PokeMQA: Programmable knowledge editing for Multi-hop Question Answering

Anonymous ACL submission

Abstract

Multi-hop question answering (MQA) is one of the challenging tasks to evaluate machine's comprehension and reasoning abilities, where large language models (LLMs) have widely achieved the human-comparable performance. Due to the dynamics of knowledge facts in real world, knowledge editing has been explored to update model with the up-to-date facts while avoiding expensive re-training or fine-tuning. Starting from the edited fact, the updated model needs to provide cascading changes in the chain of MQA. The previous art simply adopts a mix-up prompt to instruct LLMs conducting multiple reasoning tasks sequentially, including question decomposition, answer generation, and conflict checking via comparing with edited facts. However, the coupling of these functionally-diverse reasoning tasks inhibits LLMs' advantages in comprehending and answering questions while disturbing them with the unskilled task of conflict checking. We thus propose a framework, Programmable knowledge editing for Multi-hop Question Answering (PokeMQA), to decouple the jobs. Specifically, we prompt LLMs to decompose knowledge-augmented multi-hop question, while interacting with a detached trainable scope detector to modulate LLMs behavior depending on external conflict signal. The experiments on three LLM backbones and two benchmark datasets validate our superiority in knowledge editing of MQA, outperforming all competitors by a large margin in almost all settings and consistently producing reliable reasoning process.

1 Introduction

Multi-hop question answering (MQA) requires a sequence of interacted knowledge facts to reach the final answer. For instance, considering the two-hop question in Figure 1, it is necessary to deduce the intermediate answer *Inter Miami* through the fact "Messi plays for Inter Miami", and then deduce the final answer *NA* through another fact "Inter Miami is located in North America"). MQA poses a great challenge to reasoning abilities of question answering systems (Mavi et al., 2022, Chen et al., 2019, Lan et al., 2021). Thanks to the natural language comprehending and reasoning brought by large-scale pre-training, large language models (LLMs) have proven its indispensable utility in MQA tasks (Rao et al., 2022, Khalifa et al., 2023).

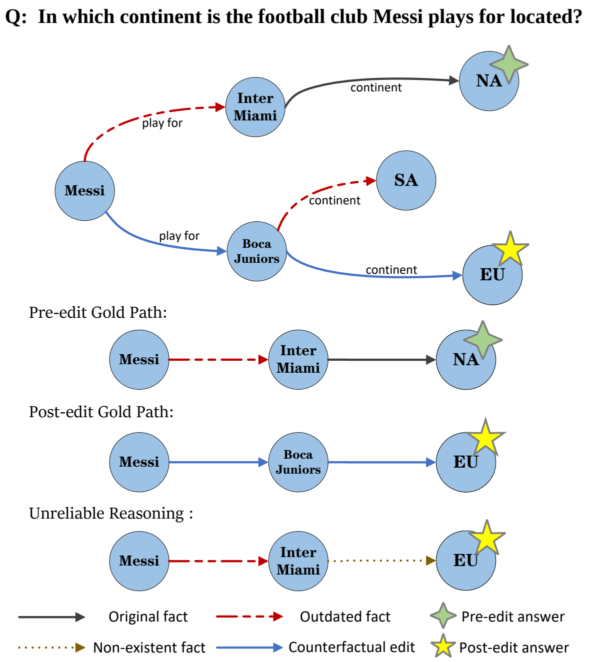


Figure 1: An example of multi-hop question answering under knowledge editing, which consists of relevant knowledge facts and three specific reasoning paths solving the two-hop question. For the unreliable reasoning, it uses a outdated and a non-existent fact and end up with the right answer *Europe*.

However, the knowledge within LLMs may be factually wrong or become invalid over time. To ensure the correctness of LLMs, technique of knowledge editing has been carried out to provide efficient and targeted updates on model behaviors (Sinitsin et al., 2020, Zhu et al., 2020, De Cao et al., 2021,). There are two popular approaches: parameter-modification based editing and memory-based editing. The former one modifies the internal model weights according to edited facts through meta-learning, fine-tuning, or knowledge locating (Meng et al., 2022a, Mitchell et al., 2021, Meng et al., 2022b). The latter approach leverages an external memory to explicitly store the edited facts (or termed as edits) and reason over them, while leaving LLMs parameters unchanged (Mitchell et al., 2022; Zheng et al., 2023a). Memory-based model editing is generally adopted due to its simpleness and agnostic to backbone LLMs.

In the context of MQA, MeLLo (Zhong et al., 2023) is first proposed by designing a multipurpose prompt to instruct LLMs conducting the reasoning tasks of question decomposition and knowledge editing sequentially. In particular, after decomposing the multi-hop questions, LLMs generate a tentative answer for each subquestion and then detect whether there exists factual conflict between tentative answer and edited facts in memory (e.g., statements of "the current British Prime Minister is Rishi Sunak" and "the current Prime Minister of the UK is Liz Truss" are factually incompatible with each other). By repeatedly prompting LLMs, MeLLo reaches the answer of multi-hop question.

However, the coupling of question decomposition and knowledge editing imposes considerable demands on LLMs to precisely perform reasoning as demonstrations in context. *First*, the knowledge editing requires LLMs to fully understand the semantics of two candidate facts and then make conflict detection based on the factual compatibility between them. In the few-shot prompting, LLMs are prone to underfit this editing logic due to inadequate supervision signals, especially when embedded in a more complex task (Khot et al., 2022), i.e. question decomposition. *Second*, within a unified prompt, the incorporation of knowledge editing instruction introduces noise to question decomposition in the similar way. Such superposed noise prevents LLMs from fully focusing on parsing the syntactic structure of multi-hop questions to precisely identify the subquestions.

Thus, we propose Programmable knowledge editing for Multi-hop Question Answering (PokeMQA), where we decouple the two essential tasks, i.e. question decomposition and knowledge editing, to alleviate burdens on LLMs while introducing auxiliary knowledge prompt to assist question decomposition. Specifically, we offload the conflict detection in knowledge editing with a programmable scope detector, which is used to detect whether a subquestion lies within the scope affected by any edited facts in semantic space (*Challenge #1*). A two-stage scope detector is designed: In pre-detection stage, we efficiently filter out a substantial number of irrelevant edits; In conflict-disambiguation stage, we perform precise retrieval on the remaining few candidate edits. Our two-stage framework provides both computational efficiency and expressiveness given the high volume of edited facts in real scenarios. The retrieved edits are used to calibrate LLMs behavior. Moreover, we propose a knowledge prompt to augment parse analysis in the process of question decomposition (*Challenge #2*). The knowledge prompt recognizes key entity from input question and retrieves its external information from a knowledge source to trig the correct decomposition.

