
ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning

Mingyang Chen¹, Linzhuang Sun², Tianpeng Li¹, Haoze Sun¹, Yijie Zhou¹,
Chenzheng Zhu¹, Haofen Wang³, Jeff Z. Pan⁴, Wen Zhang⁵, Huajun Chen⁵,
Fan Yang^{1*}, Zenan Zhou¹, Weipeng Chen¹

¹Baichuan Inc. ²University of Chinese Academy of Sciences

³Tongji University ⁴The University of Edinburgh ⁵Zhejiang University

{chenmingyang, yangfan}@baichuan-inc.com

<https://github.com/Agent-RL/ReSearch>

Abstract

Large Language Models (LLMs) have shown remarkable capabilities in reasoning, exemplified by the success of OpenAI-o1 and DeepSeek-R1. However, integrating reasoning with external search processes remains challenging, especially for complex multi-hop questions requiring multiple retrieval steps. We propose *ReSearch*, a novel framework that trains LLMs to *Reason* with *Search* via reinforcement learning without using any supervised data on reasoning steps. Our approach treats search operations as integral components of the reasoning chain, where when and how to perform searches is guided by text-based thinking, and search results subsequently influence further reasoning. We train *ReSearch* on Qwen2.5-7B(-Instruct) and Qwen2.5-32B(-Instruct) models and conduct extensive experiments. Despite being trained on only one dataset, our models demonstrate strong generalizability across various benchmarks. Analysis reveals that *ReSearch* naturally elicits advanced reasoning capabilities such as reflection and self-correction during the reinforcement learning process.

1 Introduction

In recent years, Large Language Models (LLMs) have demonstrated remarkable performance across a wide array of tasks [1, 5, 13, 39]. Beyond leveraging internal knowledge acquired during pre-training, LLMs exhibit the capability to utilize external tools, particularly search engines, to retrieve factual and time-sensitive information, thereby mitigating instances of hallucination [4, 14, 21, 25]. This capability, often referred to as Retrieval-Augmented Generation (RAG), has been the subject of extensive investigation in recent literature [2, 6, 38, 42]. Despite the effectiveness of RAG, designing robust multi-step RAG strategies applicable to complex real-world problems remains a significant challenge. This is particularly crucial, as many real-world issues are inherently complex and necessitate several steps of reasoning [23, 33, 35].

The past year has witnessed considerable advancements in LLMs’ reasoning abilities, particularly through chain-like reasoning before producing final outputs [37, 41]. This progress is exemplified by the success of OpenAI-o1 [17], and DeepSeek-R1 [5]. These developments emphasize the importance of test-time scaling in reasoning, enabling LLMs to decompose intricate problems into manageable intermediate steps [16, 28]. This reasoning capacity is also vital for the efficacy of RAG, especially when addressing complex questions that require multiple retrieval steps. Nonetheless, training LLMs

*Corresponding author

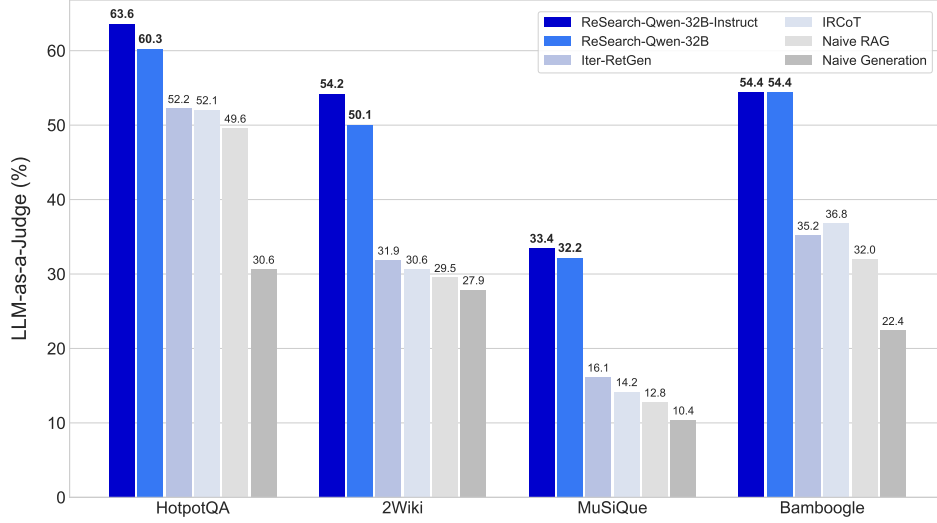


Figure 1: Comparative performance of *ReSearch* and baseline methods on benchmark datasets. All baselines are built upon Qwen2.5-32B-Instruct. See Section 3 for details.

to conduct interactive reasoning alongside information retrieval continues to present an open challenge for the research community. Most existing approaches to multi-step RAG rely on manually designed prompts or heuristics, which are not only labor-intensive but also lack scalability for more intricate problems [19, 23, 33]. Additionally, labeling reasoning steps in a multi-step RAG framework is often impractical due to the associated costs and time constraints.

