Thinker: Training LLMs in Hierarchical Thinking for Deep Search via Multi-Turn Interaction

Jun Xu[1], Xinkai Du[1], Yu Ao[1], Peilong Zhao[1], Yang Li[1], Ling Zhong[1], Lin Yuan[1], Zhongpu Bo[1], Xiaorui Wang[1], Mengshu Sun[1], Zhengke Gui[1], Dalong Zhang[1], Zhaoyang Wang[1], Wang Qiwei[1], Yangyang Hou[1], Zhiying Yin[1], Haofen Wang[2], Huajun Chen[3], Lei Liang[1]*, Jun Zhou[1]*

[1]Ant Group, Hangzhou, China
[2]Tongji University, Shanghai, China
[3]Zhejiang University, Hangzhou, China
{xujun.xj, duxinkai.dxk, mengshu.sms, leywar.liang, jun.zhoujun}@antgroup.com

Abstract

Efficient retrieval of external knowledge bases and web pages is crucial for enhancing the reasoning abilities of LLMs. Previous works on training LLMs to leverage external retrievers for solving complex problems have predominantly employed end-to-end reinforcement learning. However, these approaches neglect supervision over the reasoning process, making it difficult to guarantee logical coherence and rigor. To address these limitations, we propose Thinker, a hierarchical thinking model for deep search through multi-turn interaction, making the reasoning process supervisable and verifiable. It decomposes complex problems into independently solvable sub-problems, each dually represented in both natural language and an equivalent logical function to support knowledge base and web searches. Concurrently, dependencies between sub-problems are passed as parameters via these logical functions, enhancing the logical coherence of the problem-solving process. To avoid unnecessary external searches, we perform knowledge boundary determination to check if a sub-problem is within the LLM's intrinsic knowledge, allowing it to answer directly. Experimental results indicate that with as few as several hundred training samples, the performance of Thinker is competitive with established baselines. Furthermore, when scaled to the full training set, Thinker significantly outperforms these methods across various datasets and model sizes.

1 Introduction

Recently, large language models (LLMs) like GPT-4 have demonstrated remarkable capabilities dealing with simple tasks (OpenAI et al. 2024; Qwen et al. 2025). However, they often fall short when handling complex tasks that require in-depth investigation, information synthesis, and multi-step reasoning. They not only face challenges in planning solution pathways and integrating information from diverse sources but are also prone to hallucinations (confidently fabricating information when their knowledge is insufficient). To address these limitations, a variety of methods (Wang et al. 2025; Chen et al. 2025; Sun et al. 2025; Jin et al. 2025a; Zhang et al. 2025; Liang et al. 2025) have been proposed that focus on emulating a deep search process. Most training-based deep search models are primarily built on reinforcement learning. However, reinforcement learning struggles to constrain the problem-solving process, especially for complex or hybrid tasks. As illustrated in Figure 1, reinforcement learning based deep search methods often suffer from typical problems such as interleaved solving, unclear hierarchy, inconsistent granularity, and inefficient search. These methods often resemble an unstructured, freestyle approach. The reasoning process lacks a systematic framework, is not logically rigorous, and yields inconsistent results. In stark contrast, human experts solve complex problems using a structured thinking process: they first decompose a large problem into smaller, independent sub-problems and then tackle them one by one (Pan et al. 2025). Inspired by this observation, we propose a hierarchical thinking framework to address the limitations of existing deep search methods, improving the logic and rigor of the problem-solving process.

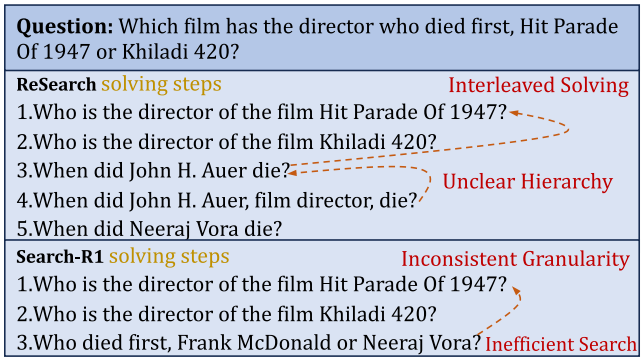


Figure 1: Typical problems with deep search methods based on reinforcement learning training.

Breadth Decomposition and Depth Solving. Direct retrieval often proves insufficient for complex multi-hop problems, as it fails to provide immediate answers. Consider, for example, the question: *"Which film has the director who died first, Hit Parade Of 1947 or Khiladi 420?"*. To address such issues, we have devised a model based on breadth decomposition and depth solving. At the breadth-level, the decomposition of

*Corresponding author.

complex problems into sub-problems is generally straightforward and typically does not require external knowledge. For instance, the aforementioned example can be decomposed into five constituent sub-problems: *"Who is the director of Hit Parade Of 1947?", "When did #1 die?", "Who is the director of Khiladi 420?", "When did #3 die?"*, and *"Which film was directed by the director who died first according to #2 and #4?"*. At the depth-level, the resolution of each individual sub-problem may require anywhere from 0 to M (maximum number of searches) retrieval operations, contingent upon its inherent complexity. The in-depth approach provides a mechanism for augmenting sub-problems with requisite external knowledge, a process crucial for achieving their solvability.

Knowledge Boundary Determination. Most prior approaches (Asai et al. 2024; Gutierrez et al. 2024) initiate retrieval for every problem or sub-problem, even when the required information is already encoded in the LLM's parametric knowledge. To prevent unnecessary and potentially noisy retrieval, we introduce a knowledge boundary determination module. Prior to knowledge boundary determination, the LLM first generates answers to the sub-problems. To ensure the precision of this knowledge boundary determination, we employ two strategies: (1) prompt-based confidence assessment and (2) likelihood-based confidence assessment. Only when both conditions are satisfied is a sub-problem deemed solvable without external retrieval, and the answer generated by the LLM is directly adopted.