Additionally, we observe that the multi-hop question answering process may use the outdated or non-existent facts, but occasionally ends up with the right answer. We refer to this situation as unreliable reasoning (as shown in Figure 1). In order to faithfully evaluate models' reasoning ability, we propose a new metric called hop-wise answering accuracy (Hop-Acc), measuring the extent how LLMs follow demonstrations, conduct question decomposition step by step, and generate desired answer to each step towards solving the multi-hop question.

2 Multi-hop Question Answering under Knowledge Editing

Notations. Following previous work (Zhong et al., 2023; Meng et al., 2022a), we denote a fact as a triplet (s, r, o), consisting of the subject s, object o, and relation r between them, such as (*Messi*, *play for*, *Inter Miami*). An edited fact (i.e., edit) is the knowledge fact that we want to update and is represented in the same form (s, r, o), such as (*Messi*, *play for*, *Boca Juniors*). We consider a multi-hop question Q, where answering Q requires sequentially querying and retrieving multiple facts. These facts are presented in the order they were queried, forming a *chain of facts* $\langle (s_1, r_1, o_1), \ldots, (s_n, r_n, o_n) \rangle$, where $s_{i+1} = o_i$ and o_n is the final answer, which uniquely represents an inter-entity path $\mathcal{P} = \langle s_1, o_1, \ldots, o_n \rangle$. It should be noted that except for s_1, all the other entities $o_1, \ldots, o_n$ in $\mathcal{P}$ do not appear in Q and need to be deduced either explicitly or implicitly through factual reasoning (like *Inter Miami* and *North America* in the multi-hop question in Figure 1). If we replace the invalid fact (s_i, r_i, o_i) with edit $e = (s_i, r_i, o_i^*)$ in a multi-hop question, due to the cascading effect caused by the edited fact, the chain of facts accordingly changes to $\langle (s_1, r_1, o_1), \ldots, (s_i, r_i, o_i^*), \ldots (s_n^*, r_n, o_n^*) \rangle$. The updated inter-entity path is $\mathcal{P}^* = \langle s_1, o_1, \ldots, o_i^*, \ldots, o_n^* \rangle$, which indicates the reasoning path to the final answer of Q has changed after being influenced by edit e.

MQA under knowledge editing. Given a set of edits $\mathcal{E} = \{e_1, \ldots, e_m\}$ and a language model f to be edited, for a multi-hop question Q, its inter-entity path becomes $\mathcal{P}^* = \langle s_1, o_1^*, \ldots, o_n^* \rangle$ after being affected by edits in $\mathcal{E}$. The goal of multi-hop question answering under knowledge editing can be formally described as producing an edited language model f_{edit} conditioned on f and $\mathcal{E}$, which can deduce the inter-entity path $\mathcal{P}^*$ and finally output the post-edit answer o_n^* to question Q. We denote $\mathcal{P}^*$ as *gold path* of Q (as shown in Figure 1). Different from the previous work, we not only evaluate whether edited model f_{edit} output the desired final answers, but also check the correctness of their intermediate reasoning paths, providing faithful MQA performance results for knowledge editing.

Edit scope. In line with our work, we make some modifications to this concept that was originally proposed by (Mitchell et al., 2022). For an edit $e = (s, r, o)$, we define the single-hop question q describing (s, r) with the answer being o as its *atomic question*. It should be noted that the atomic question corresponding to a specific edit is not unique but rather a set of semantically equivalent questions (e.g., "What is the country of origin of hockey?" and "Where did hockey originate?"). We refer to the set as the *scope* of an edit, denoted as $S(e)$. After making an edit $e = (s, r, o)$, the answers to those questions in $S(e)$ should change to o accordingly. Compared with the previous work, we define the edit scope based on the unit of atomic question, excluding the original multi-hop question, which typically has a much more complex syntactic structure. This simplified definition facilitates the programmable scope detector to learn the semantic patterns represented by $S(e)$ and then make precise edit retrieval to adjust LLMs behavior.

3 Programmable Editing in Memory of Multi-hop Question Answering

3.1 Workflow of PokeMQA

As illustrated in Figure 2, PokeMQA is a lightweight model editor that can be seamlessly integrated into any backbone LLMs, without changing parameters in the deployed language models. This empowers the language models to be robust to respond to questions based on edited facts. From initiating the editor to successfully addressing a question, the proposed procedure involves two steps as follows:

Storing edits in memory. When receiving a set of edits $\mathcal{E} = \{e_1, \ldots, e_m\}$, PokeMQA first uses manually-defined template to convert each edit triplet e into a natural language statement t (as in Zhong et al., 2023), then explicitly stores them in an external memory $\mathcal{M} = \{t_1, \ldots, t_m\}$ for query and retrieval.

Inference by checking with edit memory. Considering an input of multi-hop question, we adopt in-context learning (Brown et al., 2020) and provide a few demonstrations (i.e., input-label pairs) as the few-shot prompt to teach models to execute the following three tasks alternately: I) Identify the next subquestion (i.e., atomic question) conditioned on the input question and current inference state in LLMs; II) Detect whether this subquestion falls within the edit scope and generate answer; III) Extract the answer entity for this subquestion in LLMs. Note that this answer entity is either used to decompose the next subquestion at Step I or released as the final answer.

Particularly, we propose the programmable scope detector to detach the knowledge editing task in Step II from LLMs. Previous work (Zhong et al., 2023) generates tentative answers for each subquestion and checks the semantic conflict between tentative answer and retrieved edit in LLMs. With the slight supervision signals from the few-shot prompt, it is challenging for LLMs to compare their semantic patterns and make the correct conflict detection. In this work, the proposed scope detector takes the subquestion as input and detects whether it falls within the scope of any edit in $\mathcal{M}$. If so, the detector sends the factual conflict signal back with a chosen edit statement. The statement serves as a prompt to instruct LLMs to infer the answer from the edit. Otherwise, the factual conflict signal is empty and LLMs directly generate answer based on their internal knowledge.

