RAG-Star: Enhancing Deliberative Reasoning with Retrieval Augmented Verification and Refinement

Jinhao Jiang[1]*, Jiayi Chen[2]*, Junyi Li[4]*, Ruiyang Ren[1], Shijie Wang[3]
Wayne Xin Zhao[1]†, Yang Song[5]†, Tao Zhang[5]
[1]Gaoling School of Artificial Intelligence, Renmin University of China.
[2]Wuhan University of Science and Technology. [3]Northeastern University at Qinhuangdao.
[4]Department of Computer Science, National University of Singapore.
[5]Nanbeige Lab, BOSS Zhipin.
jiangjinhao@ruc.edu.cn, batmanfly@gmail.com

Abstract

Existing large language models (LLMs) show exceptional problem-solving capabilities but might struggle with complex reasoning tasks. Despite the successes of chain-of-thought and tree-based search methods, they mainly depend on the internal knowledge of LLMs to search over intermediate reasoning steps, limited to dealing with simple tasks involving fewer reasoning steps. In this paper, we propose **RAG-Star**, a novel RAG approach that integrates the retrieved information to guide the tree-based deliberative reasoning process that relies on the inherent knowledge of LLMs. By leveraging Monte Carlo Tree Search, RAG-Star iteratively plans intermediate sub-queries and answers for reasoning based on the LLM itself. To consolidate internal and external knowledge, we propose an retrieval-augmented verification that utilizes query- and answer-aware reward modeling to provide feedback for the inherent reasoning of LLMs. Our experiments involving Llama-3.1-8B-Instruct and GPT-4o demonstrate that RAG-Star significantly outperforms previous RAG and reasoning methods. Our codes and data are publicly available at https://github.com/RUCAIBox/RAG-Star.

1 Introduction

Despite the excellent capabilities of large language models (LLMs) (Zhao et al., 2023b), they still face significant challenges in complex reasoning tasks (*e.g.,* multi-hop question answering), which often go beyond simple, single-step problem-solving, demanding a deeper level of cognitive reasoning across multiple facts, sources, or contexts (Huang et al., 2024; Suzgun et al., 2023). Great efforts have been made to improve the reasoning effectiveness of LLMs by conducing step-by-step reasoning, exemplified by chain-of-thought (CoT) (Wei et al., 2022). However, as the number of reasoning steps grows, LLMs are often prone to introduce logical errors, factual hallucinations, or inconsistent statements (Wei et al., 2022; Lyu et al., 2023).

In fact, step-by-step reasoning in the auto-regressive generation paradigm can be described as akin to "System 1", a mode of thinking which is fast, instinctive but less accurate (Kahneman, 2011). Conversely, solving complex reasoning problems requires more in-depth, deliberative, and logical thinking, known as the "System 2" mode, which requires conscious effort to conduct massive strategic decision-making (Kahneman, 2011). To enhance the "System 2" reasoning capabilities of LLMs, prior studies have proposed to conduct deliberative generation by leveraging basic tree search algorithms (*e.g.,* Monte Carlo Tree Search (Silver et al., 2017)). However, LLMs in these studies mainly depend on their *internal knowledge* to search over intermediate reasoning steps, limited to handling problems with relatively simple reasoning process. To leverage external knowledge in model reasoning, extensive research has sought to augment LLMs with external information sources (*a.k.a.* retrieval-augmented generation, RAG) (Lewis et al., 2020b; Yao et al., 2022), while existing efforts mainly consider sequential reasoning structure, which cannot naturally support more complex reasoning structure like MCTS. Thus, we raise the following research question: *Can RAG enhance the deliberative reasoning capabilities of LLMs?*

In light of this, in this paper, we propose **RAG-Star**, a novel RAG-enhanced framework designed to improve multi-step reasoning capabilities of LLMs with deliberative planning. As the major technical contribution, RAG-Star can fully exploit the internal knowledge of LLMs to plan the multi-step reasoning, and meanwhile integrating the external retrieval to guide the internal reasoning process. To achieve this goal, we first introduce a tree-based search algorithm (*i.e.,* Monte Carlo Tree Search, MCTS) with LLMs to search over possible

* Equal contributions.
† Corresponding author.

plans for solving the problem at hand where a complete plan is composed of a sequence of sub-queries and corresponding answers. Starting from the input question (root node), RAG-Star iteratively generates and selects an appropriate sub-query and its answer (intermediate node), which aims to maximally explore the optimal sub-query path towards the final answer solely based on the inherent knowledge of LLMs. Second, different from existing deliberation methods (Wang et al., 2024a; Yao et al., 2023), RAG-Star proposes retrieval-augmented verification that involves both query- and answer-aware reward modeling, fully exploiting external sources to guide the internal deliberative reasoning. In our approach, instead of directly interfering in the reasoning process of LLMs, we consider employing RAG to refine the derived reasoning steps in MCTS, which can effectively reduce the conflicts between inherent and external knowledge, which has been a common issue when using RAG methods (Wang et al., 2024b; Gao et al., 2023).

We conduct extensive experiments to verify the effectiveness of RAG-Star based on Llama-3.1-8B-Instruct and GPT-4o. Our method outperforms the baselines by up to 18.98% and 16.19% on average for Llama-3.1-8B and GPT-4o, respectively.

