Beyond Monolithic Architectures: A Multi-Agent Search and Knowledge Optimization Framework for Agentic Search

Anonymous ACL submission

Abstract

Agentic search has emerged as a promising paradigm for complex information seeking by enabling Large Language Models (LLMs) to interleave reasoning with tool use. However, prevailing systems rely on monolithic agents that suffer from structural bottlenecks, including unconstrained reasoning outputs that inflate trajectories, sparse outcome-level rewards that complicate credit assignment, and stochastic search noise that destabilizes learning. To address these challenges, we propose **M-ASK** (Multi-Agent Search and Knowledge), a framework that explicitly decouples agentic search into two complementary roles: Search Behavior Agents, which plan and execute search actions, and Knowledge Management Agents, which aggregate, filter, and maintain a compact internal context. This decomposition allows each agent to focus on a well-defined subtask and reduces interference between search and context construction. Furthermore, to enable stable coordination, M-ASK employs turn-level rewards to provide granular supervision for both search decisions and knowledge updates. Experiments on multi-hop QA benchmarks demonstrate that M-ASK outperforms strong baselines, achieving not only superior answer accuracy but also significantly more stable training dynamics.[1]

1 Introduction

The rapid evolution of Large Language Models (LLMs) has fundamentally reshaped information retrieval, driving a paradigm shift from passive keyword matching to *Agentic Search* (Shi et al., 2025). Unlike traditional retrieval systems, agentic search systems function as autonomous decision-makers capable of iterative planning, external tool querying, and information synthesis to address complex, multi-hop user needs. Represented by Search-r1 (Jin et al., 2025), recent advancements (Song et al., 2025a,b; Zheng et al., 2025) integrate reasoning directly into LLM-based multi-round search workflows. By executing multi-step search within a single response via end-to-end optimization, these approaches significantly enhance performance on complex QA tasks—capabilities that remain beyond the reach of static search paradigms (Ma et al., 2023; Ke et al., 2024).

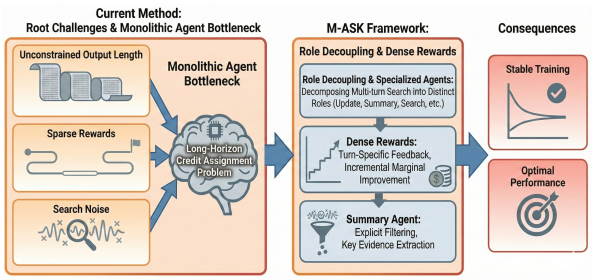


Figure 1: Challenges of current monolithic methods and the M-ASK solution. Existing agents struggle with the *long-horizon credit assignment problem* caused by unconstrained output length, sparse rewards and search noise. M-ASK addresses these bottlenecks through role decoupling and turn-level dense rewards.

However, training robust agents to navigate these dynamic search environments presents significant challenges. Prevailing approaches predominantly adopt a *monolithic* architecture that executes multi-round search within a single, continuous response. In this paradigm, the LLM shoulders the heavy burden of both trajectory planning and information processing at every iterative step of the generation. We argue that this monolithic design is structurally vulnerable to three intertwined obstacles: (i) **unconstrained output length**, where agents generate verbose reasoning chains that extend search horizons without necessarily increasing information density; (ii) **sparse rewards**, where feedback is typically delayed until task completion, hindering effective step-wise credit assignment; and (iii) **search noise**, where external tools such as search

[1]The source code for M-ASK is available at https://anonymous.4open.science/r/M-ASK-36F2.

engines introduce noise and irrelevant data into the context.

As illustrated in Figure 1, these challenges are not isolated; rather, they compound to destabilize training. The interaction between extended trajectories and sparse, outcome-only feedback creates a severe *long-horizon credit assignment* problem: optimization algorithms struggle to attribute the final reward to specific, distant tokens. This fragility is further exacerbated by search engine noise—when stochastic context infiltrate an already lengthy and sparsely rewarded episode, the learning signal becomes effectively indistinguishable from variance. Consequently, monolithic agents (Jin et al., 2025) frequently suffer from suboptimal state and high training instability (Deng et al., 2025).

To overcome the limitations of the monolithic method, we propose **M-ASK** (**M**ulti-**A**gent **S**earch and **K**nowledge), a framework that fundamentally disentangles the decision-making of search from the burden of information integration. Rather than relying on a single agent, M-ASK orchestrates a collaboration between two specialized roles:

1. **Search Behavior Agents** (including the Planning, Search, and Answer Agents), which focus exclusively on trajectory planning, interacting with search engine, and generating answers;
2. **Knowledge Management Agents** (including the Summary and Update Agents), which act as dynamic filters to prune noisy observations, update and maintain a concise internal knowledge state, and thus, constrain context length.

Crucially, M-ASK abandons the reliance on sparse feedback in favor of **turn-specific dense rewards**. By jointly optimizing all agents with immediate, turn-aware supervision, we ensure that the Knowledge Management Agents actively stabilize the state space, empowering the Search Agents to conduct more accurate planning. Consequently, this synergistic collaboration effectively mitigates the impact of noise and long horizons.

Our contributions are summarized as follows:

- We identify the compound impact of output verbosity, sparse rewards, and tool noise on agentic search, which explains why monolithic architectures struggle with training stability (Figure 1 and Section 4.3).
- We introduce M-ASK, a collaborative framework that decouples search behavior from knowledge management, utilizing dense, turn-specific rewards to enable joint optimization.
- Extensive evaluations on multi-hop QA benchmarks demonstrate that M-ASK significantly outperforms state-of-the-art baselines, delivering robust gains in accuracy and markedly improving convergence stability in multi-hop search scenarios.

2 Related Work

2.1 From Iterative RAG to Agentic Search

The paradigm of Retrieval-Augmented Generation (RAG) has evolved significantly from static "retrieve-then-read" pipelines to dynamic systems. Early iterative approaches, such as IRCoT (Trivedi et al., 2023) and Self-RAG (Asai et al., 2023), introduced feedback loops where LLMs actively decide when to retrieve. Recent advancements have formalized this into **Agentic Search**, where models plan multi-step trajectories to solve open-ended problems using external tools. Notably, emerging *Deep Research* agents, such as DeepResearcher (Zheng et al., 2025), Search-o1 (Li et al., 2025), Search-r1 (Jin et al., 2025), and the R1-Searcher series (Song et al., 2025a,b), integrate retrieval directly into the reasoning chains of LLM to tackle long-horizon QA tasks.

2.2 Context Management and Optimization

To overcome static context limitations, recent works explore dynamic optimization via uncertainty-based filtering (Jimenez Gutierrez et al., 2024; Ji et al., 2025) or explicit memory agents (Yu et al., 2025; Yan et al., 2025). More aggressively, DeepNote (Wang et al., 2024) and MemSearcher (Yuan et al., 2025) treat context as an evolving state. However, DeepNote does not support joint end-to-end optimization, while MemSearcher's monolithic design leads to coarse credit assignment due to sparse, trajectory-level rewards. M-ASK addresses these pitfalls by *decoupling* search and knowledge management into two specialized agents. Through multi-agent reinforcement learning with turn-specific dense rewards, our approach achieves precise credit assignment.

2.3 Multi-Agent Systems for Information Retrieval

To overcome the limitations of monolithic agents, multi-agent systems (MAS) commonly decompose complex tasks into sub-tasks and assign them to agents with specialized roles. General frameworks, such as MetaGPT (Hong et al., 2023) and Auto-

Gen (Wu et al., 2024), exemplify this paradigm by enabling structured role specialization and coordination among multiple agents. Within the information retrieval domain, MindSearch (Chen et al., 2024) utilizes a graph-based planner alongside parallel web searchers for query decomposition, while MMOA-RAG (Chen et al., 2025a) allocates specialized agents for query rewriting, document selection, and answer generation. Similarly, MAO-ARAG (Chen et al., 2025b) focuses on optimizing a planner agent to balance retrieval effectiveness with efficiency. Despite their potential, most search-centric MAS rely heavily on prompt engineering or standard supervised fine-tuning (SFT), often neglecting the optimization of *interaction dynamics*. Although MMOA-RAG incorporates MAPPO (Yu et al., 2022), aims to optimize a shared global reward (e.g., the F1 score of the final answer). We argue that such sparse global feedback is suboptimal for multi-turn collaboration, as it exacerbates the *credit assignment problem*, failing to accurately evaluate intermediate steps. M-ASK addresses this limitation by introducing distinct, dense reward functions for Search and Knowledge agents, ensuring that each role is optimized for its specific contribution to the collective goal.

