KnowTrace: Bootstrapping Iterative Retrieval-Augmented Generation with Structured Knowledge Tracing

Rui Li*[†][‡]
Gaoling School of Artificial Intelligence,
Renmin University of China
Beijing, China
lirui121200@ruc.edu.cn

Quanyu Dai
Huawei Noah's Ark Lab
Shenzhen, China
daiquanyu@huawei.com

Zeyu Zhang[†][‡]
Gaoling School of Artificial Intelligence,
Renmin University of China
Beijing, China
zeyuzhang@ruc.edu.cn

Xu Chen[†][‡][§]
Gaoling School of Artificial Intelligence,
Renmin University of China
Beijing, China
xu.chen@ruc.edu.cn

Zhenhua Dong
Huawei Noah's Ark Lab
Shenzhen, China
dongzhenhua@huawei.com

Ji-Rong Wen[†][‡]
Gaoling School of Artificial Intelligence,
Renmin University of China
Beijing, China
jrwen@ruc.edu.cn

Abstract

Recent advances in retrieval-augmented generation (RAG) furnish large language models (LLMs) with iterative retrievals of relevant information to handle complex multi-hop questions. These methods typically alternate between LLM reasoning and retrieval to accumulate external information into the LLM's context. However, the *ever-growing context* inherently imposes an increasing burden on the LLM to perceive connections among critical information pieces, with *futile reasoning steps* further exacerbating this overload issue. In this paper, we present **KnowTrace**, an elegant RAG framework to (1) *mitigate the context overload* and (2) *bootstrap higher-quality multi-step reasoning*. Instead of simply piling the retrieved contents, KnowTrace autonomously traces out desired knowledge triplets to organize a specific knowledge graph relevant to the input question. Such a structured workflow not only empowers the LLM with an intelligible context for inference, but also naturally inspires a reflective mechanism of *knowledge backtracing* to identify contributive LLM generations as process supervision data for self-bootstrapping. Extensive experiments show that KnowTrace consistently surpasses existing methods across three multi-hop question answering benchmarks, and the bootstrapped version further amplifies the gains.[1]

CCS Concepts

• **Computing methodologies → Natural language generation**;
• **Information systems → Question answering**.

*Work done during Rui Li's internship at Huawei Noah's Ark Lab.
[†]Beijing Key Laboratory of Research on Large Models and Intelligent Governance.
[‡]Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE.
[§]Corresponding author.
[1]The code is available at https://github.com/rui9812/KnowTrace.

Keywords

Retrieval-Augmented Generation, Large Language Models

ACM Reference Format:
Rui Li, Quanyu Dai, Zeyu Zhang, Xu Chen, Zhenhua Dong, and Ji-Rong Wen. 2025. KnowTrace: Bootstrapping Iterative Retrieval-Augmented Generation with Structured Knowledge Tracing. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3711896.3737015

1 Introduction

Recent Large language models (LLMs) [2, 5] have shown impressive performance across a variety of natural language tasks through the form of question answering. Despite their remarkable capabilities, LLMs continue to struggle with factual errors [23, 47, 55] when the input question exceeds their knowledge boundaries. As a practical solution to this problem, Retrieval-Augmented Generation (RAG) [19] empowers LLMs to incorporate external knowledge through information retrieval. One-time retrieval [8, 14] usually suffices to fulfill the knowledge needs of single-hop questions, but the complex *multi-hop questions* still remain challenging due to their demands for intensive knowledge and multi-step reasoning capabilities, thus attracting widespread attention within the research community.

To address the complex multi-hop questions, a range of recent works follow a natural strategy: extending the one-time RAG into a multi-round process [31, 36, 42, 49, 51]. Such iterative approaches interleave retrievals with LLM reasoning, periodically incorporating new information to narrow semantic gaps between the multi-hop questions and their requisite knowledge [36]. However, this workflow faces two critical challenges as presented in Figure 1. On the one hand, retrievers are not perfect and inherently inject massive redundant or misleading information into the context, imposing an increasing burden on the LLM to perceive the logical connections among critical information pieces, which are essential elements of reasoning [1]. On the other hand, not every LLM reasoning step is contributive to the answer prediction (especially when confronted with an unintelligible context), and the futile reasoning generations would trigger the retrievals of irrelevant information, thus further exacerbating the issue of context overload.

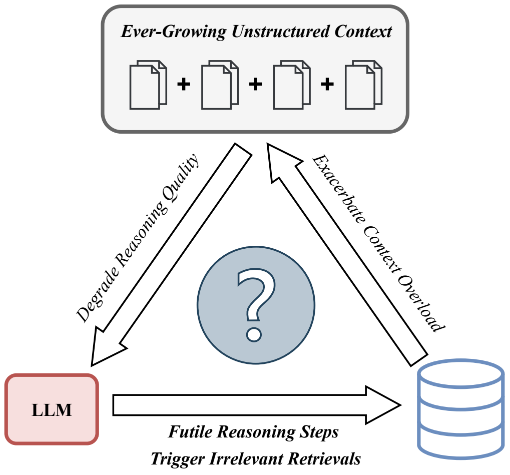

Ever-Growing Unstructured Context
Degrade Reasoning Quality
Exacerbate Context Overload
LLM
Futile Reasoning Steps
Trigger Irrelevant Retrievals

Figure 1: Two challenges of iterative RAG systems: ever-growing LLM context and non-contributive reasoning steps.

Some one-time RAG works [3, 20, 22, 27] ease the context burden with an auxiliary process of restructuring all retrieved documents, yet this process is impractical for the iterative RAG workflow, since it involves extensive LLM-driven operations (e.g., entity recognition and relation extraction) that would incur significant computational overhead for each retrieval iteration. Moreover, several recent self-training techniques [46, 53, 54] focus on enhancing the single-step generation quality of LLMs, while how to bootstrap higher-quality multi-step reasoning capabilities for the iterative RAG systems still remains critical yet underexplored.

In this paper, we move beyond existing approaches by presenting a self-bootstrapping framework called **KnowTrace**, which adopts a unique perspective of *structured knowledge tracing* to (1) tackle the context overload issue and also (2) promote higher-quality multi-step reasoning in a self-taught manner. Conceptually, we draw upon a profound insight from constructivist theory [7]: *learning is never merely about accumulating information, but also involves proactively absorbing knowledge to construct and expand one's cognitive schema.* Inspired by this principle, our framework reformulates the iterative RAG process as a coherent workflow of *knowledge graph expansion*, as illustrated in Figure 2(c). Specifically, instead of plainly stacking or intricately restructuring all retrieved contents, KnowTrace treats the LLM as an active tracker to progressively *explore-then-complete* question-relevant knowledge triplets, until tracing out a sufficient knowledge graph (KG) for the answer prediction. Such an inference workflow seamlessly empowers the LLM with an intelligible context throughout the multi-step reasoning process, which clearly reveals critical knowledge structures and, as a result, inherently enhances the LLM's reasoning quality [1]. Moreover, the transparency of KG expansion also spurs us to design a *post-hoc backtracing mechanism*, allowing KnowTrace to *retrospectively discern supportive knowledge triplets and contributive LLM generations based on the final inference.* In this way, given the positive reasoning trajectories that ultimately lead to correct answers, our framework can automatically filter out the procedural impurities (i.e., non-contributive LLM generations) to synthesize high-quality process supervision data, thereby capable of elevating the multi-step reasoning capabilities via *self-training*. To sum up, the main contributions of this work are as follows:

- We propose a self-bootstrapping iterative RAG framework called KnowTrace, which moves beyond existing methods by featuring a unique perspective of *structured knowledge tracing*.
- During inference time, KnowTrace progressively traces question-relevant knowledge triplets to endow the LLM with *an intelligible KG context*, thereby inherently raising the reasoning quality.
- By virtue of the self-organized KG structures, we further design a reflective mechanism of *knowledge backtracing*, which enables KnowTrace to retrospectively synthesize higher-quality process supervision for effective self-taught finetuning.
- We conduct extensive experiments on three standard multi-hop question answering datasets. Under different configurations of LLMs and retrieval models, KnowTrace consistently surpasses current RAG methods across all the datasets. The self-elevated version (finetuned on the self-synthesized process supervision) further amplifies the performance advantages by a clear margin.