Our main contributions can be summarized as:

• We propose RAG-Star that leverages external retrieval to enahnce the deliberative reasoning of LLMs based on their internal knowledge.

• We design an effective retrieval-augmented verification and refinement to evaluate and correct the inherent reasoning process.

• We conduct extensive experiments on several datasets, where RAG-Star significantly outperforms existing RAG and reasoning methods.

2 Related Work

Retrieval-Augmented LLMs. Augmenting large language models (LLMs) with retrieval has been extensively studied in existing literature (Lewis et al., 2020a; Borgeaud et al., 2022; Guu et al., 2020), which incorporates a differentiable retriever to provide external sources for LLMs. Furthermore, LLMs have made significant advancements in many reasoning tasks, such as code generation (OpenAI, 2023), math word problems (Zhu et al., 2023) and question answering (Brown et al., 2020). Chain-of-thought (CoT) has been reported as an emergent ability of LLMs when they are large enough (Wei et al., 2022), which encourages LLMs to generate explicit intermediate reasoning steps in reasoning rather than simply providing answers directly. To elicit or improve the multi-step reasoning capability of LLMs, several approaches seek to harness the strengths of both CoT and retrieval on knowledge-intensive complex reasoning tasks, such as multi-hop question answering (Yao et al., 2022; Zhao et al., 2023a). The rationales gained from reasoning enhance the retrieval of more relevant information, while the retrieved knowledge improves the factuality of intermediate reasoning steps. However, these approaches primarily take retrieved documents as direct input to the model, easily suffering from knowledge conflicts between the parametric knowledge of LLMs and the external sources. In contrast, our RAG-Star framework integrates tree-based search to fully explore the solution space and repurpose the retrieval information as external guidance to the reasoning process.

Enhancing LLMs with Search. Applying search on top of LLMs has been a topic of much interest. Several recent works have explored search algorithms to improve the performance of LLMs during the inference stage (Wang et al., 2024a; Zhang et al., 2024). The bitter lesson (Sutton, 2019) famously suggests that two forms of scaling, *i.e.,* learning and search, supersede all other approaches. Many studies have proven that scaling the inference-time computation can lead to substantial improvements in the performance of LLMs without training (Brown et al., 2024; Snell et al., 2024). These search algorithms, where multiple branches of outcomes are explored during search, have been widely applied in reinforcement learning algorithms (Hart et al., 1968; Silver et al., 2017) and many real-world applications such as AlphaGo (Silver et al., 2016) for their good exploration-exploitation trade-off. However, these approaches mainly rely on the internal knowledge of LLMs to search potential solutions, which might not be optimal and leads to a amount of rollouts, significantly slowing down the decoding process. In this paper, we leverage the external retrieval sources to enhance the deliberative search process with LLMs, effectively differentiate the internal reasoning and external retrieval.

3 Preliminary

In this section, we will first formally define our task and then introduce Monte Carlo Tree Search which is used in our proposed RAG-Star approach.

Task Formulation. In this work, we mainly focus on open-domain multi-hop question answering (Chen et al., 2019; Yang et al., 2018), which requires multiple steps of reasoning across different documents to answer questions. Previous work typically adopts an iterative *reason-then-generate* pipeline (Wei et al., 2022; Huang and Chang, 2023). At each step, the LLM first infers an intermediate sub-query based on the current situation and then generates possible answers to the query. Formally, given a natural language input question, at the t-th step, the LLM $\mathcal{M}_\theta$ (parameterized by θ) first deliberately reasons about a sub-query q_t, followed by generating an answer a_t based on its inherent knowledge. In some literature (Yao et al., 2022; Asai et al., 2024), retrieval-augmented generation (RAG) has been employed to improve the factuality of intermediate reasoning steps. For each sub-query q_t, the retriever retrieves top-K documents $\mathcal{D}_t = \{d_{t,k}\}_{k=1}^{K}$ from an external large-scale corpus, *e.g.,* Wikipedia, supplying them to the LLM to generate more accurate answers.

Monte Carlo Tree Search (MCTS). In existing literature (Zelikman et al., 2024; Zhang et al., 2024), MCTS builds a search tree $\mathcal{T}$ based on a policy model π_θ, which is usually the target LLM $\mathcal{M}_\theta$. Each node $s_t = [q_t, a_t, N(s_t), V(s_t)]$ represents a state comprising the sub-query q_t, its answer a_t, the number of visits $N(s_t)$, and the value function (expected reward) $V(s_t)$ for accurately answering questions, except that the root node $s_0 = [q_0]$ only contains the original input question q_0, and each edge is an action aiming to generate the next sub-query. During the search process, MCTS runs for multiple simulations. For the t-th simulation, it conducts four operations to expand the tree:

• *Selection* aims to select a node with the highest UCT (Upper Confidence bounds applied to Trees) score (Kocsis and Szepesvári, 2006) starting from the root node s_0. The UCT score of a child node with state s_t is calculated as follows:

$$UCT(s_t) = V(s_t) + w\sqrt{\frac{\ln N(p)}{N(s_t)}}, \qquad (1)$$

where w controls the exploration and exploitation, and p is the parent node of the current node s_t.

• *Expansion* explores multiple child nodes $\{s_{t+1}\}$ from the selected node s_t through repeated sampling based on the policy model π_θ.

• *Simulation* aims to perform rollout for each expanded child node s_{t+1} until the task is solved and obtain a reward r based on the rollout results.