3 Method

We propose **M-ASK**, a multi-agent framework decoupling search planning from information integration. In this section, we first formulate the problem as a *Sequential Decentralized Partially Observable Markov Decision Process* (Bernstein et al., 2002; Oliehoek et al., 2016). Second, we detail the specifications of the specialized agents. Finally, we present the framework's execution flow and the joint optimization process utilizing a turn-level reward mechanism.

3.1 Problem Formulation

We formulate the multi-hop QA task as a *Sequential Decentralized Partially Observable Markov Decision Process*. Unlike standard formulations where agents act simultaneously, our framework, M-ASK, operates in a turn-based manner. Specifically, at each discrete time step t, only one designated agent, denoted as π_{active}, is activated. This agent receives a partial observation of the environment and executes an action to advance the search process. To enable coordination across time steps, agents communicate indirectly by reading from and writing to a shared, structured knowledge state.

Structured Knowledge State Agents communicate via a shared, structured *Knowledge State* $\mathcal{K}$. We define $\mathcal{K}_t$ at time step t as:

$$\mathcal{K}_t = \begin{Bmatrix} \text{“question"} : q, \\ \text{“thinking\_trajectory"} : \mathcal{T}_t, \\ \text{“predicted\_answer"} : a_t \end{Bmatrix} \quad (1)$$

Let $\mathcal{T}_t = [\tau_1, \tau_2, \ldots, \tau_m]$ represent the evolving reasoning chain, where each element $\tau_i = \langle q_{\text{sub}}^{(i)}, a_{\text{sub}}^{(i)} \rangle$ is a tuple consisting of a **sub-query** and its associated **sub-answer** (evidence). This trajectory $\mathcal{T}_t$ explicitly records the step-by-step multi-hop inference process, serving as the logical derivation path required to deduce the final answer a_t from the original question q. Specifically, the **Planning Agent** initializes both the reasoning chain $\mathcal{T}_0$ and the answer a_0. As the multi-round search progresses, $\mathcal{T}_t$ is dynamically modified, and consequently, the answer a_t is iteratively updated based on the evolving trajectory. The specific implementation details are discussed in the following section. For a concrete instantiation of $\mathcal{K}_t$ in a multi-hop scenario, please refer to Table 5 in Appendix E.

The framework shown in Figure 2 orchestrates these agents into a cohesive workflow involving initialization, iterative refinement, and optimization (see Algorithm 1 in Appendix F for the complete pseudo-code). To better understand how this workflow operates in practice, we first present the functional roles of the agents, followed by a detailed explanation of the inference and training processes.

3.2 M-ASK Architecture

M-ASK employs a team of specialized agents, categorized by their functional roles.

3.2.1 Search Behavior Agents (SBA)

Planning Agent (A_{plan}) This agent functions as the system *initializer*. Given a user query q as input, A_{plan} leverages parametric memory to generate an initial reasoning trajectory $\mathcal{T}_0$ and a preliminary answer a_0, encapsulating them into the initial state $\mathcal{K}_0 \leftarrow \pi_{\text{plan}}(q)$.

Search Agent (A_{search}) Acting as the *navigator*, this agent iteratively evaluates the sufficiency of the thinking trajectory $\mathcal{T}_t$ with respect to the query q. Its primary role is to decide between exploration and termination. Specifically, taking the question and trajectory as input, the policy outputs an action

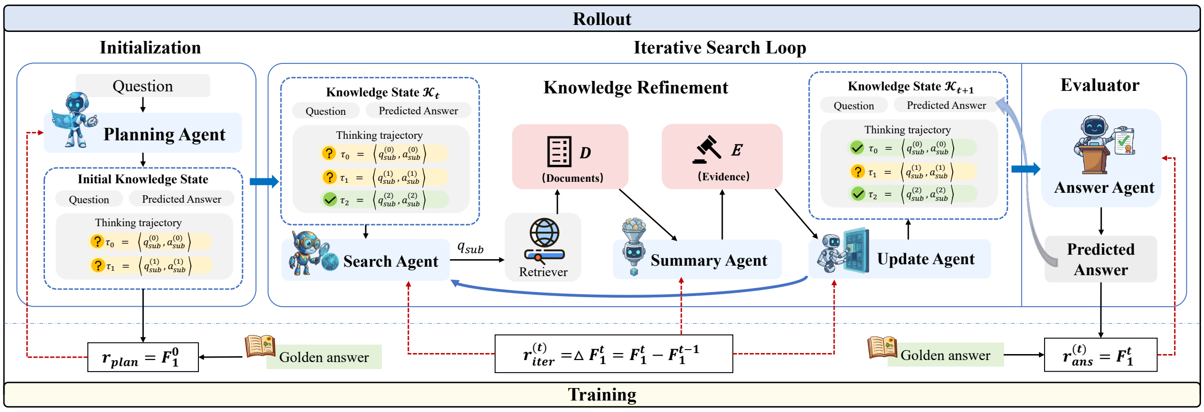


Figure 2: Overview of the M-ASK framework. (1) **Rollout**: The Planning Agent initializes the state $\mathcal{K}_0$, followed by an iterative loop where Search and Knowledge Management Agents refine the trajectory. Crucially, the Answer Agent updates the prediction after each turn. (2) **Training**: A hybrid reward mechanism assigns absolute scores (F_1^0 and F_1^t) to the Planning and Answer Agents, respectively, while the collaborative agents (Search, Summary, Update) share the marginal improvement (ΔF_1^t) to incentivize step-wise refinement.

$Act \leftarrow \pi_{\text{search}}(q, \mathcal{T}_t)$, where $Act \in \{q'_{\text{sub}}, \texttt{<end>}\}$. If expanding knowledge is necessary, it generates a specific sub-query q'_{sub}; otherwise, it outputs <end> to terminate the search loop.

Answer Agent (A_{ans}) Operating as the *solver*, this agent is responsible for generating the final prediction. Conditioned on the original question q and the accumulated reasoning trajectory $\mathcal{T}_t$ retrieved from the knowledge state $\mathcal{K}$, it synthesizes a coherent final answer a, formally defined as $a \leftarrow \pi_{\text{ans}}(q, \mathcal{T}_t)$.

3.2.2 Knowledge Management Agents (KMA)

Summary Agent (A_{sum}) This agent functions as a *filter* to distill key information. Given a sub-query q'_{sub} and a set of retrieved documents D, it extracts the pertinent evidence E while actively discarding irrelevant noise. The process is formalized as $E \leftarrow \pi_{\text{sum}}(q'_{\text{sub}}, D)$.

Update Agent (A_{upd}) Transcending the role of a passive logger, this agent acts as a *dynamic state refiner*. Its primary objective is to maintain a high-density knowledge state $\mathcal{K}$ by judiciously deciding between refining existing information or appending new findings. Given the current state $\mathcal{K}_t$, a sub-query q'_{sub}, and the retrieved evidence E, the agent outputs a discrete operation op to evolve the trajectory. The action space is designed to balance information growth and precision: (1) <Update>τ_i</Update> (**In-Place Refinement**): Targeting an existing step τ_i, this action overwrites previous hallucinations or vague information with precise evidence. (2) <Add>τ_{new}</Add> (**Expansion**): This action appends a new reasoning step only when the evidence introduces a distinct, necessary logical hop. Formally, the state transition is defined as $op, \mathcal{K}_{t+1} \leftarrow \pi_{\text{upd}}(\mathcal{K}_t, q'_{\text{sub}}, E)$.

We provide a structured summary of the functional roles and specifications for each agent in Table 4 (Appendix D), offering a clear view of their distinct mechanisms.

