# MEMSEARCHER: TRAINING LLMS TO REASON, SEARCH AND MANAGE MEMORY VIA END-TO-END REINFORCEMENT LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Traditional search agents concatenate the entire interaction history into the LLM context, preserving information integrity but producing long, noisy contexts, resulting in high computation and memory costs. In contrast, using only the current turn avoids this overhead but discards essential information. This trade-off limits the scalability of search agents. To address this challenge, we propose MemSearcher, an agent workflow that iteratively maintains a compact memory and combines the current turn with it. At each turn, MemSearcher fuses the user's question with the memory to generate reasoning traces, perform search actions, and update memory to retain only information essential for solving the task. This design stabilizes context length across multi-turn interactions, improving efficiency without sacrificing accuracy. To optimize this workflow, we introduce multi-context GRPO, an end-to-end RL framework that jointly optimize reasoning, search strategies, and memory management of MemSearcher Agents. Specifically, multi-context GRPO samples groups of trajectories under different contexts and propagates trajectory-level advantages across all conversations within them. Trained on the same dataset as Search-R1, MemSearcher achieves significant improvements over strong baselines on seven public benchmarks: +11% on Qwen2.5-3B-Instruct and +12% on Qwen2.5-7B-Instruct relative average gains. Notably, the 3B-based MemSearcher even outperforms 7B-based baselines, demonstrating that striking a balance between information integrity and efficiency yields both higher accuracy and lower computational overhead. Our code and models will be publicly available.

## 1 INTRODUCTION

Large Language Models (LLMs) (Team, 2024; Achiam et al., 2023) have demonstrated impressive performance in understanding and generating natural language, as well as in solving complex tasks in the real world. Despite their strengths, LLMs still exhibit notable shortcomings in addressing knowledge-acquisition tasks (Wei et al., 2024; He et al., 2024). These shortcomings arise from their insufficient long-tailed and up-to-date knowledge in specific domains and susceptibility to hallucinations (Xu et al., 2024; Zhang et al., 2025c).

A promising strategy to mitigate these issues is to integrate search engines with LLMs, allowing them to access external and up-to-date information. Considerable efforts have been devoted to this area in recent years. In Retrieval-Augmented Generation (RAG) methods (Gao et al., 2023; Zhao et al., 2024), a search engine is used to select relevant documents according to the the input of the LLM, and the retrieved documents are fed into the LLM context to generate the final response. While straightforward, these methods often rely on predefined pipelines (Zhou et al., 2025a; Zhu et al., 2025b) and do not fully explore the potential of LLMs in leveraging search engines. To address this limitation, search agents, which treat a search engine as a tool, have been developed.

A representative paradigm to build search agents is ReAct (Yao et al., 2023). In ReAct, the interactions between the agent and the search engine are modeled as a multi-turn conversation, which means that the entire interaction history is incorporated into the context of the agent's backbone LLM. This paradigm provides the agent with fine-grained information—including all reasoning pro-

cesses, performed actions and corresponding tool responses from previous interactions—to support more effective decision-making. However, the continuously appended interaction history leads to unbounded growth of the context of the LLM, which substantially increases the GPU memory and computational overhead.

In this paper, we introduce MemSearcher, an agentic workflow that maintains a compact, iteratively updated memory throughout interactions, preserving only the information deemed essential for addressing the user's question. At each turn, MemSearcher provides the backbone LLM with two succinct inputs, the user question and a compact memory, rather than the entire, ever-growing interaction history. The LLM first generates the reasoning trace and performs an action based on it. After the new observation is returned to the agent by the environment, the LLM then functions as a memory manager to update the memory based on the previous memory and the current interaction. Since the number of tokens in the memory is restricted by a predefined maximum length, this design keeps per-turn contexts short and stable while preserving salient facts and intermediate findings across multi-turn reasoning and interactions.

Since current LLMs have not been optimized under the MemSearcher workflow, they are not yet capable of mastering it. We employ Reinforcement Learning (RL) (Wiering & Van Otterlo, 2012) to train MemSearcher agents, which enables models to improve by leveraging their self-generated samples as optimization targets. Among RL algorithms, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) has recently emerged as the most widely adopted method, as it improves LLM abilities while optimizing the GPU memory usage of Proximal Policy Optimization (PPO) (Schulman et al., 2017). We extend vanilla GRPO to multi-context GRPO to facilitate the training of MemSearcher agents, whose trajectories consist of multiple conversations under different contexts. Specifically, multi-context GRPO propagates trajectory-level advantages to each conversation among them and subsequently treats every conversation as an independent optimization target. This extension enables a stable and scalable training for MemSearcher-based agents.

We use the same data as Search-R1 (Jin et al., 2025) to train MemSearcher from scratch on Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct, and conduct extensive evaluation on a range of public knowledge-acquisition benchmarks that need reasoning and external information retrieval. Our MemSearcher agents demonstrate significant improvement over the baseline methods, yielding 11% and 12% increases on different models, respectively. Moreover, MemSearcher also achieves higher efficiency than the ReAct paradigm. Compared to ReAct-based search agents, which exhibits a steady increase in token numbers with interaction turns, our MemSearcher agents maintain nearly constant token counts within contexts.

We summarize our main contributions as follows.

- We introduce MemSearcher, an agentic workflow that leverages the backbone LLM as a memory manager to iteratively maintain a compact memory, preserving only the essential information necessary for answering the user's question and thereby eliminating the need to append the entire interaction history to the LLM context.
- We develop search agents based on MemSearcher, and utilize multi-context GRPO, a natural extension of GRPO, to optimize LLMs to reason, leverage search engines and manage memory simultaneously. Multi-context GRPO provides end-to-end RL training for trajectories that contain multiple conversations under different contexts.
- We use the same data as Search-R1 to train our search agents. The evaluation on seven public benchmarks demonstrates the effectiveness and efficiency of our method, with two LLMs achieving average relative improvements of 11% and 12%, respectively. Compared with ReAct-based search agents, which exhibit a nearly linear increase in token numbers during interactions, MemSearcher agents maintain lower and more stable token counts.

