

SearchGym: Bootstrapping Real-World Search Agents via Cost-Effective and High-Fidelity Environment Simulation

Xichen Zhang^{1,*}, Ziyi He^{2,*}, Yinghao Zhu², Sitong Wu³,
Shaozuo Yu³, Meng Chu¹, Wenhui Zhang¹, Haoru Tan², Jiaya Jia^{1,†}

¹The Hong Kong University of Science and Technology

²The University of Hong Kong

³The Chinese University of Hong Kong

Abstract

Search agents have emerged as a pivotal paradigm for solving open-ended, knowledge-intensive reasoning tasks. However, training these agents via Reinforcement Learning (RL) faces a critical dilemma: interacting with live commercial Web APIs is prohibitively expensive, while relying on static data snapshots often introduces noise due to data misalignment. This misalignment generates *corrupted reward signals* that destabilize training by penalizing correct reasoning or rewarding hallucination. To address this, we propose **SearchGym**, a simulation environment designed to bootstrap robust search agents. SearchGym employs a rigorous generative pipeline to construct a verifiable knowledge graph and an aligned document corpus, ensuring that every reasoning task is factually grounded and strictly solvable. Building on this controllable environment, we introduce **SearchGym-RL**, a curriculum learning methodology that progressively optimizes agent policies through purified feedback, evolving from basic interactions to complex, long-horizon planning. Extensive experiments across the Llama and Qwen families demonstrate strong Sim-to-Real generalization. Notably, our Qwen2.5-7B-Base model trained within SearchGym surpasses the web-enhanced ASearcher baseline across nine diverse benchmarks by an average relative margin of 10.6%. Our results validate that high-fidelity simulation serves as a scalable and highly cost-effective methodology for developing capable search agents.

1 Introduction

Autonomous search agents, empowered by Large Language Models (LLMs) to interact with external search engines, represent a pivotal advancement in solving complex, knowledge-intensive tasks (Yao et al., 2022; Jin et al., 2025a; Sun et al., 2025; Wang et al., 2024). Unlike passive retrieval systems,

these agents can autonomously formulate queries, browse results, and synthesize information over multiple turns (Gao et al., 2025). To master such sophisticated decision-making processes, Reinforcement Learning (RL) has emerged as the standard training paradigm, enabling models to optimize their reasoning trajectories through outcome-based feedback (Guo et al., 2025; Jaech et al., 2024).

However, the advancement of search agents is fundamentally constrained by the training environment. Researchers currently face a critical dilemma between scalability and realism. On one hand, training directly with live commercial Web APIs offers high environmental fidelity but incurs prohibitive costs. As demonstrated in Table 3, conducting large-scale RL experiments is financially unsustainable; a single training epoch over 30,000 questions, with 8 rollouts per question and an average of 3 search actions per rollout, necessitates approximately 720,000 API calls. At standard commercial rates, this accumulates fees exceeding \$500 per run, rendering extensive exploration and hyperparameter tuning impractical. On the other hand, relying on static offline assets such as Wikipedia snapshots introduces significant noise due to inherent data misalignment. This noise manifests as outdated information, ambiguous queries, and factual inconsistencies between the ground truth and the retrieval corpus. Our analysis identifies these issues as the source of corrupted reward signals. Consequently, the optimization process is destabilized because agents are frequently penalized for correct reasoning or rewarded for hallucinations that coincidentally match the labels, leading to the training instability and policy collapse (Figure 3).

To bridge this gap, we introduce **SearchGym**, a cost-effective and high-fidelity simulation environment designed to bootstrap robust search agents without the costs of live web access. To resolve the training instability caused by noisy training data, SearchGym constructs a verified knowledge

*Equal contribution

†Corresponding author: Jiaya Jia (jia@cse.ust.hk)

graph and an aligned document corpus in a generative closed-loop environment. This guarantees that every reasoning task is factually grounded and strictly solvable. Within this controlled environment, we propose **SearchGym-RL**, a training methodology incorporating curriculum learning to optimize agent policies. By leveraging clean, noise-free signals, SearchGym-RL progressively guides agents from mastering basic interactions to handling complex reasoning structures. Furthermore, by training on synthetic data unseen during pre-training, we compel the agent to rely exclusively on tool execution rather than parametric memory, fostering robust search capabilities that generalize to the real world. Our main contributions are:

- (1) *Insight*: We identify *corrupted reward signals* stemming from data misalignment in existing offline training, demonstrating that high-fidelity reward signals are a necessary condition for stable RL-based search agent training.
- (2) *Methodology*: We introduce SearchGym, a closed-loop environment constructed via a rigorous pipeline spanning from knowledge graph synthesis to solvable QA generation. Leveraging this environment, we employ SearchGym-RL, a curriculum-based strategy that guarantees task solvability and provides purified feedback, enabling agents to master complex reasoning structures (e.g., compositional and parallel search) efficiently.
- (3) *Experiment*: We demonstrate robust Sim-to-Real generalization across diverse model families, including Llama 3.2, Qwen 2.5, and Qwen 3. Despite being trained in a synthetic environment with zero commercial web API costs, our Qwen2.5-7B agent surpasses the web-enhanced ASearcher baseline across 9 diverse benchmarks by an average relative margin of 10.6%. Our comprehensive experiments encompass both standard QA tasks and challenging, open-ended research benchmarks such as GAIA and xbench-DeepSearch, validating the efficacy of SearchGym and SearchGym-RL on complex real-world problems.

2 Related Work

Search agents and tool learning. The paradigm of augmenting Large Language Models (LLMs) with external knowledge has shifted from passive Retrieval-Augmented Generation (RAG), where the model receives a fixed set of retrieved con-

text, to iterative, agent-based workflows (Gao et al., 2023; Lewis et al., 2020). Early approaches rely on prompt engineering frameworks, such as ReAct (Yao et al., 2022), or Supervised Fine-Tuning (SFT) (Schick et al., 2023) to teach models how to utilize search tools. While SFT effectively bootstraps basic tool-use capabilities, it remains limited by the quality of human-annotated trajectories. Consequently, Reinforcement Learning (RL) (Guo et al., 2025; Jin et al., 2025a; Sun et al., 2025; Gao et al., 2025) has emerged as a standard for optimizing complex interaction policies, enabling agents to autonomously discover search strategies through outcome-based feedback. However, while frameworks such as ASearcher (Gao et al., 2025) validate the efficacy of training against live web engines, their reliance on commercial APIs incurs prohibitive financial costs, rendering large-scale policy optimization unsustainable.

Simulated search environments. To mitigate the prohibitive costs of online training, recent research has pivoted toward offline or simulated environments. Approaches such as Search-R1 (Jin et al., 2025a) establish a local retrieval environment based on the 2018 Wikipedia corpus to facilitate training on datasets NQ (Kwiatkowski et al., 2019) and HotpotQA (Yang et al., 2018). However, this paradigm faces two limitations: the static corpus largely overlaps with the pretraining data of modern LLMs, and the data quality within these benchmarks varies significantly. Alternatively, ZeroSearch (Sun et al., 2025) employs an LLM to simulate search engine responses. While innovative, this approach is often insufficient for training models on complex, long-horizon reasoning tasks. In contrast, SearchGym generates a grounded, verifiable, and closed-loop synthetic world. By constructing tasks from a validated knowledge graph, SearchGym ensures high-fidelity interactions and guarantees solvability for multi-hop reasoning.

3 Methodology

SearchGym constitutes a high-fidelity and cost-effective offline simulation designed for training search agents. As illustrated in Figure 1, our four-stage pipeline constructs a verifiable knowledge graph and aligned corpus, ensuring every task is logically consistent and strictly solvable. Leveraging this controlled environment, we propose SearchGym-RL, a curriculum-based reinforcement learning framework that progressively optimizes agent policies through purified feedback.