3.2.3 Inference Workflow

The inference process operates as follows:

1. **Initialization:** A_{plan} generates a "cold start" state $\mathcal{K}_0$.
2. **Iteration:** The system enters a loop controlled by A_{search}.
   - At step t, A_{search} evaluates $\mathcal{K}_t$.
   - If A_{search} generates a query, the workflow proceeds to A_{sum} (filtering) and A_{upd} (state update), producing $\mathcal{K}_{t+1}$.
3. **Termination:** The loop terminates when A_{search} outputs <end> or a maximum step limit is reached. Finally, A_{ans} is triggered to synthesize the final answer from the latest knowledge state $\mathcal{K}$.

For a microscopic view of this workflow, including how agents dynamically correct hallucinations and refine the knowledge state, please refer to the detailed case study in Table 6 (Appendix G).

3.3 M-ASK Training via Turn-Level Rewards

We employ Independent PPO (Schulman et al., 2017) for optimization. To align the SBA and KMA groups despite their different functional roles, we design a hybrid reward mechanism that differentiates between *state utilization* and *state refinement*.

State-Based Reward (Absolute F1) For agents responsible for generating solution outputs (A_{plan} and A_{ans}), the objective is to maximize the absolute quality of the current state.

$$r_{\text{plan}} = \text{F}_1(a_0, y), \quad r_{\text{ans}}^{(t)} = \text{F}_1(a_t, y) \tag{2}$$

Here, a_t is the answer synthesized from $\mathcal{K}_t$ and y is the ground truth answer. Note that during training, A_{ans} acts as an evaluator at every step t.

Transition-Based Reward (Shared Incremental Gain) Crucially, the iterative phase requires tight collaboration between the Search Behavior Agent (A_{search}) and the Knowledge Management Agents ($A_{\text{sum}}, A_{\text{upd}}$). Although they belong to different functional groups, their actions are co-dependent: effective search requires precise state updates, and useful updates depend on accurate retrieval.

To enforce this **local cooperation**, we assign a **shared incremental reward** to all agents active in the loop. It is important to distinguish the execution frequency of the Answer Agent between phases. During **inference**, the Answer Agent is triggered only *once* after the search terminates to generate the final output. However, during **training**, it assumes an additional role as an *intermediate evaluator*. It is invoked at every turn t to synthesize a temporary answer a_t based on the current state $\mathcal{K}_t$. This mechanism allows us to utilize the answer score $r_{\text{ans}}^{(t)} = \text{F}_1(a_t, y)$ (defined in Eq. 2) to measure the immediate answer quality. The iteration reward is then defined as the marginal improvement over the previous step:

$$r_{\text{iter}}^{(t)} = r_{\text{ans}}^{(t)} - r_{\text{ans}}^{(t-1)} = \text{F}_1(a_t, y) - \text{F}_1(a_{t-1}, y) \tag{3}$$

By sharing the identical $r_{\text{iter}}^{(t)}$ signal, the Search Agent (A_{search}), Summary Agent (A_{sum}), and Update Agent (A_{upd}) are jointly incentivized to maximize the marginal information gain of each turn. This mechanism binds them into a cooperative subteam, where the Search Agent learns to fetch necessary information and the Knowledge Agents learn to distill it efficiently. If A_{search} outputs `<end>`, it receives a reward of 0, ensuring the team only terminates the collaboration when further search yields no positive gain.

Optimization Objective (Parameter Sharing) To enhance sample efficiency and enable knowledge transfer across different reasoning phases, we employ a **Parameter-Shared** [2] strategy. All functional agents are instantiated from a unified LLM π_θ, distinguished solely by role-specific system instructions I_{role}.

Consequently, the optimization objective aggregates the experiences from all roles. Let u_t denote the action taken given observation o_t. The shared policy π_θ is updated to maximize:

$$\mathcal{L}(\theta) = \hat{\mathbb{E}}_t \Big[\min \Big(\rho_t \hat{A}_t, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon) \hat{A}_t \Big) \Big] \tag{4}$$

where $\rho_t = \frac{\pi_\theta(u_t|o_t)}{\pi_{\theta_{old}}(u_t|o_t)}$ is the probability ratio. Simultaneously, the shared Critic V_ϕ minimizes the unified value loss:

$$\mathcal{L}(\phi) = \hat{\mathbb{E}}_t \left[\|V_\phi(o_t) - R_t\|^2 \right] \tag{5}$$

Here, $R_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k}$ represents the cumulative discounted return, where γ is the discount factor and r denotes the role-specific rewards defined earlier.

4 Experiments

To validate the efficacy of M-ASK, we conduct comprehensive experiments to answer the following research questions: **RQ1**: Does M-ASK outperform existing monolithic and multi-agent frameworks? **RQ2**: How does M-ASK compare to the monolithic baseline (Search-r1) in terms of training stability? **RQ3**: Is it beneficial to jointly model and optimize Knowledge Management and Search Behavior? **RQ4**: Do turn-specific dense rewards provide better credit assignment than global outcome-based rewards?

4.1 Experimental Setup

Datasets We evaluate our framework on a diverse set of open-domain QA benchmarks categorized by reasoning complexity. For **Single-hop QA**, we use Natural Questions (NQ) (Kwiatkowski et al., 2019), PopQA (Mallen et al., 2022), and AmbigQA (Min et al., 2020) to test factual retrieval accuracy. For **Multi-hop QA**, to evaluate complex reasoning and trajectory planning, we employ HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), Musique (Trivedi et al., 2022), and Bamboogle (Press et al., 2022).

[2] We provide a detailed justification for adopting this parameter-shared architecture in Appendix A.

Method	Single-hop QA				Multi-hop QA					Avg
	NQ	PopQA	AmbigQA	Avg	HotpotQA	2Wiki	Musique	Bam.	Avg	All
Standard Baselines										
LLM w/o RAG	17.53	14.93	23.53	18.66	17.76	22.58	8.58	17.14	16.52	17.44
Vanilla RAG	40.60	42.74	56.20	46.51	28.33	25.91	25.20	25.28	26.18	34.89
RL-Based (Static Modular Workflow)										
RRR (Ma et al., 2023)	54.60	**50.46**	65.41	56.82	46.21	41.52	18.27	36.59	35.65	44.72
BGM (Ke et al., 2024)	54.21	49.51	65.97	56.56	46.85	37.79	17.55	37.38	34.89	44.18
MMOA-RAG (Chen et al., 2025a)	55.44	50.21	68.02	57.89	49.21	41.66	17.26	37.20	36.33	45.57
Agentic Search (Adaptive Workflow)										
Adaptive RAG (Jeong et al., 2024)	36.52	35.59	45.32	39.14	42.38	39.62	25.48	34.85	35.58	37.11
MAO-ARAG (Chen et al., 2025b)	36.82	41.85	47.03	41.90	46.65	43.96	22.38	**49.84**	40.85	41.30
DeepNote (Wang et al., 2024)	54.03	49.80	67.57	57.13	52.49	36.22	22.17	45.22	39.03	46.79
Search-r1 (Jin et al., 2025)	52.57	46.98	65.25	54.93	46.87	39.03	17.97	38.69	35.64	43.91
M-ASK (Ours)	**57.40**	50.10	**68.33**	**58.61**	**58.31**	**46.12**	**26.20**	44.18	**43.70**	**50.09**
Impv. vs Best	+1.96	-0.36	+0.31	+0.72	+5.82	+2.16	+0.72	-5.66	+2.85	+3.30

Table 1: Main performance comparison (F1 Score) on single-hop and multi-hop QA benchmarks. The best results are **bolded** and the second best are underlined. "Impv. vs Best" denotes the performance gain (blue) or drop (gray) of M-ASK compared to the best performing baseline in each column.

Baselines We evaluate M-ASK against three categories of methods: **Standard Baselines**, **RL-Based Static Workflows**, and **Adaptive Agentic Search**. Detailed implementations and settings for these baselines are provided in Appendix B.

Implementation Details Our implementation is developed based on the official `verl` repository[3]. For all experiments, we use **Qwen2.5-7B-Instruct** (Team, 2024) as the backbone LLM. We utilize the English Wikipedia as our retrieval corpus, indexed via **E5** (Wang et al., 2022) for dense retrieval. We report the standard **F1 Score** as the evaluation metric.