2 Related Work

This section reviews a broad range of relevant literature to position our proposed framework within the current research landscape. More discussions are included in Appendix A.

Multi-Hop Question Answering (MHQA). This task involves addressing multi-hop questions that require extensive knowledge and multi-step reasoning capabilities to comprehensively infer the final answers [10, 41, 50]. Different from the traditional approaches [4, 30, 32], this paper focuses on incorporating LLM reasoning with information retrieval to reason about complex multi-hop questions, which aligns with the recent RAG researches [17, 20, 22, 31, 36, 42].

Iterative RAG Methods. Retrieval-augmented generation (RAG) has been demonstrated as a promising technique to effectively boost the performance of LLMs in knowledge-intensive NLP tasks [19]. Early RAG approaches [8, 18, 56] only perform one-time retrieval, struggling to gather all essential information for the input questions, especially for the complex multi-hop questions. To ease this issue, a new series of iterative RAG methods have recently been developed [31, 36, 42, 51]. These methods typically follow such a paradigm [52] as shown in Figure 2(a): first perform LLM reasoning to generate new queries (e.g., sub-questions) for retrieval, then accumulate the newly retrieved passages along with the previously collected ones to facilitate subsequent LLM reasoning, iteratively continuing this reasoning–retrieval process until the LLM derives the final answer. Despite their prowess, these iterative methods inherently disregard the significance of the underlying knowledge structures behind the retrieved passages, which are essential elements of reasoning [1].

Structure-Enhanced RAG Methods. In light of the limitation of purely unstructured information accumulation, some latest works follow a restructuring-based paradigm as shown in Figure 2(b): they employ an auxiliary restructuring process for all retrieved passages [3, 20, 22, 27] or even entire external corpora [6, 29, 35]. However, this strategy inherently necessitates extensive LLM invocations for intricate restructuring operations such as concept recognition and refinement, leading to significant computational costs and potential knowledge loss. In contrast, *our KnowTrace framework seamlessly integrates structuring and reasoning into a coherent process, naturally enabling the acquisition of desired knowledge triplets at a low cost.*

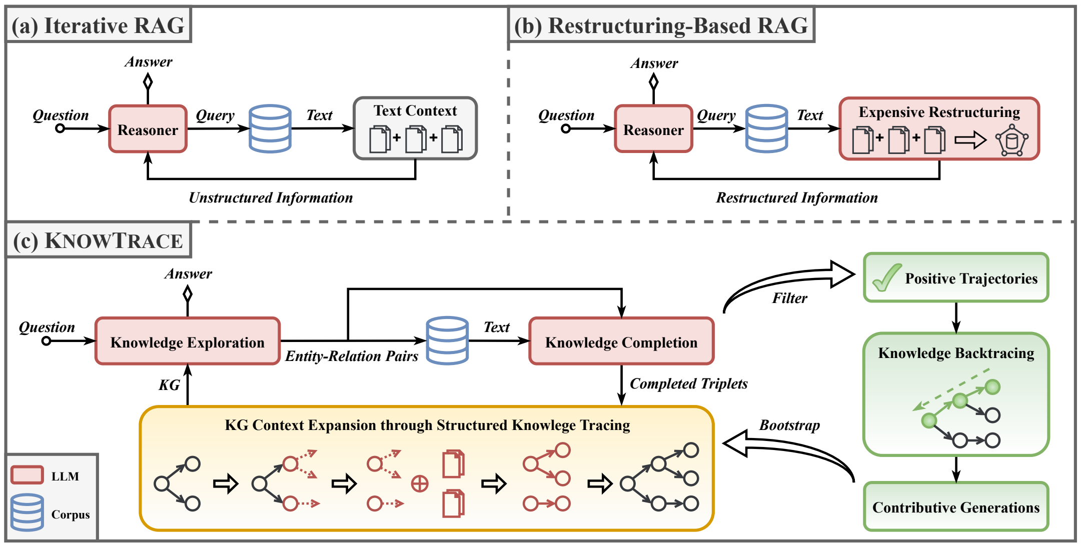


Figure 2: An overview of two representative workflows (a-b) and our KnowTrace framework (c).

In addition, several other structure-enhanced works [28, 39] focus on parsing the multi-hop questions into masked structured chains, where each masked triplet is then either directly completed based on an existing KG or rewritten as a natural language query to access relevant passages from a text database. However, these approaches heavily count on the accuracy of the initial parsing—errors at this stage can propagate—thereby requiring careful filtering operations and consistency checks [39]. Unlike them, *our KnowTrace features a more flexible workflow of adaptively tracing relevant knowledge throughout the entire multi-step reasoning process, which effectively improves both performance and robustness.*

Self-Taught Finetuning. This technique refers to an impressive way to enhance the reasoning capabilities of LLMs by training them on their self-generated correct solutions [11, 38, 53, 54]. Specifically, this self-bootstrapping process [11, 54] corresponds to a simple loop: employ an LLM to infer a set of questions; collect all the generations that yield correct answers into a dataset; finetune the base LLM on this self-generated dataset; restart to collect new generations with the newly finetuned LLM. This self-taught process is founded upon such a priori assumption: the LLM generations that lead to correct answers reveal high-quality reasoning rationales. However, in the complex scenarios of MHQA, a long-horizon multi-step reasoning trajectory, even if it eventually leads to the correct answer, typically still contains irrelevant LLM generations, which would impair the effectiveness of subsequent finetuning. *Our KnowTrace framework is endowed with a reflective backtracing mechanism to filter out these procedural impurities, offering an elegant way to self-distill higher-quality process supervision data for the self-taught finetuning.*

3 Methodology

3.1 Overview

This work introduces KnowTrace, a new iterative RAG framework that can autonomously trace out the question-relevant KG contexts in a coherent manner to bootstrap the MHQA performance of LLMs. As shown in Figure 2(c), KnowTrace alternately performs two LLM-driven operations: *knowledge exploration* and *knowledge completion*, to progressively acquire desired knowledge triplets during iterative retrievals until the expanding KG is sufficient for the LLM to output a definitive answer. We detail this inference process in Section 3.2. Moreover, by virtue of the transparent KG structures, KnowTrace further leverages a backtracing mechanism to filter out useless LLM generations (i.e., *unavailing exploration* and *extraneous completion*) from positive trajectories. In this way, KnowTrace can bootstrap its multi-step reasoning capabilities via finetuning on the self-distilled high-quality process supervision data, as described in Section 3.3.

3.2 Structured Knowledge Tracing for Inference

Given a multi-hop question q and a textual corpus C, KnowTrace actively traces out a set of q-relevant knowledge triplets from C to enrich an explicit KG context $\mathcal{G}_q = \{(e_s, r, e_o) | e_s, e_o \in \mathcal{E}_q, r \in \mathcal{R}_q\}$, in which $\mathcal{E}_q$ and $\mathcal{R}_q$ denote the sets of entities and relationships, respectively, and each triplet (e_s, r, e_o) reveals that there is a relation r between subject entity e_s and object entity e_o. The entire inference workflow of KnowTrace corresponds to an iterative *explore-then-complete* process (Algorithm 1), where each iteration involves two LLM operations: *knowledge exploration* and *knowledge completion*.