## 2 Background

### 2.1 Preliminary: ReAct

ReAct (Yao et al., 2023), which integrates reasoning and acting, has become the most popular paradigm for building LLM-based agents Jin et al. (2025); Chen et al. (2025).

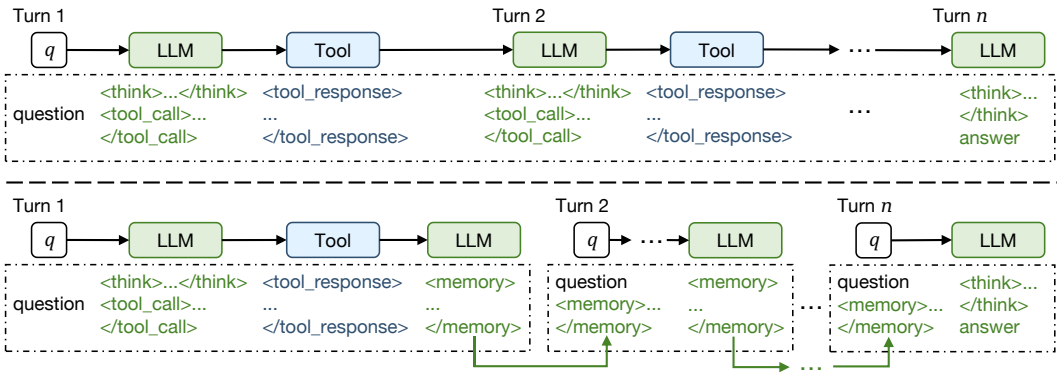



Figure 1: Comparison between ReAct (Top) and MemSearcher (Bottom). The dashed box illustrates the content included in the LLM context. While ReAct continuously appends all thoughts, actions and observations into the LLM context, MemSearcher iteratively updates a compact memory that retains only the essential information.

As shown in Figure 1 (Top), the core idea of ReAct is straightforward: a ReAct trajectory is a multi-turn conversation, and each turn is an interaction between the LLM agent and the environment, containing thought, action, and observation. At each turn, the LLM first generates a thought within `<think> </think>`, and then performs an action in `<tool_call> </tool_call>`, to interact with the environment, after which the environment provides an observation between `<tool_response> </tool_response>` in response to the performed action.

Specifically, we assume that at the $i$-th turn, the agent generates a thought $t_i$, takes an action $a_i$, and receives an observation $o_i$. In particular, $o_0 = q$ represents the observation prior to the first turn, where $q$ denotes the user's question. Then, the context to the LLM is as follows:

$$c_i = (q, t_1, a_1, o_1, \cdots, t_{i-1}, a_{i-1}, o_{i-1}). \tag{1}$$

At the $i$-th turn, the agent generates the thought $t_i$ and performs the corresponding action $a_i$, following policy $\pi(t_i, a_i | c_i)$.

### 2.2 Limitations of ReAct

Although straightforward and simple, this paradigm leads to a continuous increase in the number of tokens in the LLM context, due to its design of appending all previous thoughts, actions and observations. This increase is almost linear with the number of interaction turns, placing significant pressure on the inference of LLMs. For example, Liu et al. (2023) find that LLMs do not reliably make use of information from long contexts. Hsieh et al. (2024) demonstrate that LLMs exhibit large performance drops as the context length increases. Wu et al. (2024) reveal that LLMs show a significant accuracy drop on memorizing information across sustained multi-turn interactions. In addition, in the context of search agents, the observations are passages retrieved by the search engine, which often include substantial noise and information irrelevant to answering the user's question. This further constrains the performance and scalability of ReAct-based search agents. Moreover, the linear growth in the number of tokens leads to increased memory consumption and computational overhead. Since the computational complexity of LLMs scales as $O(n^2)$ with the number of tokens $n$, the computational cost of these search agents increases quadratically with the number of interaction turns. Consequently, more efficient and scalable approaches for building search agents need to be explored.

## 3 Method

### 3.1 Overview of MemSearcher

The MemSearcher workflow is illustrated in Figure 1 (Bottom). At the $i$-th turn, the LLM receives only two inputs: the user's question $q$, enclosed within `<question> </question>` tags, and a

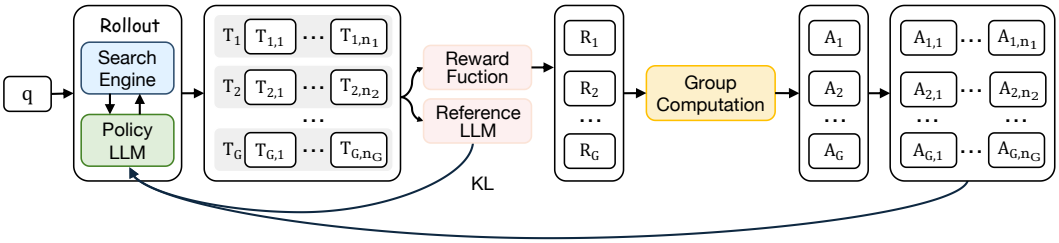



Figure 2: Multi-context GRPO. In rollout, we sample a group of trajectories $\{T_i\}_{i=1}^{G}$ for question $q$. The $i$-th trajectory $T_i$ consists of multiple conversations $\{T_{i,j}\}_{j=1}^{n_i}$ under different contexts. Then, we compute rewards $\{R_i\}_{i=1}^{G}$, and derive the trajectory-level advantages $\{A_i\}_{i=1}^{G}$ from these rewards. We propagate trajectory-level advantages to each conversation within them, *i.e.* $A_{i,j} = A_i$, and treat each conversation as an independent optimization target to update the policy LLM.

compact memory $m_{i-1}$ expressed in natural language, between `<memory> </memory>`, which encapsulates all the relevant information considered helpful to answer the question so far. In particular, the memory $m_0$ prior to the first turn is empty. Therefore, the context to the LLM at the $i$-th is formulated as:

$$c_i = (q, m_{i-1}). \tag{2}$$

After reading the user's question and the previous memory, the LLM generates a thought $t_i$ enclosed within `<think> </think>` and performs an action $a_i$ between `<tool_call> </tool_call>` tags, following policy $\pi(t_i, a_i|c_i)$. As the action is executed, the environment returns the observation $o_i$ within `<tool_response> </tool_response>` tags to the agent.