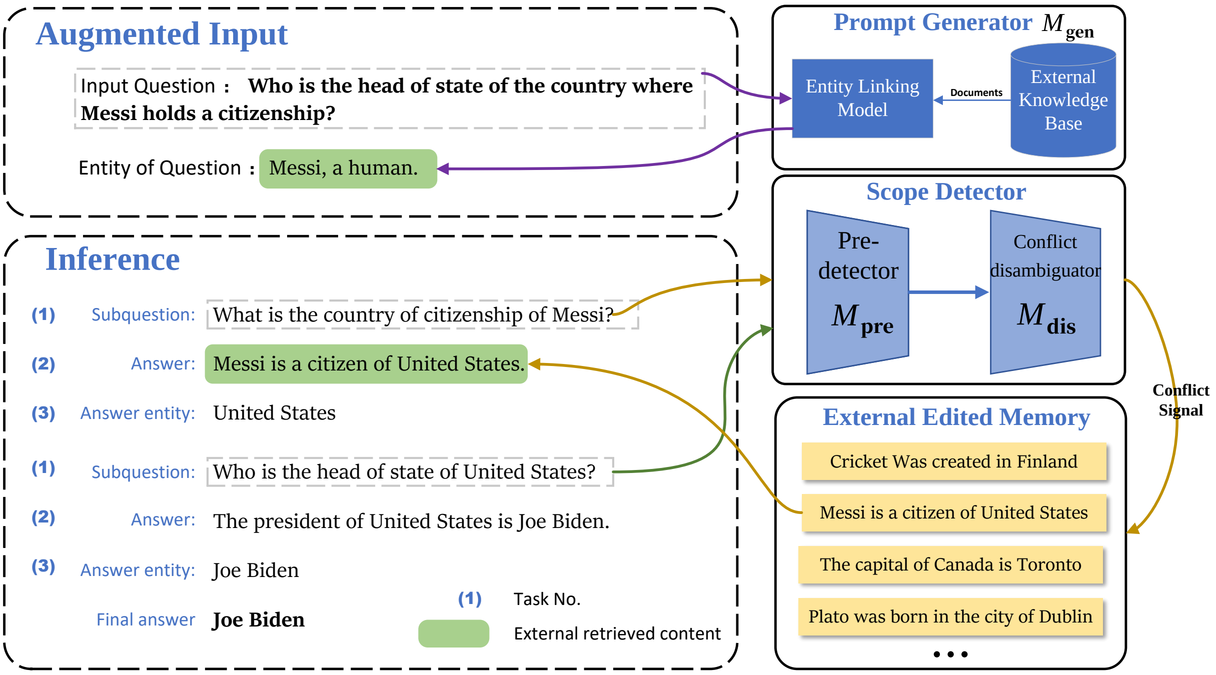


Figure 2: The illustration of our proposed method PokeMQA. PokeMQA leverages external knowledge base to construct knowledge prompt, facilitating the decomposition of the first subquestion. It then alternately executes subsequent question decomposition, knowledge editing with programmable scope detectors, and answer generation for MQA. The concrete prompts used in PokeMQA are shown in Appendix A.

In addition, we propose knowledge prompt to correct the question decomposition in Step I. In MQA, identifying the leading subquestion (i.e., the first decomposed subquestion) might be challenging due to insufficient contextual information. Specifically, given the input of multi-hop question, one lacks the explicit question entity and entity-related fact, which are available when identifying the subsequent subquestions. To address this, we innovatively employ the knowledge prompt generator to preprocess the input question. It recognizes the key entity and retrieves relevant documents from an external knowledge base to create a knowledge prompt. Then, we concatenate the input question and knowledge prompt to form an augmented input, effectively resolving the issue.

Owing to the proposed scope detector and knowledge prompt, PokeMQA allows language models to focus on question decomposing and answering, formulating a reliable reasoning path. The details of the proposed components are stated below.

3.2 Programmable Scope Detector

Motivated by (Mitchell et al., 2022), we utilize a programmable scope detector for conflict detection and design a task-specific training approach to identify effective edit scope patterns.

Architectures. The scope detector can be formally described as $g(t, q) : \mathcal{T} \times \mathcal{Q} \to [0, 1]$, which predicts the probability that an atomic question q falls into the scope of the edit statement t (in terms of the edit e). The scope detector can be implemented as arbitrary text classification models (Liu and Guo, 2019, Lu et al., 2020, Khattab and Zaharia, 2020). In our framework, considering both expressiveness and computational efficiency, we choose two lightweight, yet complementary models. The two models are denoted as g_ϕ and g_ψ, respectively. For an input pair (t, q), g_ϕ calculates the embeddings for t and q separately and models the log-likelihood by the negative squared Euclidean distance in the embedding space. Model g_ψ concatenates t and q together as a unified input for the sequence classification task.

In our framework, the models g_ϕ and g_ψ serve as pre-detector M_{pre} and conflict disambiguator M_{dis}, respectively. We combine them together to establish a *two-stage edited fact retrieval* framework. The pre-detector filters out the enormous semantically irrelevant edits from memory efficiently, while the conflict disambiguator accurately locates on candidate edit with the highest likelihood. The details are given in Appendix G. Once the detector finally believes that an input atomic question falls into the scope of any edit in $\mathcal{M}$, it retrieves the edited statement of candidate edit and sends it back along with a factual conflict signal to guide the language model generation process.

Training scope detector. According to edit memory $\mathcal{M} = \{t_1, \ldots, t_m\}$, we build up a training dataset $\mathcal{D}_{\text{train}} = \{(t_1, q_1), \ldots, (t_m, q_m)\}$. See Appendix B for more details on the construction of the dataset. To learn the scope covered by each edit statement t_i, we use a binary cross-entropy loss with negative sampling (Mikolov et al., 2013) as the training objective:

$$\mathcal{L} = -\log g(t_i, q_i) - \mathbb{E}_{q_n \sim P_n(q)} \left[\log(1 - g(t_i, q_n))\right], \tag{1}$$

where P_n is a negative sampling distribution and we set it to a uniform distribution over each mini-batch. Note that M_{pre} and M_{dis} are trained separately using the above supervised learning setting.