Knowledge Exploration. During this phase, KnowTrace leverages the LLM's planning capability [51] to determine the action for each iteration: either generate a definitive answer as the final prediction or continue to explore more relevant knowledge for KG expansion. Formally, at the l-th iteration ($1 \leq l \leq L$), KnowTrace integrates the question q and the KG $\mathcal{G}_q^{l-1}$ acquired from the previous $l-1$ iterations (the initial $\mathcal{G}_q^0$ is empty) into an instruction prompt I_{exp}. This prompt is designed to elicit such a coherent generation from an LLM M: first, M self-assesses whether $\mathcal{G}_q^{l-1}$ is sufficient to derive the final answer, and accordingly sets a boolean FLAG; if the FLAG is set to true, M then directly outputs the answer a, along with a chain of thought t [45] to reveal the reasoning rationales; otherwise, M

Algorithm 1 Inference Process of KnowTrace

```
Require: base LLM $M$; prompts $I_{\texttt{exp}}$ and $I_{\texttt{com}}$; corpus $C$
Input: question $q$
Output: KG context $\mathcal{G}_q$; final thought $t$; final prediction $a$
1: $\mathcal{G}_q^0 \leftarrow \emptyset$
2: for $l$ from 1 to $L$ do
3:     // Knowledge Exploration Phase in Equation (1)
4:     $\{\texttt{FLAG}, \mathcal{P}\} \leftarrow M\left(I_{\texttt{exp}}(q, \mathcal{G}_q^{l-1})\right)$
5:     if FLAG then
6:         // Chain-of-Thought Reasoning
7:         $\mathcal{P}$ includes the thought $t$ and the prediction $a$
8:         return $\mathcal{G}_q^{l-1}, t, a$
9:     else
10:        $\mathcal{P}$ includes a set of entity-relation pairs $\{(e_i, r_i)\}_{i=1}^P$
11:        // Parallelizable Inner Loop
12:        for $i$ from 1 to $P$ do
13:            $(e_i, r_i)$ serves as a query to retrieve $C_{(e_i,r_i)}^N$ from $C$
14:            // Knowledge Completion Phase in Equation (2)
15:            $\mathcal{T}_{(e_i,r_i)} \leftarrow M\left(I_{\texttt{com}}\left(e_i, r_i, C_{(e_i,r_i)}^N\right)\right)$
16:        // KG Context Expansion
17:        $\mathcal{G}_q^l \leftarrow \bigcup_{i=1}^P \mathcal{T}_{(e_i,r_i)} \cup \mathcal{G}_q^{l-1}$
```

Algorithm 2 Self-Bootstrapping Process of KnowTrace

```
Require: labeled dataset $\mathcal{D} = \{(q_d, \hat{a}_d)\}_{d=1}^D$
Input: base LLM $M$
Output: bootstrapped LLM $M_K$
1: $M_0 \leftarrow M$
2: for $k$ from 1 to $K$ do
3:     $\mathcal{D}_k \leftarrow \emptyset$
4:     for $d$ from 1 to $D$ do
5:         // Inference Process of KnowTrace (Algorithm 1)
6:         $\{\mathcal{G}_{q_d}, t_d, a_d\} \leftarrow \texttt{KnowTrace}(M_{k-1}, q_d)$
7:         if $a_d == \hat{a}_d$ then
8:             Collect all $\{(I_{\texttt{exp}}(\cdot), \mathcal{P})\}$ and $\{(I_{\texttt{com}}(\cdot), \mathcal{T}_*)\}$
9:             // Knowledge Backtracing (Section 3.3)
10:            $\mathcal{S}_{q_d} \leftarrow \texttt{Backtracing}(\mathcal{G}_{q_d}, [t_d, \hat{a}_d])$
11:            // Filter Out Unavailing Exploration
12:            $\mathcal{P}^+ \leftarrow \texttt{Filter}(\mathcal{P}, \mathcal{S}_{q_d})$
13:            // Filter Out Extraneous Completion
14:            $\mathcal{T}_*^+ \leftarrow \texttt{Filter}(\mathcal{T}_*, \mathcal{S}_{q_d})$
15:            $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{(I_{\texttt{exp}}(\cdot), \mathcal{P}^+)\} \cup \{(I_{\texttt{com}}(\cdot), \mathcal{T}_*^+)\}$
16:    // Finetune Base LLM on the Augmented Dataset
17:    $M_k \leftarrow \texttt{Train}(M, \mathcal{D}_k)$
18: return $M_K$
```

then provides a specific exploration guidance on how to expand the current KG—it adaptively determines the expansion points (i.e., entities) and the corresponding directions (i.e., relations), forming a set of entity-relation pairs that indicate the knowledge desired for the next reasoning step. We formulate this process as follows:

$$\{\texttt{FLAG}, \mathcal{P}\} = M\left(I_{\texttt{exp}}\left(q, \mathcal{G}_q^{l-1}\right)\right), \tag{1}$$

where $\mathcal{P}$ is either the final prediction $[t, a]$ or the KG expansion guidance $\{(e_i, r_i)\}_{i=1}^P$ conditioned on the self-generated FLAG as described above. Note that M can create new entities as the expansion points, rather than only selecting from the current KG. We refer to such entities as *initial entities*, as they typically correspond to the expansion beginnings of different components. After this phase, we then utilize each (e_i, r_i) as the query to retrieve N relevant passages from the textual corpus C, denoted as $C_{(e_i,r_i)}^N$, while also employing this pair to guide the subsequent knowledge completion phase.

Knowledge Completion. Given the entity-relation pair (e_i, r_i) as well as the retrieved passages $C_{(e_i,r_i)}^N$, KnowTrace further harnesses the LLM's language understanding capability to purposefully grasp key knowledge from the unstructured text. Formally, with a completion instruction $I_{\texttt{com}}$ that receives (e_i, r_i) and $C_{(e_i,r_i)}^N$, the LLM M is prompted to generate (e_i, r_i)-conditioned knowledge triplets:

$$\mathcal{T}_{(e_i,r_i)} = M\left(I_{\texttt{com}}\left(e_i, r_i, C_{(e_i,r_i)}^N\right)\right). \tag{2}$$

If the passages $C_{(e_i,r_i)}^N$ do not contain relevant information to (e_i, r_i), M can return an empty string. Each pair (e_i, r_i) may also correspond to multiple knowledge triplets, i.e., $|\mathcal{T}_{(e_i,r_i)}| > 1$, showcasing the underlying relation mapping properties [21, 44] behind the natural language text. After completing each (e_i, r_i) , KnowTrace induces a new set of knowledge triplets $\mathcal{T} = \bigcup_{i=1}^P \mathcal{T}_{(e_i,r_i)}$, offering a more comprehensive KG context $\mathcal{G}_q^l = \mathcal{G}_q^{l-1} \cup \mathcal{T}$ for the next iteration.

Knowledge Prompting Strategy. Since our KnowTrace endows the LLM with a self-organized KG context throughout the entire inference process, one essential consideration lies in how to integrate the KG information into the LLM's prompt. On this matter, we investigate three strategies to describe $\mathcal{G}_q$ for the prompt $I_{\texttt{exp}}$:

- *KG-to-Triplets:* directly represent $\mathcal{G}_q$ with the knowledge triplets.
- *KG-to-Paths:* connect the knowledge triplets that share common subject/object entities to form paths as the descriptions of $\mathcal{G}_q$.
- *KG-to-Texts:* leverage an additional generative model to rewrite the KG triplets into natural language texts as the descriptions.