Dual Representation based Reasoning. Prior deep search approaches (Wang et al. 2025; Chen et al. 2025; Sun et al. 2025; Jin et al. 2025a; Lewis et al. 2020; Li et al. 2025a), when handling both problems and sub-problems, are restricted to plain text representations, thereby preventing them from effectively utilizing high-quality structured knowledge bases. To address this issue, we introduce four logical forms tailored to frequently encountered problems (see Appendix B (Xu et al. 2025)). Each logical form comprises two components: a natural language segment (Step) and a logical function expression segment (Action). For generic retrievers like E5 and BGE-M3, we can directly employ the content within the logical form's Step. Conversely, for structured knowledge retrievers, the Action portion of the logical form can be utilized. Crucially, the logical form facilitates the propagation of dependencies among interdependent sub-problems. For example, consider *"Step1: Who is the director of Hit Parade Of 1947? Action1: Retrieval(s=s1:film['Hit Parade Of 1947'], p=p1:director, o=o1:director)"* followed by *"Step2: When did #1 die? Action2: Retrieval(s=o1, p=p2:deathtime, o=o2:deathtime)"*. The variables $\#1$ and $o1$ enable seamless transfer of text and expressions from the outcome of the first planning step. In this manner, the logical form enhances the logic, rigor, and stability of the LLM's reasoning.

In summary, the main contribution of this work is as follows:

- We introduce a hierarchical deep search method that enhances logical rigor in complex problem-solving through multi-turn interaction. The method innovatively employs a dual representation of natural language and logical functions for effective knowledge base and web retrieval during breadth decomposition and depth solving, while a knowledge boundary detection module minimizes unnecessary external searches to boost overall performance.
- Our model significantly outperforms established baselines across various datasets and model sizes. Furthermore, our approach demonstrates remarkable sample efficiency, achieving performance competitive with established baselines while using only a few hundred training samples.

2 Related Work

LLMs often lack domain-specific knowledge and are prone to hallucinations. To mitigate these limitations, external knowledge bases and search engines are widely integrated to supply external information. Several retrieval-augmented generation (RAG) methods have been proposed (Jeong et al. 2024; Islam et al. 2024; Tang et al. 2025; Borgeaud et al. 2022; Dong et al. 2023; Luo et al. 2024; Li et al. 2025b; Yang et al. 2025; Reichman and Heck 2024). However, these methods are reliant on a monolithic, single-pass retrieval step, which often yields responses based on incomplete or superficial evidence. To mitigate these shortcomings, deep search methods have been developed. These methods enhance RAG by operationalizing an iterative search-read-reason-refine cycle. This process persists until a predefined termination criterion is met, thereby optimizing for more comprehensive and accurate outcomes. Existing deep search strategies can be broadly categorized into two paradigms (Jeong et al. 2024; Islam et al. 2024; Tang et al. 2025; Wang et al. 2025; Chen et al. 2025; Sun et al. 2025; Wu et al. 2024): (1) API-driven systems that leverage task decomposition and planning, often employing multi-agent architectures to orchestrate retrieval decisions; and (2) Interaction-trained models, where LLMs are fine-tuned, typically via reinforcement learning, for more seamless and effective retriever interaction. Despite these innovations, a paramount challenge lies in the opacity of their intermediate reasoning steps, which impedes verifiability and can compromise the logical integrity of the solution path.

3 Approach

We propose a deep search method based on hierarchical thinking and multi-turn interaction, as shown in Figure 2.

3.1 Breadth Decomposition and Depth Solving

Complex multi-hop problems usually need to be broken down to be solved. Our method decomposes problems into two parts: breadth decomposition, which ensures that the main problem and sub-problems remain logical and precise, and in-depth solving, which ensures that sub-problems are provided with sufficient knowledge to be solved.

Breadth Decomposition We decompose complex problems breadth-wise into n atomic granularity sub-problems, with our decomposition instruction template detailed in Appendix L (Xu et al. 2025). We define four logical form functions (Retrieval, Math, Deduce, and Output), each dedicated to handling specific tasks: retrieval-focused problems, mathematical computation and causal reasoning tasks, and results aggregation functions. As shown in Figure 2, our breadth decomposition divides the original question into five logical