After receiving $o_i$, MemSearcher overwrites the previous memory to a updated one for the next turn. The LLM are asked to carefully reads $o_i$ and incorporates any new information that helps to answer the question, while preserving all relevant details from the previous memory $m_{i-1}$. The resulting memory are denoted as $m_i$.

Different from ReAct, which continuously concatenates all historical thoughts, actions and observations into the LLM context, MemSearcher compresses only the essential information into a compact memory. Since the number of tokens in the memory never exceeds a predefined maximum length, MemSearcher maintains the context within a few thousands of tokens while retaining important information through iterative updates of the memory. This process continues iteratively, until the maximum number of interactions is reached or sufficient information is gathered and the LLM generates a final answer as its action.

Specifically, under the setting of search agents designed to solve knowledge-acquisition tasks by leveraging search engines as tools, action $a_i$ takes one of the following two forms: (1) providing a final answer in `\boxed{}` to the user's question and terminating the interactions, (2) issuing a search engine call with a query to obtain additional information to answer the question. If the latter is chosen, the observation $o_i$ is the relevant passages retrieved from search engines in response to the search query.

### 3.2 RL Training Algorithm

In this subsection, we introduce multi-context GRPO, the training algorithm of our MemSearcher agents. Figure 2 illustrates the overview of multi-context GRPO.

We use end-to-end reinforcement learning (RL) to train our MemSearcher agents, since it allows models to evolve themselves through their self-generated samples. In contrast, Supervised Fine-Tuning (SFT) requires costly, carefully curated high-quality trajectories, such as Li et al. (2025a); Sun et al. (2025b); Wu et al. (2025); Schick et al. (2023). For RL, we utilize Group Relative Policy Optimization (GRPO) (Shao et al., 2024), as it optimizes the memory usage of Proximal Policy Optimization (PPO) (Schulman et al., 2017) and has recently become the most widely adopted RL algorithm for RLVR due to its effectiveness (Guo et al., 2025).

Vanilla GRPO samples a group of trajectories $\{T_1, T_2, \cdots, T_G\}$ for each question $q$, and then optimizes the policy model $\pi_\theta$ by maximizing the following objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim D, \{T_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot|q)] \\ \frac{1}{G}\sum_{i=1}^{G}\left(\min\left(r_i(\theta)A_i, \text{clip}\left(r_i(\theta), 1-\epsilon, 1+\epsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right), \tag{3}$$

where

$$r_i(\theta) = \frac{\pi_\theta(T_i|q)}{\pi_{\theta_{old}}(T_i|q)}. \tag{4}$$

$A_i$ represents the normalized advantage, calculated by using the rewards $\{R_1, R_2, \cdots, R_G\}$ within each group:

$$A_i = \frac{R_i - mean(\{R_1, R_2, \cdots, R_G\})}{std(\{R_1, R_2, \cdots, R_G\})}. \tag{5}$$

In the training of MemSearcher, each trajectory consists of multiple conversations under different LLM contexts. Therefore, we extend the vanilla GRPO algorithm to a natural extension, multi-context GRPO, as illustrated in Figure 2. Specifically, we assume that trajectory $T_i$ contains $n_i$ conversations, represented as $\{T_{i,1}, T_{i,2}, \cdots, T_{i,n_i}\}$. According to Section 3.1, the $j$-th conversation can be represented as:

$$T_{i,j} = \begin{cases} (q, m_{i,j-1}, t_{i,j}, a_{i,j}, o_{i,j}, m_{i,j}), & \text{if } j = 1, 2, \cdots, n_i - 1 \\ (q, m_{i,j-1}, t_{i,j}, a_{i,j}), & \text{if } j = n_i \end{cases} \tag{6}$$

where memories $m_{i,j-1}$ and $m_{i,j}$, thought $t_{i,j}$ and action $a_{i,j}$ are generated by the policy model, and observation $o_{i,j}$ is the retrieved text from the search engine.

We compute reward $R_i$ for each trajectory, and calculate its advantage $A_i$ within the group using Equation 5. Then, we uniformly propagate this advantage to all conversations whithin the trajectory, and use each conversation as an independent target to optimize the policy model. The training objective is formulated as:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim D, \{T_{i,j}\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot|q, m_{i,j-1})] \\ \frac{1}{\sum_{i=1}^{G} n_i}\sum_{i=1}^{G}\sum_{j=1}^{n_i}\left(\min\left(r_{i,j}(\theta)A_{i,j}, \text{clip}\left(r_{i,j}(\theta), 1-\epsilon, 1+\epsilon\right)A_{i,j}\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right), \tag{7}$$

where

$$r_{i,j}(\theta) = \frac{\pi_\theta(T_{i,j}|q, m_{i,j-1})}{\pi_{\theta_{old}}(T_{i,j}|q, m_{i,j-1})} \text{ and } A_{i,j} = A_i. \tag{8}$$

Notably, conversation $t_{i,j}$ consists of tokens from both the policy model and the search engine. Following previous RL-based search agents, such as Search-R1 (Jin et al., 2025) and ReSearch (Chen et al., 2025), we use loss masking for the tokens from the search engine, ensuring the policy gradient objective is computed only over model-generated tokens and thereby stabilizing RL training.

### 3.3 Reward Modeling

The reward serves as the primary training signal in RL, guiding the optimization process of models. During the training of MemSearcher, we only adopt a simple reward function on the generated samples. Similar to DeepSeek-R1 (Guo et al., 2025), our reward function considers two parts: format reward and answer reward.