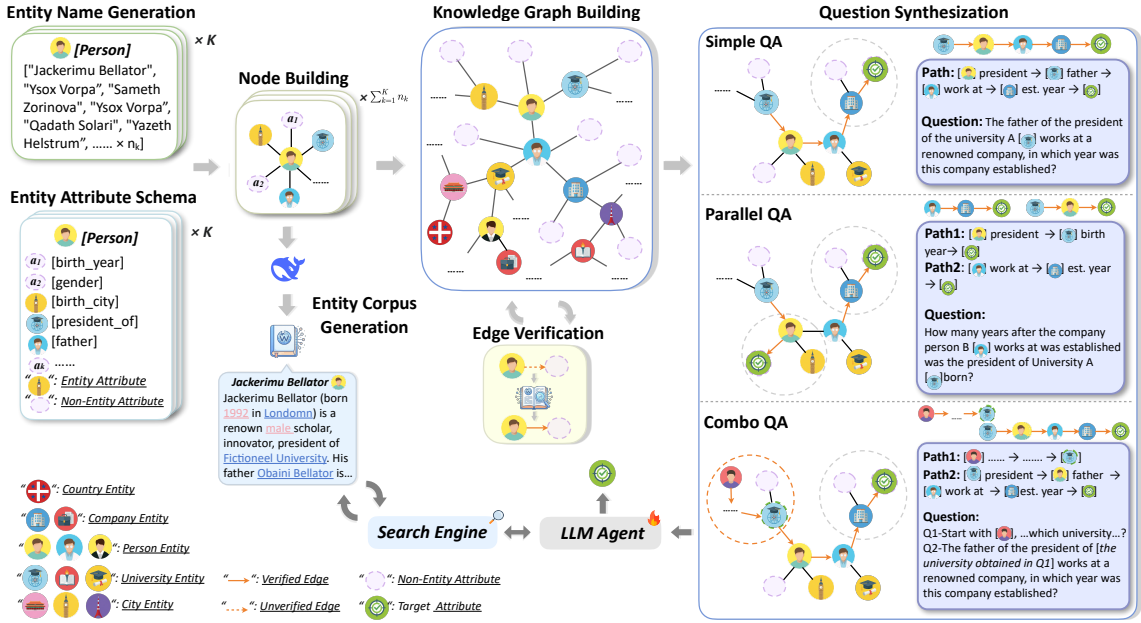


Figure 1: An overview of the SearchGym pipeline. From a verified knowledge graph (left), we sample reasoning paths of varying structures. These paths are verbalized into complex, solvable Question-Answering pairs (right), categorized as Simple, Parallel, or Combo QA. This creates a closed-loop environment where the LLM agent interacts with a search engine to retrieve evidence and solve the tasks. The legend defines the visual elements.

3.1 SearchGym Simulation Environment

To address the lack of rigor in existing offline methods, we formalize the data generation process. Let the synthetic retrieval environment be defined as a tuple $\mathcal{W} = \langle \mathcal{G}, \mathcal{D} \rangle$, where \mathcal{G} is a structured knowledge graph and \mathcal{D} is the associated searchable corpus. All LLM prompts employed in our data generation process and generation details are provided in Appendix D.

Knowledge graph synthesis. First, we construct a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ based on a predefined schema \mathcal{S} . The schema defines valid entity types (e.g., Person, Organization) and relational constraints. We programmatically populate the vertex set \mathcal{V} with diverse, fictional yet plausible entities and establish directed edges \mathcal{E} to represent semantic relationships. This construction ensures that \mathcal{G} exhibits a realistic topological structure while remaining fully controllable.

Entity corpus generation. We construct a searchable document corpus $\mathcal{D} = \{d_v \mid v \in \mathcal{V}\}$. For each entity v , we employ an LLM M_{gen} to synthesize a Wikipedia-style document $d_v = M_{\text{gen}}(v, \mathcal{N}_v)$, which incorporates the entity’s attributes and its local graph neighborhood \mathcal{N}_v . To simulate a web environment, each document d_v is associated with a unique URL, allowing agents to perform distinct Search and Access actions.

Edge verifiability and filtering. A critical innovation of SearchGym is the enforcement of learnability through retrievability. For every edge $e = (u, v) \in \mathcal{E}$, we generate search queries Q_e (specifically, we set $|Q_e| = 15$). We execute these queries against \mathcal{D} using a retrieval engine \mathcal{R} . An edge is retained in the verified subgraph $\mathcal{G}^* \subseteq \mathcal{G}$ if and only if the target document d_v is retrieved by at least 5 of these queries:

$$(u, v) \in \mathcal{E}^* \iff |\{q \in Q_e \mid d_v \in \text{Top-}K(\mathcal{R}(q, \mathcal{D}))\}| \geq 5. \quad (1)$$

This filtering ensures the reasoning path is discoverable via search, decoupling the agent’s reasoning capability from stochastic retrieval failures and ensuring clean training signals.