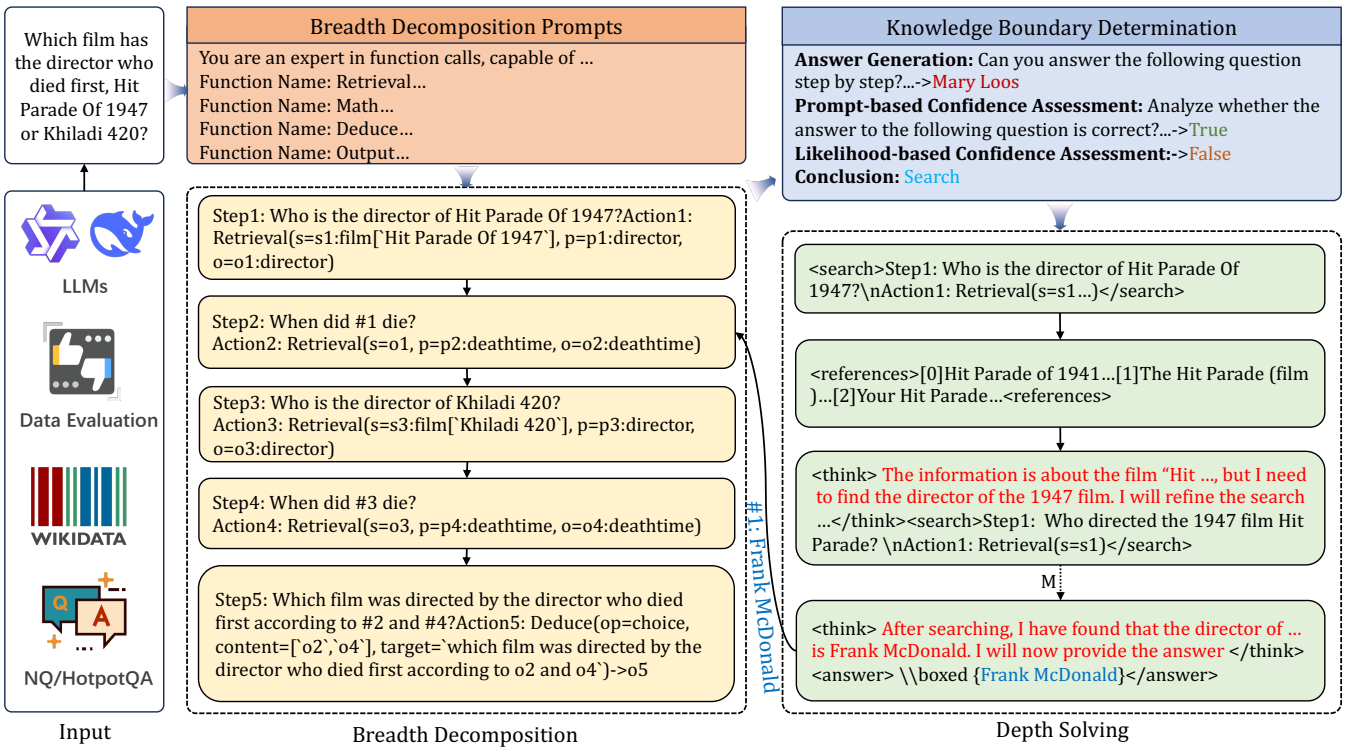


Figure 2: Overview of the Thinker model that uses hierarchical thinking through a multi-turn reasoning process. During problem breadth decomposition, all sub-problems are obtained in a single decomposition pass, where each sub-problem is an atomic problem that can be solved independently. Herein, the terms **Step** and **Action** maintain semantic consistency, both denoting such a sub-problem. Within problem breadth decomposition, Step employs $\#n$ for answer propagation of the n-th sub-problem, while Action binds variables in logical function (e.g., o_n, s_n) for variable transmission. By determining the knowledge boundary of the sub-problem, it is decided whether to utilize the base model's answer or to generate a deep retrieval. During depth solving of sub-problems, the system sequentially executes retrieval, focusing, and reasoning in iterative processes until either the sub-problem answer is obtained or the maximum solving attempt threshold is activated.

forms of atomic granularity, each independently solvable. The dependency variables are propagated between logical forms by function variable $\#n$, o_n, and s_n. This approach ensures logically coherent question-solving while maintaining consistent sub-question granularity. To ensure compatibility with both natural language and logical form retrievers during sub-question solving, each sub-question adopts dual representations, **Step** and **Action**, that maintain semantic equivalence.

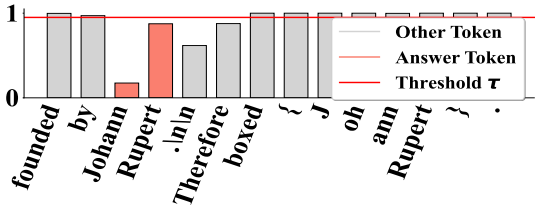
Depth Solving Constrained by sub-problem representations and retriever capabilities, many sub-problems defy resolution through single-turn retrievals. We have engineered a depth-oriented resolution strategy for the *Retrieval* sub-problem, as detailed in the right part of Figure 2. During depth solving, we prompt the LLM to iteratively search across multi-level and multi-dimensional contexts. The iterative process continues until one of the following conditions is met: (1) the maximum number of turns is reached, or (2) the model generates a conclusive response enclosed between the designated answer tokens <answer> and </answer>. Before conducting depth solving of the *Retrieval* sub-problem, we first perform a knowledge boundary determination on the sub-problem to assess whether the LLM can directly answer it. If the LLM can confidently answer the current sub-problem, there is no need for depth solving, and the answer can be directly obtained; otherwise, a depth solving process will be initiated. After retrieving relevant content, we conduct focusing and reasoning analysis, allowing the model to determine contextually whether to initiate the next action. If a subsequent action is required, a new logical form is generated; otherwise, the sub-question answer is produced. This process iterates interactively until either the maximum number of turns is reached or the sub-question answer is obtained.

3.2 Knowledge Boundary Determination

Exhaustive retrieval for every sub-problem introduces significant computational overhead and noise, causing more hallucinations. To mitigate these challenges, we introduce the **Generate First, Then Assess** strategy. This approach mandates that the model first attempts to formulate a direct answer to the sub-problem using its internal knowledge, as guided by the prompt detailed in Appendix M (Xu et al. 2025). Subsequently, the reliability of this initial response is evaluated via a hybrid confidence assessment that combines

both prompt-based and likelihood-based methods. The final confidence is determined to be *True* if and only if both methods independently yield a *True* assessment. If the confidence level is deemed sufficient, this internally generated answer is adopted as the final solution.

Prompt-based Confidence Assessment. We introduce a prompt-based methodology for confidence assessment that employs introspective verification. This interactive procedure enables the model to leverage its internal knowledge for the iterative self-assessment of its generated answers, culminating in a binary classification of *True* or *False*.



Question: AutoTrader.co.za is a site for the company … that was founded by whom?
Ground Truth: John Madejski
Answer Generation: Johann Rupert
Prompt-based Confidence Assessment: True
Likelihood-based Confidence Assessment: False

Figure 3: The probability distribution of the answer tokens.

Likelihood-based Confidence Assessment. Likelihood-based confidence assessment evaluates the reliability of a model's generated response by leveraging the probabilities of its output tokens. As illustrated in Figure 3, given a sub-problem input $\boldsymbol{x}$, the model first extracts the critical content enclosed within \boxed{} from its response. This extracted content is formulated as a target sequence $\boldsymbol{y} = \{y_1, y_2, \ldots, y_T\}$, where T is the sequence length. During the auto-regressive generation process, each token y_t $(t = 1, 2, \ldots, T)$ is associated with a conditional probability $P(y_t \mid \boldsymbol{x}, y_{<t})$. This term represents the likelihood of generating y_t given the input $\boldsymbol{x}$ and the preceding context $y_{<t}$. While these token-level probabilities can, in principle, serve as a basis for confidence evaluation, a naive averaging of these values is an inadequate strategy. This is because high-frequency tokens (e.g., "the", "of") inherently possess high generation probabilities but contribute minimally to assessing the model's certainty regarding the core semantic content. To more effectively capture moments of high uncertainty during generation, we adopt a minimum probability criterion. We define the confidence score C as the lowest probability among all tokens in the target sequence:

$$C = \min\{p(y_1 \mid \boldsymbol{x}), p(y_2 \mid \boldsymbol{y}_{<2}, \boldsymbol{x}), \ldots, p(y_T \mid \boldsymbol{y}_{<T}, \boldsymbol{x})\}$$

The reliability of a generated answer is assessed by comparing its confidence score, C against a predefined threshold, τ. An answer is classified as reliable (*True*) if $C \geq \tau$, and unreliable (*False*) otherwise.