4.2 Main Results Analysis (RQ1)

To answer **RQ1**, we evaluate M-ASK against competitive baselines across seven datasets. As shown in Table 1, M-ASK achieves the highest average F1 score (**50.09**), outperforming both monolithic and multi-agent frameworks.

Dominance on Complex QA and Generalization. M-ASK demonstrates superior capability on complex reasoning tasks, particularly on **HotpotQA**, where it surpasses the best baseline by a substantial margin of **+5.82**. This validates that our collaborative framework is far more effective at handling complex multi-hop queries than standard methods. Furthermore, M-ASK generalizes robustly to unseen out-of-domain datasets (e.g., **2Wiki**, **Musique**), consistently exceeding other Agentic baselines. This suggests M-ASK learns abstract, transferable strategies for query decomposition and filtering rather than merely overfitting to training patterns.

[3] `https://github.com/volcengine/verl`

Analysis of Bamboogle Performance. A notable exception is observed on **Bamboogle**, where M-ASK trails MAO-ARAG. We attribute this to two primary factors. First, the test set of Bamboogle is significantly smaller (only 125 queries) than other benchmarks, introducing potential statistical variance. Second, MAO-ARAG utilizes frozen pre-trained answer generators, which may answer factoid questions based on their existing parametric knowledge, whereas M-ASK is optimized to ground answers strictly in retrieved evidence. Nevertheless, M-ASK remains highly competitive, trailing the second-best method (DeepNote) by only $\sim$1.0 point while outperforming all other baselines, reaffirming the robustness of our policy despite these factors.

Comparison with Monolithic and Modular Frameworks. A clear performance hierarchy is observed, further answering RQ1 regarding framework efficacy. The **monolithic** agent, Search-r1, lags significantly (43.91) due to *unconstrained context growth* and accumulated search noise. This structural limitation is addressed by **modular** frameworks like DeepNote (46.79), which achieves the second-best results by decoupling knowledge management. However, M-ASK further outperforms DeepNote (+3.30). Unlike DeepNote's disjoint modules, M-ASK employs *end-to-end joint optimization*, ensuring that the Search and Knowledge Management agents are collaboratively updated to maximize the final reasoning reward.

4.3 Training Stability and Convergence (RQ2)

To investigate the training dynamics and stability of our framework compared to monolithic approaches, we visualize the training curves on the HotpotQA dataset in Figure 3. Furthermore, to rigorously quantify robustness, we conducted 10 independent training runs for both methods and recorded the rate of "model collapse"—defined as the performance score dropping to near zero and failing to recover. These statistics are summarized in Table 2.

Method	Collapse Rate (over 10 runs) @ 200 Steps	@ 500 Steps	@ 1000 Steps
Search-r1	1/10 (10%)	7/10 (70%)	9/10 (90%)
M-ASK (Ours)	**0/10 (0%)**	**0/10 (0%)**	**0/10 (0%)**

Table 2: Training stability analysis comparing the rate of model collapse (performance degrading to ≈ 0) at different training stages across 10 independent runs.

Catastrophic Collapse in Monolithic Agents. As shown in Table 2 and Figure 3(a), the monolithic Search-r1 agent exhibits extreme volatility. Consistent with the findings in (Deng et al., 2025), we observe that Search-r1 is prone to collapse during the mid-to-late stages of training. Specifically, while only 10% of runs failed at 200 steps, the collapse rate escalated to **90%** as training progressed to 1000 steps. We attribute this to the interaction between long horizons and sparse rewards. As the agent explores, it inevitably encounters retrieval noise. In a monolithic architecture, this noise accumulates in the context without intermediate correction. When the optimization algorithm attempts to assign credit based solely on the final outcome, the noisy gradients frequently push the policy into degenerate states from which it cannot recover.

Analysis of Response Length and Mechanism. The key to M-ASK's superior stability (0% collapse rate) is elucidated in Figure 3(b). While the monolithic Search-r1 suffers from "context bloating"—rapidly converging to over 1000 tokens—M-ASK maintains a remarkably low and stable average output length. This conciseness is not merely a cosmetic difference but a direct indicator of three structural advantages that stabilize training: (1) **Simplified Functionality:** Decoupling ensures atomic tasks with short generation horizons, significantly reducing optimization complexity compared to monolithic models; (2) **Precise Credit Assignment:** Short outputs allow turn-specific dense rewards to provide immediate, high-confidence feedback, unlike global rewards that dilute over long trajectories; and (3) **Effective Noise Filtering:** Knowledge Management Agents actively prune noise to prevent error accumulation in the knowledge state.

Collectively, these factors eliminate the gradient variance caused by noisy, long contexts, and sparse reward. This structural robustness enables M-ASK to achieve the stable, monotonic convergence illustrated in Figure 3(c), even in the absence of supervised warm-starting.

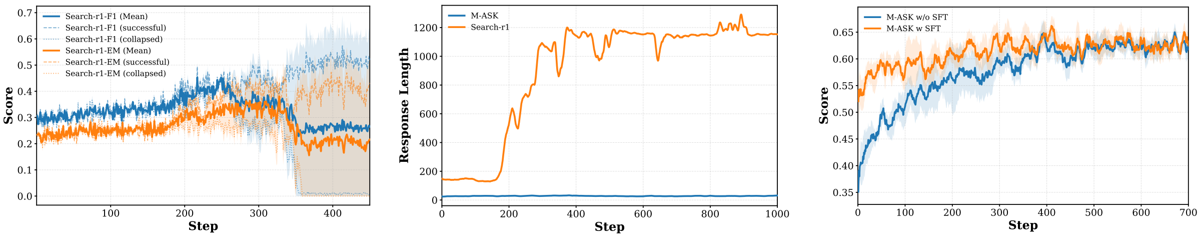
4.4 Ablation Studies (RQ3 & RQ4)

To validate the contribution of individual components in M-ASK, we conduct ablation studies and observing the impact across different task complexities. The results are summarized in Table 3.

4.4.1 Impact of Collaborative Knowledge Management (RQ3)

To answer whether explicitly modeling knowledge management is beneficial, we evaluate the variant **w/o KMA**. In this setting, we remove the *Summary* and *Update* agents; the *Search Behavior Agent* interacts directly with the raw search engine output and appends full documents to the context. This setup simulates the unconstrained information flow of a standard monolithic interaction. However, unlike the fully monolithic *Search-r1*, this variant retains the underlying multi-agent paradigm and independent optimization methods, thereby strictly isolating the impact of the missing knowledge management module.

Analysis. The results show a consistent performance degradation across both single-hop (avg. $\Delta - 2.58\%$) and multi-hop (avg. $\Delta - 2.86\%$) benchmarks. This confirms that without the active filtering provided by the KMA group, the search agent is overwhelmed by noise in the retrieval results, leading to hallucinations or distracted reasoning. We note a slight anomaly on **2WikiMultiHopQA** ($+0.46\%$), where the unfiltered model performs marginally better. However, this is outweighed by the severe drops in more complex datasets like **Musique** (-6.81%) and **HotpotQA** (-3.25%). Overall, explicit knowledge management is crucial for stabilizing the reasoning trajectory against retrieval noise.



(a) Search-r1 Training Dynamics (b) Average Response Length (c) M-ASK Training Dynamics

Figure 3: Training curves on HotpotQA. In **(a)** and **(c)**, solid lines represent the mean across multiple runs, while shaded regions indicate the variance. **(a)** Search-r1 exhibits high instability and frequent mode collapse. **(b)** Evolution of average response length; M-ASK remains concise while Search-r1 suffers from context bloating. **(c)** M-ASK demonstrates stable convergence.