Question-answer synthesis from verified paths. Finally, we synthesize complex Question-Answering (QA) pairs by sampling reasoning paths from \mathcal{G}^* . A path $\mathcal{P} = (v_0, e_1, v_1, \dots, v_k)$ represents a logic chain of length k . We categorize paths into three structural types:

- (1) **Simple QA:** Linear paths where $v_i \rightarrow v_{i+1}$.
- (2) **Parallel QA:** Multiple independent paths $\mathcal{P}_1, \mathcal{P}_2$ merging at a terminal condition, requiring information synthesis.
- (3) **Combo QA:** Nested paths where the answer to \mathcal{P}_1 parametrizes the query for \mathcal{P}_2 .

The generator M_{gen} verbalizes these paths into natural language questions Q and answers A , guaranteeing every task (Q, A) is solvable within \mathcal{D} .

Data statistics. The synthesized environment consists of approximately 3,600 nodes, each corresponding to a generated document. From this corpus, our pipeline yields over 41,000 unique QA pairs, stratified by complexity as shown in Table 1. For evaluation, we sample a representative subset of 642 questions (462 Simple, 134 Parallel, 46 Combo). We designate this high-quality evaluation set as SearchGymBench. These instances span 1 to 12 hops, explicitly testing the capacity for sustained, long-horizon reasoning.

Hops	Simple QA	Parallel QA	Combo QA
1-3 hops	20,384	2,913	-
4-6 hops	11,264	2,019	-
>6 hops	-	1,870	2,622
Total	31,648	6,802	2,622

Table 1: Distribution of generated questions by type and reasoning hops.

3.2 SearchGym-RL Training

Our methodology focuses on training search agents within the SearchGym environment to master a realistic action space consisting of Search and Access. This process is driven by two core components: a reinforcement learning framework for policy optimization, and a curriculum learning strategy designed to progressively enhance agent capabilities.

Action Space. To mimic web browsing and decouple retrieval from reading, we define two primitives: **Search**(q) returns a list of snippets with URLs, while **Access**(u) retrieves the full document content d_u . This design not only simulates web browsing but also enforces rigorous reasoning, compelling the agent to evaluate relevance based on summaries before committing to detailed reading.

Reinforcement learning within SearchGym. Our agent addresses problems via a multi-turn reasoning process, generating a trajectory \mathcal{T} of intermediate thoughts and actions that culminates in a final answer. Unlike single-step generation, this iterative process allows for complex problem decomposition. The quality of each trajectory \mathcal{T} is measured by a terminal reward $R(\mathcal{T})$, determined by the correctness of the final answer. Specifically, we define the reward using the token-level F1 score, which assesses the token overlap between the prediction and the ground truth.

To optimize the policy π_θ , we employ Group Relative Policy Optimization (GRPO) (Shao et al., 2024), building on recent advancements (Jin et al., 2025a; Sun et al., 2025; Gao et al., 2025). GRPO operates on a group of N trajectories $\{\mathcal{T}_i\}_{i=1}^N$ sampled from the policy. For each trajectory, it computes a normalized advantage \hat{A}_i by standardizing the terminal F1 rewards within the group. The policy parameters θ are then updated by maximizing the following clipped surrogate objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{T}_i \sim \pi_{\theta_{\text{old}}}} \left[\min \left(\rho_i(\theta) \hat{A}_i, \right. \right. \\ \left. \left. \text{clip}(\rho_i(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_i \right) \right. \\ \left. - \beta D_{KL}(\pi_\theta(\mathcal{T}_i) \parallel \pi_{\text{ref}}(\mathcal{T}_i)) \right] \quad (2)$$

where $\rho_i(\theta) = \pi_\theta(\mathcal{T}_i) / \pi_{\theta_{\text{old}}}(\mathcal{T}_i)$ denotes the importance sampling ratio, ϵ is the clipping hyperparameter, and the D_{KL} term imposes a penalty against a reference policy π_{ref} to stabilize training.

