with little reference selected from MQuAKE and thus can be generalized to other datasets.

# References	1	2	3	4	5
ChatGPT-3.5	68.9	80.2	78.6	79.3	81.2
ChatGPT-4	78.7	81.0	81.0	78.0	81.9
GPT4-o	80.8	67.9	70.5	71.0	82.3

Table 7: We compare the performance of the few-shot demonstration given different numbers of human reference examples. The result is on MQuAKE-CF dataset.

How many demonstrations are needed to tackle the ripple effect in knowledge editing? We answer this question in Table 8. We experimented with using 1, 2, 5, 10, and 20 demonstrations to evaluate the performance. As the number of contexts increases, computational complexity also rises. Our goal is to identify an optimal number of demonstrations that elicit high-quality model outputs. As shown in Table 8, using 5 demonstrations appears to be an effective choice. While using 20 demonstrations offers a slight improvement in accuracy, considering the trade-off between accuracy gains and the cost of time and computational resources, 5 demonstrations represent a practical and efficient choice.

5.7.2 Demonstration Refinement

As discussed in our methodology, we employ cosine similarity to select demonstrations. During the generation process, we initially generate 20 candidate contexts and then select 5 of these based on cosine similarity. This approach results in approximately a 5% performance increase compared to random selection, underscoring the importance of incorporating cosine similarity into our method.

# Demonstrations	1	2	5	10	20
GPT-J	69.5	75.8	81.7	80.9	82.0
Vicuna-7B	51.9	62.8	87.3	85.8	88.4

Table 8: compare the performance across different numbers of demonstrations using the GPT-J and Vicuna-7B models, evaluated on the MQuAKE-CF dataset.

6 Conclusion

RIPPLECOT has demonstrated superior performance relative to existing approaches. Through our experiments, we have highlighted the importance of our method in amplifying the ripple effect, as well as its flexibility in integration with other existing methods. Additionally, our analysis of chain-of-thought generation provides valuable insights for automatic generation. By combining the inherent flexibility and improved efficacy of in-context editing, our method can significantly streamline the process of knowledge updating, facilitating more accurate and contextually relevant model responses across various domains.

Limitations

Our study, while promising, has several notable limitations that should be addressed in future work:

- **Limited Dataset Scope.** There are limited benchmarks for analyzing ripple effects, especially for multiple edits. We conducted experiments on only two datasets. We hope that, in the future, a larger dataset will be developed, encompassing various scenarios such as questions related to several parallel facts, to enable a more comprehensive evaluation.
- **Assumption of LLM Capabilities.** Our approach assumes that the employed LLMs possess sufficient capabilities to handle knowledge editing and chain-of-thought (CoT) reasoning. However, if sub-optimal LLMs are used, the effectiveness of the proposed methods may be compromised, leading to diminished overall performance.
- **Bias in Edits.** The creation of multiple edits to simulate real-life scenarios may inadvertently introduce biases. These biases might not accurately reflect the complexity and variability of natural

knowledge updates. It is crucial to develop more objective and systematic methods for generating edits to ensure the authenticity and relevance of the scenarios used in experiments.

Addressing these limitations will be vital for advancing the field of knowledge editing and improving the effectiveness and reliability of methods like RIPPLECOT in real-world applications.

Potential Negative Social Impact

Our commitment to ethical research practices guided our methodology and implementation throughout the study, however, RIPPLECOT may raise the following negative impacts:

Firstly, we acknowledge the importance of ensuring the accuracy and integrity of information in language models. The ability to edit knowledge within these models must be approached with caution to prevent the propagation of misinformation. This means that our approach may be maliciously employed to distort, manipulate, or propagate misinformation. We raise this potential negative social impact here to highlight the need for stringent safeguards and monitoring mechanisms. Researchers and practitioners utilizing RIPPLECOT must implement robust verification processes to ensure that only accurate and verified information is introduced into language models.

Secondly, we are aware of the potential biases that may be introduced through manual edits and the limitations of the datasets used. This means that using LLMs or the limited current datasets on knowledge editing might inherit the biases in LLMs or the current datasets.

By addressing these potential negative social impacts, we aim to contribute to the responsible advancement of knowledge editing technologies, ensuring they are used to enhance the reliability and effectiveness of language models in various applications.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments and feedback. This work was supported in part by Johns Hopkins University Institute for Assured Autonomy (IAA) with grants 80052272 and 80052273, National Science Foundation (NSF) under grants CNS-21-31859, CNS-21-12562, CNS-19-37786, CNS-19-37787, and CNS-18-54000, as well as Army Research Office (ARO) under grant No. W911NF2110182. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, ARO, or JHU-IAA.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Yingfa Chen, Zhengyan Zhang, Xu Han, Chaojun Xiao, Zhiyuan Liu, Chen Chen, Kuai Li, Tao Yang, and Maosong Sun. 2024. Robust and scalable model editing for large language models. *arXiv preprint arXiv:2403.17431*.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2023. Evaluating the ripple effects of knowledge editing in language models. *arXiv preprint arXiv:2307.12976*.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. *arXiv preprint arXiv:2210.03329*.