For our framework, we demonstrate that the *KG-to-Triplets* strategy offers the dual advantages of simplicity and efficacy in Section 4.5.

Connections to Current RAG Methods. As shown in Figure 2, existing iterative RAG approaches such as IRCoT [42], Self-Ask [31] and ReAct [51] periodically generate new queries for retrieval, and simply accumulate all retrieved passages to form the LLM context. In comparison, our designed framework accentuates the importance of structured thinking, which reformulates the textual accumulation process as a coherent workflow of knowledge structure expansion to empower the LLM with intelligible contexts. Furthermore, recent structure-enhanced methods rely on either intricate restructuring process or rigid question-parsing operations, while our KnowTrace adopts a unique perspective of structured knowledge tracing with dual merits in efficiency and flexibility. More notably, we highlight that our KnowTrace is a bootstrapping framework, which can self-distill high-quality process supervision from positive trajectories to stimulate self-taught finetuning, as described in the next section. More discussions on the related works are included in Appendix A.

3.3 Reflective Knowledge Backtracing for Self-Taught Finetuning

Self-taught finetuning is an attractive technique, in which the LLM can post-refine its own performance without human intervention. In line with the concept of self-training [25], recent works [38, 54] improve the reasoning capabilities of LLMs by training them on their own generations that ultimately lead to the correct answers. Nevertheless, despite its effectiveness for one-time generation tasks, this process is inherently flawed when applied to recent RAG systems in the MHQA scenario: for a complex multi-hop question, even when the final prediction is correct, a long-horizon reasoning trajectory typically still contains useless LLM generations, which would diminish the efficacy of subsequent finetuning process.

To address this limitation, the key challenge lies in how to remove the useless LLM generations and distill the contributive ones in the positive trajectories. With the perspective of structured knowledge tracing, our KnowTrace progressively organizes a transparent KG for the input question throughout the multi-step reasoning process. This structured workflow allows us to design a post-hoc backtracing mechanism to retrospectively discern the contributive generations based on the final inference. In this way, KnowTrace can synthesize higher-quality process supervision to stimulate the self-training.

Formally, given a KnowTrace inference sample $(q, \mathcal{G}_q, [t, a])$ that yields the correct answer (i.e., the prediction a matches the ground-truth answer $\hat{a}$), the supporting knowledge essentially corresponds to a subgraph $\mathcal{S}_q \subseteq \mathcal{G}_q$ that exactly supports the final prediction. In light of this, we adopt a simple yet effective backtracing mechanism to identify $\mathcal{S}_q$: first, since the ground-truth label $\hat{a}$ could verify the rationality of the final outputs (i.e., $[t, a]$) [45, 48], we accordingly select the entities that appear in $[t, a]$ as the *target entities*; then, we trace back along the graph structure of $\mathcal{G}_q$ from these target entities to the *initial entities* (defined in Section 3.2), thereby inducing the expected subgraph $\mathcal{S}_q$ that consists of all the supporting knowledge triplets. Based on this subgraph, we can naturally filter out the non-contributive generations in the multi-step reasoning process:

- (*Unavailing Exploration*) For the generations $\mathcal{P}$ in Equation (1), we filter out $(e_i, r_i) \in \mathcal{P}$ (or even entire $\mathcal{P}$) that fails to produce any supporting knowledge triplets in $\mathcal{S}_q$.
- (*Extraneous Completion*) For the generations $\mathcal{T}_*$ in Equation (2), we filter out the completed triplets (or even entire $\mathcal{T}_*$) that do not support the final prediction (i.e., not included in $\mathcal{S}_q$).

By virtue of this process, our framework automatically removes the procedural impurities in the positive trajectories, thus synthesizing higher-quality process supervision data for self-training. As shown in Algorithm 2, KnowTrace incorporates the backtracing mechanism to bootstrap its reasoning capabilities through a simple loop: (1) collect reasoning trajectories that lead to correct answers from a labeled MHQA dataset; (2) distill the contributive generations into a training dataset with the backtracing mechanism; (3) finetune the base LLM on this dataset; (4) restart this process to synthesize more data using the improved LLM for the next round of finetuning, until the performance plateaus. In this way, the evident KG structures acquired by our KnowTrace not only directly enhance the inference quality, but also offer a natural way to self-synthesize high-quality supervision data for effective bootstrapping. In Figure 3, we show a specific toy example of the inference and bootstrapping procedures.

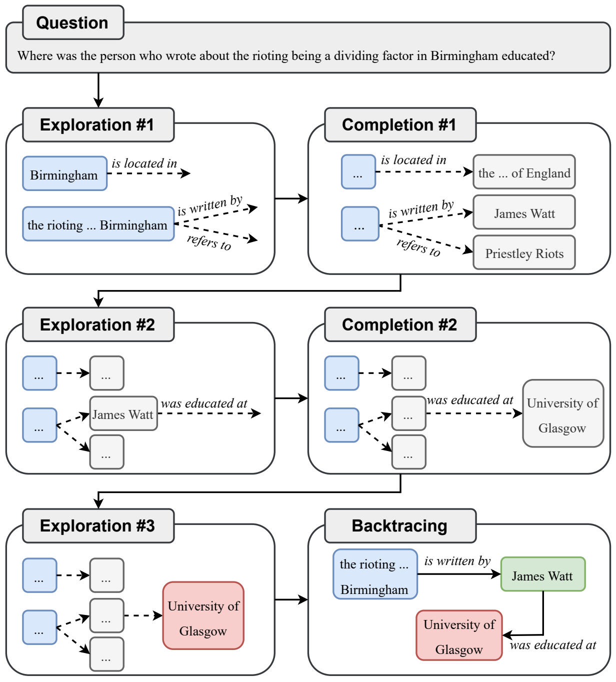


Figure 3: An example of KnowTrace's inference and backtracing process. The generated texts are included in Appendix D.

Connections to Current Bootstrapping Methods. The standard self-training approaches such as STaR [54], ReST [38] and RFT [53] finetune LLMs on their own generations that yield correct answers. A concurrent work, InstructRAG [46], directly utilizes this process to improve the one-time RAG systems. Nevertheless, when applying this workflow to the multi-step reasoning scenarios of MHQA, its efficacy is inherently undermined by the irrelevant generations that do not contribute to the final predictions. Our framework addresses this limitation by employing a reflective backtracing mechanism to retrospectively synthesize higher-quality supervision data, and we demonstrate that this mechanism is indispensable for the efficacy of self-taught finetuning (Section 4.4). To the best of our knowledge, KnowTrace is the first iterative RAG framework that can effectively elevate the multi-step reasoning capabilities via self-bootstrapping.

4 Experiments

To comprehensively demonstrate the effectiveness of our proposals, we conduct extensive experiments, which are outlined as follows:

- Firstly, we compare the basic KnowTrace with a range of RAG approaches in the MHQA task (using two mainstream LLMs as reasoning backbones), showcasing the facilitative effect of structured knowledge tracing on inference. (Section 4.3)
- Secondly, we employ the designed reflective backtracing mechanism to bootstrap a new version of KnowTrace, and validate the superiority and rationality of our design from both performance and statistical perspectives. (Section 4.4)
- Last but not least, we provide a detailed efficiency analysis, and investigate the effect of configuring different retrieval methods and prompting strategies on the performance. (Section 4.5)

Table 1: Evaluation results on three multi-hop question answering datasets. We adopt two advanced LLMs as the backbones, and select $N = 5$ most relevant passages for each retrieval. The best results are in bold, and the second best results are underlined.