Method	**Single-hop Datasets**								**Multi-hop Datasets**									
	NQ		PopQA		AmbigQA		**Avg.**		HotpotQA		2Wiki		Musique		Bam.		**Avg.**	
	F1	Δ	F1	Δ	F1	Δ	**F1**	Δ	F1	Δ	F1	Δ	F1	Δ	F1	Δ	**F1**	Δ
M-ASK (Ours)	57.40	–	50.10	–	68.33	–	**58.61**	–	58.31	–	46.12	–	26.20	–	44.18	–	**43.70**	–
Ablation Studies																		
w/o KMA	54.93	-2.47	48.36	-1.74	64.79	-3.54	**56.03**	**-2.58**	55.06	-3.25	46.58	+0.46	19.39	-6.81	42.33	-1.85	**40.84**	**-2.86**
w/o T-L Reward	54.95	-2.45	47.12	-2.98	62.04	-6.29	**54.70**	**-3.91**	42.57	-15.74	33.27	-12.85	11.95	-14.25	26.10	-18.08	**28.47**	**-15.23**

Table 3: Ablation study. **Avg.** columns show the aggregated performance on Single-hop (3 datasets) and Multi-hop (4 datasets) benchmarks. Δ represents the performance drop (gray) or gain (blue) compared to the full model.

4.4.2 Efficacy of Turn-Specific Dense Rewards (RQ4)

To assess the necessity of our reward shaping mechanism, we evaluate **w/o T-L Reward**. In this variant, we disable the turn-level incremental rewards (Eq.2 and Eq.3) and instead optimize all agents using only the final outcome reward (Global F1) at the end of the episode, similar to the strategy used in MMOA-RAG (Chen et al., 2025a).

Analysis. This ablation reveals a critical insight into the optimization dynamics of multi-turn search. **Single-hop Resilience:** On single-hop datasets, the performance drop is moderate (avg. $\Delta - 3.91\%$) because the short trajectories typical of these tasks make the final reward sufficient for credit assignment. **Multi-hop Collapse:** In stark contrast, performance collapses on multi-hop datasets (avg. $\Delta - 15.23\%$), with the score on **HotpotQA** plummeting by over 15 points.

This disparity highlights the **turn-level credit assignment problem**. In frameworks like MMOA-RAG, all agents across different turns share an identical global outcome reward. This ambiguity obscures the specific contribution of individual steps, making it difficult to discern which action actually drove the success or failure. Consequently, agents struggle to optimize intermediate sub-goals in long-horizon tasks. In contrast, M-ASK's turn-specific ΔF1 reward provides immediate, dense feedback, enabling agents to accurately lock in optimal policies for each reasoning hop.

5 Conclusion

In this paper, we addressed the structural instability of monolithic agentic search caused by unconstrained contexts, sparse rewards, and search noise. We proposed **M-ASK**, a multi-agent framework that decouples search planning from knowledge management, enabling synergistic collaboration via turn-specific dense rewards.

Empirical evaluations across seven benchmarks demonstrate that M-ASK consistently outperforms state-of-the-art baselines, particularly in complex multi-hop scenarios. Crucially, our analysis reveals that decoupling these roles significantly enhances training stability compared to end-to-end RL approaches. These findings suggest that explicit role specialization and intermediate supervision are critical for scaling agentic search to more open-ended and noisy real-world environments. Future work will explore extending M-ASK to heterogeneous model architectures and broader tool-use scenarios beyond information retrieval.

Limitations

Despite the effectiveness of M-ASK, several limitations remain. First, the collaborative multi-agent workflow inherently increases the frequency of LLM invocations compared to single-pass approaches, as each reasoning step requires discrete inference calls for search, summarization, and update modules. This sequential interaction pattern inevitably raises computational costs and may pose challenges for latency-sensitive applications. Second, our evaluation is currently confined to textual QA tasks; the framework's generalizability to other complex domains, such as code generation or multimodal reasoning, remains to be explored. Finally, the efficacy of our parameter-sharing strategy relies on the inherent capacity of the backbone LLM to handle diverse role instructions, and performance on smaller-scale architectures warrants further investigation.

Ethics Statement

This work builds upon large language models to facilitate multi-agent collaboration for textual question answering. All datasets used in our experiments are publicly available and do not contain any personally identifiable information. Nevertheless, outputs generated by LLMs may reflect biases present in their pretraining data or produce incorrect information. Users and practitioners should exercise caution when applying the proposed framework to real-world scenarios, especially in domains where misinformation or biased content could cause harm. We do not foresee additional ethical risks beyond those commonly associated with contemporary language model research.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840.

Yiqun Chen, Hangyu Mao, Jiaxin Mao, Shiguang Wu, Tianle Zhang, Bin Zhang, Wei Yang, and Hongxing Chang. 2022. Ptde: Personalized training with distilled execution for multi-agent reinforcement learning. *arXiv preprint arXiv:2210.08872*.

Yiqun Chen, Lingyong Yan, Weiwei Sun, Xinyu Ma, Yi Zhang, Shuaiqiang Wang, Dawei Yin, Yiming Yang, and Jiaxin Mao. 2025a. Improving retrieval-augmented generation through multi-agent reinforcement learning. *arXiv preprint arXiv:2501.15228*.

Yiqun Chen, Erhan Zhang, Lingyong Yan, Shuaiqiang Wang, Jizhou Huang, Dawei Yin, and Jiaxin Mao. 2025b. Mao-arag: Multi-agent orchestration for adaptive retrieval-augmented generation. *arXiv preprint arXiv:2508.01005*.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Jiangning Liu, Wenwei Zhang, Kai Chen, and Feng Zhao. 2024. Mindsearch: Mimicking human minds elicits deep ai searcher. *arXiv preprint arXiv:2407.20183*.

Wenlong Deng, Yushu Li, Boying Gong, Yi Ren, Christos Thrampoulidis, and Xiaoxiao Li. 2025. On grpo collapse in search-r1: The lazy likelihood-displacement death spiral. *arXiv preprint arXiv:2512.04220*.

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

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*.

Yuelyu Ji, Rui Meng, Zhuochun Li, and Daqing He. 2025. Memory-aware and uncertainty-guided retrieval for multi-hop question answering. *arXiv preprint arXiv:2503.23095*.

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

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

Zixuan Ke, Weize Kong, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Bridging the preference gap between retrievers and llms. *arXiv preprint arXiv:2401.06954*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

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

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*.

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

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. *arXiv preprint arXiv:2004.10645*.

Frans A Oliehoek, Christopher Amato, and 1 others. 2016. *A concise introduction to decentralized POMDPs*, volume 1. Springer.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51.

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

Zhengliang Shi, Yiqun Chen, Haitao Li, Weiwei Sun, Shiyu Ni, Yougang Lyu, Run-Ze Fan, Bowen Jin, Yixuan Weng, Minjun Zhu, Qiujie Xie, Xinyu Guo, Qu Yang, Jiayi Wu, Jujia Zhao, Xiaqiang Tang, Xinbei Ma, Cunxiang Wang, Jiaxin Mao, and 7 others. 2025. Deep research: A systematic survey. *Preprint*, arXiv:2512.02038.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025a. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

Huatong Song, Jinhao Jiang, Wenqing Tian, Zhipeng Chen, Yuhuan Wu, Jiahao Zhao, Yingqian Min, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025b. R1-searcher++: Incentivizing the dynamic knowledge acquisition of llms via reinforcement learning. *arXiv preprint arXiv:2505.17005*.

Qwen Team. 2024. Qwen2 technical report. *arXiv preprint arXiv:2412.15115*.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 10014–10037.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Ruobing Wang, Qingfei Zhao, Yukun Yan, Daren Zha, Yuxuan Chen, Shi Yu, Zhenghao Liu, Yixuan Wang, Shuo Wang, Xu Han, and 1 others. 2024. Deepnote: Note-centric deep retrieval-augmented generation. *arXiv preprint arXiv:2410.08821*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z Pan, Hinrich Schütze, and 1 others. 2025. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and 1 others. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*.

Qianhao Yuan, Jie Lou, Zichao Li, Jiawei Chen, Yaojie Lu, Hongyu Lin, Le Sun, Debing Zhang, and Xianpei Han. 2025. Memsearcher: Training llms to reason, search and manage memory via end-to-end reinforcement learning. *arXiv preprint arXiv:2511.02805*.