• *Backpropagation* operation leverages the reward r of the child node to update the expected reward $V(s_t)$ of nodes along the path from the root node to the current node:

$$N_{new}(s_t) = N_{old}(s_t) + 1, \qquad (2)$$

$$V_{new}(s_t) = \frac{V_{old}(s_t)N_{old}(s_t) + r}{N_{new}(s_t)}, \qquad (3)$$

where $N_{old}(s_t)$ and $V_{old}(s_t)$ are the number of visits and value function at last iteration, respectively.

4 Approach

4.1 Overview

RAG has been an indispensable technique to address the inherent knowledge limitations of LLMs, effectively integrating requisite information and grounding to reliable sources (Lewis et al., 2020a; Guu et al., 2020). However, existing work mainly utilizes RAG to provide supplementary knowledge, while overlooking a thorough investigation of RAG on enhancing the inherent reasoning capabilities of LLMs. To address this, we propose RAG-Star, a framework to fully harness the potential of internal knowledge in LLMs for multi-step reasoning guided by the external retrieval.

Our RAG-Star framework contains two major technical steps. First, we propose *tree-based subquery generation* to perform deliberative reasoning with MCTS, totally relying on the inherent knowledge of LLMs. Second, we design *retrieval-augmented verification* capitalizing on RAG to assist in guiding the reasoning based on the external knowledge. Under this framework, RAG-Star first selects a node from the tree to explore (Section 4.2), then generates the next sub-query and answers for obtaining new child nodes (Section 4.3), and computes a reward to the expanded nodes (Section 4.4). Finally, it backpropagates the reward to update their parent nodes on the tree (Section 4.4). This process will iterate until the task is solved. Next, we will describe each step in detail.

4.2 Node Selection

To answer multi-hop questions, our framework will iterate the tree-based search process multiple times to gradually generate inference solutions in a step-by-step way. In our work, the solution is composed of a sequence of intermediate sub-queries and the

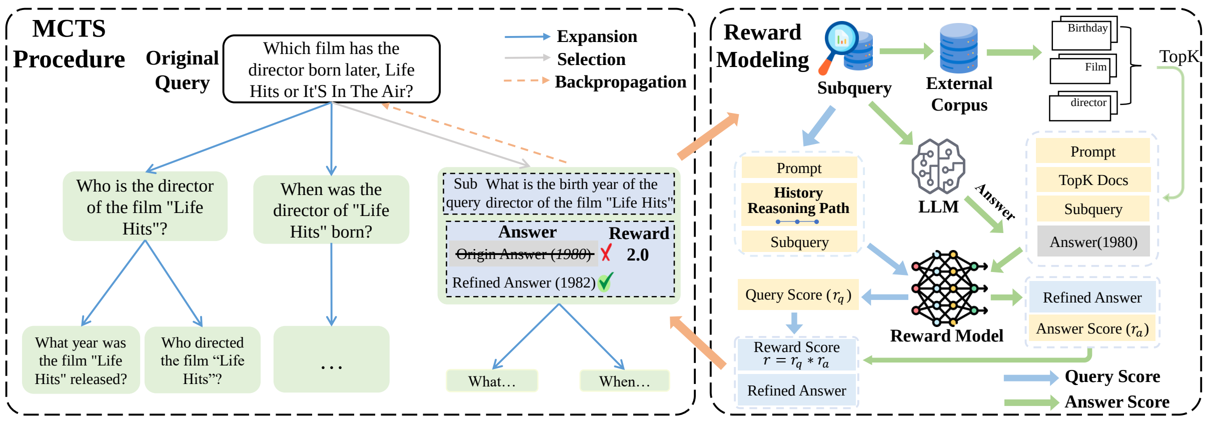

Figure 1: Overall framework of our proposed RAG-Star approach.

associated answers. At each iteration, it first selects an appropriate node from the current tree for the next exploration or expansion. The selection operation is based on the node values computed through the reward modeling and backpropagation steps.

Specifically, starting from the root node s_0 (*i.e.,* the input question q_0), our RAG-Star model selects one node with the highest score from its child nodes, and then sequentially selects the next best child node layer-by-layer along the tree until reaching a leaf node, *i.e.,* the terminal state indicating the final answer. To better balance exploration and exploitation, we use the UCT algorithm (Kocsis and Szepesvári, 2006) to calculate the score of each node according to its number of visits $N(s)$ and expected reward $V(s)$ in Eq. 1.

4.3 Plan Expansion

After selecting the current node, it expands the search tree by repeatively sampling multiple child nodes as *plan* based on the policy model π_θ. Specially, the expansion process involves two steps, *i.e.,* sub-query generation and answer deduction.

Sub-query Planning. To generate the next sub-query as plan, our approach first builds the context information by concatenating states from the root node to the current selected node, and then instructs the policy model to sample the next sub-query based on the context information. Formally, given the node s_t, we can extract a path from the root node s_0 to the current node s_t, denoted by $\mathcal{H} = \{q_0; \langle q_1, a_1\rangle; ...; \langle q_t, a_t\rangle\}$, where q_0 is the original input question and each $\langle q_i, a_i\rangle$ pair denotes the planned sub-query and its answer verified by our retrieval-augmented varification (Section 4.4). We convert this path into the context information, and feed it to the policy model π_θ to generate the next sub-query $q_{t+1} = \pi_\theta(\mathcal{H})$. During inference, we employ repeated sampling to sample sub-queries by m_q times to fully exploit the policy model's inherent capabilities and obtain m_q new expanded sub-queries.