Methods	`LLaMA3-8B-Instruct`						`GPT-3.5-Turbo-Instruct`					
	HotpotQA		**2Wiki**		**MuSiQue**		**HotpotQA**		**2Wiki**		**MuSiQue**	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
RA-CoT	.206	.314	.194	.255	.132	.203	.308	.429	.272	.364	.164	.258
ReAct	.270	.382	.232	.324	.204	.308	.354	.471	.336	.483	.232	.370
IRCoT	.324	.425	.286	.372	.240	.332	.442	.565	.374	.519	.278	.385
Self-Ask	.252	.367	.218	.325	.186	.275	.352	.468	.328	.464	.204	.323
Iter-RetGen	.304	.393	.264	.347	.228	.317	.426	.542	.368	.495	.246	.372
GE-Reasoning	.330	.426	.298	.353	.202	.295	.454	.587	.408	.519	.232	.354
SG-Prompt	.328	.411	.306	.369	.236	.342	.448	.583	.430	.537	.254	.369
ERA-CoT	.344	.435	.294	.365	.242	.346	.460	.592	.432	.543	.268	.376
KnowTrace	**.386**	**.479**	**.342**	**.403**	**.280**	**.387**	**.516**	**.633**	**.476**	**.582**	**.304**	**.425**

4.1 Experimental Setup

Datasets. We evaluate our KnowTrace over three standard MHQA benchmarks under the open-domain setting: HotpotQA [50], 2Wiki-MultihopQA (2Wiki) [10], and MuSiQue [41]. We use the same data splits as previous works [20, 42] for evaluation. In order to create the open-domain setting, we follow IRCoT [42] to collect all candidate passages (including supporting and distractor passages) as the retrieval corpus for each dataset. See Appendix B for more details.

Metrics. We calculate the Exact Match (EM) and F1 score as metrics. The EM accuracy is the proportion of correct answers in the test set, where a prediction is deemed correct if it exactly matches one of the ground-truth answers. The F1 score evaluates the overlap between the tokens in the prediction and the answer. We apply normalization to both the predictions and the answers when computing these two metrics, following the implementation of [17, 51].

Baselines. We compare KnowTrace with a series of current RAG methods, which can be classified into three categories: (1) *one-time retrieval-augmented* chain-of-thought reasoning [45], i.e., RA-CoT; (2) *iterative RAG methods*: IRCoT [42], ReAct [51], Self-Ask [31], and Iter-RetGen [36]; (3) *structure-enhanced RAG methods*: SG-Prompt [20], GE-Reasoning [28], and ERA-CoT [22]. Here, we describe two representative baselines that are highly relevant to our framework. As an unstructured RAG approach, IRCoT [42] interleaves retrieval-augmented chain-of-thought reasoning and reasoning-guided retrieval until the final answer is reported or the maximum allowed number of reasoning steps is reached. A recent restructuring-based work, ERA-CoT [22], uncover the knowledge structures behind the textual passages with a fully LLM-driven process: first, it identifies all entities involved in the text; then, it extracts both explicit and implicit relations between entities; next, it scores the reliability of the relations and removes those falling below a predefined threshold; after this intricate process, it performs the final answer prediction. More descriptions of all baselines can be found in Appendix C.

4.2 Implementation Details

Backbones. We use `LLaMA3-8B-Instruct` [5] as the base LLM M for main experiments, and also employ `GPT-3.5-Turbo-Instruct` [26] to investigate the effect of distinct LLM backbones. The specific LLM prompts (i.e., $I_{\mathtt{exp}}$ and $I_{\mathtt{com}}$) are included in our code repository. For each instruction, we provide four simple examples shared across all datasets to elicit the LLMs' instruction-following capabilities [2]. We set the temperature of 0.0 when calling the OpenAI's API, and use greedy decoding for LLaMA, to avoid random sampling [33].

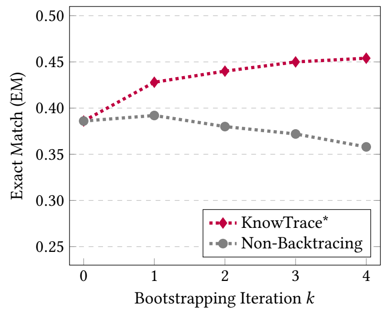
Retrievers. Under the open-domain setting, we investigate three different retrieval models to verify the compatibility of our proposal, including BM25 [34], DPR [16], and Contriever [13]. We implement BM25 retrieval with Elasticsearch [9], and employ BEIR framework [40] for DPR and Contriever. In main experiments, we retrieve the top $N = 5$ most relevant passages for each query with BM25, and also vary N to $\{10, 20, 30, 50\}$ for further analysis.

Self-taught finetuning. For each dataset, we randomly sample 5,000 question-answer pairs to form $\mathcal{D}$ in Algorithm 2. During the bootstrapping process, we use the designed backtracing mechanism to distill contributive generations and construct a finetuning dataset. The statistical characteristics are analyzed in Section 4.4. On top of the base LLM, we train two distinct LoRA adapters [12] to specialize the capabilities of knowledge exploration and knowledge completion, respectively. We tune the training epoch in $\{1, 2, 3\}$, batch size in $\{32, 64, 128\}$, and learning rate in $\{1e{-}5, 5e{-}5, 1e{-}4, 3e{-}4\}$. We repeat the bootstrapping iteration until the performance plateaus.

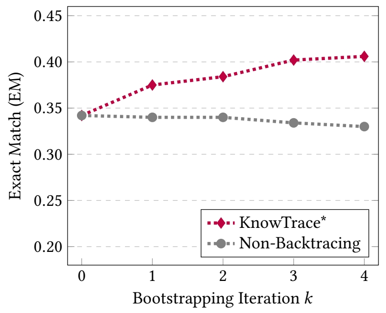
4.3 Inference Performance Comparison

Table 1 summarized the main experimental results on three standard MHQA datasets. First, whether using `LLaMA3-8B-Instruct` or `GPT-3.5-Turbo-Instruct` as the LLM backbones, iterative RAG methods, especially IRCoT, significantly outperform the one-time RA-CoT, confirming that multi-round retrievals can indeed enhance the inference quality of the LLMs for the open-domain MHQA task. Second, two recent restructuring-based methods, i.e., SG-Prompt and ERA-CoT, conduct one-time retrieval and strive to restructure all retrieved passages. Despite these methods retrieving only once (due to the complexity of restructuring process), they still perform comparably to or even better than the iterative methods, indicating the rationality of structured context for LLM inference.

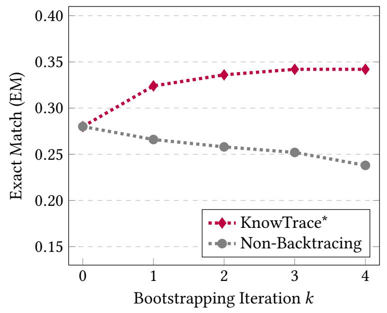
Beyond all these methods, our framework adopts a unique perspective of structured knowledge tracing to seamlessly integrate knowledge structuring with LLM reasoning. One can observe that



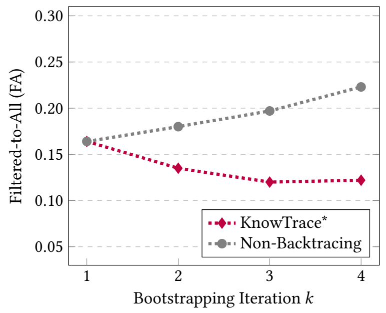
(a) EM vs. k on HotpotQA



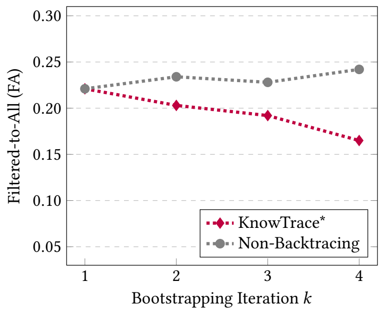
(b) EM vs. k on 2Wiki



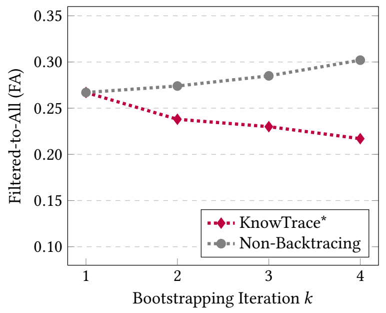
(c) EM vs. k on MuSiQue



(d) FA vs. k on HotpotQA



(e) FA vs. k on 2Wiki



(f) FA vs. k on MuSiQue

Figure 4: EM results (a-c) and FA ratios (d-f) in each bootstrapping iteration. KnowTrace* is the bootstrapped version based on the knowledge backtracing mechanism, and Non-Backtracing version is derived through the vanilla self-training process [54].