Model selection. In practice, we observed that using the traditional classification metric (accuracy) to validate the detector's performance can often result in underfitting. We believe this is due to the unique characteristics of the conflict detection task. Thus, we define two novel task-specific metrics to select detector models and guide early stopping during training: *Success Rate* and *Block Rate*. The *Success Rate* measures the accuracy to retrieve the correct edit statement t_i for a target question q_i from a set of candidates:

$$SR = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1} \left[\bigwedge_{(t,q) \in \mathcal{D}_{val}} (g(t_i, q_i) \geq g(t, q_i)) \right], \tag{2}$$

where $\mathbb{1}(\cdot)$ is the indicator function, N is the size of validation set $\mathcal{D}_{val}$, and $\bigwedge$ denotes the AND gate. For the target pair (t_i, q_i), the retrieval is precise if and only if its detection likelihood is higher than the other pairs (t, q_i), which are synthesized by replacing the target edit statement t_i with candidates from $\mathcal{D}_{val}$. On the other hand, metric *Block Rate* quantifies the extent of detector models to inhibit the unrelated edit statements for a target question:

$$BR = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1} \left[\bigwedge_{(t,q) \in \mathcal{D}_{val}^{-}} (g(t, q_i) < 0.5) \right], \tag{3}$$

where $\mathcal{D}_{val}^{-} = \mathcal{D}_{val} - \{(t_i, q_i)\}$. Intuitively, a higher value of SR suggests that the scope detector is able to retrieve the desired edit statements for more atomic questions, while a higher value of BR implies that fewer atomic questions are mistakenly categorized into the edit scope of the irrelevant edits. We use these two metrics to evaluate detectors M_{pre} and M_{dis}, and return the optimal-performing detectors on validation set (i.e., having the highest sum of SR and BR). We empirically find that these two metrics performs better serving as the indicator of early stopping (Yao et al., 2007).

3.3 Knowledge Prompt Generator

To identify the leading subquestion during question decomposition, we propose knowledge prompt generator M_{gen}, which aims to provide the additional valuable contextual information. Specifically, we employ ELQ (Li et al., 2020), a fast end-to-end entity linking model. It recognizes the key entity, i.e., the named entity in the input question Q, links the entity to Wikidata, and subsequently retrieves the related knowledge facts from Wikidata (Vrandečić and Krötzsch, 2014).

The retrieved knowledge facts from the Wikidata are the valuable contextual information for the question decomposition and the knowledge facts are stored as triplets (s, r, o) in Wikidata. We adopt the following strategy to only preserve the commonsense facts from the vast knowledge base. For simplicity, we consider two *basic membership properties* $\mathcal{R} = [r_1, r_2]$ as our interested relations, where r_1=*instance of*, r_2=*subclass of*. Each entity in Wikidata possesses at least one of the relations. These two relations typically provide infallible commonsense facts related to the entity. Thus, for a key entity s_i, we randomly choose (s_i, r_1, o_1) or (s_i, r_2, o_2) as the retrieval fact. After retrieving, we use a manually-defined template to convert both key entity and retrieval fact into a knowledge prompt to augment the input question Q. For instance, as shown in Figure 2, we recognize the key entity *Messi* and retrieve the knowledge fact *(Messi, instance of, human)*. After composing them together, we finally get the knowledge prompt *Entity of Question: Messi, a human*.

4 Experimental Setup

We evaluate our approach on MQUAKE (Zhong et al., 2023), which is a knowledge editing benchmark. It includes MQUAKE-CF-3K based on counterfactual edits, and MQUAKE-T with temporal knowledge updates. These datasets consist of a number of k-hop questions ($k \in \{2, 3, 4\}$), each of them is associated with one or more edits. More statistics can be found in Appendix C.

4.1 Evaluation Metrics

Multi-hop accuracy (Zhong et al., 2023). It measures the accuracy of the (edited) language models in answering multi-hop questions.

Hop-wise answering accuracy (Hop-Acc). In order to avoid the potential interference caused by unreliable reasoning, we propose the Hop-Acc to checks the correctness of intermediate reasoning path when evaluating MQA performance. Specifically, for a multi-hop question Q, since the question decomposition prompt is completely structured, language models are able to state the intermediate answer of subquestion in a concise, parseable way. Thus, the chain of intermediate answer $\langle s_1, o_1, \ldots, o_n \rangle$ can be parsed from the inference content as the deduced path $\mathcal{P}$. We argue that a multi-hop question is fully solved by language models only if the deduced path $\mathcal{P}$ is exactly the same as the *gold path* $\mathcal{P}^*$ (defined in Section 2), i.e., the novel metric measures the accuracy of reasoning path for multi-hop questions, which is only available for sequential question decomposition.

4.2 Baselines Methods & Language Models

We take four knowledge editing methods as baselines, including parameter updating methods, **FT** (Zhu et al., 2020), **ROME** (Meng et al., 2022a), **MEMIT** (Meng et al., 2022b) and memory-based method **MeLLo** (Zhong et al., 2023). More implementation details are in Appendix E. So far there is still no enough evidence to prove whether chain-of-thought (COT) prompting (Wei et al., 2022) or question decomposition (QD) prompting (Press et al., 2022) is more effective. Thus, to ensure fair and comprehensive comparisons, except for memory-based editors (MeLLo, PokeMQA) that relies on question decomposition, we report the performance of other parameter updating methods under both COT and QD prompting[1].

We conduct experiments on the following three base language models: **LLaMa-2-7B** (Touvron et al., 2023) is a powerful open-source pre-trained large language model, implemented by Huggingface Transformers library (Wolf et al., 2020); **Vicuna-7B** (Chiang et al., 2023) is trained by fine-tuning LLaMA, implemented by Fastchat library (Zheng et al., 2023b); **GPT-3.5-turbo-instruct** (Ouyang et al., 2022) is a variant of the most capable GPT-3.5 series model, GPT-3.5-turbo (ChatGPT), which is used for legacy completion.