Reinforcement learning (RL) has emerged as a promising avenue for enhancing reasoning capabilities without the need for supervised data regarding reasoning steps [5, 24]. This approach holds potential for training LLMs to exhibit reasoning skills solely based on simple reward signals derived from final outcomes. Recent advancements in RL-based training for LLMs have demonstrated significant improvements in complex reasoning tasks, where models learn to decompose problems into manageable steps through trial and error rather than explicit instruction. Models such as DeepSeek-R1 have shown that rule-based reward functions can effectively guide LLMs to develop sophisticated reasoning patterns autonomously. Despite these successes, current approaches primarily focus on enhancing internal reasoning capabilities, with limited exploration of how to effectively combine this reasoning process with external knowledge retrieval.

In this paper, we propose a novel framework for training LLMs to *Reason with Search* via reinforcement learning, which we term ***ReSearch***. The reasoning chain in this framework is not only composed of text-based thinking (i.e., enclosed by `<think>` `</think>`) as DeepSeek-R1, but also search query (i.e., enclosed by `<search>` `</search>`) and retrieval results (i.e., enclosed by `<result>` `</result>`). We treat the search operation as part of the chain-like reasoning process, and the search operation will interact with text-based thinking. Specifically, when and how to perform search will be steered by previous text-based thinking and the search results will influence subsequent text-based thinking. In the framework, we don’t provide any supervised data on reasoning steps for LLMs to imitate, instead, we leverage reinforcement learning (i.e., GRPO) to incentivize LLMs to perform reasoning with search.

We train *ReSearch* from scratch on Qwen2.5-7B(-Instruct) and Qwen2.5-32B(-Instruct), and conduct extensive experiments on multi-hop question answering benchmarks that need multi-step reasoning and multiple information retrieval. Our trained models show significant absolute improvements range from 8.9% to 22.4% over the baselines, as shown in Figure 1. Furthermore, our training is only conducted on one specific training set, and trained models are evaluated on multiple benchmarks, showing the generalizability of our framework. Our contributions are as follows:

- By emphasizing the interaction between reasoning and search, we propose a novel framework *ReSearch* that using reinforcement learning to train LLMs to reason with search from scratch, without any supervised data on reasoning steps.

- We train *ReSearch* on different scales of models, and conduct extensive experiments on multi-hop question answering benchmarks, showing the effectiveness of this framework. The trained models show significant generalizability and potential for more realistic scenarios.
- By analyzing the training process, we demonstrate that *ReSearch* can effectively elicit reasoning capabilities with search progressively itself, and that reasoning abilities such as reflection and self-correction can be incentivized without relying on any pre-defined heuristics.

2 Method

Drawing inspiration from the success of OpenAI-o1 and DeepSeek-R1 in learning to reason, we incorporate search operation into the reasoning process and train LLMs from scratch using reinforcement learning (i.e., GRPO) without any labeled data on reasoning chains, making LLMs learn to Reason with Search (*ReSearch*). In this section, we first show the details of training *ReSearch*, dive into the details of the GRPO and how to conduct rollout with search during reinforcement learning (§2.1). Then, we demonstrate the prompt template design directing the LLMs to generate the defined format of rollout (§2.2), and finally, we introduce the reward modeling for guiding the optimization of reinforcement learning (§2.3).

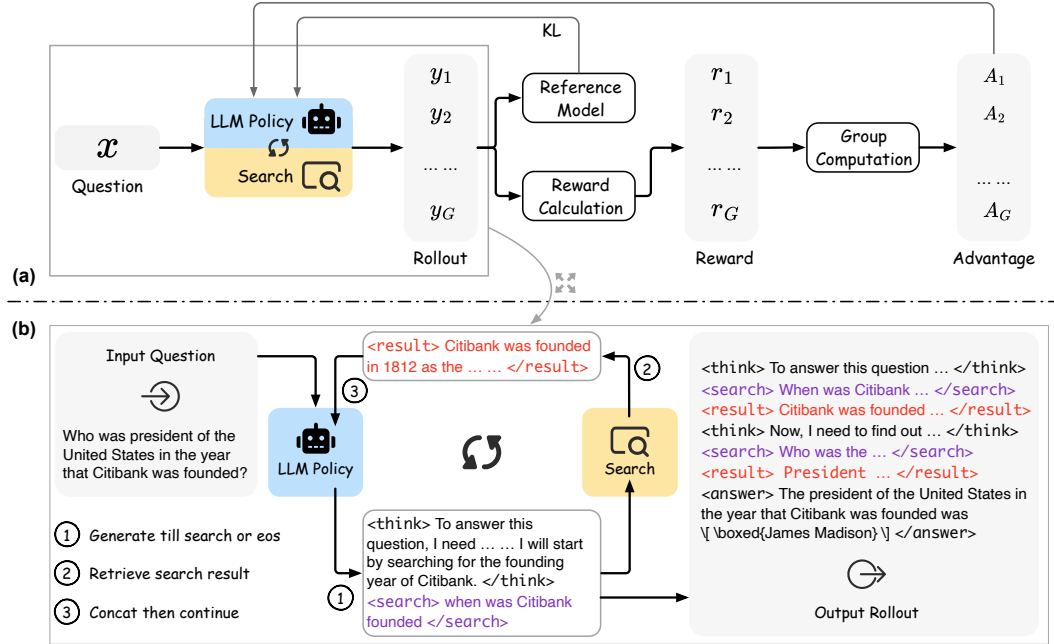


Figure 2: The training overview of *ReSearch*. (a) The GRPO pipeline. (b) The details of the rollout generation process.