- **Format Reward**: It checks whether the rollout correctly follows our predefined format, including the correctness of usage of tags and the existence of `\boxed{}` in the answer.
- **Answer Reward**: A rule-based reward assesses the correctness of the model's response. It is calculated by using the F1 score between the final answer inside `\boxed{}` and the ground truth.

Table 1: Performance comparison. Exact Match (EM) is used as the evaluation metric. The best performance is highlighted in **bold**, while the second-best performance is indicated with an underline. Among these methods, R1-Searcher and ZeroSearch interact with the realistic web environment during their evaluation, while other methods, including MemSearcher, interact only with local knowledge base. MemSearcher based on Qwen2.5-3B-Instruct achieves a higher average score than other methods based on Qwen2.5-7B-Instruct.

| Methods | NQ | TriviaQA | PopQA | HotpotQA | 2wiki | Musique | Bamboogle | Avg. |
|---|---|---|---|---|---|---|---|---|
| **Qwen2.5-3B-Instruct** | | | | | | | | |
| Direct Answer | 10.6 | 28.8 | 10.8 | 14.9 | 24.4 | 2.0 | 2.4 | 13.4 |
| CoT | 2.3 | 3.2 | 0.5 | 2.1 | 2.1 | 0.2 | 0.0 | 1.5 |
| IRCoT | 11.1 | 31.2 | 20.0 | 16.4 | 17.1 | 6.7 | 24.0 | 18.1 |
| RAG | 34.8 | 54.4 | 38.7 | 25.5 | 22.6 | 4.7 | 8.0 | 27.0 |
| Search-o1 | 23.8 | 47.2 | 26.2 | 22.1 | 21.8 | 5.4 | 32.0 | 25.5 |
| Search-R1 | 34.1 | 54.5 | 37.8 | 32.4 | 31.9 | 10.3 | 26.4 | 32.5 |
| ReSearch | 20.4 | 33.5 | 17.3 | 35.6 | 39.3 | 17.3 | 37.6 | 28.7 |
| AutoRefine | 43.6 | 59.7 | 44.7 | 40.4 | 38.0 | 16.9 | 33.6 | 39.6 |
| ZeroSearch | 41.4 | 57.4 | 44.8 | 27.4 | 30.0 | 9.8 | 11.1 | 31.7 |
| MemSearcher | **47.0** | **63.8** | **47.9** | **43.9** | **43.5** | **17.9** | **42.4** | **43.8** |
| **Qwen2.5-7B-Instruct** | | | | | | | | |
| Direct Answer | 13.4 | 40.8 | 14.0 | 18.3 | 25.0 | 3.1 | 12.0 | 18.1 |
| CoT | 4.8 | 18.5 | 5.4 | 9.2 | 11.1 | 2.2 | 23.2 | 10.6 |
| IRCoT | 22.4 | 47.8 | 30.1 | 13.3 | 14.9 | 7.2 | 22.4 | 23.9 |
| RAG | 34.9 | 58.5 | 39.2 | 29.9 | 23.5 | 5.8 | 20.8 | 30.4 |
| Search-o1 | 15.1 | 44.3 | 13.1 | 18.7 | 17.6 | 5.8 | 29.6 | 20.6 |
| Search-R1 | 39.3 | 61.0 | 39.7 | 37.0 | 41.4 | 14.6 | 36.8 | 38.5 |
| ReSearch | 40.9 | 63.7 | 44.6 | 43.5 | 47.6 | 22.3 | 42.4 | 43.6 |
| R1-Searcher | 40.4 | 52.2 | 41.0 | 44.2 | **51.3** | 15.8 | 36.8 | 40.2 |
| ZeroSearch | 43.6 | 65.2 | **48.8** | 34.6 | 35.2 | 18.4 | 27.8 | 39.1 |
| MemSearcher | **52.7** | **68.1** | 47.8 | **50.8** | 48.6 | **25.8** | **48.8** | **48.9** |

The reward function is formulated as:

$$R = \begin{cases} 0, & \text{if format is incorrect} \\ 0.1, & \text{if format is correct but F1}(a_{\text{pred}}, a_{\text{gold}}) \text{ is 0} \\ \text{F1}(a_{\text{pred}}, a_{\text{gold}}), & \text{if format is correct and F1}(a_{\text{pred}}, a_{\text{gold}}) \text{ is not 0} \end{cases} \tag{9}$$

where $a_{\text{pred}}$ is the final answer extracted from the model's response, $a_{\text{gold}}$ is the ground truth, and $\text{F1}(a_{\text{pred}}, a_{\text{gold}})$ is the F1 score between $a_{\text{pred}}$ and $a_{\text{gold}}$.

# 4 Experiments

## 4.1 Experiment Setups

**Baselines.** We compare MemSearcher against three categories of baseline methods, including: (1) Inference without retrieval, such as Direct inference and Chain-of-Thought (CoT) reasoning (Wei et al., 2022); (2) Inference with Retrieval, such as RAG (Lewis et al., 2020), IRCoT (Trivedi et al., 2022a), and Search-o1 (Li et al., 2025b); (3) RL-based search agents, such as Search-R1 (Jin et al., 2025), ReSearch (Chen et al., 2025), AutoRefine (Shi et al., 2025), R1-Searcher (Song et al., 2025), and ZeroSearch (Sun et al., 2025a). Among these baselines, R1-Searcher and ZeroSearch interact with the realistic web environment via Google Web Search during their evaluation.

**Benchmarks and Evaluation Metrics.** We compare MemSearcher and the baseline methods on a range of public benchmarks that encompass search with reasoning challenges, such as Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2022), Bamboogle (Press et al., 2022), Musique (Trivedi et al., 2022b), HotpotQA (Yang et al., 2018), and 2WikiMultiHopQA (Ho et al., 2020). We use Exact Match (EM) as the evaluation metric, where the prediction is correct if it matches the ground truth answer exactly.