Anna Huang et al. 2008. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, volume 4, pages 9–56.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2023. Catastrophic jailbreak of open-source llms via exploiting generation. *arXiv preprint arXiv:2310.06987*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

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

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2018. 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)*.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Jinyuan Wang, Junlong Li, and Hai Zhao. 2023. Self-prompted chain-of-thought on large language models for open-domain multi-hop reasoning. *arXiv preprint arXiv:2310.13552*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

Zhen Wang. 2022. Modern question answering datasets and benchmarks: A survey. *arXiv preprint arXiv:2206.15030*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Derong Xu, Ziheng Zhang, Zhihong Zhu, Zhenxi Lin, Qidong Liu, Xian Wu, Tong Xu, Xiangyu Zhao, Yefeng Zheng, and Enhong Chen. 2024. Editing factual knowledge and explanatory ability of medical large language models. *arXiv preprint arXiv:2402.18099*.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

A Appendix

A.1 Example Prompts and Generation Results

```
{Demonstrations}
New Fact: Lou Pearlman is a citizen of India, The
      capital of India is Taloga
Question: What is the capital of the country to
      which Lou Pearlman belonged?

New fact: the author of Misery is Richard Dawkins.
Question: What is the nationality of the author of
      Misery.
Thought: The author of Misery is Richard Dawkins.
      Richard Dawkins is a citizen of United Kingdom
      . Therefore, the nationality of the author of
      Misery is British.
Answer: British

New fact: The capital of United States of America is
       El Campu.
Question: What is the capital city of the country
      that Michael Feinstein is a citizen of ?
Thought: Michael Feinstein is a citizen of United
      States of America. The capital of United
      States of America is El Campu. Thus, the
      capital city of the country that Michael
      Feinstein is a citizen of is El Campu.
Answer: El Campu

Thought: Lou Pearlman is a citizen of India. The
      capital of the country of which Lou Pearlman
      is a citizen is Taloga.
Answer: Taloga
```

A.2 Comparison of Different CoT Methods

Our evaluation in Table 9 also shows that RIPPLECOT outperforms the standard CoT approach, Base-COT with "think step by step," as well as other advanced CoT methods, i.e., Self-generated-COT (Wang et al., 2023) that prompts the model to split a complex question into several sub-questions, while Least-to-most-COT (Zhou et al., 2022) let the model generate and arrange the sub-questions from easy to hard. We also compare more advanced prompting such as Self-consistency (Wang et al., 2022) by generating several candidates and performing majority voting. RippleCOT also outperforms plain Self-consistency, demonstrating the significance of CoT in knowledge editing. Self-consistency essentially is self-ensembling which be combined with RippleCOT. We demonstrate that RippleCOT can be further boosted by self-consistency, as shown in the Table 9.

Method	SG-COT	LtM-COT	SC	RIPPLECOT	RIPPLECOT +SC
GPT-J	54.1	74.9	34.2	87.3	90.1

Table 9: Comparison of different Chain-of-Thought (CoT) methods. SG-COT: Self-Generated CoT, LtM-COT: Least-to-Most CoT, SC: Self-Consistency, RIPPLECOT: Your system.

Method	GPT-3 (175B)	Claude-3.5 sonnet	GPT-4o	GPT-4-0125
BaseCoT	72.3	75.8	79.1	79.1
RippleCoT	89.7	85.2	87.9	88.6

Table 10: Performance comparison between BaseCoT and RippleCoT across different models.

The results indicate that RippleCOT is well-customized for knowledge editing. This is because RippleCOT creates thoughts that break down questions based on relationships, i.e., key components in knowledge editing, helping the model learn how to solve multi-hop questions in knowledge editing. In contrast, other methods let the model determine how to divide the questions. If the model makes an error during the early stages of problem decomposition, it can affect the following steps and lead to an incorrect final answer.

A.3 Safety Evaluation

We conducted a jailbreak attack (Huang et al., 2023) before and after applying RippleCOT, and found that the attack success rate is unchanged (92%) for the MaliciousInstruct dataset with the Vicuna-7B model under their setting w/o sys. prompt. This is because RippleCOT does not alter any model parameters during editing, thus it does not affect the model's safety level.

A.4 More baseline with large models

We added three large models, GPT-4-0125 (1.8T), GPT-4o and Claude-3.5 sonnet (while the exact parameter size isn't specified, it is the latest high-performing large model), to the table below, in addition to GPT-3 (175B), which have already been included in our paper. We observe that RippleCOT performs well on both smaller and larger models, whereas BaseCOT, which relies solely on the model's reasoning ability, is effective only for larger models with enhanced reasoning capabilities. The results is shown in Table 10.