3.3 Focusing and Reasoning

During knowledge boundary determination, we ascertain whether a sub-question requires retrieval. After executing retrieval, we need to assess if the current retrieval results can adequately address the sub-problem. Therefore, we have designed Focusing and Reasoning modules to fulfill this purpose. Focusing and Reasoning are primarily designed to analyze retrieved references, with their data construction methodology as detailed in Appendix N (Xu et al. 2025). Subsequently, based on the answer and reason from the Output section of Appendix N (Xu et al. 2025), we employ LLMs to perform conditional restructuring, ultimately embedding the restructured content into the designated <think> within the depth-solving module.

3.4 Multi-Turn Interactive Training

We conduct supervised fine-tuning of the deep search reasoning process through multi-turn interaction data. Detailed descriptions of the complete training samples and the corresponding data construction methodology can be found in Appendix K (Xu et al. 2025) and Appendix A (Xu et al. 2025), respectively. The supervised fine-tuning procedure for multi-turn interactions closely follows that of the single-turn setting. Specifically, each entire interaction sequence, represented as $[S, U_1, A_1, ..., U_n, A_n]$, is concatenated. Where S_i, U_i, and A_i denote the System, User, and Assistant contents, respectively. The cross-entropy loss is computed solely over the assistant's response tokens $(A_1 ... A_n)$, after which the losses are averaged. This approach is computationally efficient due to its compatibility with parallel processing, and it maximizes the utility of the data by incorporating supervision at every assistant turn. This supervised learning strategy enables the model to acquire and adapt to specific response styles, thereby facilitating customization for specialized domains such as medicine, law, and finance.

4 Experiments

4.1 Experimental Settings

Benchmarks. Our experiments are conducted on 7 widely-used datasets. The dataset comprises two primary categories: (1) Single-Hop QA: NQ (Kwiatkowski et al. 2019), TriviaQA (Joshi et al. 2017), and PopQA (Mallen et al. 2023). (2) Multi-Hop QA: HotpotQA (Yang et al. 2018), 2WikiMultiHopQA (Ho et al. 2020), Musique (Trivedi et al. 2022), and Bamboogle (Press et al. 2023). Our evaluation set maintains methodological consistency with established prior works (Chen et al. 2025; Jin et al. 2025a).
Comparison Methods. We establish a three-tiered evaluation model comprising the following baselines: (1) **Non-Retrieval Paradigms** Naive Generation: Direct answer synthesis without external knowledge integration; Chain-of-Thought (CoT): Explicit cognitive pathway formalization through sequential reasoning traces (Wei et al. 2022). (2) **Retrieval-Augmented Architectures** Naive RAG: Standard retrieval-generation pipeline without iterative optimization (Lewis et al. 2020); IRCoT: Multi-cycle retrieval-reasoning coordination with dynamic feedback mechanisms (Trivedi et al. 2023); Search-o1: Agent-mediated search workflow integration in reason-

LLMs	Methods	Single-Hop QA			Multi-Hop QA				Avg
		NQ†	TriviaQA*	PopQA*	HotpotQA†	2Wiki*	MuSiQue*	Bamboogle*	
Qwen2.5-3B-Instruct	Naive Generation	0.106	0.288	0.108	0.149	0.244	0.020	0.024	0.134
	CoT	0.023	0.032	0.005	0.021	0.021	0.002	0.000	0.015
	Search-o1	0.238	0.472	0.262	0.221	0.218	0.054	0.320	0.255
	IRCoT	0.111	0.312	0.200	0.164	0.171	0.067	0.240	0.181
	Naive RAG	0.348	0.544	0.387	0.255	0.226	0.047	0.080	0.270
	R1-Gen	0.210	0.449	0.171	0.208	0.275	0.06	0.192	0.224
	Search-R1	0.341	0.545	0.378	0.324	0.319	0.103	0.264	0.325
	ZeroSearch	0.414	0.574	0.448	0.274	0.300	0.098	0.111	0.317
	StepSearch	-	-	-	0.345	0.320	0.174	0.344	-
	ReSearch	0.352	0.544	0.421	0.356	0.331	0.159	0.280	0.349
	Thinker(ours)	**0.439**	**0.598**	**0.469**	**0.400**	**0.469**	**0.214**	**0.424**	**0.430**
Qwen2.5-7B-Instruct	Naive Generation	0.134	0.408	0.140	0.183	0.250	0.031	0.120	0.181
	CoT	0.048	0.185	0.054	0.092	0.111	0.022	0.232	0.106
	Search-o1	0.151	0.443	0.131	0.187	0.176	0.058	0.296	0.206
	IRCoT	0.224	0.478	0.301	0.133	0.149	0.072	0.224	0.226
	Naive RAG	0.349	0.585	0.392	0.299	0.235	0.058	0.208	0.304
	R1-Gen	0.270	0.537	0.199	0.237	0.292	0.072	0.293	0.271
	Search-R1	0.393	0.610	0.397	0.370	0.414	0.146	0.368	0.385
	ZeroSearch	0.436	**0.652**	**0.488**	0.346	0.352	0.184	0.278	0.391
	StepSearch	-	-	-	0.386	0.366	0.226	0.400	-
	ReSearch	0.407	0.611	0.423	0.419	0.412	0.205	0.400	0.411
	Thinker(ours)	**0.450**	0.642	0.484	**0.421**	**0.469**	**0.221**	**0.480**	**0.452**

Table 1: EM performance of different models. The best performance is set in bold. †/* represents in-domain / out-domain datasets. In contrast to other baselines, StepSearch and ReSearch employ the Musique dataset for training.

ing processes (Li et al. 2025a). (3) **Reinforcement Learning Models** R1-Gen: Pure RL-optimized generation without search engine interfacing (DeepSeek-AI et al. 2025); Search-R1: Multimodal trajectory optimization combining search interactions with reasoning steps (Jin et al. 2025a); ZeroSearch: Supervised LLM transformation into dual-function retrieval module (relevant/noisy document generation) (Sun et al. 2025); StepSearch: PPO-based search LLM training with incremental exploration (Wang et al. 2025).