2.1 Reinforcement Learning

When handling complex multi-step tasks needing retrieval (i.e., multi-step RAG), reasoning is crucial for steering multiple retrieval (i.e., search) operations, mainly on when and how to perform search. It’s challenging to collect labeled reasoning data with search for supervised fine-tuning LLMs to imitate how to reason with search. Fortunately, reinforcement learning has shown impressive performance in training LLMs to conduct reasoning, which can elicit reasoning capabilities from LLMs without any supervised data. In general, the main idea behind reinforcement learning here is to sample multiple reasoning-with-search chains (i.e., rollouts) and optimize the policy (i.e., LLMs) to maximize the probability of generating rollouts with higher rewards, as described in Figure 2.

Group Relative Policy Optimization Specifically, in this work, we use Group Relative Policy Optimization (GRPO) as the learning algorithm, which estimate the baseline from a group of rollouts

instead of training a separate critic model in Proximal Policy Optimization (PPO). Given an existing policy $\pi_{\theta_{\text{old}}}$ and an reference policy $\pi_{\theta_{\text{ref}}}$, base on G rollouts $\tau = \{y_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|x)$ for each input $x \sim \mathcal{D}$, the objective of GRPO is to optimize the policy π_{θ} by maximizing the following objective:

$$\mathcal{J}(\theta) = \mathbb{E}_{x \sim \mathcal{D}, \{y_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{G} \sum_{i=1}^G \left[\min \left(\frac{\pi_{\theta}(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)} A_i, \text{clip} \left(\frac{\pi_{\theta}(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}, 1 - \epsilon, 1 + \epsilon \right) A_i \right) - \beta \mathbb{D}_{\text{KL}}(\pi_{\theta} || \pi_{\theta_{\text{ref}}}) \right] \right], \quad (1)$$

where $A_i = (r_i - \text{mean}(\{r_j\}_{j=1}^G)) / \text{std}(\{r_j\}_{j=1}^G)$ is the normalized advantage of the i -th rollout in current group, ϵ is the clipping ratio, and β is the KL loss coefficient. Moreover, a KL divergence penalty is added to the objective to prevent the policy from deviating too much from the original reference policy LLMs. The illustration of GRPO is shown in Figure 2(a).

Rollout with Search Compared with conventional rollout that only contains text-based thinking as reasoning, the rollout in *ReSearch* also contains search queries and retrieval results. We use `<search>` and `</search>` to enclose the search queries and `<result>` and `</result>` to enclose the retrieval results, and such instruction is described in the prompt templates, which will be introduced later in §2.2. The rollout process is an iterative process between text-based thinking, search queries, and retrieval results as described in Figure 2(b). Specifically, when the generation process encounters `</search>` tag, the query between the last `<search>` and current `</search>` tags will be used as the search query to retrieve relevant factual information, and the retrieval results will be enclosed by `<result>` and `</result>` tags. Then, existing rollout concated with the retrieval results will be used as the next input to generate following response iteratively, until the generation encounters end-of-sentence (eos) tag (i.e., `<endoftext>` or `<im_end>` in Qwen-2.5 Models).

Retrieval Result Masking In original GRPO, the loss is calculated by all the generated tokens in the whole rollout. However, in *ReSearch*, the rollout contains retrieval results, which are not generated by the training policy, but retrieved by the search environment. Therefore, we mask the retrieval results in the loss calculation to avoid the training policy from being biased towards the retrieval results. That is, during the computation of Equation 1, we only consider the tokens in the text-based thinking and the search queries, and ignore the tokens in the retrieval results.

2.2 Training Template

Since we orchestrate the rollout process by identifying our defined special tags (e.g., stopping at `</search>` and transferring control to the search environment), it is crucial for policy LLMs to generate output in the defined format. To guide the LLMs in understanding this rollout format—specifically, the tags indicating when the search operation is invoked—we created two prompt templates: one for the base (i.e., pre-trained) model and another for the instruction-tuned model. As shown in Table 1, inspired by DeepSeek-R1, these templates are designed to be simple and concise, ensuring that the model can act as a natural progression during the reinforcement learning process. Specifically, for the *base model*, this template, filled with a specific user question, will be used as direct input to the LLMs. For the *instruction-tuned model*, its prompt template serves as the system prompt, utilized in conjunction with the corresponding chat template of the instruction-tuned LLM.

2.3 Reward Modeling

During reinforcement learning of *ReSearch*, there is no supervised reasoning data, and we only use a simple reward on rollouts to guide the optimization of LLMs. Experimentally, only rule-based reward function is enough to successfully elicit capabilities of reasoning with search for LLMs. Our reward function considers following two parts: answer reward and format reward.

- **Answer Reward:** We calculate the correctness of the final answer in `\boxed{\}` and the ground truth answer via F1 score.
- **Format Reward:** We check whether the rollout correctly follows our defined format as described in the prompt templates, mainly checking the correctness of tags and existence of `\boxed{\}` in the answer.

Table 1: Prompt templates for training from base model and instruction-tuned model. For the base model, **prompt** will be replaced with the actual question. For the instruction-tuned model, this template is used as the system prompt.