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

A Rationale for Parameter Sharing Strategy

In the M-ASK framework, we employ a parameter-sharing strategy where a single Large Language Model (LLM) backbone π_θ serves as the underlying policy for all functional agents (i.e., $A_{\text{plan}}, A_{\text{search}}, A_{\text{sum}}, A_{\text{upd}}, A_{\text{ans}}$), distinguished solely by role-specific system instructions. We adopt this design based on three critical considerations:

1. Theoretical Foundation in MARL Parameter sharing is a well-established and effective paradigm in Multi-Agent Reinforcement Learning (MARL) (Rashid et al., 2020; Lowe et al., 2017; Yu et al., 2022; Chen et al., 2022). In cooperative settings, sharing parameters allows agents to learn a unified representation of the state space, which often leads to faster convergence and improved training stability compared to maintaining independent policies. By optimizing a single set of parameters on the aggregated experiences of all roles, the model can efficiently generalize across the diverse phases of the search and reasoning process.

2. Computational and Storage Efficiency Unlike traditional RL agents based on small Multi-Layer Perceptrons (MLPs), agents in our framework are initialized with LLMs containing billions of parameters. Maintaining independent policy networks for five distinct agents would result in a linear increase in memory consumption ($\mathcal{O}(N)$), rendering the training process computationally prohibitive and difficult to deploy. Parameter sharing reduces the storage requirement to $\mathcal{O}(1)$, significantly lowering the barrier for training and inference. This efficiency is paramount for complex RAG systems like M-ASK, allowing us to allocate resources toward longer context windows rather than redundant model weights.

3. Inherent Multi-Task Capability of LLMs LLMs inherently possess strong multi-task capabilities, enabling them to perform distinct tasks based on contextual instructions (prompts) without modifying their internal weights. In M-ASK, different agents (e.g., the *Search Agent* deciding on queries vs. the *Summary Agent* extracting evidence) share fundamental reasoning competencies, such as reading comprehension and logical deduction. Parameter sharing leverages this synergy: skills learned in the *summarization* task can implicitly enhance the *answer generation* capability via shared representations. By conditioning the shared π_θ on role-specific prompts I_{role}, we effectively project the model's general capabilities into specific functional subspaces, achieving role specialization without architectural redundancy.

B Implementation Details of Baselines

We compare M-ASK against three categories of methods:

- **Standard Baselines:** *LLM w/o RAG* (closed-book), *Vanilla RAG* (standard retrieve-then-generate).
- **RL-Based (Static Modular Workflow):** *RRR* (Ma et al., 2023) (RL-based query reformulation), and *BGM* (Ke et al., 2024) (RL-based document selection), and *MMOA-RAG* (Chen et al., 2025a) (multi-agent RL).
- **Agentic Search (Adaptive Workflow):** *Adaptive RAG* (Jeong et al., 2024) (adaptive workflow), *MAO-ARAG* (Chen et al., 2025b) (planner-executors optimization), *DeepNote* (Wang et al., 2024) (knowledge-management) and *Search-r1* (Jin et al., 2025) (monolithic agent optimized via RL reasoning training).

To ensure a fair comparison, all baselines and our M-ASK method are unified under the same experimental setting. We utilize `Qwen2.5-7B-Instruct` as the backbone model. Specifically, for components within the baselines that do not require training, we employ the pre-trained version of `Qwen2.5-7B-Instruct`. For components requiring training, we fine-tune them based on the `Qwen2.5-7B-Instruct` initialization. Detailed implementation notes for each baseline are provided below:

Standard Baselines

- **LLM w/o RAG:** This represents the closed-book setting where the LLM directly generates answers based on its internal parametric knowledge without accessing external corpora.
- **Vanilla RAG:** A standard retrieve-then-generate pipeline. It retrieves the top-5 relevant documents based on the query and concatenates them with the prompt to generate the answer using the pre-trained LLM.

Static Modular Workflow (RL-Based)

- **RRR** (Ma et al., 2023): This method employs the PPO algorithm to end-to-end train a query rewriter module. In our reproduction, to prevent the performance bottleneck potentially caused by a frozen generator, we also fine-tuned the answer generation module, ensuring it is well-trained alongside the rewriter.

- **BGM** (Ke et al., 2024): BGM uses PPO to train a document selection module. Although the original implementation may utilize a frozen generator, we fine-tuned the answer generation module in our reproduction to align with the robust setting of RRR and maximize overall performance.

- **MMOA-RAG** (Chen et al., 2025a): This framework utilizes Multi-Agent PPO (MAPPO) (Yu et al., 2022) to train three distinct agents—a query rewriter, a document selector, and an answer generator—guided by a shared final reward mechanism. We adopted `Qwen2.5-7B-Instruct` as the backbone model for this multi-agent framework, strictly following the original logic while maintaining consistency with our unified setting.

Agentic Search (Adaptive Workflow)

- **Adaptive RAG** (Jeong et al., 2024): This method trains a classifier to dynamically determine which of three distinct retrieval workflows (Directly Answer, Simple RAG, or Iterative RAG) should handle a given query. In our implementation, we fine-tuned `Qwen2.5-7B-Instruct` to serve as this classifier, while utilizing the pre-trained version of `Qwen2.5-7B-Instruct` for executing the subsequent static workflows.

- **MAO-ARAG** (Chen et al., 2025b): MAO-ARAG features a hierarchical multi-agent architecture composed of a planner and multiple executors, sharing a modular structure similar to our M-ASK. The original work utilizes PPO with a shared final reward to train specifically the planner agent. Consistent with our unified setting, the planner module is a fine-tuned `Qwen2.5-7B-Instruct`, while the executor modules employ the pre-trained version of the same model.

- **DeepNote** (Wang et al., 2024): DeepNote focuses on optimizing knowledge management within QA tasks through Direct Preference Optimization (DPO) (Rafailov et al., 2023). In our reproduction, we upgraded the data generation model from `gpt-4o-mini` (Hurst et al., 2024) (used in the original paper) to `gpt-4o` (Hurst et al., 2024) to construct higher-quality DPO training data.

- **Search-r1** (Jin et al., 2025): Unlike modular designs, Search-r1 adopts a monolithic architecture where multi-turn search and reasoning processes occur within a single response generation. It is optimized via RL using outcome-based rewards. For this baseline, we obtained the experimental results using the code provided in the author's official open-source repository[4].

C Prompt for Different Agents

C.1 Prompt for Planning Agent

The following is the full system prompt used for decomposing queries. The `[EXAMPLE_PROMPT]` placeholder represents the few-shot examples injected at runtime.

System Prompt

```
You are a knowledgeable reasoning assistant.
Your task is to decompose a given question
into multiple sub-questions based purely
on your own knowledge (do not use external
tools), and answer each sub-question based
on your own knowledge. If you do not know
the answer, write 'unkown' exactly.

Output all results strictly in the
following format for easy parsing:
<q1>sub-question text</q1>
<a1>answer text</a1>
<q2>sub-question text</q2>
<a2>answer text</a2>

Continue numbering sequentially until done.
At the end, output your final conclusion
answer to the original question in the tag
<predicted_answer>...</predicted_answer>,
and inside this tag follow the style of:
[EXAMPLE_PROMPT]
```

User Input

```
Query: {query}
```

[4] https://github.com/PeterGriffinJin/Search-R1

C.2 Prompt for Search Agent

System Prompt

```
You are a search query generation assistant.
Your input contains:
- The original question
- A thinking_trajectory (list of sub-questions and their sub-answers)
- A search_history (list of past search queries and their summaries)

Note:
- The search_history represents queries YOU have already searched in previous turns.
- You MUST avoid generating queries that are identical to or semantically similar to any query in search_history.
- Only generate a new query if it provides new, distinct information that has not yet been searched.

Your task:
1. If any sub-question in thinking_trajectory has an answer 'unkown' or unclear, generate a specific, searchable query to find the missing information.
2. If you believe additional sub-questions are needed to answer the original question, generate a specific searchable query for that.
3. If no further search is needed, output <end>.

Output only one of the following two formats:
<search>Your query here</search>
<end>

Do not include any extra text, explanations, or other tags.
```

User Input

```
Original question: {question}

Thinking trajectory:
{trajectory_text}

Search history:
{history_text}

Output only one of the following two formats:
<search>Your query here</search>
<end>

Do not include any extra text, explanations, or other tags.
```