Hierarchical skill acquisition via curriculum

Training agents on complex, multi-hop tasks from scratch is challenging due to sparse rewards, which can destabilize the learning process. To address this, we employ a two-stage curriculum that leverages the structured difficulty of tasks within SearchGym to progressively build agent capabilities.

- *Stage 1: Foundational Skill Acquisition.* We first train the agent on Simple QA tasks with shorter reasoning chains (1-6 hops). This stage focuses on mastering core skills, such as query formulation, document parsing, and sequential evidence, ensuring a stable start to training.
- *Stage 2: Advanced Reasoning Development.* Once the agent reaches a baseline proficiency, we introduce more complex Parallel and Combo QA tasks and increase the proportion of long-horizon problems (6-12 hops). This challenges the agent to develop advanced abilities like problem decomposition and information synthesis, preparing it for real-world complexity.

4 Experimental Setups

Training datasets. We optimize the policy using the SearchGym Synthetic Corpus, comprising approximately 41,000 verifiable tasks. We strictly isolate the held-out SearchGymBench from training to ensure that performance reflects generalized reasoning capabilities rather than memorization.

Search tools. Following ASearcher (Gao et al., 2025), we employ three distinct retrieval settings.

First, the SearchGym Environment utilizes a local index of our synthetic corpus \mathcal{D} for training process. Second, the Local Wikipedia setting employs the 2018 snapshot (Kwiatkowski et al., 2019) for standard QA evaluation. Third, the Live Web setting integrates a commercial Search API to address open-ended, challenging benchmarks. Detailed configurations are provided in Appendix E.2.

Baselines. To evaluate the effectiveness of our approach, we compare it against the following baselines: (1) Inference without Retrieval: Base instruct models performing direct generation without external tools. (2) Inference with Retrieval: Standard retrieval-augmented generation (RAG) (Lewis et al., 2020). (3) RL-Based Methods: Search-R1 (Jin et al., 2025a), ZeroSearch (Sun et al., 2025) and ASearcher (Gao et al., 2025). We conduct these comparisons across a diverse suite of backbones, including the Qwen 2.5 series (1.5B/7B, Base/Instruct), Qwen 3 (4B/8B), and Llama 3.2 3B Instruct, to demonstrate that our method’s efficacy is robust to scale and transferable across architectures.

Evaluation benchmarks. Our evaluation spans 10 diverse benchmarks to comprehensively assess agent capabilities. We categorize these into: (1) Single-hop QA: Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023), testing factual recall and simple lookup. (2) Multi-hop QA: 2WikiMultiHopQA(2Wiki) (Ho et al., 2020), HotpotQA (Yang et al., 2018), Bamboogle (Press et al., 2023), and Musique (Trivedi et al., 2022), evaluating complex reasoning chains. (3) Deep Research: GAIA (Mialon et al., 2023) and xbench-DeepSearch(xbench) (Chen et al., 2025), which serve as proxies for open-ended, real-world research tasks. (4) Pure Tool-Use: SearchGymBench, our held-out test suite comprising 642 complex tasks that are unseen during pretraining, thereby strictly evaluating tool-use capabilities rather than parametric memory. Following established protocols (Gao et al., 2025), for Bamboogle, GAIA, and xbench-DeepSearch, we use their full test sets. For GAIA specifically, we utilize the 103 examples from the text-only validation subset (Li et al., 2025). For all other benchmarks, we use 1,000 randomly sampled instances.

Evaluation metrics. Following ASearcher’s evaluation protocol (Gao et al., 2025), we employ their robust LLM-as-Judge to assess semantic correctness given the open-ended tasks. Specifically,

Qwen-2.5-72B-Instruct serves as the universal evaluator to verify consistency with the ground truth. We report Pass@1 for standard QA and Pass@4 for complex, long-horizon tasks such as GAIA and xbench-DeepSearch. Detailed evaluation prompts and protocols are provided in Appendix E.4.

Implementation details. We train all models for 5 epochs using the AReal framework (Fu et al., 2025), reporting results from the best-performing checkpoint. For both training and evaluation, the retrieval module returns the top-5 relevant documents for every search action. We set the maximum interaction turns to 16 for training and 64 for evaluation. For training, we employ a maximum token limit of 1,024 with temperature 1.0 during training, whereas evaluation uses 4,096 tokens, temperature 0.6, and Top- p 0.95. Consistent with recent studies (Gao et al., 2025), we keep all other hyperparameters unchanged. A comprehensive list of hyperparameters is detailed in Appendix E.5.