KnowTrace consistently outperforms all the baselines on both evaluation metrics (i.e., EM and F1) across the three datasets. For example, compared with IRCoT and ERA-CoT, when `LLaMA3-8B-Instruct` is selected as the base LLMs, KnowTrace achieves approximately 5.3% and 4.3% average absolute EM gains, respectively. When the base LLMs are switched to `GPT-3.5-Turbo-Instruct`, the gains increase to 6.7% and 4.5% accordingly. This advanced performance showcases the superiority of our design perspective at the inference level. We further present the self-bootstrapping characteristic of our framework in the next subsection.

4.4 Effectiveness of Knowledge Backtracing

The knowledge backtracing mechanism enables KnowTrace to synthesize a high-quality finetuning dataset for self-bootstrapping. We refer to the bootstrapped version as KnowTrace*. For validating its effectiveness, we follow the vanilla self-taught finetuning workflow [54] to derive a *non-backtracing* version as the baseline.

On the one hand, we compare their inference performance (EM) in each bootstrapping iteration. On the other hand, we also consider such a statistical indicator: during data collection in each bootstrapping iteration, since the backtracing mechanism can naturally identify which LLM generations in the positive trajectories should be filtered, we then calculate the ratio of the tokens that should be filtered to all output tokens. We refer to this ratio as FA (Filtered-to-All). A larger FA means that the collected positive trajectories contain more useless generations, indicating an inferior quality of the finetuning dataset. We use this ratio to measure the proportion of noisy data in each self-training iteration.

Table 2: Cost statistics of KnowTrace and two baselines. #Tok is the average count of passage tokens processed by LLMs per question; #Time is the average inference time per question.

	HotpotQA		2Wiki		MuSiQue	
Method	#Tok	#Time	#Tok	#Time	#Tok	#Time
IRCoT	1.4k	5s	1.8k	7s	2.0k	8s
ERA-CoT	2.7k	16s	3.2k	20s	3.5k	22s
KnowTrace	1.6k	6s	1.9k	7s	2.1k	9s

Figure 4 presents the EM results and FA ratios of the two versions. In terms of the inference performance (a-c), we observe that the backtracing-guided KnowTrace* achieves performance gains during self-training, while the non-backtracing version instead undergoes notable performance decline, which we attribute to its disregard for the non-contributive LLM generations in the positive trajectories. We use the FA indicator to support this attribution from a statistical perspective (d-f). For the fist bootstrapping iteration ($k = 1$), there are more than 15% (even 26.7%) useless generations in the collected positive trajectories. Indiscriminately finetuning on such noisy data results in a negative synergistic effect: the noisy data undermine the generation quality, which in turn causes the generated positive trajectories to contain more noise. Our backtracing mechanism is able to filter out this type of noise, making it indispensable for the self-bootstrapping characteristic of our KnowTrace framework. We highlight that this mechanism is naturally built on the self-acquired KGs, thereby further confirming the rationality and superiority of our design perspective of structured knowledge tracing.

Table 3: EM results for the models using three different retrieval methods. We commonly select $N = 5$ most relevant passages for each retrieval, and set `LLaMA3-8B-Instruct` as backbones.

	HotpotQA			2Wiki			MusiQue		
Method	BM25	DPR	Contriver	BM25	DPR	Contriver	BM25	DPR	Contriver
IRCoT	.324	.252	.332	.286	.214	.280	.240	.126	.252
ERA-CoT	.344	.286	.348	.294	.220	.312	.242	.134	.246
KnowTrace	**.386**	**.320**	**.398**	**.342**	**.246**	**.354**	**.280**	**.176**	**.288**

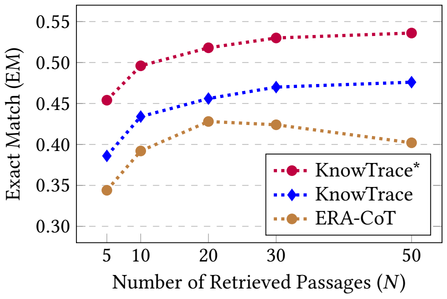


Figure 5: EM results with varying N on HotpotQA.

4.5 More Analysis

Cost Analysis. We include a detailed cost analysis for KnowTrace and two representative baselines (i.e., IRCoT and ERA-CoT). The statistics are summarized in Table 2. In terms of the inference time and the token quantity, KnowTrace is clearly more efficient than the restructuring-based ERA-CoT, and comparable to the simplest iterative IRCoT. Combined with the MHQA results in Table 1, one can observe that our framework is capable of achieving superior effectiveness without compromising the efficiency. Moreover, our designed backtracing mechanism does not require additional LLM calls, and automatically synthesizes high-quality data for the self-taught finetuning. In this way, the bootstrapping cost aligns with standard parameter-efficient finetuning (approximately 2–3 hours on an NVIDIA A100 GPU).

Retrieval Models. We validate the compatibility of our framework across different retrievers. Specifically, in addition to BM25 used in Table 1, we further conduct experiments with two other retrievers: DPR and Contriever. Table 3 reports the EM results of KnowTrace and two representative baselines (i.e., IRCoT and ERA-CoT) under these three retrievers. One can observe that our proposal consistently outperforms both baselines on all the datasets, regardless of the type of retrieval model. This superior performance demonstrates the general applicability of our approach on various retrievers.

Number of Retrieved Passages. We further investigate the effect of the number of retrieved passages (i.e., N). Figure 5 presents the EM results of two KnowTrace versions and one restructuring-based baseline (i.e., ERA-CoT) with varying N on HotpotQA dataset. From this figure, one can observe that our models consistently surpass the baseline by a clear margin across all the values of N. Moreover, ERA-CoT exhibits performance degradation when N is relatively large (i.e., more than 20), which we attribute to the absence of explicit reasoning guidance during the sophisticated restructuring process. In contrast, the performance of both KnowTrace versions improves until saturation as we increase the value of N. This stronger and more stable performance demonstrates the effectiveness of seamlessly integrating reasoning and structuring within our framework.

Table 4: EM/F1 results for KnowTrace with three different knowledge prompting strategies.