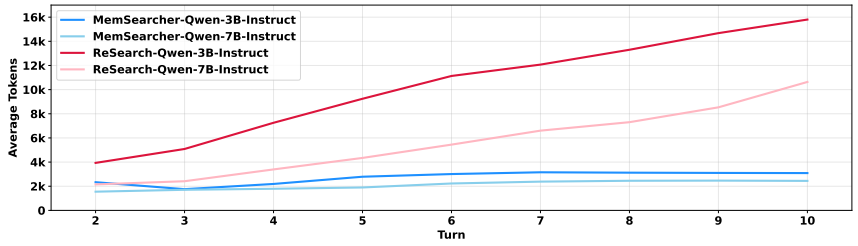



Figure 3: Comparison of the average token number in the LLM context between MemSearcher and ReAct-based ReSearch.

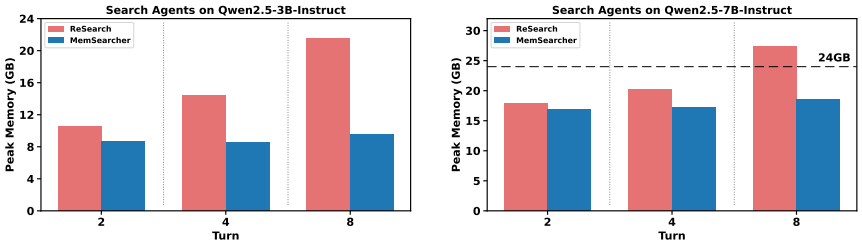



Figure 4: Peak GPU memory usage (GB) comparison between MemSearcher and ReSearch.

**Implementation Details.** We conduct our training and evaluation on Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct. We use the 2018 Wikipedia dump (Karpukhin et al., 2020) as the knowledge source and E5 (Wang et al., 2022) as the retriever. We conduct the training based on verl (Sheng et al., 2025), constrain the model to an 8K context window, and set the maximum tokens in the memory to 1,024 tokens. For training, we follow Search-R1, using its fully open training data, including the training splits of two datasets, NQ (Kwiatkowski et al., 2019) and HotpotQA (Yang et al., 2018), to form the dataset for training. Within the datasets used for training, NQ is a general question answering dataset, while HotpotQA is a multi-hop question answering dataset. For evaluation, we systematically test MemSearcher and the baseline methods on seven datasets, covering both in-domain and out-of-domain scenarios. This setup enables us to rigorously assess not only how well the models generalize to questions that resemble the training distribution, but also how robust they are when applied to domains that differ from the training data. Such a comprehensive evaluation provides deeper insights into the effectiveness of our approach under varied conditions.

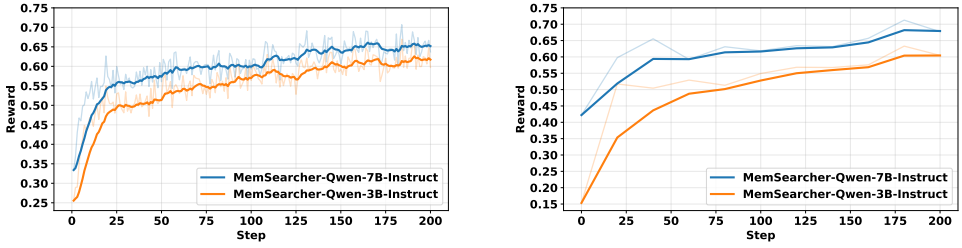
### 4.2 Main Results

In Table 1, we provide a comprehensive performance comparison between MemSearcher and the baseline methods across the evaluated benchmarks. Several key observations can be drawn from these results: (1) When trained on the same datasets as Search-R1, MemSearcher consistently outperforms the baseline methods, demonstrating the superior effectiveness of our method. These performance improvements are consistently observed across both in-distribution benchmarks such as NQ and HotpotQA, and out-of-distribution benchmarks, such as TriviaQA, PopQA, 2WikiMultiHopQA, Musique and Bamboogle. (2) Remarkably, even when using a smaller backbone model, *i.e.* Qwen2.5-3B-Instruct, MemSearcher achieves an average EM score 43.8 on the seven benchmarks, higher than those of the baseline methods based on the lager model, *i.e.* Qwen2.5-7B-Instruct, suggesting that MemSearcher makes more effective use of model capacity. (3) Furthermore, MemSearcher surpasses the baseline methods that rely on the realistic web search engine. Specifically, MemSearcher achieves superior performance compared to R1-Searcher and ZeroSearch, both of which depend on Google Web Search to retrieve external information during their evaluation.

In addition to the improvement in performance, MemSearcher also achieves superior token efficiency compared with ReAct-based search agents, since it eliminates the need to append all historical thoughts, actions and observations into the LLM context, as discussed in Section 3. To validate this, we record the number of tokens in the LLM contexts of MemSearcher and ReAct-based ReSearch

Table 2: Comparison between models with and without training. Exact Match (EM) is used as the evaluation metric. The better performance is highlighted in **bold**.

| Methods | General QA | | | Multi-Hop QA | | | | |
|---|---|---|---|---|---|---|---|---|
| | NQ | TriviaQA | PopQA | HotpotQA | 2wiki | Musique | Bamboogle | Avg. |
| **Qwen2.5-3B-Instruct** | | | | | | | | |
| *w/o* training | 16.4 | 23.8 | 22.5 | 11.9 | 11.0 | 3.7 | 11.2 | 14.4 |
| *w/* training | **47.0** | **63.8** | **47.9** | **43.9** | **43.5** | **17.9** | **42.4** | **43.8** |
| **Qwen2.5-7B-Instruct** | | | | | | | | |
| *w/o* training | 22.1 | 41.2 | 23.5 | 27.4 | 27.8 | 11.6 | 27.2 | 25.8 |
| *w/* training | **52.7** | **68.1** | **47.8** | **50.8** | **48.6** | **25.8** | **48.8** | **48.9** |




(a) Training Reward (b) Validation Reward

Figure 5: Training and validation reward during training. The validation is conducted on a part of development set of HotpotQA with 100 randomly selected samples, and conducted every 20 steps during training. The curves are smoothed for clarity.

at each turn and calculate their average across the evaluated datasets. The results are illustrated in Figure 3. Compared to ReSearch, which exhibits an almost linear increase in token consumption during the interaction process, MemSearcher maintains substantially lower and more stable token counts. The efficiency gain is primarily attributed to the design of MemSearcher, which iteratively updates a compact memory as context to preserve only the essential information for the question throughout the interactions. Moreover, we present the peak GPU memory usage comparison in Figure 4. We can observe that MemSearcher supports more scalable and cost-efficient multi-turn interactions in search agents.