4.3 Implementation Details

We finetune the pre-detector g_ϕ and conflict disambiguator g_ψ based on **DistilBERT** (Sanh et al., 2019). Note that the $\mathcal{D}_{\text{train}}$ used for fine-tuning does not contain any edit statements t that appear during testing. We provide detailed fine-tuning setting in Appendix D.

To evaluate performance under varying numbers of edits, we conduct stratified sampling (Parsons, 2014) of the dataset according to hops of questions to construct edit batches of different sizes, which ensures the proportion of questions with different hops is relatively the same within each edit batch. We inject all the edits within a batch simultaneously[2] (Wang et al., 2023).

It should be noted that we conduct experiments related to parameter updating methods exclusively on the open-source LLM (**LLaMa-2-7B**), while the memory-based editing methods are comprehensively evaluated across all language models. (More details about experiments in Appendix H).

5 Performance Analysis

5.1 Main Results

PokeMQA is effective and reliable. We report our main results in Table 1. The results demonstrate that PokeMQA outperforms all baselines by a large margin in almost all settings. Moreover, PokeMQA achieves the highest **Hop-Acc** across all settings, which strongly supports our view that the coupling of question decomposition and conflict detection places too much burden on LLMs, thus negatively impacting their inference abilities. PokeMQA significantly addresses the issue of unreliable reasoning, further improving MQA performance under knowledge editing. Meanwhile, achieving a high **Hop-Acc** indicates that PokeMQA's reasoning process is more rational and can serve as a more reliable explanation for model predictions, enhancing the interpretability of LLMs in MQA. Furthermore, PokeMQA can scale effectively with current mainstream LLMs, such as GPT-3.5-turbo-instruct (175B), without the need for additional training.

MeLLo is a potential alternative. The related results about MeLLo suggest that it is undoubtedly a strong competitor. In the head-to-head comparisons on LLaMa-2-7B, MeLLo achieves (1/10) op-

[1] These methods executes the entire process by itself, without the need for repeatedly prompting.

[2] Since ROME is not able to perform batch edit, we sequentially inject edits within a batch.

Method	MQUAKE-CF-3K						MQUAKE-T			
	1 edited		100 edited		All edited		1 edited		All edited	
	Acc.	Hop-Acc	Acc.	Hop-Acc	Acc.	Hop-Acc	Acc.	Hop-Acc	Acc.	Hop-Acc
LLaMa-2										Size: 7B
FT_{COT}	22.3	-	2.13	-	OOM	-	47.32	-	3.75	-
FT	28.2	7.3	2.37	0.03	OOM	OOM	56.48	33.89	1.02	0.37
ROME_{COT}	11.17	-	2.87	-	2.77	-	28.96	-	14.4	-
ROME	13.13	5.37	3.5	0.03	3.63	0.1	24.89	17.99	1.71	0.32
MEMIT_{COT}	11.83	-	9.23	-	5.57	-	36.88	-	31.58	-
MEMIT	14.97	6.43	9.4	2.47	2.3	0.37	30.89	23.98	25.21	20.13
MeLLo	33.57	9.9	20.0	10.07	17.33	9.9	**97.7**	0.21	62.58	3.96
PokeMQA (Ours)	**44.13**	**30.6**	**37.33**	**27.83**	**32.83**	**23.87**	75.43	**60.44**	**74.36**	**60.22**
Vicuna										Size: 7B
MeLLo	22.7	7.03	12.83	6.77	10.9	6.7	42.24	1.12	19.86	1.28
PokeMQA (Ours)	**45.83**	**34.8**	**38.77**	**31.23**	**31.63**	**25.3**	**74.57**	**55.19**	**73.07**	**55.09**
GPT-3.5-turbo-instruct										Size: 175B
MeLLo	57.43	28.8	40.87	28.13	35.27	25.3	**88.12**	52.84	74.57	53.53
PokeMQA (Ours)	**67.27**	**56.37**	**56.0**	**49.63**	**48.87**	**39.77**	78.16	**68.09**	**76.98**	**67.88**

Table 1: Evaluation results on MQUAKE-CF-3K and MQUAKE-T. The best result is indicated in **Bold**. 'n edited' represent the number of multi-hop questions within each edit batch, i.e., the size of edit batch; 'Acc' and 'Hop-Acc' respectively denotes the **Multi-hop accuracy** and **Hop-wise answering accuracy** discussed in Section 4.1; 'COT' means that the current method uses chain-of-thought prompt, otherwise the question decomposition prompt; '-' means the metric is not applicable to the current method.

timal result and (7/10) sub-optimal results. Surprisingly, MeLLo also achieves the best performance in two settings (In MQUAKE-T, with single instance edited, 97.7 on LLaMa-2-7B and 88.12 on GPT-3.5-turbo-instruct). But through a detailed analysis of the reasoning processes, we discover that MeLLo resolves most multi-hop questions by exploiting *shortcut reasoning patterns* (See an example in Appendix I), which can be considered as a form of underfitting to the prompt. A clear evidence is that the accuracy on LLaMa-2-7B with weaker inference ability is higher than on GPT-3.5-turbo. Meanwhile, it is undeniable that MeLLo's performance benefits significantly from the increased capabilities of LLMs, suggesting that on a stronger LLM in the future, MeLLo might further narrow the performance gap with PokeMQA.

Parameter-updating may not be the answer to knowledge editing. In line with previous researches (Zhong et al., 2023, Onoe et al., 2023), parameter updating methods fail catastrophically at answering multi-hop questions, indicating that the injected knowledge cannot be flexibly applied to inference by the edited model. FT achieves the best performance among these updating methods with single instance edited, but the impact of a slightly larger edit batch on its performance can already be devastating. ROME performs worse than MEMIT in all settings, which is consistent with the fact that MEMIT is an improved version of ROME. MEMIT displays a certain level of robustness to the size of edit batch, but it still fails under thousands of edited facts[3]. In summary, our results indicate that parameter updating methods can hardly meet the desiderata of knowledge editing applications.

MQA under knowledge editing remains challenging. As shown in Figure 3 (Middle, Right), PokeMQA consistently maintains state-of-the-art performance across multi-hop questions of varying difficulty levels while significantly surpassing other competitors in producing reliable reasoning. But depressingly, the increasing difficulty of the questions also has a significant negative impact on PokeMQA's performance. Combining more facts to generate more complex reasoning processes poses a double challenge in terms of edited fact retrieval accuracy and language model reasoning capabilities. Currently, PokeMQA is not fully capable of addressing these challenges, indicating that this task remains challenging for future knowledge editing methods.