Answer Deduction. After planning the sub-query, we further instruct the policy model to generate an answer to explore the internal knowledge of LLMs. Specially, for each planned sub-query q_{t+1}, we directly feed the historical context $\mathcal{H}$ and sub-query into the policy model to generate a candidate answer by leveraging the inherent knowledge encoded in its parameters as follows:

$$a_{t+1} = \pi_\theta(\mathcal{H}, q_{t+1}). \quad (4)$$

In this process, we do not consider the external knowledge from RAG to avoid knowledge conflicts. We aim to fully exploit the potential of the internal knowledge of LLMs without interference from external information, differing from previous retrieval-augmented work (Lewis et al., 2020b; Yao et al., 2022) that might suffer from knowledge conflicts and interference. After obtaining the answer, we can store each $\langle q_{t+1}, a_{t+1}\rangle$ pair in the corresponding node state, which will be subsequently used for reward modeling.

When completing the plan expansion process, we can obtain m_q child nodes for every parent node, each of which contains a sub-query q_{t+1} and its answer a_{t+1}.

4.4 Reward Modeling and Backpropagation

Traditional MCTS methods require to perform expensive rollout from the current node until the task ends to evaluate the expanded nodes. In our work, following previous work on process-supervised reward modeling (Setlur et al., 2024; Lightman et al., 2024), we propose *retrieval-augmented verification and refinement* by using external knowledge to verify the consistency between the model output and retrieved information. Specifically, we employ reward models to assign an estimated reward r to the expanded node, which effectively quantifies the effectiveness of the policy model in successfully answering the input question if continually reasoning from the current node. Next, we introduce the involved two kinds of reward scores, namely *answer-aware reward* and *query-aware reward*.

Answer-aware Reward. We first introduce the answer-aware reward in the verification process. First, given a sub-query q_{t+1}, we follow existing methods (Lewis et al., 2020a) to retrieve top-K documents $\mathcal{D}_{t+1} = \{d_{t+1,k}\}_{k=1}^{K}$ from the external corpus. Based on the retrieved documents, we then employ the reward model to assign an *answer-aware reward* r_a to the currently generated answer a_{t+1} from the internal knowledge of LLMs. Specifically, there are overall three cases for the knowledge consistency between a_{t+1} and $\mathcal{D}_{t+1}$ with different rewards:

$$r_a = \begin{cases} 1, & \text{if } a_{t+1} \text{ cannot be verified by } \mathcal{D}_{t+1} \\ 2, & \text{if } a_{t+1} \text{ is in conflict with } \mathcal{D}_{t+1} \\ 3, & \text{if } a_{t+1} \text{ is aligned with } \mathcal{D}_{t+1} \end{cases}$$

Note that in the second case (*i.e.,* a_{t+1} is in conflict with $\mathcal{D}_{t+1}$), we assign a moderate score 2 to the answer because we will refine a_{t+1} with a new potential answer $\tilde{a}_{t+1}$ from the external knowledge $\mathcal{D}_{t+1}$ to support the policy model to continually reason from the current node. However, if the answer a_{t+1} cannot be verified by the external knowledge, we will assign the lowest score 1 to the answer, avoiding the policy model from exploring the potentially risky solution space.

Query-aware Reward. In addition to evaluating the consistency of the generated answer with external knowledge, we employ the reward model to provide a *query-aware reward* r_q for measuring the plausibility of the planned sub-query q_{t+1} based on the historical context information from the root node to current node, *i.e.,* $\mathcal{H} = \{q_0; \langle q_1, a_1\rangle; ...; \langle q_t, a_t\rangle\}$. If the sub-query evaluated by the reward model is logically inconsistent with the history plan, the score r_q is set to 0; otherwise, it is set to 1. Therefore, the final reward r for the expanded node s_{t+1} is computed as $r = r_a \cdot r_q$. This step aims to prevent the policy model from continuing to reason along illogical sub-queries.

After obtaining the final reward for the newly expanded node, we backpropagate the reward to update the value of nodes from the root node s_0 to the current node s_{t+1}. For each node $s_0, s_1, ..., s_{t+1}$ in the path, its number of visits $N(s)$ and the value $V(s)$ will be updated according to Eq. 2. These updated values are used in the UCT algorithm in Eq. 1 to guide the node selection at the next iteration.

4.5 Reward Model Training

In the reward modeling process, the capacity of the reward model critically influences the search process and ultimate answer accuracy. However, utilizing close-source model API or very large LLMs incurs substantial computational costs for deployment. Hence, we adopt a knowledge distillation technique to transfer capabilities from an advanced LLM, which usually has more parameters, to a relatively smaller model. This involves two phases: data synthesis and instruction fine-tuning.

During data synthesis, we mix up training sets from our evaluation datasets to maintain diversity. First, we adopt in-context learning to instruct the policy model to generate a CoT format solution and then break down into multiple sub-steps, each incorporating the input question, accumulated reasoning paths, and a sub-query specific to the current step. To further ensure diversity, only one random step from each sample is selected for subsequent instruction data creation. We then employ a more advanced LLM (*i.e.,* GPT-4o-mini) combined with a retrieval system to evaluate the sub-query and its answer for each step (Section 4.4), and filter the output that fails to meet the format criteria. Finally, we compile a dataset of intermediate steps and their query and answer rewards from an advanced LLM. In the instruction fine-tuning phase, we utilize the synthetic samples to fine-tune a smaller LLM (*i.e.,* Llama-3.1-8B-Instruct), thereby enhancing its capabilities in reward modeling.