### 4.3 Further Analysis

#### 4.3.1 Do we need RL Training?

To investigate the impact of RL training on the performance of MemSearcher, we perform a comparative analysis. The baselines are Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct models, both of which are integrated with the MemSearcher workflow but do not undergo RL training. As shown in Table 2, the models without RL training demonstrate a pronounced performance degradation across all evaluated benchmarks. This observation highlights the necessity of RL training in equipping models with the ability to effectively interact with both the search engine and memory, thereby enhancing their overall functionality and task-solving ability.

#### 4.3.2 Training and Validation Reward.

We present the curves of training and validation reward in Figure 5, which offer an intuitive view of the models' learning dynamics during training. For the validation, we construct a validation dataset by randomly sampling 100 examples from the development set of HotpotQA. We conduct validation at fixed intervals, specifically every 20 training steps. The observed reward patterns reveals the following two phases of learning: (1) Early stage (first 25 steps). In this phase, the reward increases sharply. This improvement indicates that the models rapidly acquire the fundamental ability to interact effectively with the search engine and memory. (2) Later stage (after 25 steps). In contrast, the reward grows at a more gradual pace. This improvement suggests that the models are refining their strategy, progressively enhancing its capacity to exploit the search engine and manage memory.

The difference between these two stages underscores the transition from basic skill acquisition to more advanced optimization of reasoning behaviors.

# 5 Related Work

## 5.1 Large Language Models with Search Engines

Although Large Language Models (LLMs) (Team et al., 2025; Comanici et al., 2025; Zeng et al., 2025) have made significant progress in solving complex tasks in the real world (Guo et al., 2024), they often lack knowledge in specific domains (Peng et al., 2023; Li et al., 2023). To address these issues, Retrieval-Augmented Generation (RAG) integrates search engines (Xiao et al., 2024; Zhuang et al., 2024) with LLMs to provide relevant external information. In a typical RAG pipeline (Lewis et al., 2020; Yue et al., 2024; Xiong et al., 2025), a search engine first selects relevant documents based on the input query, and then the retrieved content is fed into an LLM to produce responses. Previous studies on RAG guides LLMs through processes such as search query generation and decomposition (Yu et al., 2022; Press et al., 2022). Although RAG enhances the performance of LLMs, it faces challenges related to the retrieval of irrelevant information (Zhu et al., 2025a; Jin et al., 2024a) and the absence of sufficiently useful context (Jiang et al., 2023). In addition to RAG, another approach to integrate external search engines with LLMs is to treat search engines as tools and LLMs as agents, named search agent (Zong et al., 2024). For example, ReAct (Yao et al., 2023) integrates search into the reasoning process by interleaving it with Chains-of-Thought (CoT) (Wei et al., 2022) steps. Recent studies (Jin et al., 2025; Chen et al., 2025; Zheng et al., 2025) develop agentic reinforcement learning (RL) (Zhang et al., 2025b) for search agents, based on multi-turn chat. Although effective, current RL-based search agents (Wu et al., 2025; Tao et al., 2025) primarily adhere to the ReAct workflow, lacking the exploration of more efficient paradigms.

## 5.2 Context Management

Most LLM agents utilize ReAct (Yao et al., 2023) for context management, which incorporates the entire interaction history between the LLM and the environment into the LLM context. While simple, it leads to prolonged token sequences and reduced efficiency. To address these issues, memory mechanisms are proposed to manage the context of LLMs. RAG-style memory systems (Jimenez Gutierrez et al., 2024; Zhong et al., 2024) treat memory as an external knowledge source, similar to that in RAG, and use predefined management strategies to store, integrate and retrieve relevant information (Zhu et al., 2023). Token-level memory systems (Jin et al., 2024b; Zhou et al., 2025b; Orlicki, 2025) equip models with explicit, trainable context managers and optimize them via SFT or RL algorithms such as PPO (Schulman et al., 2017), allowing agents to regulate their memory at the token level. For example, Wang et al. (2024) and Wang et al. (2025) maintain a fixed set of latent tokens serves as memory, and Yang et al. (2024) equip LLMs with a forget-resistant memory for evolving context. MemAgent (Yu et al., 2025) reforms long-context processing as an agent task, maintains a token-level memory alongside the LLM to compress long-context inputs into more concise, informative summaries. Structured memory systems (Zeng et al., 2024) organize and encode information in structured representation, such as knowledge graph in Zep (Rasmussen et al., 2025), the atomic memory units in A-MEM (Xu et al., 2025), and the hierarchical graph-based memory in Mem0 (Chhikara et al., 2025) and G-Memory (Zhang et al., 2025a). In this paper, we utilize the backbone LLM of search agents as a memory manager, and optimize it for reasoning, action, and memory management via end-to-end multi-context GRPO algorithm.

# 6 Conclusion

In this paper, we propose MemSearcher, an agentic workflow that retains a compact memory as LLM context throughout the interaction process between the agent and the environment, thereby eliminating the need to append all historical thoughts, actions and observations, as in the ReAct paradigm. We utilize a natural extension of GRPO, namely multi-context GRPO, to optimize search agents based on our workflow in an end-to-end fashion. These agents demonstrate superior performance across a range of public benchmarks compared with previous ReAct-based baselines, while maintaining nearly constant token consumption during interactions with the environment, highlighting more scalable and cost-efficient multi-turn interactions in search agents.