Strategy	HotpotQA	2Wiki	MusiQue
KG-to-Triplets	.386/.479	.342/.403	.280/.387
KG-to-Paths	.382/.465	.334/.392	.286/.398
KG-to-Texts	.376/.471	.320/.386	.274/.383

Knowledge Prompting Strategies. Since this work highlights the significance of structured knowledge structures (i.e., KGs) for LLM inference, a critical concern lies in how to incorporate the KG structures into LLM prompts. Here, we investigate three prompting strategies: *KG-to-Triplets*, *KG-to-Paths*, and *KG-to-Texts*, respectively corresponding to elementary triplets, connected paths, and natural language texts, as described in Section 3.2. Table 4 reports the EM and F1 results of our framework with these three strategies. For fair comparisons, we utilize `LLaMA3-8B-Instruct` as the base LLM and BM25 as the retriever. One the one hand, the simplest *KG-to-Triplets* works well, while connecting independent triplets into paths (i.e., *KG-to-Paths*) does not lead to consistent gains. We observe that the path extraction process typically duplicates some triplets, which could distract LLM inference [15, 37]. On the other hand, converting KGs back into plain text with the LLM (i.e., *KG-to-Texts*) also results in inferior performance, which we attribute to the absence of priori structural templates in the prompts. For example, when adopting the *KG-to-Triplets* strategy, one can directly inform the LLM that its contexts take the triplet form of (*subject*, *relation*, *object*). In this way, *KG-to-Triplets* exhibits the dual advantages of simplicity and effectiveness, since it avoids information duplication and also offers structural priors, making it the main choice for our framework.

5 Discussion and Conclusion

Limitations. Despite enjoying dual merits in inference and bootstrapping, this work still has a few limitations. First, the applicability of our design perspective in other complex scenarios, such as mathematics and decision-making tasks, has yet to be explored. Second, although our KnowTrace can retrospectively distill high-quality data for bootstrapping, how to proactively correct erroneous trajectories without finetuning remains an open challenge.

Conclusion. This work introduces KnowTrace, an elegant iterative RAG framework that incorporates a new perspective of *structured* *knowledge tracing* to enhance the LLM's multi-step reasoning capabilities for the MHQA task. Based on this perspective, KnowTrace empowers the LLM with intelligible KG structures to facilitate its inference, and also employs a reflective backtracing mechanism to self-synthesize high-quality supervision data for self-bootstrapping. Extensive experiments on three standard MHQA benchmarks comprehensively validate the rationality and superiority of our design.

Acknowledgments

This work is supported in part by National Natural Science Foundation of China (No. 62422215 and No. 62472427), Major Innovation & Planning Interdisciplinary Platform for the "DoubleFirst Class" Initiative, Renmin University of China, Public Computing Cloud, Renmin University of China, fund for building world-class universities (disciplines) of Renmin University of China, the Outstanding Innovative Talents Cultivation Funded Programs 2024 of Renmin University of China, and Huawei Innovation Research Programs. We gratefully acknowledge the support from Mindspore[2], CANN (Compute Architecture for Neural Networks) and Ascend AI Processor used for this research.

References

[1] Martin D Braine. 1978. On the relation between the natural logic of reasoning and standard logic. *Psychological review* 85, 1 (1978), 1.
[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, Vol. 33. 1877–1901.
[3] Xiaoxia Cheng, Zeqi Tan, and Weiming Lu. 2024. Information Re-Organization Improves Reasoning in Large Language Models. *arXiv preprint arXiv:2404.13985* (2024).
[4] Zhenyun Deng, Yonghua Zhu, Yang Chen, Michael Witbrock, and Patricia Riddle. 2022. Interpretable AMR-based question decomposition for multi-hop question answering. *arXiv preprint arXiv:2206.08486* (2022).
[5] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).
[6] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).
[7] Catherine Twomey Fosnot. 2013. *Constructivism: Theory, perspectives, and practice.* Teachers College Press.
[8] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).
[9] Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine.* " O'Reilly Media, Inc.".
[10] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *Proceedings of the 28th International Conference on Computational Linguistics.* 6609–6625.
[11] Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-STaR: Training Verifiers for Self-Taught Reasoners. In *First Conference on Language Modeling.*
[12] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations.*
[13] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised Dense Information Retrieval with Contrastive Learning. *Transactions on Machine Learning Research* (2022).
[14] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research* 24, 251 (2023), 1–43.
[15] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surveys* 55, 12 (2023), 1–38.
[16] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing.* 6769–6781.
[17] Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024. SuRe: Summarizing Retrievals using Answer Candidates for Open-domain QA of LLMs. In *The Twelfth International Conference on Learning Representations.*
[18] Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115* (2022).
[19] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems.*
[20] Ruosen Li and Xinya Du. 2023. Leveraging Structured Information for Explainable Multi-hop Question Answering and Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP.* 6779–6789.
[21] Rui Li, Chaozhuo Li, Yanming Shen, Zeyu Zhang, and Xu Chen. 2024. Generalizing Knowledge Graph Embedding with Universal Orthogonal Parameterization. In *Proceedings of the 41st International Conference on Machine Learning*, Vol. 235. 28040–28059.
[22] Yanming Liu, Xinyue Peng, Tianyu Du, Jianwei Yin, Weihao Liu, and Xuhong Zhang. 2024. ERA-CoT: Improving Chain-of-Thought through Entity Relationship Analysis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 8780–8794.
[23] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 9802–9822.
[24] Kanishka Misra, Cicero Nogueira dos Santos, and Siamak Shakeri. 2023. Triggering Multi-Hop Reasoning for Question Answering in Language Models using Soft Prompts and Random Walks. *arXiv preprint arXiv:2306.04009* (2023).
[25] Kamal Nigam and Rayid Ghani. 2000. Analyzing the Effectiveness and Applicability of Co-training. In *Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management.* 86–93.
[26] OpenAI. 2022. Introducing chatgpt. *https://openai.com/blog/chatgpt* (2022).
[27] Pranoy Panda, Ankush Agarwal, Chaitanya Devaguptapu, Manohar Kaul, and Prathosh A P. 2024. HOLMES: Hyper-Relational Knowledge Graphs for Multi-hop Question Answering using LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 13263–13282.
[28] Jinyoung Park, Ameen Patel, Omar Zia Khan, Hyunwoo J Kim, and Joo-Kyung Kim. 2023. Graph-guided reasoning for multi-hop question answering in large language models. *arXiv e-prints arXiv:2311.09762* (2023).
[29] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph Retrieval-Augmented Generation: A Survey. *arXiv preprint arXiv:2408.08921* (2024).
[30] Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. *arXiv preprint arXiv:2002.09758* (2020).
[31] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and Narrowing the Compositionality Gap in Language Models. In *Findings of the Association for Computational Linguistics: EMNLP.* 5687–5711.
[32] Peng Qi, Haejun Lee, Oghenetegiri Sido, Christopher D Manning, et al. 2020. Answering open-domain questions of varying reasoning steps from text. *arXiv preprint arXiv:2010.12527* (2020).
[33] Matthew Renze and Erhan Guven. 2024. The effect of sampling temperature on problem solving in large language models. *arXiv preprint arXiv:2402.05201* (2024).
[34] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.
[35] Bhaskarjit Sarmah, Benika Hall, Rohan Rao, Sunil Patel, Stefano Pasquali, and Dhagash Mehta. 2024. HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction. *arXiv preprint arXiv:2408.04948* (2024).
[36] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing Retrieval-Augmented Large Language Models with

[2]https://www.mindspore.cn

Iterative Retrieval-Generation Synergy. In *Findings of the Association for Computational Linguistics: EMNLP*. 9248–9274.

[37] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large Language Models Can Be Easily Distracted by Irrelevant Context. In *International Conference on Machine Learning*, Vol. 202. 31210–31227.

[38] Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T. Parisi, Abhishek Kumar, Alexander A. Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. 2024. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. *Transactions on Machine Learning Research* (2024).

[39] Xin Su, Tiep Le, Steven Bethard, and Phillip Howard. 2023. Semi-Structured Chain-of-Thought: Integrating Multiple Sources of Knowledge for Improved Language Model Reasoning. *arXiv preprint arXiv:2311.08505* (2023).