5 Experiments

5.1 Experimental Setup

Datasets and Evaluation Metrics. We select four typical complex multi-hop question-answering datasets, *i.e.,* HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), MusiQue (Trivedi et al., 2022), and StrategyQA (Geva et al., 2021). For evaluation metrics, we use Exact Match (EM), F1 score, and Cover Exact Match (Cover EM), where Cover EM measures whether the ground truth answer is covered in the generated answer. We randomly select 100 samples from the whole validation sets of each dataset as our final test set for all baselines and our method.

Baselines. We compare **RAG-Star** to the following two types of baselines based on GPT-4o and Llama-3.1-8B-Instruct:

• **Vanilla prompting methods** including direct prompting, Chain-of-Thought (CoT), and standard RAG. Direct prompting instructs the model to directly generate answers and CoT incorporates intermediate reasoning steps, which are all based on the inherent knowledge of LLMs. Standard RAG first retrieves documents from Wikipedia based on DPR (Karpukhin et al., 2020) as prompts and then generates the final answers.

• **Improved RAG methods** including Iterative RAG (Xu et al., 2024), Judge-then-retrieve (Asai et al., 2024), and Generate-then-retrieve (Wang et al., 2023). We reimplement all of these baselines in our experiments. Iterative RAG iteratively decomposes the input question into sub-queries for retrieval and generation, ultimately ensembling all intermediate answers; Judge-then-retrieve first decides whether the retrieval is needed, autonomously deciding to utilize either internal or external knowledge to aid in generation; Generate-then-retrieve first generates an initial answer used for retrieving more documents relevant to the question and then generates the final answer based on documents.

Implementation Details. We use a closed-source model (GPT-4o) and an open-source model (Llama-3.1-8B-Instruct) as our policy models to measure the performance of the RAG-Star framework. For the reward models, we use GPT-4o-mini and a fine-tuned Llama-3.1-8B-Instruct. For HotpotQA, we only use the abstract of articles in Wikipedia 2017 dump as the retrieval corpus following Yang et al. (2018), while for other datasets, we use the whole articles in Wikipedia 2018 dump (Karpukhin et al., 2020). Moreover, for the retrieval model, we use FAISS for index building and BGE-large-en-v1.5 (Xiao et al., 2023) for dense passage retrieval. For all retrieval-based baselines, we retrieve top-5 documents and employ greedy search for decoding with a temperature of 0. For RAG-Star, we set the maximum number of simulations to 50 and a maximum of 6 layers. In UCT algorithm, the weight w to control the exploration and exploitation is set to 0.2. We also retrieve top-5 documents and sample three sub-queries at a time ($m_q = 3$) with temperature 1.0 and top-p sampling where $p = 1.0$. For answer generation, we sample an answer using a temperature of 0.9 and top-p sampling set to 1.0.

5.2 Main Results

Table 1 shows the results of RAG-Star and other baselines across four representative multi-hop question answering datasets.

Firstly, it can be observed that relatively smaller models (*e.g.,* Llama-3.1-8B-Instruct) show limited performance on these knowledge-intensive reasoning tasks, achieving below 10% across three metrics in MusiQue. Although the Chain-of-Thought technique can slightly improve the answer recall (*e.g.,* Cover EM scores of Llama-3.1-8B-Instruct and GPT-4o in MusiQue increase from 3.0% and 19.0% to 16.0% and 27.0%, respectively), the model is prone to generating substantial irrelevant information in the output, decreasing the overall performance (*e.g.,* F1 score of Llama-3.1-8B-Instruct drops from 21.9% to 7.1% on 2WikiMultihopQA).

Secondly, based on the standard RAG, GPT-4o achieves substantial improvement in HotpotQA (*e.g.,* Cover EM increases from 47.0% to 57.0%) but exhibits a large decline in StrategyQA (*e.g.,* Cover EM from 73.0% to 62.0%), suggesting a potential conflict between external sources and internal knowledge of LLMs. We speculate the reason might be that using the retrieved information directly as input incorporates some noises and makes the LLM lost in the useful information. Therefore, by controlling the utilization of internal and external knowledge, Judge-then-Retrieve can significantly alleviate this issue (*e.g.,* Cover EM from 62.0% to 74.0% in StrategyQA). However, these approaches still present limited or even negative improvements in complex tasks (*e.g.,* Cover EM from 19.0% to 16.0% in MusiQue), necessitating effective methods to consolidate external and internal knowledge.