5 Experimental Results

We present a comprehensive evaluation of SearchGym-RL across diverse benchmarks and model architectures. The results demonstrate that agents trained in our high-fidelity offline simulation not only master standard retrieval tasks but also generalize effectively to complex, open-ended research problems without requiring expensive online interactions during training.

Performance on standard QA benchmarks. Table 2 details the performance on single-hop and multi-hop QA datasets. Our method consistently outperforms all baseline approaches. Notably, we achieve a relative improvement of 10.4% over Search-R1, which relies on static Wikipedia snapshots under the Qwen-2.5-7B-Base model. This improvement confirms that the clean and verifiable logic in SearchGym provides effective training signals compared to noisy static datasets.

Robustness across model architectures and scales. SearchGym-RL maintains a significant lead over baselines regardless of the underlying model family (Qwen, Llama) or size (3B to 8B). As shown in Table 2, on the Llama-3.2-3B backbone, SearchGym-RL achieves an average score of 54.96, representing a relative improvement of approximately 80% over both Search-R1 and ZeroSearch. This demonstrates that the high-fidelity data from SearchGym is a universally effective driver of performance.

Method	Single-Hop QA			Multi-Hop QA				Avg.
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	Musique	Bamboogle	
<i>Qwen-2.5-3B-Base/Instruct</i>								
Direct Inference	10.60	28.80	10.80	14.90	24.40	2.00	2.40	13.41
RAG	34.80	44.40	28.70	25.50	22.60	4.70	8.00	24.10
Search-R1-base	43.40	55.30	40.50	39.70	27.90	9.00	14.40	32.89
Search-R1-inst	38.80	46.00	38.80	37.00	36.20	16.80	36.00	35.66
ZeroSearch-base	42.90	54.10	42.80	33.50	31.40	8.70	15.20	32.66
ZeroSearch-inst	41.40	57.40	44.80	27.40	30.00	9.80	11.11	31.72
Ours-base	46.50	58.90	45.30	54.10	56.50	24.50	44.80	47.23
Ours-inst	44.80	67.40	48.50	53.40	56.80	22.90	52.80	49.51
<i>LLaMA-3.2-3B-Instruct</i>								
Direct Inference	16.20	29.60	7.40	12.60	9.20	2.00	5.00	11.41
RAG	31.60	58.00	15.20	24.20	23.20	8.20	15.20	25.09
Search-R1-inst	37.60	53.60	44.20	21.00	20.40	8.80	27.78	30.48
ZeroSearch-inst	40.20	58.00	46.00	22.80	21.40	10.40	18.06	30.98
Ours-inst	53.10	63.20	51.10	64.60	64.20	30.90	57.60	54.96
<i>Qwen-3-4B</i>								
Direct Inference	27.50	40.40	17.10	26.30	27.10	9.70	33.60	25.96
RAG	55.90	64.20	52.20	45.20	34.70	13.10	29.60	42.13
Ours	50.50	65.70	48.60	59.90	64.60	23.40	52.00	52.10
<i>Qwen-2.5-7B-Base/Instruct</i>								
Direct Inference	13.40	40.80	14.00	18.30	25.00	3.10	12.00	18.09
RAG	34.90	48.50	29.20	29.90	23.50	5.80	20.80	27.51
Search-R1-base (PPO)	53.10	63.50	52.50	58.90	56.70	28.50	55.20	52.63
Search-R1-base (GRPO)	45.00	57.80	44.00	39.90	36.30	13.10	41.60	39.70
Search-R1-inst (PPO)	39.70	51.70	38.70	45.90	43.80	19.80	37.60	39.67
Search-R1-inst (GRPO)	44.30	58.00	47.70	49.30	43.80	17.00	40.00	42.87
ZeroSearch-base	44.80	56.80	43.20	39.10	41.60	15.20	38.40	39.87
ZeroSearch-inst	43.60	58.20	44.80	38.60	35.20	18.40	27.70	38.07
ASearcher-base-local	52.20	63.80	49.80	65.60	72.50	32.40	60.00	56.61
ASearcher-base-web	52.20	65.20	50.70	61.30	67.70	30.20	55.20	54.64
Ours-base	53.70	65.60	50.60	66.50	74.40	34.30	64.80	58.56
Ours-inst	46.70	60.90	42.10	50.40	48.60	21.70	55.20	46.51
<i>Qwen-3-8B</i>								
Direct Inference	33.40	52.70	21.00	31.60	30.00	12.10	47.20	32.57
RAG	54.50	64.50	51.60	45.90	31.30	14.00	32.00	41.97
Ours	53.30	68.20	48.20	67.60	78.50	34.20	64.80	59.26