Prompt Template For Base Model
A conversation between User and Assistant. The user asks a question, and the assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. During thinking, the assistant can invoke the wikipedia search tool to search for fact information about specific topics if needed. The reasoning process and answer are enclosed within <code><think></code> <code></think></code> and <code><answer></code> <code></answer></code> tags respectively, and the search query and result are enclosed within <code><search></code> <code></search></code> and <code><result></code> <code></result></code> tags respectively. For example, <code><think></code> This is the reasoning process. <code></think></code> <code><search></code> search query here <code></search></code> <code><result></code> search result here <code></result></code> <code><think></code> This is the reasoning process. <code></think></code> <code><answer></code> The final answer is <code>\boxed{answer here}</code> <code></answer></code> . In the last part of the answer, the final exact answer is enclosed within <code>\boxed{}</code> with latex format. User: prompt . Assistant:
System Prompt Template For Instruction-Tuned Model
You are a helpful assistant that can solve the given question step by step with the help of the wikipedia search tool. Given a question, you need to first think about the reasoning process in the mind and then provide the answer. During thinking, you can invoke the wikipedia search tool to search for fact information about specific topics if needed. The reasoning process and answer are enclosed within <code><think></code> <code></think></code> and <code><answer></code> <code></answer></code> tags respectively, and the search query and result are enclosed within <code><search></code> <code></search></code> and <code><result></code> <code></result></code> tags respectively. For example, <code><think></code> This is the reasoning process. <code></think></code> <code><search></code> search query here <code></search></code> <code><result></code> search result here <code></result></code> <code><think></code> This is the reasoning process. <code></think></code> <code><answer></code> The final answer is <code>\boxed{answer here}</code> <code></answer></code> . In the last part of the answer, the final exact answer is enclosed within <code>\boxed{}</code> with latex format.

Specifically, for the final reward of a rollout:

$$r = \begin{cases} f1(a_{\text{pred}}, a_{\text{gt}}), & \text{if f1 score is not 0} \\ 0.1, & \text{if f1 score is 0 and format is correct} \\ 0, & \text{if f1 score is 0 and format is incorrect} \end{cases} \quad (2)$$

where a_{pred} is the final answer in `\boxed{}` and a_{gt} is the ground truth answer, and $f1(a_{\text{pred}}, a_{\text{gt}})$ is the F1 score between a_{pred} and a_{gt} .

3 Experiments

3.1 Experiment Setup

To evaluate the effectiveness of *ReSearch*, we conduct extensive experiments mainly on multi-hop question answering benchmarks that need multi-step reasoning and multiple information retrieval. Our *ReSearch* is trained from Qwen2.5-7B, Qwen2.5-7B-Instruct, Qwen2.5-32B and Qwen2.5-32B-Instruct [39]. During training, we only use the data from training set of MuSiQue [32], since it has various types of multi-hop questions and constructed via fine-grained quality control.

Benchmarks We use four standard benchmarks on multi-hop question answering tasks, including HotpotQA [40], 2WikiMultiHopQA [7], MuSiQue [32], and Bamboogle [19]. Specifically, HotpotQA, 2WikiMultiHopQA, and MuSiQue are constructed among wikipedia or wikidata [34], via different multi-hop mining strategies with crowd-sourcing, while Bamboogle is manually constructed dataset with 2-hop questions, where all questions are sufficiently difficult to be unanswerable by a popular internet search engine. Our evaluation is conducted on the full dev set of HotpotQA, 2WikiMultiHopQA, and MuSiQue, and the test set of Bamboogle, including 7405, 12576, 2417, 125 samples respectively. Note that we discard the context documents from the original datasets for

HotpotQA, 2WikiMultiHopQA, and MuSiQue, and only use the question and answer pairs for evaluation. We use an open-ended retrieval environment based on wikipedia to retrieve the background knowledge for all the datasets, which we introduce later.

Baselines We first compare *ReSearch* with two naive baselines: (1) *No RAG*: Use corresponding instruction-tuned model to generate answer directly without any RAG, and (2) *Naive RAG*: A naive retrieval-based setting that concatenate the retrieval results with question and then generate answer directly. Furthermore, we also consider two approaches focusing on improving multi-step RAG: (3) *Iter-RetGen* [23]: A method synergizes retrieval and generation in an iterative manner, and (4) *IRCoT* [33]: An interleaving method, which use retrieval and the chain-of-thought (CoT) guide each other. Since these methods are prompt-based, we use instruction-tuned models in same size as our *ReSearch* to implement them for fair comparison.

Evaluation Metrics For evaluate the correctness of the final answer, we first use Exact Match (*EM*) where the prediction is correct if it matches the ground truth answer exactly. However, such exact match is too strict for our setting, since the retrieval environment is open-ended and the result is described by natural language. Therefore, we also consider LLM-as-a-judge (*LJ*) for automatic evaluation, where we use gpt-4o-mini with our defined judge prompt to score the correctness of the final answer. Such judge prompt is shown in Appendix A.

Implementation Details We conduct our training and evaluation on Qwen2.5-7B, Qwen2.5-7B-Instruct, Qwen2.5-32B and Qwen2.5-32B-Instruct. The reinforcement learning framework is built on verl [26]. We only use the training set (19938 samples) of MuSiQue for training, and the number of training epochs is 2. The retrieval environment is based on FlashRAG [10], a standard toolkit for RAG research. We use E5-base-v2 [36] as the retriever and Wikipedia data from Dec. 2018 as the knowledge base [11]. All the corpus indexing and embedding has been preprocessed by FlashRAG. During the rollout in training and evaluation, we retrieve top-5 results for each query. For baseline methods, we use the implementation from FlashRAG. For details about model training, please refer to Appendix B.