Implementation Details. To maintain consistency with previous work, we selected NQ and HotpotQA as our training datasets. Using the data construction and evaluation frameworks outlined in Appendices A and C (Xu et al. 2025), we obtained a total of 71K training samples. All comparison methods use the same base model. We employ E5-base-v2 (Wang et al. 2024) as the retriever and utilize Wikipedia data from December 2018 as the knowledge base (Karpukhin et al. 2020). All corpus indexing and embedding processes are pre-processed using FlashRAG (Jin et al. 2025b). Except for ReSearch, which retrieves the top 5 documents for each question, all other methods retrieve the top 3 documents. For additional implementation details, please refer to Appendix D (Xu et al. 2025).

4.2 Main Results

We evaluated our model on seven widely used benchmark datasets. Here, to maintain consistency with the retriever components of other baseline methods, we conduct plain text retrieval using the *"Step"* content from the logical form. Table 1 presents the comparisons between our model and other baselines. As the results show, our model significantly outperforms the baselines at both the 3B and 7B scales. The improvement is particularly notable for the 3B model, which achieves an average 7.9% gain over the Research baseline. This is because the 3B model's low-quality RL rollouts struggled to produce correct answers. Compared to baselines without retrieval, the performance of our model exceeds Naive Generation and CoT by an average of 27. 1% and 34. 6%, respectively, in seven datasets. The primary reason for the improvement is that, within these datasets, retrieval is essential, as LLMs struggle to directly answer these questions accurately without access to relevant information. Compared to retrieval-augmented methods, our model outperforms Search-o1, IRCoT, and Naive RAG by average margins of 24.6%, 22.6%, and 14.8%, respectively. These enhancements are primarily attributed to our model's implementation of breadth decomposition and depth solving, enabling it to learn how to leverage the retriever more effectively—particularly during deep retrieval, where the generated queries align well with the retriever's capabilities. In comparison to reinforcement learning-based approaches, it can be observed that our model outperforms the previous state-of-the-art model, ReSearch, by an average of 4.1% in EM score across seven datasets. Specifically, it improves by an average of 4.5% on single-hop datasets and by an average of 3.9% on multi-hop datasets. The primary reason is that our breadth decomposition and depth solving model enables the decomposition of questions into atomic granularity, thereby reducing the complexity involved in model retrieval and answering.

The logical coherence and rigor of different deep search models are evaluated using four defined metrics: Logical Hierarchy (Hier), Interleaved Solving (Intrlv), Granularity Consistency (Gran), and Search Efficiency (Eff) (see Appendix E (Xu et al. 2025) for detailed definitions). For this

evaluation, GPT-4 is employed to assess the reasoning processes on the HotpotQA test set. The results, presented in Table 2, indicate that the performance of Thinker in logical coherence and rigor is significantly superior to that of ReSearch and Search-R1 across these dimensions.

Methods	Hier	Intrlv	Gran	Eff	Overall
Search-R1	0.813	0.955	0.852	0.903	0.638
ReSearch	0.872	0.967	0.877	0.922	0.705
Thinker	**0.975**	**0.989**	**0.955**	**0.958**	**0.904**

Table 2: The accuracy of logical coherence and rigor across different models on HotpotQA.

4.3 Model Ablation Studies

To evaluate the contribution of each component within Thinker (Qwen2.5-7B-Instruct), we conduct an ablation study across seven datasets (cf. Table 3). The removal of the depth solving component causes the most significant performance degradation, resulting in a 3.7% drop in the average EM score. This is primarily because depth solving retrieves sufficient external knowledge for each sub-problem, which is crucial for improving its solution accuracy. In contrast, ablating the focusing and reasoning module results in a slight performance decrease. Furthermore, while removing the knowledge boundary determination has a minimal impact on overall performance, it significantly reduces unnecessary retrievals (cf. Table 7). A similar observation applies to the logical function (the Action part of a sub-problem): although its contribution to overall performance is negligible, its role is indispensable for enhancing performance within the graph-retrieval-enabled framework.

	Avg
Thinker	0.452
- depth solving	0.415
- knowledge boundary determination	0.447
- focusing & reasoning	0.445
- logical function	0.449

Table 3: Model ablation studies (average EM score).

4.4 Sensitivity Analysis

To validate the robustness of Thinker (Qwen2.5-7B-Instruct), we conduct a series of sensitivity analysis experiments. As shown in Table 4, we evaluate the performance of Thinker with varying numbers of training samples. We observe that even with only 1% of the data, which corresponds to just a few hundred samples, the performance is already close to SOTA method (ReSearch Avg: 0.411). As the sample size increases, the performance also shows a gradual improvement. This also demonstrates that our multi-turn interaction approach significantly reduces training costs.

Table 5 illustrates the impact of the maximum search depth per sub-problem on EM performance. The results indicate a significant performance degradation when the model is limited to a single search. In contrast, the performance stabilizes once the maximum search count is two or greater. Notably, we find that single-hop datasets are insensitive to retrieval depth. This is primarily because the sub-problems in these tasks are relatively straightforward, allowing the majority of answers to be retrieved directly.

Dataset	1%	10%	20%	50%	80%	100%
NQ	0.388	0.441	0.439	0.453	0.452	0.450
TriviaQA	0.602	0.614	0.620	0.627	0.625	0.642
PopQA	0.438	0.469	0.472	0.478	0.475	0.484
HotpotQA	0.377	0.408	0.412	0.414	0.412	0.421
2Wiki	0.413	0.438	0.461	0.471	0.470	0.469
MuSiQue	0.200	0.212	0.214	0.216	0.221	0.221
Bamboogle	0.424	0.432	0.448	0.472	0.456	0.480
Avg	**0.406**	0.431	0.438	0.447	0.444	0.452

Table 4: Impact of the number of training samples.