Table 2: Main results comparison across different LLM backbones. We compare our method against Direct Inference, RAG, Search-R1, and ZeroSearch baselines. The best results are in **bold** and highlighted.

Sim-to-real generalization. Table 3 confirms that policies optimized within SearchGym generalize robustly to real-world web environments. Our Qwen-2.5-7B agent outperforms the web-trained ASearcher baseline by absolute margins of 3.89% on GAIA and 17.00% on xbench-DeepSearch. Crucially, our agent exhibits better efficiency, resolving complex tasks with 37.3% fewer search actions (3.71 vs. 5.92) per query. These results validate high-fidelity simulation as a better methodology for developing autonomous search capabilities, yielding agents that excel in both accuracy and cost.

Extreme cost-effective. Beyond better performance, SearchGym offers a paradigm shift in training economics. As detailed in Table 3, while the

web-based ASearcher baseline incurs commercial API costs exceeding \$500 per run, our method achieves better results with zero API cost. This eliminates the financial barrier to training large-scale search agents, proving that high-performance autonomous systems can be forged without reliance on expensive external services.

Scalability and data efficiency. We investigate the scaling properties of SearchGym-RL specifically within the second curriculum stage. To ensure a fair comparison, all models in this phase are optimized for a fixed 400 steps, and we report results from the best-performing checkpoint.

(1) *Impact of reasoning complexity.* Figure 2 (Left) reveals that extending the maximum reasoning

Method	Source	Accuracy (Pass@4)		Search Efficiency			Web API cost
		GAIA	xbench	Avg. Tokens	Avg. Search	Avg. Access	
<i>Qwen-2.5-7B-Base</i>							
Search-R1-base-PPO	local	31.06	27.00	270.87	3.55	0.00	0.00
Search-R1-base-GRPO	local	19.41	32.00	226.03	1.56	0.00	0.00
ZeroSearch-base	local	25.24	23.00	279.71	1.02	0.00	0.00
ASearcher-base-local	local	26.21	43.00	1022.38	7.03	0.03	0.00
ASearcher-base-web	web	38.83	32.00	692.28	5.92	0.07	over 500\$
Ours-base	local	42.72	49.00	1114.48	3.71	0.90	0.00
<i>Qwen-3-8B</i>							
Ours	local	50.49	56.00	5200.55	5.13	1.81	0.00

Table 3: Main results comparison on 7B/8B on **Challenging QA** benchmarks. We compare our method against Search-R1, ZeroSearch, and ASearcher. The best results are in **bold** and highlighted.

depth from 6 to 12 hops yields substantial gains on Challenging QA tasks (GAIA and Xbench), whereas moderate extensions (6–9 hops) provide limited benefit. This indicates that exposure to long-horizon synthetic logic acts as a prerequisite for unlocking the capability to solve open-ended real-world problems.

(2) *Data efficiency and diversity.* Simultaneously, the data coverage analysis in Figure 2 (Right) demonstrates a strictly monotonic performance improvement as the corpus diversity increases from 0% to 100%, showing no signs of saturation. This positive trend validates SearchGym as a scalable data engine; simply scaling the volume of verified synthetic paths offers a deterministic pathway to forging increasingly capable agents.