3.2 Main Results

The main results of baselines and *ReSearch* are demonstrated in Table 2, and we show the methods based on LLMs with different sizes respectively. From the main results, we can draw the following observations:

Effectiveness of *ReSearch* Compared with all the baselines, *ReSearch* achieves significant improvements on all the benchmarks, which demonstrates the effectiveness of our proposed framework. Specifically, among all the benchmarks, the average improvement of *ReSearch* over the best baseline is **15.81%** in exact match and **17.56%** in LLM-as-a-judge, for Qwen2.5 model with 7B parameters. For Qwen2.5 model with 32B parameters, the average improvement is **14.82%** in exact match and **15.46%** in LLM-as-a-judge.

Comparison between base and instruction-tuned models We train *ReSearch* from both base and instruction-tuned models with 7B and 32B parameters respectively, and note that they are all trained using reinforcement learning from scratch without any supervised fine-tuning. From the results, we can observe training from the instruction-tuned model can further improve the performance of *ReSearch*. Such observation is consistent among all the benchmarks and model sizes.

Generalization Ability During reinforcement learning, *ReSearch* learns the ability of reasoning with search, which is independent of specific knowledge or multi-hop patterns, and such ability is generalizable. Our model *ReSearch* is only trained on the training set of MuSiQue dataset, but from the results, we can observe that it can generalize to other benchmarks with different question types and structures, which demonstrates the generalization ability of *ReSearch*.

3.3 Further Analysis

We investigate the important metrics during training *ReSearch* in this section. Specifically, the response length and number of search operations during training are shown in Figure 3 respectively.

Table 2: Exact Match (EM, %) and LLM-as-a-Judge (LJ, %) results on multi-hop question answering benchmarks. The best results are highlighted in bold, and the best results across baselines are underlined.

Model	HotpotQA		2Wiki		MuSiQue		Bamboogle	
	EM	LJ	EM	LJ	EM	LJ	EM	LJ
Qwen2.5-7B(-Instruct)								
Naive Generation	19.18	30.64	25.76	27.87	3.76	10.38	10.40	22.40
Naive RAG	31.90	49.59	25.78	29.52	6.21	12.78	20.80	32.00
Iter-RetGen	<u>34.36</u>	<u>52.22</u>	<u>27.92</u>	<u>31.86</u>	<u>8.69</u>	<u>16.14</u>	21.60	35.20
IRCoT	30.33	52.06	21.57	30.65	6.99	14.19	<u>24.80</u>	<u>36.80</u>
<i>ReSearch-Qwen-7B</i>	40.57	60.26	44.67	50.06	21.68	32.19	43.20	54.40
<i>ReSearch-Qwen-7B-Instruct</i>	43.52	63.62	47.59	54.22	22.30	33.43	42.40	54.40
Qwen2.5-32B(-Instruct)								
Naive Generation	24.63	38.26	27.23	29.68	6.12	14.23	18.40	29.60
Naive RAG	36.46	55.73	30.38	34.87	9.27	15.97	23.20	40.80
Iter-RetGen	<u>39.81</u>	<u>58.80</u>	<u>33.64</u>	<u>38.22</u>	<u>12.49</u>	<u>20.11</u>	29.60	44.80
IRCoT	28.44	55.44	13.53	29.50	7.82	18.20	<u>31.20</u>	<u>47.20</u>
<i>ReSearch-Qwen-32B</i>	42.77	64.27	38.52	45.59	26.40	37.57	54.40	66.40
<i>ReSearch-Qwen-32B-Instruct</i>	46.73	67.70	44.90	50.30	26.40	38.56	56.80	67.20

The curve of training reward and validation reward are shown in Figure 4. The validation is conducted on a part of development set of MuSiQue dataset with 100 random samples, and conducted every 10 steps during training.

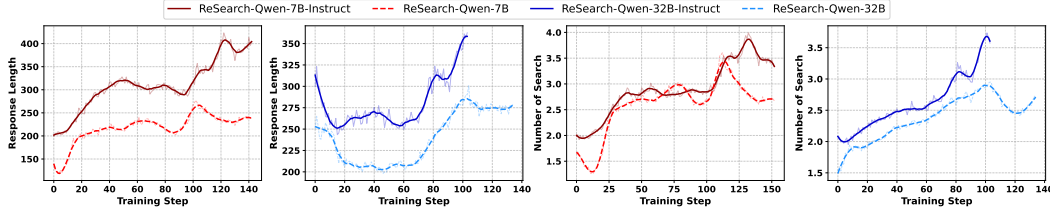


Figure 3: Response length and number of search operations during training.

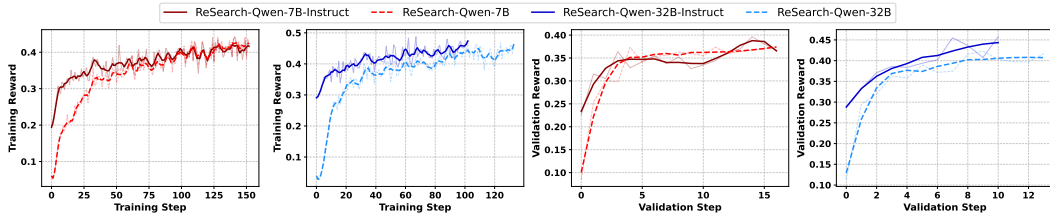


Figure 4: Training and validation reward during training.