5.2 Ablation Study

We conduct ablation experiments to investigate how the two detachable components M_{dis} and M_{gen} improves PokeMQA and analyze their necessity. The results are shown in Table 2 and Figure 3 (Left). Based on the experimental results, we find that the

[3]In MQUAKE-CF-3K, there are 2786 different edits with all instances edited

M_{dis}	M_{gen}	GPT-3.5-turbo-instruct				LLaMa-2-7B				Vicuna-7B			
		MQUAKE-CF-3K		MQUAKE-T		MQUAKE-CF-3K		MQUAKE-T		MQUAKE-CF-3K		MQUAKE-T	
		1 edited	All edited	1 edited	All edited	1 edited	All edited	1 edited	All edited	1 edited	All edited	1 edited	All edited
-	-	49.0	29.93	67.99	55.67	29.33	19.47	59.31	52.19	27.37	16.43	54.23	48.39
√	-	49.0	34.27	**68.09**	67.77	29.33	22.87	59.31	59.1	27.37	19.37	54.23	54.12
-	√	56.07	33.83	68.04	56.32	**30.6**	20.3	**60.44**	53.21	**34.8**	22.23	**55.19**	49.68
√	√	**56.37**	**39.77**	**68.09**	**67.88**	**30.6**	**23.87**	**60.44**	**60.22**	**34.8**	**25.3**	**55.19**	**55.09**

Table 2: Ablation study results of PokeMQA and its variants in terms of Hop-Acc. We also provide the results in terms of Acc. in Appendix A.

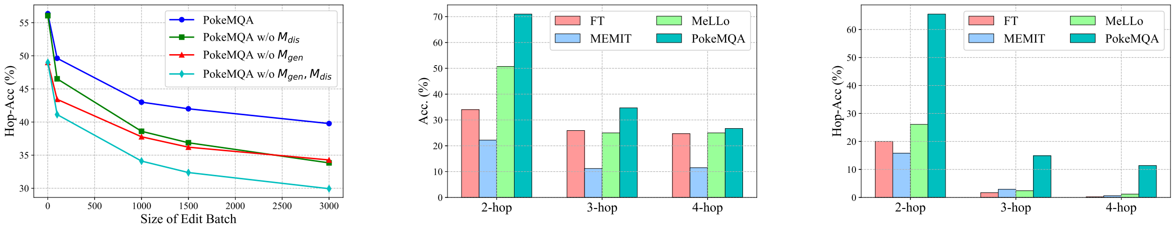


Figure 3: Left: Hop-Acc across multiple variants of PokeMQA in MQUAKE-CF-3K on GPT-3.5-turbo-instruct with edit batches of different sizes. Middle, Right: In MQUAKE-CF-3K, Acc. and Hop-Acc on multi-hop questions with different hop counts, with single instance edited on LLaMa-2-7B. We also provide the extra results for the two experiments in Appendix A.

two components indeed enhance PokeMQA and summarize two conclusive findings as follows:

Use M_{gen} selectively. As shown in Table 2, Figure 3 (Left), the knowledge prompt generator M_{gen} improves PokeMQA performance in almost all settings. Although the above results verify its effectiveness, we have to point out that the performance gain is much more significant in MQUAKE-CF-3K compared to MQUAKE-T. Our view is that this is because MQUAKE-T is constructed based on real fact updates in recent years, so the key entity in the input question may be familiar to the latest pre-trained LLMs. Consequently, they can recognize the key entity and access entity-related knowledge relatively easily, even without additional contextual information. Due to the extra computation cost of M_{gen}, we recommend using M_{gen} selectively depending on the specific application.

M_{dis} is indispensable for large-scale editing. As shown in Table 2, Figure 3 (Left), the relative performance gain from the M_{dis} in terms of Hop-Acc gradually increases with larger edit batch. Meanwhile, we calculate the average number of predictions of M_{dis} (2.746 in MQUAKE-CF-3K and 4.565 in MQUAKE-T both on LLaMa-2-7B with all instances edited). The results indicate that by incorporating the M_{dis}, a tiny additional number of predictions greatly boosts MQA performance under large edit batch. We conclude that M_{dis} can greatly enhance the robustness of PokeMQA to large-scale editing with almost no additional computational cost, which is indispensable to maintain the applicability of PokeMQA in real scenarios.

6 Related Work

Knowledge editing methods. Knowledge editing focuses on updating factual knowledge to language models and a lot of related research has been carried out. Most of these methods predict updates to the weights of the base model by knowledge locating or meta-learning and then locally modify parameters (Mitchell et al., 2021, Meng et al., 2022b). Another part preserves parameters and explicitly stores edit instances (Mitchell et al., 2022, Zhong et al., 2023). Recent work has identified the limitations of existing editing methods through theoretical analysis (Hase et al., 2023) and performance evaluation (Onoe et al., 2023). Our work focuses on addressing one of these challenging tasks: MQA under knowledge editing scenarios.

7 Conclusion

In this work, we propose a novel programmable knowledge editing method (PokeMQA) to improve MQA performance and address unreliable reasoning. PokeMQA leverages a scope detector to align LLMs' behavior with edited facts and incorporates auxiliary knowledge prompt to enrich contextual information. Extensive experiments across three LLMs show that PokeMQA helps LLMs answer multi-hop questions in a precise and reliable way.

Limitations

In this work, we did not design a task-specific architecture for the scope detector to achieve higher fact retrieval accuracy and mitigate the pressure that context length imposes on the LLMs reasoning capabilities when handling complex multi-hop questions.

Besides, although memory-based editing shows great potential for controlled editing and large-scale editing, the way it stores edit instances makes it extremely vulnerable to attacks such as memory injection. Therefore, memory-based editing needs to be supported by reliable security technology to reduce its risk in real scenarios.

References

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.

Jifan Chen, Shih-ting Lin, and Greg Durrett. 2019. Multi-hop question answering via reasoning chains. *arXiv preprint arXiv:1910.02610*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213*.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023. Few-shot reranking for multi-hop qa via language model prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15882–15897.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. *arXiv preprint arXiv:2105.11644*.