Dataset	D=1	D=2	D=3	D=4	D=5
NQ	0.427	0.438	0.437	0.440	0.450
TriviaQA	0.627	0.632	0.627	0.630	0.642
PopQA	0.464	0.470	0.470	0.470	0.484
HotpotQA	0.375	0.420	0.410	0.408	0.421
2Wiki	0.421	0.456	0.460	0.455	0.469
MuSiQue	0.181	0.230	0.225	0.225	0.221
Bamboogle	0.408	0.448	0.464	0.448	0.480
Avg	0.415	0.442	0.442	0.439	0.452

Table 5: Impact of the maximum number of depth searches.

To evaluate the performance of the knowledge boundary determination, we conduct an analysis on the test sets of TriviaQA and Bamboogle, with the results presented in Table 7. In our evaluation, EM=1 signifies a correct prediction, whereas EM=0 signifies an incorrect one. In cases where EM=1, we assume that all decisions not to retrieve are accurate, as the LLM successfully answers the sub-question using its internal knowledge, which results in an overall accuracy of 1 for this subset. Conversely, in the EM=0 samples, there is a notable reduction in sub-questions for which retrieval is deemed unnecessary. In summary, the integration of the knowledge boundary determination module leads to 16.0% and 17.8% reductions in search queries on TriviaQA and Bamboogle, respectively. However, for instances where EM=0, assessing the correctness of a no-retrieval determination is challenging. To calculate its accuracy under this condition, we employ a dual verification process using Qwen2.5-72B-Instruct and DeepSeek-R1-Distill-32B. A no-retrieval determination is considered correct only if both models concur that the answer generated by Thinker based on its internal knowledge is accurate. As shown in Table 7, our knowledge boundary determination achieves an accuracy of 96.9% on TriviaQA and 97.8% on Bamboogle.

4.5 Thinker with KAG

Our dual representation of sub-problems enables better utilization of high-quality knowledge bases. To validate this hypothesis, we create KAG-Thinker by applying the Thinker

Methods	Size	Retriever	HotpotQA		MusiQue		2Wiki		Avg	
			EM	F1	EM	F1	EM	F1	EM	F1
Search-R1	7B	BGE-M3	0.545	0.676	0.307	0.403	0.517	0.589	0.456	0.556
ReSearch	7B	BGE-M3	0.538	0.671	0.322	0.423	0.555	0.633	0.472	0.576
Thinker (ours)	7B	BGE-M3	0.538	0.664	0.337	0.442	0.596	0.657	0.490	0.588
KAG-Thinker (ours)	7B	Hybrid Graph Retriever	**0.568**	**0.698**	**0.350**	**0.469**	**0.644**	**0.711**	**0.520**	**0.626**

Table 6: Performance of different models within a self-built corpus. All retrievers retrieve the top 3 documents.

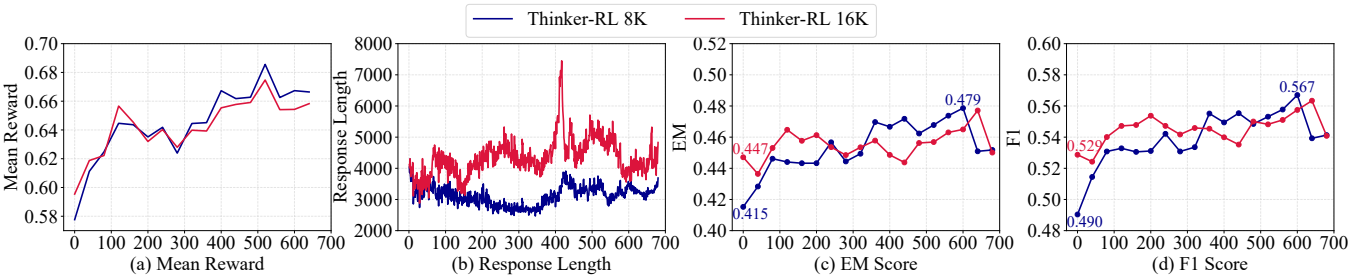


Figure 4: Performance of Thinker with Reinforcement Learning vs. Training Steps. (a) Average reward during the RL training. (b) Average model response length. (c) Average EM score across seven datasets. (d) Average F1 score across seven datasets.

Dataset	EM = 1		EM = 0		All	
	KBD	Acc	KBD	Acc	KBD	Acc
TriviaQA	0.216	1.000	0.065	0.794	0.160	0.969
Bamboogle	0.234	1.000	0.124	0.938	0.178	0.978

Table 7: Retrieval rate reduction from knowledge boundary determination (KBD).

model within the KAG framework (Liang et al. 2024). The multi-turn interactive reasoning framework of KAG-Thinker is depicted in Appendix G (Xu et al. 2025). To maintain consistency with KAG, we continue to use its test set, with 1,000 examples each from HotpotQA, Musique, and 2Wiki (consistent with HippoRAG (Gutierrez et al. 2024)). The data for our self-built corpus are entirely derived from the supporting facts of these three datasets, totaling 26,990 documents. Within the self-built corpus, and utilizing the same retriever and retrieved documents, our Thinker model still outperforms ReSearch and Search-R1 across three multi-hop datasets. For a fair comparison against the baselines, the original Thinker model could only perform pure text retrieval with BGE-M3 for natural language steps in its logical form, even for operations like Deduce and Math. However, the KAG framework robustly supports these operations. Table 6 illustrates that KAG-Thinker, operating within the KAG framework, shows a marked improvement over Thinker, with average EM and F1 scores increasing by 3.0% and 3.8%, respectively. KAG-Thinker achieves these enhancements by leveraging two key features from the KAG framework: the hybrid graph retriever (HGR) and native support for Math and Deduce.