C.3 Prompt for Summary Agent

System Prompt

```
You are an evidence extraction assistant.
You will be given:
- A search query
- Several candidate documents related to that query

Your task:
1. From the provided documents, extract the key evidence that directly answers the query.
2. If no key evidence exists in the documents, output 'No useful information'.

Output exactly in one of the two formats:
<evidence>your key evidence here</evidence>
<evidence>No useful information</evidence>

Do not include explanations, reasoning steps, or any text outside the <evidence> tags.
```

User Input

```
Query: {query}
Candidate Documents: {docs_text}
```

C.4 Prompt for Update Agent

System Prompt

```
You are an update detection assistant.
You are given:
- An original question and its thinking_trajectory (sub-questions and their answers)
- The predicted_answer derived from thinking_trajectory
- A new question and its evidence

Your task:
1. If the new question and evidence can replace one existing sub-question-answer pair in thinking_trajectory, output <Update>ti</Update> where ti is the key.
2. If the new question and evidence contains information not present in thinking_trajectory, output <Add>t(n+1)</Add> where n is current number of t's.
3. Output strictly one of these formats and no other text.
```

User Input

```
Original question: {orig_question}
Thinking trajectory: {trajectory_text}
New query: {new_query}
Evidence: {new_evidence}
```

C.5 Prompt for Answer Agent

System Prompt

```
You are a final answer generation
assistant.
You are given:
- An original question
- A thinking_trajectory (list of
sub-questions and their answers)

Your task:
1. Use the provided thinking_trajectory's
answers to determine the final answer to
the original question.
2. You must only output in the following
strict format as shown in [EXAMPLE_PROMPT]

IMPORTANT: Output must be:
<predicted_answer>your answer
here</predicted_answer>
No extra text, no reasoning.
```

User Input

```
Question: {question}
Thinking trajectory: {trajectory_text}
```

D Detailed Agent Specifications

In this section, we provide comprehensive specifications for the multi-agent architecture employed in M-ASK. Due to space constraints in the main text, we present the granular details of the Search Behavior Agents (SBA) and Knowledge Management Agents (KMA) here. Table 4 formally defines the specific roles, input-output interfaces, and action spaces for each agent, elucidating their operational logic within the iterative retrieval process.

E Example of Structured Knowledge State

To facilitate a better understanding of the data structure defined in Eq. 1, we provide a concrete example of the Structured Knowledge State $\mathcal{K}_t$.

Consider a multi-hop question q: *"Who is the current CEO of the company that developed ChatGPT?"*. Table 5 illustrates the state $\mathcal{K}_t$, where the agent has successfully decomposed the query and retrieved the necessary evidence chain.

F Joint Training Algorithm Details

In this section, we provide the comprehensive pseudo-code for the M-ASK joint training process, complementing the methodology described in Section 3.3. Algorithm 1 details the execution flow of the **Parameter-Shared** strategy, where a unified policy π_θ is employed across all functional roles (Planning, Search, Summary, Update, and Answer).

The procedure consists of two distinct phases:

- **Data Collection Phase:** Agents interact sequentially to solve the task. Notably, the Answer Agent serves as an intermediate evaluator at each step t, calculating the marginal performance gain ΔF1 to provide dense, turn-specific supervision.
- **Unified Optimization Phase:** Experiences from all roles are stored in a mixed replay buffer $\mathcal{B}$. The shared model parameters θ and critic ϕ are then jointly updated via PPO, maximizing the objective function derived from these heterogeneous interaction trajectories.

G Detailed Case Study: Structured Knowledge State Evolution

The trajectory presented in Table 6 offers a microscopic view of how M-ASK addresses the structural limitations of monolithic agents. The execution process exhibits high rationality and ingenuity in three critical aspects:

1. Correction of Parametric Hallucinations via "Targeting". A pervasive challenge in agentic search is the "cold start" problem, where models must plan without initial observations. In the **Init** phase, the Planning Agent relies on its parametric memory and incorrectly predicts "Guadalajara." Crucially, M-ASK does not treat this initial plan as ground truth but as a *search target*. In **Turn 1**, the Update Agent identifies a conflict between the parametric "Guadalajara" and the retrieved non-parametric evidence ("Puebla"). Instead of attempting to reconcile the two or hedging, the agent prioritizes external evidence, executing an `<Update>` action to overwrite the hallucination. This demonstrates a robust **error-correction mechanism** that prevents initial errors from propagating through the reasoning chain.

2. Deep Verification of Temporal Constraints. The transition to **Turn 2** highlights the Search Agent's ability to handle complex constraints. The query contains a specific temporal condition ("1943"). While a naive retriever might stop after finding the entity "Puebla," the Search Agent

Module	Agent	Role	Description & Action Space	Input / Output Formulation
SBA	**Planning**	*Initializer*	Leverages parametric memory to generate an initial reasoning trajectory $\mathcal{T}_0$ and a preliminary answer a_0, encapsulating them into the initial state.	**In:** Query q **Out:** $\mathcal{K}_0 \leftarrow \pi_{\text{plan}}(q)$
	Search	*Navigator*	Evaluates information sufficiency to decide between exploration and termination. 1) Generate sub-query q'_{sub} to expand knowledge. 2) Output <end> to terminate the loop.	**In:** Query q, Trajectory $\mathcal{T}_t$ **Out:** $Act \leftarrow \pi_{\text{search}}(\mathcal{K}_t)$, where $Act \in \{q'_{\text{sub}},$ <end>$\}$
	Answer	*Solver*	Synthesizes the final answer prediction conditioned on the original question and the accumulated reasoning trajectory.	**In:** Query q, Trajectory $\mathcal{T}_t$ **Out:** $a \leftarrow \pi_{\text{ans}}(q, \mathcal{T}_t)$
KMA	**Summary**	*Filter*	Distills pertinent evidence from retrieved documents while actively discarding irrelevant noise to ensure information density.	**In:** Sub-query q'_{sub}, Documents D **Out:** $E \leftarrow \pi_{\text{sum}}(q'_{\text{sub}}, D)$
	Update	*Dynamic Refiner*	Judiciously decides how to integrate evidence to evolve the trajectory: • <Update>τ_i</Update> (**In-Place Refinement**): Overwrites hallucinations/vague steps. • <Add>τ_{new}</Add> (**Expansion**): Appends necessary logical hops.	**In:** State $\mathcal{K}_t$, sub-query q'_{sub}, Evidence E **Out:** $op, \mathcal{K}_{t+1} \leftarrow \pi_{\text{upd}}(\mathcal{K}_t, q'_{\text{sub}}, E)$

Table 4: Comprehensive specifications of the agents within the M-ASK framework, detailing their roles, functional mechanisms, and input/output formalizations.

Key	Value (Content)
`"question"` (q)	Who is the current CEO of the company that developed ChatGPT?
`"thinking_trajectory"` ($\mathcal{T}_t$)	(τ_1): $\langle q_{sub}^{(1)} \rangle$: Which company developed ChatGPT? $\langle a_{sub}^{(1)} \rangle$: OpenAI developed ChatGPT, an AI chatbot launched in November 2022. (τ_2): $\langle q_{sub}^{(2)} \rangle$: Who is the current CEO of OpenAI? $\langle a_{sub}^{(2)} \rangle$: Sam Altman is the CEO of OpenAI.
`"predicted_answer"` (a_t)	Sam Altman

Table 5: A example of the Knowledge State $\mathcal{K}_t$. The trajectory $\mathcal{T}_t$ contains a sequence of sub-query and sub-answer pairs that logically support the final answer.

actively verifies the historical context ("significant event... in 1943"). The subsequent `<Add>t2</Add>` action is highly rational: the agent recognizes that the "professionalization of the league" is distinct background information that supports the answer's validity. By appending this as a new step (t_2), the system constructs a logical evidence chain: *Entity Existence* (t_1) + *Historical Context* (t_2).

3. Active Prevention of Context Bloating. The most significant display of ingenuity occurs in **Turn 3**. The agent retrieves specific confirmation linking "Puebla" to the "Athletic Club." In standard monolithic frameworks (e.g., Search-r1), this new retrieval would typically be appended to the context, causing the input length to grow linearly with search steps. In contrast, M-ASK employs **In-Place Refinement**. The Update Agent recognizes that this information is a more precise version of t_1. It triggers `<Update>t1` to refine the query and answer (adding "specifically Puebla") rather than creating a redundant t_3. This mechanism maintains a concise state space ($\mathcal{K}_3$), effectively solving the "unconstrained output length" challenge and ensuring the final reasoning is performed on a high-density, noise-free context.