Finally, our approach outperforms all baselines across most metrics in four datasets. RAG-Star introduces a "System 2"-like slow and deliberative thinking process and employs RAG to verify and guide the multi-step reasoning process. By employing retrieval-augmented verification, the reward model can effectively encourage the model towards plausible sub-query nodes or avert from

Method	HotpotQA			2WikiMultihopQA			MusiQue			StrategyQA		
	EM	CEM	F1	EM	CEM	F1	EM	CEM	F1	EM	CEM	F1
Llama-3.1-8B-Instruct	14.0	25.0	26.0	9.0	29.0	21.9	2.0	3.0	3.9	63.0	65.0	63.0
+ Chain-of-Tought	20.0	38.0	26.3	4.0	32.0	7.1	4.0	16.0	6.6	55.0	69.0	55.0
+ Standard RAG	40.0	48.0	52.8	17.0	23.0	26.1	11.0	11.0	15.5	63.0	64.0	63.0
+ Iterative RAG	26.0	31.0	36.9	22.0	23.0	26.0	7.0	11.0	15.9	61.0	63.0	61.0
+ Generate-then-Retrieve	34.0	44.0	49.4	21.0	30.0	26.6	13.0	17.0	19.4	63.0	67.0	63.0
+ Judge-then-Retrieve	39.0	48.0	53.9	18.0	26.0	26.8	10.0	10.0	16.0	58.0	63.0	58.0
+ RAG-Star w Llama RM	42.0	44.0	54.4	34.0	38.0	42.0	13.0	18.0	22.2	**71.0**	**72.0**	**71.0**
+ RAG-Star w GPT RM	**46.0**	**49.0**	**60.0**	**38.0**	**43.0**	**46.8**	**22.2**	**27.0**	**30.7**	67.6	69.0	67.6
GPT-4o	43.0	47.0	56.7	36.0	42.0	45.7	13.0	19.0	24.3	70.0	73.0	70.0
+ Chain-of-Tought	36.0	49.0	56.8	38.0	55.0	53.9	20.0	27.0	29.6	37.0	79.0	37.0
+ Standard RAG	47.0	57.0	63.7	25.0	26.0	31.2	14.0	18.0	20.6	45.0	62.0	45.0
+ Iterative RAG	47.0	**59.0**	63.3	19.0	24.0	26.3	15.0	26.0	25.5	32.0	74.0	32.0
+ Generate-then-Retrieve	44.0	57.0	62.0	29.0	36.0	37.5	23.0	28.0	31.0	50.0	68.0	50.0
+ Judge-then-Retrieve	44.0	50.0	58.6	28.0	29.0	32.2	14.0	16.0	22.8	**72.0**	74.0	**72.0**
+ RAG-Star w Llama RM	**48.0**	54.0	66.3	47.0	**68.0**	**62.8**	25.0	36.0	39.0	61.0	**86.0**	61.0
+ RAG-Star w GPT RM	**48.0**	57.0	**68.6**	**48.0**	63.0	61.7	**29.0**	**40.0**	**43.5**	60.0	81.0	60.0

Table 1: Evaluation results on four representative multi-hop question answering tasks. "RM" is short for reward model. The **bold** and underline fonts denote the best and second best results in each dataset, respectively.

Method	GPT-4o		Llama3.1-8B	
	CEM	F1	CEM	F1
RAG-Star (Ours)	84.0	68.3	75.0	73.3
w/o Query Score	82.0	68.0	71.0	69.0
w/o Answer Score	80.0	66.3	66.0	65.3
w/o Retrieval	78.0	67.3	67.0	66.0
w/o Refine	77.0	68.2	70.0	68.1

Table 2: Ablation study in StrategyQA.

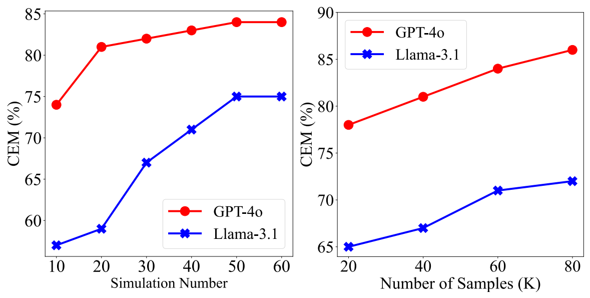

Figure 2: Cover EM performance on the StrategyQA *w.r.t.* the number of simulations (**Left**) or the number of training data (**Right**).

potential risky nodes. For example, equipped with our RAG-Star framework, Llama-3.1-8B-Instruct achieves higher scores in two challenging reasoning datasets, *i.e.,* 2WikiMultihopQA and MusiQue, significantly beyond all baseline methods.

5.3 Further Analysis

We report further analysis in StrategyQA with randomly selected 100 samples – we have similar findings in other datasets.

Ablation Study. To validate the effectiveness of our proposed framework, we conduct an ablation analysis of its key design elements. We design four variants: (1) *w/o Retrieval* removes the retrieved documents in reward modeling; (2) *w/o Refine* does not refine the conflict answer with retrieved documents in reward modeling; (3) *w/o Query Reward* removes the query-aware reward r_q for scoring; and (4) *w/o Answer Reward* removes the answer-aware reward r_a for scoring. We show the results in Table 2. It is clear that all the variants perform worse than the original method, indicating the effectiveness of each component in our framework. Specifically, the performance of *w/o Retrieval* drops significantly for Llama-3.1-8B, indicating that using external knowledge for verification can be highly beneficial for the inherent reasoning of LLMs. Similarly, *w/o Refine* leads to a decline in model performance, which highlights the importance of repurposing external sources for correcting the errors in the model's reasoning process. Moreover, both *w/o Query Reward* and *w/o Answer Reward* variants lead to a substantial performance decline, which suggests that the consistency and logical plausibility of intermediate sub-queries and answers are both critical for the model to plan the correct path towards the final answer.