Response Length We define the response length as the total number of tokens in a model’s output, excluding retrieval results, which can be interpreted as the test-time cost of reasoning. From the first two figures in Figure 3, it is evident that the response length generally increases throughout the training process. Instruction-tuned models exhibit longer response lengths compared to base models for both the 7B and 32B parameters. Additionally, for the 32B models, the response length initially decreases during the first 20 training steps before increasing again after approximately the 60th step. This observed behavior may be attributed to the 32B models having more inherent knowledge, leading

them to generate longer responses initially, and do not utilize retrieval. However, after receiving some guidance from reward signals early in the training, they begin to leverage searches more frequently, thereby reducing their reliance on generated knowledge.

Number of Search Operations We also count the average number of search operations in one rollout during training, which is shown in the last two figures in Figure 3. As depicted in the figures, the number of search operations shows consistent growth throughout the training process. Such trend indicates that for complex multi-hop questions, the model gradually learns to use search to retrieve relevant information iteratively multiple times.

Training and Validation Reward We show the training and validation reward during reinforcement learning of *ReSearch* in Figure 4. The reward increases sharply during the first 20 training steps, and then gradually increases. The cold-start reward is higher for instruction-tuned models for both 7B and 32B models. For 7B models, the rewards converge to a similar level for both base and instruction-tuned models, while for 32B models, the training reward of instruction-tuned models is higher than that of base models.

3.4 Case Study

Table 3: A case study of *ReSearch* during training.

Question: Who won the 2018 presidential election in the country where the political party of Martín Ramírez Pech operates? Answer: Andrés Manuel López Obrador
<pre> <think> To find the answer, I first need to determine the country in which Martín Ramírez Pech's political party operates. I will search for "Martín Ramírez Pech" to find out this information. </think> <search> Martín Ramírez Pech </search> <result> "Martín Ramírez", Martín Ramírez Martín Ramírez (January 30, 1895 – February 17, 1963) was ... \n "Martín Ramírez", and \$134,500 in New York in 2011. Martín Ramírez Martín Ramírez (January 30, 1895 – ... \n "Martín Ramírez", including brown paper bags, scraps of examining-table paper, and book pages ... </result> <think> It appears that I made a mistake in my search query. Instead of searching for "Martín Ramírez Pech", the results are showing information about a different "Martín Ramírez". To find out the right information, I need to look for Martín Ramírez Pech specifically and the political party associated with him. I will modify the search query to "Martín Ramírez Pech political party" for the correct information. </think> <search> Martín Ramírez Pech political party </search> <result> "Cuauhtémoc Blanco", of Morelos. In the 2015 legislative elections, he won in a closely contested election ... \n "Labor Party (Mexico)", for PT and 75 deputies and 16 senators for the PES. The alliance has received ... \n "Graco Ramírez", Graco Ramírez Graco Luis Ramírez Garrido Abreu (born 26 June 1949, in Villahermosa, Tabasco) is a Mexican ... </result> <think> Now, we see that Martín Ramírez Pech is associated with the political party known as the Labor Party (PT) in Mexico. I now need to continue this line of reasoning to find out who won the presidential election in 2018 in Mexico. </think> <search> 2018 Mexican presidential election </search> <result> "2018 Mexican general election", (INE) officially declared the new process underway on 8 September 2017. The presidential election was won by Andrés Manuel López Obrador of ... \n "2018 Mexican general election", irregularities including reports of vote-buying, spending beyond legal campaign ... \n "2018 Mexican general election", 2018 Mexican general election General elections were held in Mexico on 1 July 2018 ... </result> <think> According to the search results, the 2018 Mexican presidential election was won by Andrés Manuel López Obrador as the candidate for the National Regeneration Movement (MORENA). </think> <answer> The final answer is \boxed{Andrés Manuel López Obrador} </answer> </pre>

To give a more intuitive understanding of the utility of *ReSearch*, we show a case study in Table 3. This case is from the reinforcement learning process of Qwen2.5-32B-Instruct model. Where text enclosed by `<think>` and `</think>`, `<search>` and `</search>`, and `<answer>` and `</answer>` are generated by the model, and the text enclosed by `<result>` and `</result>` are retrieved from the retrieval environment. For clarity, we use "... .." to represent the truncation of the retrieval results. From this case, we can see that the model can effectively break down the complex question and conduct reasoning within `<think>` and `</think>`. Such reasoning process is crucial for guiding when and what to search, and leading to the final answer in a multi-step manner.

Self-elicited Reflection In addition, we also observe reflection phenomenon in the model’s response. As depicted in the second thinking step in Table 3, the model states, “**I made a mistake**”, recognizing that the previous search query failed to retrieve useful information. It then corrects itself in the third thinking step by generating a more effective search query to obtain the relevant information. Note that such reflection ability is not explicitly trained or designed in the prompt templates, but is naturally elicited from the model itself during reinforcement learning.