[40] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[41] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics* 10 (2022), 539–554.

[42] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 10014–10037.

[43] Siyuan Wang, Zhongyu Wei, Jiarong Xu, Taishan Li, and Zhihao Fan. 2023. Unifying structure reasoning and language model pre-training for complex reasoning. *arXiv preprint arXiv:2301.08913* (2023).

[44] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 1112–1119.

[45] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*.

[46] Zhepei Wei, Wei-Lin Chen, and Yu Meng. 2025. InstructRAG: Instructing Retrieval-Augmented Generation via Self-Synthesized Rationales. In *The Thirteenth International Conference on Learning Representations*.

[47] Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural Text Generation With Unlikelihood Training. In *8th International Conference on Learning Representations*.

[48] Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun Liu, Xin Guo, Junzhe Wang, Honglin Guo, Wei Shen, Xiaoran Fan, Yuhao Zhou, Shihan Dou, Xiao Wang, Xinbo Zhang, Peng Sun, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Training Large Language Models for Reasoning through Reverse Curriculum Reinforcement Learning. In *Forty-first International Conference on Machine Learning*.

[49] Diji Yang, Jinmeng Rao, Kezhen Chen, Xiaoyuan Guo, Yawen Zhang, Jie Yang, and Yi Zhang. 2024. IM-RAG: Multi-Round Retrieval-Augmented Generation Through Learning Inner Monologues. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 730–740.

[50] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2369–2380.

[51] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*.

[52] Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2023. Answering Questions by Meta-Reasoning over Multiple Chains of Thought. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 5942–5966.

[53] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825* (2023).

[54] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. STaR: Bootstrapping Reasoning With Reasoning. In *Advances in Neural Information Processing Systems*.

[55] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the AI ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219* (2023).

[56] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774* (2021).

A Discussion on More Related Works

This section provides a detailed discussion about more structured-enhanced works to thoroughly showcase the novelty of our design.

Several works [24, 43] first extracts information structures from Wikipedia documents and then mask specific entities to construct a pre-training dataset, aiming to imbue the language models with structural reasoning capabilities. In contrast, this paper focuses on enhancing the LLMs' multi-step reasoning capabilities to facilitate the MHQA task with the aid of multi-round information retrieval.

GE-Reasoning [28] and Semi-Structured CoT [39] focus on parsing input questions into masked structured chains and subsequently fill each incomplete knowledge triplet based on a pre-defined KG or a plain text database. However, such approaches heavily count on the accuracy of the initial parsing stage, thus necessitating careful filtering operations and consistency checks [39]. In contrast, our framework adaptively traces out desired knowledge triplets throughout the multi-step reasoning process, rather than solely relying on one-time question parsing. This flexible workflow can effectively reduce error propagation and enhance robustness.

To sum up, our designed framework adopts a unique perspective of structured knowledge tracing to progressively acquire question-related knowledge triplets during the multi-step reasoning process. This perspective enables more flexible LLM inference and does not rely on additional structural training or one-time question parsing. Moreover, the evident KG contexts acquired during inference also provide a simple yet effective way (i.e., knowledge backtracing) to self-distill high-quality supervision data for self-taught finetuning. Therefore, our KnowTrace framework enjoys dual advantages in the both processes of inference and bootstrapping.

B Datasets

HotpotQA. This is a comprehensive dataset collected from English Wikipedia, comprising approximately 113k crowd-sourced questions. The unique characteristic of HotpotQA lies in its construction, which requires answers to be derived from the introductory paragraphs of two distinct Wikipedia articles. For each question, the dataset includes the corresponding golden paragraphs from these articles, along with a curated list of sentences identified by crowd-workers as supporting evidence necessary to accurately resolve the query. Note that our evaluation is conducted under open-domain setting [42], and thus does not use these golden information.

2WikiMultihopQA (2Wiki). This dataset consists of complex 2-hop questions that require either compositional reasoning or comparative analysis. Both structured and unstructured information from Wikipedia and Wikidata are combined for data construction.

MuSiQue. The multi-hop questions in this dataset is constructed by carefully selecting and composing single-hop questions obtained from a large collection of single-hop questions. In terms of difficulty, MuSiQue is more challenging since it contains 2 to 4 hop questions.

Table 5: A toy example of KnowTrace inference from the MuSiQue dataset.

Question: Where was the person who wrote about the rioting being a dividing factor in Birmingham educated?

[Iteration 1]

```
# Knowledge Memory: None
# Knowledge Exploration
Whether the collected knowledge triplets are sufficient: No
What entity-relation pairs to retrieve in the current step:
- The rioting being a dividing factor in Birmingham: Find out who wrote about this
 ↪  topic and what this rioting refers to.
- Birmingham: Find out where Birmingham is located.
# Retrieve Relevant Textual Passages From Wikipedia
# Knowledge Completion
(James Watt, wrote, the rioting being a dividing factor in Birmingham);
(James Watt, is, an industrialist);
(the rioting being a dividing factor in Birmingham, refers to, Priestley Riots);
(Birmingham, is located in, the West Midlands region of England)
```

[Iteration 2]

```
# Knowledge Memory: The Knowledge Triplets Acquired in Iteration 1
# Knowledge Exploration
Whether the collected knowledge triplets are sufficient: No
What entity-relation pairs to retrieve in the current step:
- James Watt: Find out which school James Watt attended.
# Retrieve Relevant Textual Passages From Wikipedia
# Knowledge Completion
(James Watt, was educated at, University of Glasgow)
```

[Iteration 3]

```
# Knowledge Memory: The Knowledge Triplets Acquired in Iteration 1 and 2
# Knowledge Exploration
Whether the collected knowledge triplets are sufficient: Yes
Thought: James Watt wrote about the rioting being a dividing factor in Birmingham.
 ↪  He was educated at the University of Glasgow.
Answer: University of Glasgow
```

C Baselines

RA-CoT [45]. This is the simplest approach, which conducts one-time retrieval with the input question as the query, and then utilize the retrieved passages to elicit the chain-of-thought generation.

ReAct [51]. This approach integrates reasoning, action, and observation steps in an iterative process until a final answer is reached. Actions in this process include generating queries to search for relevant information or concluding with a final answer. Observations are formed by concatenating the results from these actions and serve as inputs for subsequent reasoning steps.

Self-Ask [31]. This approach follows a step-by-step workflow that breaks down the multi-hop questions and retrieve related information for each sub-question. To improve the decomposition quality, all retrieved passages are prepended to the original question [52].

Iter-RetGen [36]. This approach concatenates the LLM generations from previous iterations with the original question to retrieve more relevant knowledge for the subsequent LLM inference.

SG-Prompt [20]. This approach first constructs a semantic graph structures through information extraction from all retrieved text, and then leverages this symbolic information (including entities and semantic relations) to enhance the inference quality.

D A Toy Example of KnowTrace

To better illustrate the workflow of our KnowTrace framework, we present a succinct example of its inference and backtracing process, complementing the high-level presentation in Figure 2.

The inference example of KnowTrace is shown in Table 5. Based on the transparent KG structure traced out in this example, one can naturally trace back from the answer entity *University of Glasgow* to identify the following supporting subgraph: *(James Watt, wrote, the rioting being a dividing factor in Birmingham)*; *(James Watt, was educated at, University of Glasgow)*. In this manner, our framework allows for removing unavailing exploration (e.g., "- *Birmingham: Find out where Birmingham is located*") and extraneous completion (e.g., *(James Watt, is, an industrialist)*), thereby self-distilling higher-quality process supervision data for more effective bootstrapping.