Effect of Simulation Scaling. Typically, scaling the simulation iterations will lead to a higher

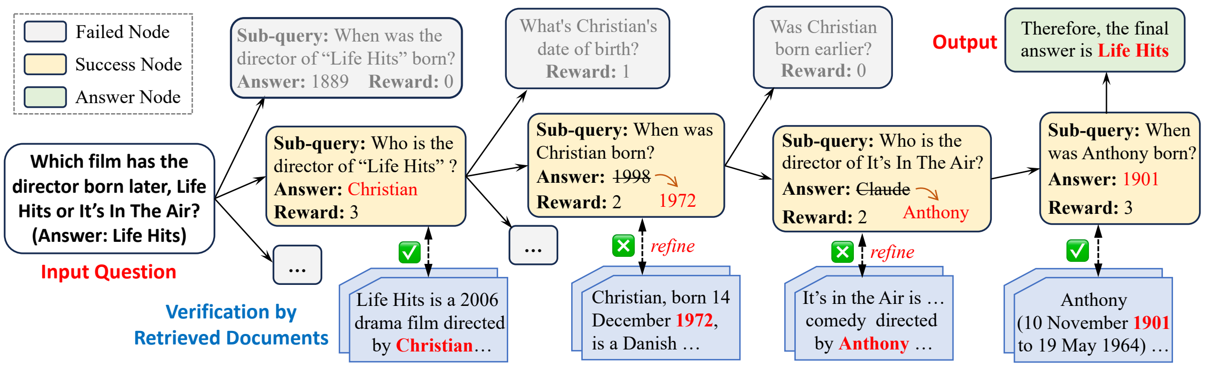
Failed Node
Success Node
Answer Node
Which film has the director born later, Life Hits or It's In The Air? (Answer: Life Hits)
Input Question
Sub-query: When was the director of "Life Hits" born? Answer: 1889 Reward: 0
Sub-query: Who is the director of "Life Hits"? Answer: Christian Reward: 3
Verification by Retrieved Documents
Life Hits is a 2006 drama film directed by Christian…
What's Christian's date of birth? Reward: 1
Sub-query: When was Christian born? Answer: ~~1998~~ 1972 Reward: 2
refine
Christian, born 14 December 1972, is a Danish …
Was Christian born earlier? Reward: 0
Sub-query: Who is the director of It's In The Air? Answer: ~~Claude~~ Anthony Reward: 2
refine
It's in the Air is … comedy directed by Anthony …
Output
Therefore, the final answer is Life Hits
Sub-query: When was Anthony born? Answer: 1901 Reward: 3
Anthony (10 November 1901 to 19 May 1964) …

Figure 3: A qualitative example showing the deliberative reasoning process of RAG-Star in 2WikiMultihopQA.

level of task-solving capability. To explore the relationship between simulation scaling and the final performance of RAG-Star, we test our model under different maximum simulation iterations. Specifically, we vary the maximum simulation rounds in a set $\{10, 20, 30, 40, 50, 60\}$, and evaluate Llama and GPT-4o in StrategyQA with GPT-4o-mini as the reward model. The results are presented in the Figure 2. We can see that as the maximum number of simulation increases, the model's performance gradually improves, although the average time consumed also rises to some extent. This highlights that scaling the test-time computation can further promote more thorough exploration and exploitation by the policy model within the search space. However, as the number of simulations further increases, the performance of the policy model tends to be saturated. Due to the limitation of inherent knowledge, the policy model cannot benefit a lot from conducting more simulations.

Effect of Reward Model. In our framework, the reward model is used to assess the logical plausibility of the sub-query and the consistency between the output answer and external sources. In this part, we aim to explore how to train open-source reward models (*i.e.,* Llama-3.1-8B-Instruct) to achieve performance comparable to closed-source LLMs (*i.e.,* GPT-4o-mini) by varying amounts of training data from 20K to 80K. Specifically, we employ different amounts of training data to fine-tune Llama-3.1-8B-Instruct and use the fine-tuned model to evaluate the sub-query and its answer. As shown in Figure 2, we can see that as the amount of training data increases, the reward model can achieve more accurate verification quality, significantly benefiting the planning and reasoning of the policy model. However, the performance gains tend to saturate at later stages, necessitating instruction tuning data with higher diversity and quality.

Method	NS	TT	RT	CEM
CoT	-	30.48	0.00	34%
RAG	-	10.38	8.44	24%
Iterative RAG	-	33.73	28.51	20%
Judge-then-Retrieve	-	24.18	11.55	28%
Generate-then-Retrieve	-	17.25	10.54	28%
RAG-Star (Ours)	10	96.30	66.54	32%
	30	143.23	99.83	36%
	50	157.29	103.62	40%
	80	200.37	130.92	42%

Table 3: Comparison of time costs per question and the final achieved performance. Here, **NS** represents the number of simulations per question, **TT** and **RT** represent the total inference time and retrieval time per question, respectively.