4 Related Work

4.1 Reinforcement Learning with LLMs

Reinforcement learning [30], which aims to maximize the expected return of an agent’s policy through interactions with the environment, has emerged as a crucial technique for LLMs, from aligning with human values to enhancing reasoning capabilities. A significant development was Reinforcement Learning from Human Feedback (RLHF) [18], which uses Proximal Policy Optimization (PPO) [22] with reward models trained on human preferences. Several methods have since improved upon PPO, including Direct Preference Optimization (DPO) [20], Simulated Preference Optimization (SimPO) [15], and Group Relative Policy Optimization (GRPO) [24]. Recently, reinforcement learning has demonstrated remarkable success in enhancing reasoning capabilities, as evidenced by notable achievements such as OpenAI-o1 [17], DeepSeek-R1 [5], and Kimi k1.5 [31]. However, the reasoning capabilities of LLMs under RAG settings remain largely unexplored. Meanwhile, several concurrent works have also begun to investigate reinforcement learning for enhancing LLM reasoning with tool use [9, 29].

4.2 Retrieval-Augmented Generation for LLMs

RAG techniques [8, 12] augment LLMs by retrieving external knowledge and incorporating it into the generation process. Extensive research has been conducted in this area, encompassing various aspects such as retriever optimization [27, 38], query refinement [3], and self-reflection mechanisms [2]. For more complex scenarios, particularly in addressing multi-hop questions, iterative RAG models [23, 33] have been developed that alternately perform retrieval-enhanced generation and generation-enhanced retrieval. Additionally, supervised learning approaches have been explored using annotated trajectories of multi-step retrieval [21, 35]. Despite these advances, the application of reinforcement learning to enhance reasoning capabilities within RAG-style tool-augmented environments remains an open and under-explored research direction.

5 Conclusion

In this paper, we introduced *ReSearch*, a novel framework that trains LLMs to reason with search via reinforcement learning without requiring any supervised data on reasoning steps. Our approach integrates search operations as integral components of the reasoning chain, where text-based thinking guides when and how to perform searches, and search results subsequently influence further reasoning. Through extensive experiments on multiple multi-hop question answering benchmarks, we demonstrated that *ReSearch* achieves significant improvements over baseline methods. The results also indicate the framework’s potential for more realistic scenarios. Analysis of the training process revealed that *ReSearch* naturally elicits advanced reasoning capabilities such as reflection and self-correction, without relying on pre-defined heuristics. This work highlights the effectiveness of integrating reasoning and search operations through reinforcement learning, offering a promising direction for developing more capable and reliable LLM-based systems for complex multi-hop tasks.

References

- [1] Anthropic. Claude 3.7 sonnet and claude code, 2025. URL <https://www.anthropic.com/news/claude-3-7-sonnet>.
- [2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*. OpenReview.net, 2024.
- [3] Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. RQ-RAG: learning to refine queries for retrieval augmented generation. *CoRR*, abs/2404.00610, 2024.
- [4] Mingyang Chen, Haoze Sun, Tianpeng Li, Fan Yang, Hao Liang, Keer Lu, Bin Cui, Wentao Zhang, Zenan Zhou, and Weipeng Chen. Facilitating multi-turn function calling for llms via compositional instruction tuning. *CoRR*, abs/2410.12952, 2024.
- [5] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025.
- [6] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997, 2023.
- [7] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *COLING*, pages 6609–6625. International Committee on Computational Linguistics, 2020.
- [8] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*, 1(2):4, 2022.
- [9] Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025.
- [10] Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *CoRR*, abs/2405.13576, 2024.
- [11] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781. Association for Computational Linguistics, 2020.
- [12] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

- [13] Mingan Lin, Fan Yang, Yanjun Shen, Haoze Sun, Tianpeng Li, Tao Zhang, Chenzheng Zhu, Tao Zhang, Miao Zheng, Xu Li, Yijie Zhou, Mingyang Chen, Yanzhao Qin, Youquan Li, Hao Liang, Fei Li, Yadong Li, Mang Wang, Guosheng Dong, Kun Fang, Jianhua Xu, Bin Cui, Wentao Zhang, Zenan Zhou, and Weipeng Chen. Baichuan alignment technical report. *CoRR*, abs/2410.14940, 2024.
- [14] Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn LLM agents. In *NeurIPS*, 2024.
- [15] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.
- [16] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *CoRR*, abs/2501.19393, 2025.
- [17] OpenAI. Learning to reason with LLMs, 2024. URL <https://openai.com/index/learning-to-reason-with-llms>.
- [18] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- [19] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *EMNLP (Findings)*, pages 5687–5711. Association for Computational Linguistics, 2023.
- [20] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.
- [21] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.
- [22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.
- [23] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *EMNLP (Findings)*, pages 9248–9274. Association for Computational Linguistics, 2023.
- [24] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.
- [25] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving AI tasks with chatgpt and its friends in hugging face. In *NeurIPS*, 2023.
- [26] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. *CoRR*, abs/2409.19256, 2024.
- [27] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: retrieval-augmented black-box language models. In *NAACL-HLT*, pages 8371–8384. Association for Computational Linguistics, 2024.
- [28] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314, 2024.