## 7 Ethics Statement

We affirm our strict adherence to the ICLR Code of Ethics in all aspects of this work. Our research does not involve human participants, personal data, or sensitive information. The datasets used (*e.g.*, NQ, HotpotQA, and other public QA benchmarks) are entirely publicly available and widely used within the research community, ensuring no privacy or security concerns arise.

Our experiments are focused on algorithmic innovation in context management and reinforcement learning (RL) for large language model (LLM)-based agents. While our methods aim to improve computational efficiency and accuracy, care should be taken if applying them in domains where model outputs may have real-world impact. We advise thorough assessment of fairness, reliability, and bias mitigation in any such downstream use. Finally, we intend to make our code and models publicly available to promote transparency and foster ongoing research integrity.

## 8 Reproducibility Statement

We are committed to ensuring the reproducibility of our work. All datasets used in our experiments are publicly available and described in Section 4.1 of this paper. Implementation details, including models, hyperparameters, training procedures, and evaluation metrics, are described in Section 4.1 as well as in Appendix A.2. Additionally, we will release all code and trained models.

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1), 2023.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Yancheng He, Shilong Li, Jiaheng Liu, Yingshui Tan, Weixun Wang, Hui Huang, Xingyuan Bu, Hangyu Guo, Chengwei Hu, Boren Zheng, et al. Chinese simpleqa: A chinese factuality evaluation for large language models. *arXiv preprint arXiv:2411.07140*, 2024.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569, 2024.

Bowen Jin, Jinsung Yoon, Jiawei Han, and Sercan O Arik. Long-context llms meet rag: Overcoming challenges for long inputs in rag. *arXiv preprint arXiv:2410.05983*, 2024a.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. Disentangling memory and reasoning ability in large language models. *arXiv preprint arXiv:2411.13504*, 2024b.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pp. 6769–6781, 2020.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

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.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025a.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*, 2025b.

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, pp. 374–382, 2023.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 7, 2022.

José I Orlicki. Beyond words: A latent memory approach to internal reasoning in llms. *arXiv preprint arXiv:2502.21030*, 2025.

Cheng Peng, Xi Yang, Aokun Chen, Kaleb E Smith, Nima PourNejatian, Anthony B Costa, Cheryl Martin, Mona G Flores, Ying Zhang, Tanja Magoc, et al. A study of generative large language model for medical research and healthcare. *NPJ digital medicine*, 6(1):210, 2023.

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

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Yaorui Shi, Sihang Li, Chang Wu, Zhiyuan Liu, Junfeng Fang, Hengxing Cai, An Zhang, and Xiang Wang. Search and refine during think: Autonomous retrieval-augmented reasoning of llms. *arXiv preprint arXiv:2505.11277*, 2025.

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. *arXiv preprint arXiv:2503.05592*, 2025.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*, 2025a.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, et al. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. *arXiv preprint arXiv:2505.16834*, 2025b.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, et al. Webshaper: Agentically data synthesizing via information-seeking formalization. *arXiv preprint arXiv:2507.15061*, 2025.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Qwen Team. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022a.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022b.

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. *arXiv preprint arXiv:2212.03533*, 2022.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, et al. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*, 2024.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending memoryllm with scalable long-term memory. *arXiv preprint arXiv:2502.00592*, 2025.

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

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*, 2024.

Marco A Wiering and Martijn Van Otterlo. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3):729, 2012.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, et al. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*, 2025.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 641–649, 2024.

Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, et al. Rag-gym: Optimizing reasoning and search agents with process supervision. *arXiv preprint arXiv:2502.13957*, 2025.

Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*, 2024.

Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, et al. Memory3: Language modeling with explicit memory. *arXiv preprint arXiv:2407.01178*, 2024.

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. *arXiv preprint arXiv:1809.09600*, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. Generate rather than retrieve: Large language models are strong context generators. *arXiv preprint arXiv:2209.10063*, 2022.

Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. Inference scaling for long-context retrieval augmented generation. *arXiv preprint arXiv:2410.04343*, 2024.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

Ruihong Zeng, Jinyuan Fang, Siwei Liu, and Zaiqiao Meng. On the structural memory of llm agents. *arXiv preprint arXiv:2412.15266*, 2024.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *arXiv preprint arXiv:2506.07398*, 2025a.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025b.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: A survey on hallucination in large language models. *Computational Linguistics*, pp. 1–46, 2025c.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4):1–60, 2024.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*, 2025.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024.

Yingli Zhou, Yaodong Su, Youran Sun, Shu Wang, Taotao Wang, Runyuan He, Yongwei Zhang, Sicong Liang, Xilin Liu, Yuchi Ma, et al. In-depth analysis of graph-based rag in a unified framework. *arXiv preprint arXiv:2503.04338*, 2025a.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025b.

Rongzhi Zhu, Xiangyu Liu, Zequn Sun, Yiwei Wang, and Wei Hu. Mitigating lost-in-retrieval problems in retrieval augmented multi-hop question answering. *arXiv preprint arXiv:2502.14245*, 2025a.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023.

Zulun Zhu, Tiancheng Huang, Kai Wang, Junda Ye, Xinghe Chen, and Siqiang Luo. Graph-based approaches and functionalities in retrieval-augmented generation: A comprehensive survey. *arXiv preprint arXiv:2504.10499*, 2025b.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. Efficientrag: Efficient retriever for multi-hop question answering. *arXiv preprint arXiv:2408.04259*, 2024.

Chang Zong, Yuchen Yan, Weiming Lu, Jian Shao, Eliot Huang, Heng Chang, and Yueting Zhuang. Triad: A framework leveraging a multi-role llm-based agent to solve knowledge base question answering. *arXiv preprint arXiv:2402.14320*, 2024.

# A APPENDIX

## A.1 LLM USAGE

We used OpenAI's ChatGPT to help polish the language and improve the readability of the manuscript. Specifically, ChatGPT was used for grammar checking and sentence rephrasing. We list our prompt for using OpenAI's ChatGPT to help polish writing as follows.