Analysis of Computational Expenses. We conduct further analysis of computational costs concerning different numbers of simulations on 2WikiMultihopQA. We randomly sample 50 questions to compute the average time and performance. Specifically, we compute the total inference time and retrieval time separately. We show the results in Table 3. Firstly, our proposed method obtains the best performance when scaling the inference-time. Secondly, we observe that the overall time increases with the number of simulations, and performance continually improves, aligning with the recent test-time scaling law (Snell et al., 2024). Thirdly, retrieval time accounts for more than half of the total time. Further analysis indicates that retrieval can be optimized at the hardware level, such as through GPU acceleration and distributed retrieval. We plan to enhance this aspect in future versions.

5.4 Case Study

To facilitate understanding of the entire workflow of our proposed RAG-Star, we present a qualitative analysis in 2WikiMultihopQA. Throughout the search process, the LLM initializes the input question as root node and conducts multiple simulations, eventually reaching the terminal leaf node, which can be vividly represented as a tree. As shown in Figure 3, after selecting the first query (*i.e., Who is the director of "Life Hits"?*), the model expands multiple children nodes by repeated sampling. At the next iteration, the model refines the generated answer (*i.e., 1998*) for the sub-query ("*When was Christian born?*") based on retrieved documents and the reward model returns an overall score of 2. By iterating the multi-step reasoning and retrieval-augmented verification processes for several rounds, the model outputs the final answer (*i.e., Life Hits*). In the task-solving process, the policy model generates an answer to the current sub-query based on its internal knowledge, which might be erroneous due to the limited pre-training corpus in time or the memorization mistakes. Therefore, the external knowledge can be beneficial to validate the correctness of inherent knowledge of LLMs, effectively guiding the model to plan a reasonable path.

6 Conclusion

In this work, we proposed RAG-Star, a novel RAG approach for leveraging external retrieval technique to enhance the multi-step reasoning capabilities of LLMs. RAG-Star employed Monte Carlo Tree Search to search intermediate sub-queries and corresponding answers. Moreover, RAG-Star introduced retrieval-augmented verification to evaluate the plausibility and consistency of the planned sub-queries and answers based on a query-aware and an answer-aware reward. At each iteration, RAG-Star conducted node selection, plan expansion, reward modeling, and reward backpropagation sequentially to consolidate the internal knowledge of LLMs and external knowledge from RAG. Extensive experiments on several datasets showed that our proposed RAG-Star outperforms the traditional RAG and reasoning methods.

Limitations

Despite the great efforts that we have made, the experimental analysis is still limited due to the massive computational cost of tree-based search approaches. We will investigate into more types of complex reasoning tasks and datasets. In our model, we only leverage Monte Carlo Tree Search to conduct our deleberative reasoning process. we may consider investigate more kinds of search algorithms to verify the generalization and robustness of our proposed framework. Moreover, the performance of our model is affected by the feedback quality provided by the reward model. Therefore, a well-trained and performant reward model is important for guiding the reasoning process. We will consider other fine-tuning strategies and more LLMs in reward modeling.

Acknowledgements

This work was partially supported by National Natural Science Foundation of China under Grant No. 92470205 and 62222215, and Outstanding Innovative Talents Cultivation Funded Programs 2022 of Renmin University of China.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Bradley C. A. Brown, Jordan Juravsky, Ryan Saul Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *CoRR*, abs/2407.21787.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jifan Chen, Shih-Ting Lin, and Greg Durrett. 2019. Multi-hop question answering via reasoning chains. *CoRR*, abs/1910.02610.

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*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? A question answering benchmark with implicit reasoning strategies. *Trans. Assoc. Comput. Linguistics*, 9:346–361.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2):100–107.

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, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1049–1065. Association for Computational Linguistics.

Zhen Huang, Zengzhi Wang, Shijie Xia, Xuefeng Li, Haoyang Zou, Ruijie Xu, Run-Ze Fan, Lyumanshan Ye, Ethan Chern, Yixin Ye, Yikai Zhang, Yuqing Yang, Ting Wu, Binjie Wang, Shichao Sun, Yang Xiao, Yiyuan Li, Fan Zhou, Steffi Chern, Yiwei Qin, Yan Ma, Jiadi Su, Yixiu Liu, Yuxiang Zheng, Shaoting Zhang, Dahua Lin, Yu Qiao, and Pengfei Liu. 2024. Olympicarena: Benchmarking multi-discipline cognitive reasoning for superintelligent AI. *CoRR*, abs/2406.12753.

Daniel Kahneman. 2011. Thinking, fast and slow. *Farrar, Straus and Giroux*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020a. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

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. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, IJCNLP 2023 -Volume 1: Long Papers, Nusa Dua, Bali, November 1 - 4, 2023*, pages 305–329. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *OpenAI Blog*.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314.

Richard Sutton. 2019. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554.

Chaojie Wang, Yanchen Deng, Zhiyi Lv, Zeng Liang, Jujie He, Shuicheng Yan, and Bo An. 2024a. Q*: Improving multi-step reasoning for llms with deliberative planning. *CoRR*, abs/2406.14283.

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö Arık. 2024b. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. *arXiv preprint arXiv:2410.07176*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9414–9423. Association for Computational Linguistics.

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 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding.

Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2024. Search-in-the-chain: Interactively enhancing large language models with search for knowledge-intensive tasks. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 1362–1373. ACM.

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, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint*.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking. *CoRR*, abs/2403.09629.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, et al. 2024. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023a. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5823–5840. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023b. A survey of large language models. *CoRR*, abs/2303.18223.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. 2023. Solving math word problems via cooperative reasoning induced language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4471–4485. Association for Computational Linguistics.