- [29] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *CoRR*, abs/2503.05592, 2025.
- [30] Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999.
- [31] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.
- [32] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554, 2022.
- [33] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *ACL (1)*, pages 10014–10037. Association for Computational Linguistics, 2023.
- [34] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014.
- [35] Junjie Wang, Mingyang Chen, Binbin Hu, Dan Yang, Ziqi Liu, Yue Shen, Peng Wei, Zhiqiang Zhang, Jinjie Gu, Jun Zhou, Jeff Z. Pan, Wen Zhang, and Huajun Chen. Learning to plan for retrieval-augmented large language models from knowledge graphs. In *EMNLP (Findings)*, pages 7813–7835. Association for Computational Linguistics, 2024.
- [36] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *CoRR*, abs/2212.03533, 2022.
- [37] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.
- [38] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *CoRR*, abs/2401.15884, 2024.
- [39] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024.
- [40] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pages 2369–2380. Association for Computational Linguistics, 2018.
- [41] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. In *NeurIPS*, 2022.
- [42] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense text retrieval based on pretrained language models: A survey. *ACM Trans. Inf. Syst.*, 42(4):89:1–89:60, 2024.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: We claim the scope of this paper in the beginning of Abstract and Instruction, and at the end of the Introduction, we show the contribution of our paper.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We discuss the limitations of our work in Appendix C.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: This work mainly focus on the empirical evaluation, and the theoretical analysis is not included.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: The main algorithm and training regime are provided in the Method section, and the implementation details are also provided in Section B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have provided the code and usage instructions in the supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The training setup is provided in Section 3.1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: In our experiments, the results conducted by 3 independent runs, with different random seeds, and we report the mean performance across these runs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide the compute resources in the Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We follow the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss the broader impacts of our work in Appendix D.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly credit the creators of the assets, and respect the license and terms of use. For benchmark datasets, HotpotQA is distributed under the Apache-2.0 license, MuSiQue is distributed under CC-BY-4.0 license, Bamboogle is distributed under MIT license, and 2WikiMultiHopQA is distributed under Apache-2.0 license. For the code, verl is distributed under Apache-2.0 license, and FlashRAG is distributed under MIT license.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: This paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A Prompt for LLM-as-a-Judge

Prompt for Extracting Scenarios

You will be given a question and its ground truth answer list where each item can be a ground truth answer. Provided a `pred_answer`, you need to judge if the `pred_answer` correctly answers the question based on the ground truth answer list. You should first give your rationale for the judgement, and then give your judgement result (i.e., correct or incorrect).

Here is the criteria for the judgement:

1. The `pred_answer` doesn't need to be exactly the same as any of the ground truth answers, but should be semantically same for the question.
2. Each item in the ground truth answer list can be viewed as a ground truth answer for the question, and the `pred_answer` should be semantically same to at least one of them.

question: {question}
ground truth answers: {gt_answer}
pred_answer: {pred_answer}

The output should in the following json format:

```
```json
{
 "rationale": "your rationale for the judgement, as a text",
 "judgement": "your judgement result, can only be 'correct'
 or 'incorrect'"
}
```

Your output:

## B Implementation Details

Our training is conducted on  $8 \times 8$  Nvidia H800 GPUs, with full parameter optimization and gradient checkpointing. We show some important parameter settings in Table 4.

Table 4: Implementation details of *ReSearch*.

Parameter	Value
Learning Rate	1e-6
Train Batch Size	256
Number of Training Epochs	2
Number of Rollout	5
Rollout Temperature	1.0
KL Loss Coefficient	0.001
Clip Ratio	0.2

## C Limitation

While our work demonstrates promising results in training LLMs to reason with search, there are some limitations to consider. Our current framework primarily focuses on scenarios where the



answers are relatively concise and can be objectively verified through simple metrics like F1 score. This approach may not generalize well to tasks requiring longer, more nuanced responses, where more sophisticated reward modeling would be necessary to effectively guide the reinforcement learning process. Additionally, like many existing works in retrieval-augmented generation, our study utilizes Wikipedia as the primary knowledge base for RAG operations, following the common practice due to the availability of standardized open-source knowledge bases. This limitation means we have not yet explored the framework’s effectiveness with other types of specialized or domain-specific knowledge bases that might be more appropriate for certain applications. Future work could investigate extending our approach to handle more complex response types and diverse knowledge sources beyond general encyclopedic knowledge.

## **D Broader Impact**

Our work on *ReSearch* has several potential positive societal impacts. By improving the ability of LLMs to reason with search capabilities, this framework can enhance the accuracy and reliability of AI systems in knowledge-intensive tasks, particularly benefiting fields such as education, scientific research, and fact-checking. The framework’s ability to break down complex questions into manageable steps while verifying information through external sources could help reduce the spread of misinformation and improve the quality of AI-assisted decision-making. Furthermore, the self-reflective capabilities that emerge during training could lead to more transparent and explainable AI systems, fostering greater trust between users and AI technologies. However, we acknowledge potential concerns that warrant consideration. The increased efficiency in information retrieval and processing could lead to higher computational resource consumption, potentially contributing to environmental impacts through increased energy usage. Additionally, while our framework improves accuracy, there remains a small possibility of reinforcing existing biases present in the search results or knowledge bases used for training. To mitigate these concerns, we recommend implementing energy-efficient training strategies and regularly auditing the search sources used in the system. Overall, we believe the benefits of more accurate, transparent, and reliable AI systems outweigh these manageable risks, particularly as the technology continues to evolve with appropriate safeguards and monitoring mechanisms in place.