Belinda Z Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. *arXiv preprint arXiv:2010.02413*.

Gang Liu and Jiabao Guo. 2019. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338.

Zhibin Lu, Pan Du, and Jian-Yun Nie. 2020. Vgcn-bert: augmenting bert with graph embedding for text classification. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part I 42*, pages 369–382. Springer.

Vaibhav Mavi, Anubhav Jangra, and Adam Jatowt. 2022. A survey on multi-hop question answering and generation. *arXiv preprint arXiv:2204.09140*.

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*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

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.

Yasumasa Onoe, Michael JQ Zhang, Shankar Padmanabhan, Greg Durrett, and Eunsol Choi. 2023. Can lms learn new entities from descriptions? challenges in propagating injected knowledge. *arXiv preprint arXiv:2305.01651*.

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.

Van L Parsons. 2014. Stratified sampling. *Wiley StatsRef: Statistics Reference Online*, pages 1–11.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Dattaraj J Rao, Shraddha S Mane, and Mukta A Paliwal. 2022. Biomedical multi-hop question answering using knowledge graph embeddings and language models. *arXiv preprint arXiv:2211.05351*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitriy Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. *arXiv preprint arXiv:2004.00345*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, et al. 2023. Easyedit: An easy-to-use knowledge editing framework for large language models. *arXiv preprint arXiv:2308.07269*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

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, Mariama Drame, Quentin Lhoest, and Alexander Rush. 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.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. 2007. On early stopping in gradient descent learning. *Constructive Approximation*, 26:289–315.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023a. 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. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena.

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

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 Prompt & Supplement Results for PokeMQA

A demonstration example used in prompt is shown in Table 5. The supplement ablation results are shown in Table 6 and Figure 4.

B Details of Training Dataset Construction

To train our scope detector, including pre-detector g_ϕ and conflict disambiguator g_ψ, we construct a training dataset $\mathcal{D}$. Specifically, we first extract edit triples from MQUAKE-CF and filter out the part sharing the same (s, r) with fact triples appeared in MQUAKE-CF-3K and MQUAKE-T, constructing a edit dataset $\mathcal{D}_e = \{e_1, \ldots, e_n\}$. Then we use manually-defined template to convert each edit triple e into a natural language statement s and get a edit statement dataset $\mathcal{D}_{state} = \{t_1, \ldots, t_n\}$. Finally we design a prompt consisting of instruction and demonstrations (shown in Table 7) and prompt Vicuna-13B (Chiang et al., 2023) to generate three diversely phrased atomic questions for each $s \in \mathcal{D}_{state}$, building a training dataset $\mathcal{D} = \{(t_1, q_1), \ldots, (t_n, q_n)\}$. It should be noted that when computing SR and BR, or sampling negative instances, instances with the same statement t should not be considered.

C Multi-hop Question Answering Dataset Statistics

Table 3 contains the statistics for the two benchmark datasets used in our experiments.

	#Edits	2-hop	3-hop	4-hop	Total
MQUAKE-CF-3K	1	513	356	224	1093
	2	487	334	246	1067
	3	-	310	262	572
	4	-	-	268	268
	All	1000	1000	1000	3000
MQUAKE-T	1 (All)	1421	445	2	1868

Table 3: Statistics of datasets used in experiments

Besides, In MQUAKE-CF-3K, there are 2786 different edits with all instances edited; In MQUAKE-T, there are 96 different edits with all instances edited;

D Details about Scope Detector finetuning

We finetune the pre-detector g_ϕ and conflict disambiguator g_ψ based on **DistilBERT** (Sanh et al., 2019) and the checkpoint is *distilbert-base-cased* from Huggingface Transformers library (Wolf et al., 2020). We take $SR + BR - 1$ as the indicator of early stopping.

To fine-tune pre-detector g_ϕ, the learning rate is set as $1e^{-5}$ with Adam optimizer (Kingma and Ba, 2014), the batch size is set as 1024, and the number of negative samples is 20; To fine-tune conflict disambiguator g_ψ, the learning rate is set as $1e^{-5}$ with Adam optimizer, the batch size is set as 256 and the number of negative samples is 1. And the dataset split is 80%/20% for training and validation, without the need of testing.

E Implementation Details of Baselines

In our experiments, the parameter updating knowledge editing methods, including **FT**, **ROME** and **MEMIT** is implemented by EasyEdit library (Wang et al., 2023). We basically follow the default hyperparameter settings on **LLaMa-2-7B** in library, and make a slight adjustment to ensure the effectiveness of these methods in different experimental settings. We modify the learning rate for ROME and target editing layer for MEMIT, the detailed modifications is shown in Table 4.

Method	MQUAKE-CF-3K			MQUAKE-T	
	1 edited	100 edited	All edited	1 edited	All edited
ROME	$5e^{-1}$	$5e^{-5}$	$5e^{-6}$	$5e^{-1}$	$1.5e^{-1}$
MEMIT	4,5,6,7,8	5,6,7	7	4,5,6,7,8	4,5,6,7,8

Table 4: Detailed hyperparameter modification for ROME and MEMIT.

F Licensing

Vicuna-7B (v1.1) and distilbert-base-cased are released under the Apache License 2.0. LLaMa-2-7B is licensed under the LLAMA 2 Community License. ELQ, ROME, MEMIT, FT are released under the MIT license.

G Two-stage edited fact Retrieval

Algorithm 1 Two-stage Edited Fact Retrieval.

Input: edited memory $\mathcal{M}$, pre-detector g_ϕ, conflict disambiguator g_ψ, atomic question q

Initialize candidate set $\mathcal{Z} = \varnothing$
Initialize final set $\mathcal{F} = \varnothing$
/* Pre-detection stage */
for all $t_i \in \mathcal{M}$ **do**
  if $g_\phi(t_i, q) \geq 0.5$ **then** $\mathcal{Z} = \mathcal{Z} \bigcup \{t_i\}$
end for
if $|\mathcal{C}| = 1$ **then**
  return t_{i^*}, where $t_{i^*} \in \mathcal{Z}$
end if
/* Conflict-disambiguation stage */
for all $t_i \in \mathcal{C}$ **do**
  if $g_\psi(t_i, q) \geq 0.5$ **then** $\mathcal{F} = \mathcal{F} \bigcup \{t_i\}$
end for
if $\mathcal{F} \neq \varnothing$ **then**
  return t_{i^*}, where $i^* = \arg\max_i g_\psi(t_i, q)$, $t_i \in \mathcal{F}$
end if

H Details about Experiments

Because MQUAKE regard a *chain of facts* as an instance and there are three diversely phrased multi-hop questions Q for each instance, we follow (Zhong et al., 2023), if any of the three questions is considered solved in terms of the specific metric, the instance is considered correct.