Algorithm 1: M-ASK Joint Training (Parameter Shared)

Input: Data $\mathcal{D}$, Unified Model π_θ, Critic V_ϕ, Role Instructions $I = \{I_{\text{plan}}, I_{\text{search}}, \dots\}$, Batch B

Output: Optimized Parameters θ^*, ϕ^*

Initialize θ, ϕ; Initialize unified replay buffer $\mathcal{B}$;

for *iteration* $i = 1, \dots, M$ **do**

    `// 1. Data Collection Phase`

    while $|\mathcal{B}| < B$ **do**

        Sample $(q, y) \sim \mathcal{D}$;

        `// Phase I: Initialization`

        $\mathcal{K}_0 \leftarrow \pi_\theta(q; I_{\text{plan}})$; $a_0 \leftarrow \mathcal{K}_0$.predicted_answer;

        $S_{prev} \leftarrow \text{F1}(a_0, y)$;

        Add transition $(I_{\text{plan}}, q, \mathcal{K}_0, r = S_{prev})$ to $\mathcal{B}$;

        `// Phase II: Iterative Collaboration`

        $t \leftarrow 0$;

        while $t < T_{max}$ **do**

            `// Step 1: SBA Decision`

            $Action \leftarrow \pi_\theta(\mathcal{K}_t; I_{\text{search}})$; `// Prompt as Searcher`

            if $Action ==$ <end> **then**

                Add transition $(I_{\text{search}}, \mathcal{K}_t,$ <end>$, r = 0)$ to $\mathcal{B}$;

                break;

            end

            $q'_{sub} \leftarrow Action$;

            `// Step 2: KMA Execution`

            $D \leftarrow \text{SearchEngine}(q'_{sub})$;

            $E \leftarrow \pi_\theta(q'_{sub}, D; I_{\text{sum}})$; `// as Summarizer`

            $op, \mathcal{K}_{next} \leftarrow \pi_\theta(\mathcal{K}_t, q'_{sub}, E; I_{\text{upd}})$; `// as Updater`

            `// Step 3: Evaluation`

            $a_{next} \leftarrow \pi_\theta(\mathcal{K}_{next}; I_{\text{ans}})$; `// as Answerer`

            $S_{curr} \leftarrow \text{F1}(a_{next}, y)$;

            $\Delta\text{F1} \leftarrow S_{curr} - S_{prev}$;

            `// Step 4: Store Mixed Experiences`

            Add $(I_{\text{search}}, \mathcal{K}_t, q'_{sub}, r = \Delta\text{F1})$ to $\mathcal{B}$;

            Add $(I_{\text{sum}}, q'_{sub}, D, E, r = \Delta\text{F1})$ to $\mathcal{B}$;

            Add $(I_{\text{upd}}, \mathcal{K}_t, E, op, r = \Delta\text{F1})$ to $\mathcal{B}$;

            Add $(I_{\text{ans}}, \mathcal{K}_{next}, a_{next}, r = S_{curr})$ to $\mathcal{B}$;

            $\mathcal{K}_{t+1} \leftarrow \mathcal{K}_{next}$; $S_{prev} \leftarrow S_{curr}$; $t \leftarrow t + 1$;

        end

    end

    `// 2. Unified Optimization Phase`

    Compute GAE on unified buffer $\mathcal{B}$ using shared Critic V_ϕ;

    for *epoch* $k = 1, \dots, K$ **do**

        Sample mixed mini-batches from $\mathcal{B}$;

        Update θ, ϕ via PPO maximizing joint objective $\mathcal{L}(\theta)$;

    end

    Clear buffer $\mathcal{B}$;

end

Target Question: What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943?
Golden Answer: **Puebla**

Turn	Role	Full Action Output & Structured Knowledge State ($\mathcal{K}_t$)
Init	**Planning Agent**	<q1> What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? </q1> <a1> Guadalajara </a1> <predicted_answer> Guadalajara </predicted_answer>
	State $\mathcal{K}_0$ (Initial Structure):	"question": What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? "thinking_trajectory": [$[\tau_1]$ $\langle q^{(1)}_{sub} \rangle$: What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? $\langle a^{(1)}_{sub} \rangle$: **Guadalajara** *(Parametric Hallucination)*] "predicted_answer": **Guadalajara**
Turn 1	**Search Agent**	<search> What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? </search>
	Search Engine	*[External Tool] Retrieved 5 document fragments (∼100 words each) containing keyword matches for "Mexican city", "Athletic Club", "Primera Fuerza", "1943".*
	Summary Agent	<evidence> Puebla, which has a city of Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza. </evidence>
	Update Agent	<Update>τ_1</Update> (**Corrects Hallucination**)
	Answer Agent	<predicted_answer> Puebla </predicted_answer>
	State $\mathcal{K}_1$ (Corrected Answer):	"question": What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? "thinking_trajectory": [$[\tau_1]$ $\langle q^{(1)}_{sub} \rangle$: What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? $\langle a^{(1)}_{sub} \rangle$: **Puebla, which has a city of Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza.**] "predicted_answer": **Puebla** *(Updated via retrieval)*
Turn 2	**Search Agent**	<search> What is a significant event or fact about Puebla A.C.'s football team's participation in the Primera Fuerza in 1943? </search>
	Search Engine	*[External Tool] Retrieved 5 document fragments (∼100 words each) related to "Puebla A.C.", "Primera Fuerza participation", "1943 events".*
	Summary Agent	<evidence> Puebla A.C.'s football team played in the Primera Fuerza prior to the professionalization and development of the Primera División de México in 1943. </evidence>
	Update Agent	<Add>τ_2</Add> (Appends Context)
	Answer Agent	<predicted_answer> Puebla </predicted_answer>
	State $\mathcal{K}_2$ (Expanded Context):	"question": What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? "thinking_trajectory": [$[\tau_1]$ $\langle q^{(1)}_{sub} \rangle$: What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? $\langle a^{(1)}_{sub} \rangle$: Puebla, which has a city of Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza. , $[\tau_2]$ $\langle q^{(2)}_{sub} \rangle$: **What is a significant event or fact about Puebla A.C.'s football team's participation in the Primera Fuerza in 1943?** $\langle a^{(2)}_{sub} \rangle$: **Puebla A.C.'s football team played in the Primera Fuerza prior to the professionalization and development of the Primera División de México in 1943.**] "predicted_answer": **Puebla**
Turn 3	**Search Agent**	<search> What Mexican city, specifically Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza in 1943? </search>
	Search Engine	*[External Tool] Retrieved 5 document fragments verifying "Puebla", "Puebla A.C.", "Primera Fuerza 1943".*
	Summary Agent	<evidence> Puebla, specifically Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza in 1943. </evidence>
	Update Agent	<Update>τ_1</Update> (**Refines Existing Step**)
	Answer Agent	<predicted_answer> Puebla </predicted_answer>
	State $\mathcal{K}_3$ (Refined Trajectory):	"question": What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? "thinking_trajectory": [$[\tau_1]$ $\langle q^{(1)}_{sub} \rangle$: What Mexican city, **specifically Puebla**, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza in 1943? $\langle a^{(1)}_{sub} \rangle$: **Puebla, specifically Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza in 1943.** , $[\tau_2]$ $\langle q^{(2)}_{sub} \rangle$: What is a significant event or fact about Puebla A.C.'s football team's participation in the Primera Fuerza in 1943? $\langle a^{(2)}_{sub} \rangle$: Puebla A.C.'s football team played in the Primera Fuerza prior to the professionalization and development of the Primera División de México in 1943.] "predicted_answer": **Puebla**
Turn 4	**Search Agent**	<end> (Termination Triggered)

Final Prediction: **Puebla** **(✓Matches Golden Answer)**

Table 6: Full execution log for the query: *"What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943?"*. The **State** rows explicitly show the three components of $\mathcal{K}_t$. The trajectory steps are indexed as $[\tau_i]$, allowing the Update Agent to target specific steps (e.g., <Update>τ_1</Update>).