4.6 Thinker with Reinforcement Learning

To explore the potential of the Thinker model, we apply reinforcement learning initialized from the optimal SFT model. For implementation details, please see Appendix D (Xu et al. 2025). As illustrated in Figure 4, our experiments show significant improvements. Figure 4(a) plots the training reward curve, smoothed with a 40-step moving average, while Figure 4(b) tracks the average response length. Figure 4(c) and (d) show the average EM and F1 scores on seven evaluation datasets. The application of reinforcement learning yields a substantial performance gain over the SFT baseline, increasing the average EM score from 0.452 to 0.479. For a detailed performance comparison on each dataset, please refer to Appendix J (Xu et al. 2025). This improvement is strongly correlated with the training reward, demonstrating RL's ability to unlock latent model potential. Moreover, experiments with output length constraints show that a more restrictive 8K limit, despite initial underperformance, ultimately outperforms the 16K counterpart, effectively promoting response conciseness. Compared to deep search methods based on a pure reinforcement learning algorithm, our SFT-then-RL methodology not only preserves the logical consistency of reasoning but also significantly boosts the model's performance. Ultimately, Thinker-RL performs inference in the same way as Thinker.

5 Conclusion

In this paper, we propose a model for interactive thinking and deep reasoning to solve complex problems. We find that our model significantly outperforms previous methods across various datasets and model sizes, and notably improves the logical rigor in solving complex tasks. Through ablation studies and sensitivity analyses, we discover that our proposed methods can achieve near-SOTA performance even when trained on only a few hundred samples. Our approach offers a systematic reasoning framework, making it particularly well-suited for domains where logical rigor is paramount, including medical diagnostics (cf. Appendix O (Xu et al. 2025)), legal compliance, and financial risk assessment.

References

Asai, A.; Wu, Z.; Wang, Y.; Sil, A.; and Hajishirzi, H. 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.

Borgeaud, S.; Mensch, A.; Hoffmann, J.; Cai, T.; Rutherford, E.; Millican, K.; van den Driessche, G.; Lespiau, J.; Damoc, B.; Clark, A.; de Las Casas, D.; Guy, A.; Menick, J.; Ring, R.; Hennigan, T.; Huang, S.; Maggiore, L.; Jones, C.; Cassirer, A.; Brock, A.; Paganini, M.; Irving, G.; Vinyals, O.; Osindero, S.; Simonyan, K.; Rae, J. W.; Elsen, E.; and Sifre, L. 2022. Improving Language Models by Retrieving from Trillions of Tokens. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvári, C.; Niu, G.; and Sabato, S., eds., *ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, 2206–2240. PMLR.

Chen, M.; Li, T.; Sun, H.; Zhou, Y.; Zhu, C.; Wang, H.; Pan, J. Z.; Zhang, W.; Chen, H.; Yang, F.; Zhou, Z.; and Chen, W. 2025. ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning. arXiv:2503.19470.

DeepSeek-AI; Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.

Dong, G.; Li, R.; Wang, S.; Zhang, Y.; Xian, Y.; and Xu, W. 2023. Bridging the KB-Text Gap: Leveraging Structured Knowledge-aware Pre-training for KBQA. In Frommholz, I.; Hopfgartner, F.; Lee, M.; Oakes, M.; Lalmas, M.; Zhang, M.; and Santos, R. L. T., eds., *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM 2023, Birmingham, United Kingdom, October 21-25, 2023*, 3854–3859. ACM.

Gutierrez, B. J.; Shu, Y.; Gu, Y.; Yasunaga, M.; and Su, Y. 2024. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Ho, X.; Duong Nguyen, A.-K.; Sugawara, S.; and Aizawa, A. 2020. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In Scott, D.; Bel, N.; and Zong, C., eds., *Proceedings of the 28th International Conference on Computational Linguistics*, 6609–6625. Barcelona, Spain (Online): International Committee on Computational Linguistics.

Islam, S. B.; Rahman, M. A.; Hossain, K. S. M. T.; Hoque, E.; Joty, S.; and Parvez, M. R. 2024. Open-RAG: Enhanced Retrieval Augmented Reasoning with Open-Source Large Language Models. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y., eds., *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, 14231–14244. Association for Computational Linguistics.

Jeong, S.; Baek, J.; Cho, S.; Hwang, S. J.; and Park, J. 2024. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In Duh, K.; Gómez-Adorno, H.; and Bethard, S., eds., *NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, 7036–7050. Association for Computational Linguistics.

Jin, B.; Zeng, H.; Yue, Z.; Yoon, J.; Arik, S.; Wang, D.; Zamani, H.; and Han, J. 2025a. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. arXiv:2503.09516.

Jin, J.; Zhu, Y.; Dou, Z.; Dong, G.; Yang, X.; Zhang, C.; Zhao, T.; Yang, Z.; and Wen, J.-R. 2025b. FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, 737–740. New York, NY, USA: Association for Computing Machinery. ISBN 9798400713316.

Joshi, M.; Choi, E.; Weld, D.; and Zettlemoyer, L. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In Barzilay, R.; and Kan, M.-Y., eds., *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1601–1611. Vancouver, Canada: Association for Computational Linguistics.

Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; and Yih, W.-t. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In Webber, B.; Cohn, T.; He, Y.; and Liu, Y., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6769–6781. Online: Association for Computational Linguistics.

Kwiatkowski, T.; Palomaki, J.; Redfield, O.; Collins, M.; Parikh, A.; Alberti, C.; Epstein, D.; Polosukhin, I.; Devlin, J.; Lee, K.; Toutanova, K.; Jones, L.; Kelcey, M.; Chang, M.-W.; Dai, A. M.; Uszkoreit, J.; Le, Q.; and Petrov, S. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7: 452–466.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Li, X.; Dong, G.; Jin, J.; Zhang, Y.; Zhou, Y.; Zhu, Y.; Zhang, P.; and Dou, Z. 2025a. Search-o1: Agentic Search-Enhanced Large Reasoning Models. *CoRR*, abs/2501.05366.