The experiments, data, language models in the paper are all in English. We run all experiments on a machine with four NVIDIA A40 GPU. One run of our experiments takes about 15 GPU hours. For all experiments, we use greedy decoding strategy to get the output in text space of language models for reproducibility and report a single run result due to the limited computational resources.

I Shortcut Reasoning

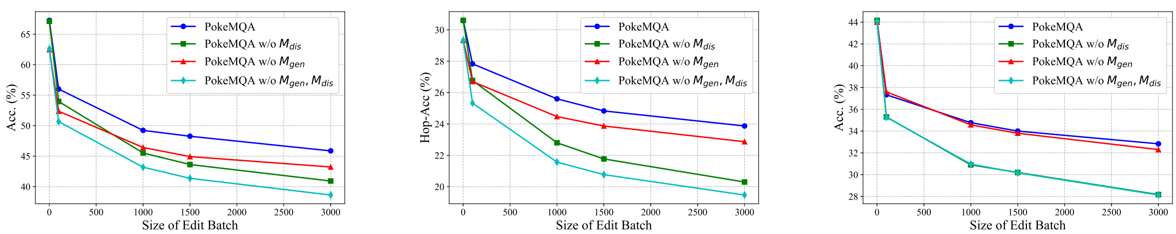
Given an example from Table 8, although MeLLo appears to successfully combine two facts to arrive at the final answer, its reasoning process does not adhere to the task logic demonstrated in the prompt, which can be regarded as an underfitting to question decomposition.

Question: What is the capital city of the country of citizenship of Ivanka Trump's spouse?
Entity of Question: Ivanka Trump, a human.
Subquestion: Who is Ivanka Trump's spouse?
Generated answer: Ivanka Trump's spouse is Jared Kushner.
According to Generated answer, the entity of Subquestion is: Jared Kushner
Subquestion: What is the country of citizenship of Jared Kushner?
Generated answer: Jared Kushner is a citizen of Canada.
According to Generated answer, the entity of Subquestion is: Canada
Subquestion: What is the capital city of Canada?
Generated answer: The capital city of Canada is Ottawa.
According to Generated answer, the entity of Subquestion is: Ottawa
Final answer: Ottawa

Table 5: A in-context demonstration example used in our PokeMQA prompt, here we omit the remaining three demonstrations. This color indicate that this part is constructed after being retrieved by knowledge prompt generator from external knowledge base. This color indicate that this part is retrieved by scope detector from external memory.

M_{dis}	M_{gen}	GPT-3.5-turbo-instruct				LLaMa-2-7B				Vicuna-7B			
		MQUAKE-CF-3K		MQUAKE-T		MQUAKE-CF-3K		MQUAKE-T		MQUAKE-CF-3K		MQUAKE-T	
		1 edited	All edited	1 edited	All edited	1 edited	All edited	1 edited	All edited	1 edited	All edited	1 edited	All edited
-	-	62.6	38.63	76.87	65.63	44.07	28.13	74.68	65.90	44.03	26.27	73.72	66.27
√	-	62.47	43.23	**77.09**	**78.53**	44.0	32.3	74.68	73.82	44.03	29.87	73.72	72.38
-	√	67.13	40.93	76.93	66.06	**44.17**	28.17	**75.43**	66.6	**45.9**	28.2	**74.57**	67.29
√	√	**67.27**	**45.87**	76.98	78.16	44.13	**32.83**	**75.43**	**74.36**	45.83	**31.63**	**74.57**	**73.07**

Table 6: Ablation study results of PokeMQA and its variants in terms of Acc.



(a) GPT-3.5-turbo-instruct (Acc.)
(b) LLaMa-2-7B (Hop-Acc)
(c) LLaMa-2-7B (Acc.)

Figure 4: Hop-Acc and Acc. across multiple variants of PokeMQA in MQUAKE-CF-3K on GPT-3.5-turbo-instruct and LLaMa-2-7B with edit batches of different sizes.

Please generate three different phrased Questions for each fact

Fact: The univeristy where Bob Dylan was educated is University of Minnesota.
Question 1: What is the name of the educational institution where Bob Dylan studied?
Question 2: What is the name of the university where Bob Dylan was educated?
Question 3: At which university did Bob Dylan receive his education?

Fact: The capital of United Kingdom is Angri.
Question 1: What is the name of the capital city of United Kingdom?
Question 2: Which city serves as the capital of United Kingdom?
Question 3: In which city is the capital of United Kingdom located?

Fact: basketball was created in the country of Spain.
Question 1: Where is basketball originated from?
Question 2: What is the name of the country of origin of basketball?
Question 3: Where did the sport of basketball originate?

Fact: John Coltrane is married to Anne Hathaway.
Question 1: Who is John Coltrane's spouse?
Question 2: To whom is John Coltrane married?
Question 3: Which person is married to John Coltrane?

Fact: Miranda Kerr is a citizen of Australia.
Question 1: Which country is Miranda Kerr a citizen of?
Question 2: What is the name of the country that Miranda Kerr belongs to?
Question 3: What is Miranda Kerr's nationality?

Table 7: The rephrased question generation prompt we used for dataset construction. The prompt contains one instruction and some in-context demonstrations.

Question: What is the name of the current head of state of Buckcherry's country of origin?

Subquestion: What is the country of origin of the band Buckcherry?

Generated answer: Buckcherry's country of origin is United States of America.

Retrieved fact: The name of the current head of state in United States of America is Joe Biden.

Retrieved fact does not contradict to generated answer, so the intermediate answer is: Joe Biden

Final answer: Joe Biden

Table 8: An example of shortcut reasoning that occurs in the inference process of GPT-3.5-turbo-instruct.