**Prompt for Using OpenAI's ChatGPT to Help Polish Writing**

```
Below is a paragraph from an academic paper. Polish the
   writing to meet the academic style,improve the spelling,
   grammar, clarity, concision and overall readability.
   Furthermore, list all modification and explain the
   reasons to do so in markdown table. \\
Paragraph: {paragraph}
```

## A.2 TRAINING HYPERPARAMETERS

We train MemSearcher agents with full parameter optimization and gradient checkpointing. We show some important training hyperparameters in Table 3.

Table 3: Training details of MemSearcher.

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

## A.3 DETAILS OF EVALUATED DATASETS

We evaluate MemSearcher agents on the following public question answering datasets:

- **Natural Questions (NQ)** (Kwiatkowski et al., 2019), a QA dataset with questions consisting of real anonymized, aggregated queries issued to the Google search engine.
- **TriviaQA** (Joshi et al., 2017), a large scale challenging dataset with relatively complex, compositional questions, requireing more reasoning to find answers.
- **PopQA** (Mallen et al., 2022), a dataset consisting of 14k questions covering factual information in the long tail.
- **Bamboogle** (Press et al., 2022), a manually constructed dataset with multi-hop questions, where all questions are sufficiently difficult to be unanswerable by a single search engine call.
- **Musique** (Trivedi et al., 2022b), a QA dataset with 25K questions that by require proper multi-hop reasoning.
- **HotpotQA** (Yang et al., 2018), a dataset with 113k Wikipedia-based question-answer pairs, where the questions requires finding and reasoning over multiple supporting documents to answer, and and not constrained to any existing knowledge bases or knowledge schemas.
- **2WikiMultiHopQA** (Ho et al., 2020), a multi-hop QA dataset that uses both structured and unstructured data. It is challenging for multi-hop models and explicitly ensures that multi-hop reasoning is required.

### A.4 CASE STUDY

To provide a more intuitive understanding of the utility of MemSearcher, we present a case study in Table 4. This case is drawn from the evaluation of MemSearcher based on Qwen2.5-7B-Instruct. The text enclosed by <think> and </think>, <tool_call> and </tool_call>, as well as <memory> and </memory> is generated by the model. The text enclosed by <tool_response> and </tool_response> is retrieved from the search engine. This case demonstrates that the model can effectively maintain a compact memory, retaining only the essential information necessary to solve the question.

Table 4: A case study of MemSearcher.

**Question:** What was the form of the language that the last name Sylvester comes from, used in the era of Rotrude's father, later known as?
**Answer:** Medieval Latin

**Turn 1:** <think> To answer this question, I need to find out which language the last name Sylvester comes from. Let's start by searching Wikipedia for this information. </think>
<tool_call> {"name": "wikipedia_search", "arguments": {"query": "Last name Sylvester comes from what language"}} </tool_call>
<tool_response> "Sylvester (singer)", Sylvester (singer) Sylvester James Jr. (September 6, 1947December 16, 1988), who used the stage name of Sylvester, was an American singer-songwriter. Primarily active in ... ... </tool_response>
<memory> The last name Sylvester comes from the Latin language. </memory>

**Turn 2:** <think> The memory mentions that the last name Sylvester comes from the Latin language. However, it does not provide any information about the form of the Latin language used in the era of Rotrude's father, later known as. I need to find out which era Rotrude's father lived in. </think>
<tool_call> {"name": "wikipedia_search", "arguments": {"query": "Who was Rotrude's father, and what was he later known as?"}} </tool_call>
<tool_response> "Rotrude", Rotrude Rotrude (or sometimes referred to as Hruodrud/Hruodhaid) (775/778 – 6 June 810) was a Frankish princess, the second daughter of Charlemagne from his marriage to Hildegard. Few clear records ... ... </tool_response>
<memory> The last name Sylvester comes from the Latin language. Rotrude was a Frankish princess, the second daughter of Charlemagne from his marriage to Hildegard. </memory>

**Turn 3:** <think> The memory mentions that the last name Sylvester comes from the Latin language. However, it does not provide any information about the form of the Latin language used in the era of Rotrude's father, later known as. I need to find out who Charlemagne was. </think>
<tool_call> {"name": "wikipedia_search", "arguments": {"query": "Who was Charlemagne"}} </tool_call>
<tool_response> "Charlemagne", Charlemagne Charlemagne or Charles the Great (2 April 742 – 28 January 814), numbered Charles I, was King of the Franks from 768, King of the Lombards from 774, and Holy Roman Emperor from ... ... </tool_response>
<memory> The last name Sylvester comes from the Latin language. Rotrude was a Frankish princess, the second daughter of Charlemagne from his marriage to Hildegard. Charlemagne was later known as Holy Roman Emperor. </memory>

**Turn 4:** <think> The last name Sylvester comes from the Latin language. Charlemagne was later known as Holy Roman Emperor. The question asks what was the form of the Latin language used in the era of Rotrude's father, later known as. Since Rotrude's father was Charlemagne, and Charlemagne was later known as Holy Roman Emperor, we need to find out what was the form of the Latin language used in his era. I will use the wikipedia_search function to find this information. </think>
<tool_call> {"name": "wikipedia_search", "arguments": {"query": "Form of Latin language used in the era of Charlemagne"}} </tool_call>
<tool_response> "Medieval Latin", Medieval Latin Medieval Latin was the form of Latin used in Roman Catholic Western Europe during the Middle Ages. In this region it served as the primary written language, though local languages were also written to ... ... </tool_response>
<memory> The last name Sylvester comes from the Latin language. Rotrude was a Frankish princess, the second daughter of Charlemagne from his marriage to Hildegard. Charlemagne was later known as Holy Roman Emperor. The form of Latin language used in the era of Charlemagne was Medieval Latin. </memory>

**Turn 5:** <think> The memory mentions that the form of Latin language used in the era of Charlemagne was Medieval Latin. </think>
The answer to the question is \boxed{Medieval Latin}.