Li, Z.-Z.; Zhang, D.; Zhang, M.-L.; Zhang, J.; Liu, Z.; Yao, Y.; Xu, H.; Zheng, J.; Wang, P.-J.; Chen, X.; Zhang, Y.; Yin, F.; Dong, J.; Guo, Z.; Song, L.; and Liu, C.-L. 2025b. From System 1 to System 2: A Survey of Reasoning Large Language Models. *ArXiv*, abs/2502.17419.

Liang, L.; Bo, Z.; Gui, Z.; Zhu, Z.; Zhong, L.; Zhao, P.; Sun, M.; Zhang, Z.; Zhou, J.; Chen, W.; Zhang, W.; and Chen, H. 2025. KAG: Boosting LLMs in Professional Domains

via Knowledge Augmented Generation. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, 334–343. New York, NY, USA: Association for Computing Machinery. ISBN 9798400713316.

Liang, L.; Sun, M.; Gui, Z.; Zhu, Z.; Jiang, Z.; Zhong, L.; Qu, Y.; Zhao, P.; Bo, Z.; Yang, J.; Xiong, H.; Yuan, L.; Xu, J.; Wang, Z.; Zhang, Z.; Zhang, W.; Chen, H.; Chen, W.; and Zhou, J. 2024. KAG: Boosting LLMs in Professional Domains via Knowledge Augmented Generation. *CoRR*, abs/2409.13731.

Luo, H.; E, H.; Tang, Z.; Peng, S.; Guo, Y.; Zhang, W.; Ma, C.; Dong, G.; Song, M.; Lin, W.; Zhu, Y.; and Luu, A. T. 2024. ChatKBQA: A Generate-then-Retrieve Framework for Knowledge Base Question Answering with Fine-tuned Large Language Models. In Ku, L.; Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, 2039–2056. Association for Computational Linguistics.

Mallen, A.; Asai, A.; Zhong, V.; Das, R.; Khashabi, D.; and Hajishirzi, H. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 9802–9822. Toronto, Canada: Association for Computational Linguistics.

OpenAI; Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; et al. 2024. GPT-4 Technical Report. arXiv:2303.08774.

Pan, Q.; Ji, W.; Ding, Y.; Li, J.; Chen, S.; Wang, J.; Zhou, J.; Chen, Q.; Zhang, M.; Wu, Y.; and He, L. 2025. A Survey of Slow Thinking-based Reasoning LLMs using Reinforced Learning and Inference-time Scaling Law. *CoRR*, abs/2505.02665.

Press, O.; Zhang, M.; Min, S.; Schmidt, L.; Smith, N. A.; and Lewis, M. 2023. Measuring and Narrowing the Compositionality Gap in Language Models. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, 5687–5711. Association for Computational Linguistics.

Qwen; Yang, A.; Yang, B.; Zhang, B.; Hui, B.; et al. 2025. Qwen2.5 Technical Report. arXiv:2412.15115.

Reichman, B. Z.; and Heck, L. 2024. Retrieval-Augmented Generation: Is Dense Passage Retrieval Retrieving? *CoRR*, abs/2402.11035.

Sun, H.; Qiao, Z.; Guo, J.; Fan, X.; Hou, Y.; Jiang, Y.; Xie, P.; Zhang, Y.; Huang, F.; and Zhou, J. 2025. ZeroSearch: Incentivize the Search Capability of LLMs without Searching. arXiv:2505.04588.

Tang, X.; Gao, Q.; Li, J.; Du, N.; Li, Q.; and Xie, S. 2025. MBA-RAG: a Bandit Approach for Adaptive Retrieval-Augmented Generation through Question Complexity. In Rambow, O.; Wanner, L.; Apidianaki, M.; Al-Khalifa, H.; Eugenio, B. D.; and Schockaert, S., eds., *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, 3248–3254. Association for Computational Linguistics.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10: 539–554.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In Rogers, A.; Boyd-Graber, J. L.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, 10014–10037. Association for Computational Linguistics.

Wang, L.; Yang, N.; Huang, X.; Yang, L.; Majumder, R.; and Wei, F. 2024. Improving Text Embeddings with Large Language Models. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 11897–11916. Bangkok, Thailand: Association for Computational Linguistics.

Wang, Z.; Zheng, X.; An, K.; Ouyang, C.; Cai, J.; Wang, Y.; and Wu, Y. 2025. StepSearch: Igniting LLMs Search Ability via Step-Wise Proximal Policy Optimization. arXiv:2505.15107.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E. H.; Le, Q. V.; and Zhou, D. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713871088.

Wu, J.; Feng, M.; Zhang, S.; Che, F.; Wen, Z.; and Tao, J. 2024. Beyond Examples: High-level Automated Reasoning Paradigm in In-Context Learning via MCTS. *CoRR*, abs/2411.18478.

Xu, J.; Du, X.; Ao, Y.; Zhao, P.; Li, Y.; Zhong, L.; Yuan, L.; Bo, Z.; Wang, X.; Sun, M.; Gui, Z.; Zhang, D.; Wang, Z.; Wang, Q.; Hou, Y.; Yin, Z.; Wang, H.; Chen, H.; Liang, L.; and Zhou, J. 2025. Thinker: Training LLMs in Hierarchical Thinking for Deep Search via Multi-Turn Interaction. arXiv:2511.07943.

Yang, L.; Yu, Z.; Cui, B.; and Wang, M. 2025. ReasonFlux: Hierarchical LLM Reasoning via Scaling Thought Templates. *CoRR*, abs/2502.06772.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In Riloff, E.; Chiang, D.; Hockenmaier, J.; and Tsujii, J., eds., *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2369–2380. Brussels, Belgium: Association for Computational Linguistics.

Zhang, D.; Xu, J.; Zhou, J.; Liang, L.; Yuan, L.; Zhong, L.; Sun, M.; Zhao, P.; Wang, Q.; Wang, X.; Du, X.; Hou, Y.; Ao, Y.; Wang, Z.; Gui, Z.; Yi, Z.; Bo, Z.; Wang, H.; and Chen, H. 2025. KAG-Thinker: Interactive Thinking and Deep Reasoning in LLMs via Knowledge-Augmented Generation. arXiv:2506.17728.