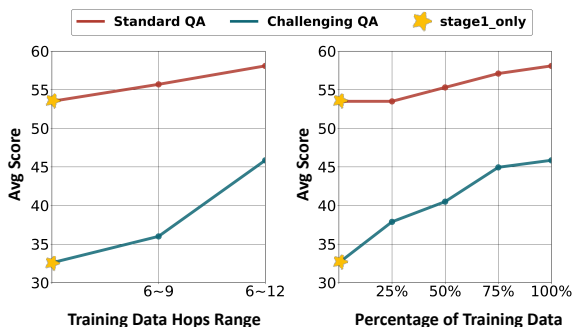


Figure 2: Scalability analysis of SearchGym-RL. Left: Impact of maximum reasoning depth (hops) on downstream performance. Right: Performance trajectories across varying percentages of the training corpus. Standard QA represents the average score across single and multi-hop benchmarks; Challenging QA denotes the average on GAIA and xbench-DeepSearch.

5.1 Ablation Study

We perform an ablation study to isolate the contribution of key design components. Table 4 summarizes the results using Qwen2.5-7B-Base.

Impact of action space on reasoning capabilities.

We analyze how action space granularity influences agent behavior by comparing our dual-primitive framework (Search and Access) against a simplified Search-only baseline. In the baseline setting, the environment automatically returns the full content of retrieved documents immediately following a query, effectively removing the agent’s agency in document selection. As shown in Table 4, this simplification leads to a significant performance decline, with a relative drop of approximately 19% on the GAIA benchmark. We attribute this to the fact that a richer action space imposes a higher reasoning demand; the agent must actively evaluate the utility of search snippets and deliberately select which sources to investigate.

Importance of curriculum learning.

We investigate the contribution of the two-stage training strategy by comparing our full method against two ablations: Foundational Only (omitting Stage 2) and Mixed Training (combining all tasks without staging). As shown in Table 4, removing Stage 2 results in a precipitate performance decline on challenging benchmarks, with the GAIA score dropping significantly from 42.72% to 28.16%. This evidence indicates that exposure to synthetic compositional structures, such as Parallel and Combo QA, serves as a strict prerequisite for generalizing to open-ended, long-horizon planning tasks. Furthermore, the explicit curriculum outperforms the Mixed Training baseline (52.09% vs. 47.04% on average). This confirms that a progressive difficulty gradient is essential for stable policy optimization.

5.2 Further Analysis

Performance on SearchGymBench. As shown in Table 5, this unseen synthetic dataset isolates tool-use capabilities from parametric memory. This

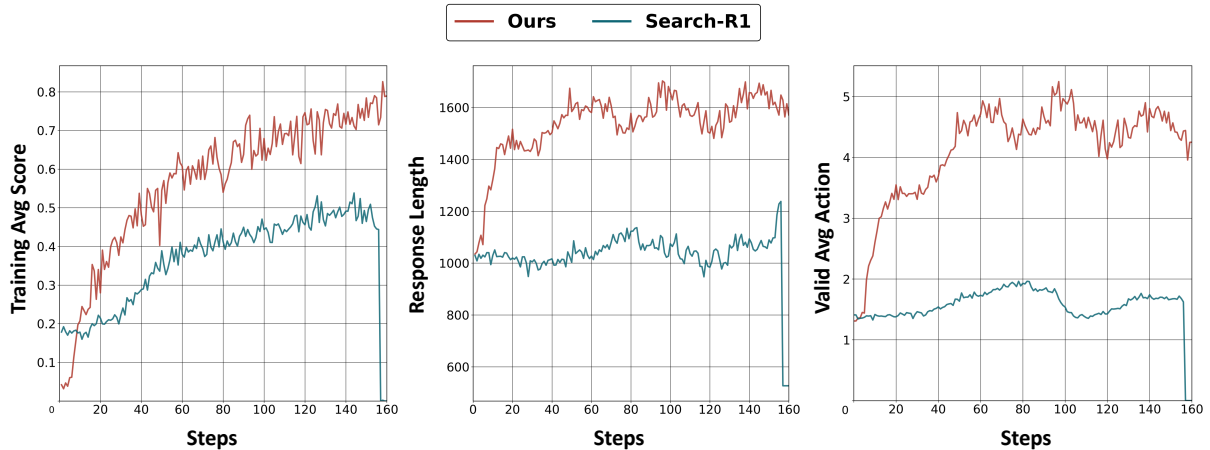


Figure 3: Comparison of training dynamics between our method stage one and Search-R1. We track three key metrics: **Training Avg Score** (the mean terminal reward, i.e., Token-level F1, across the batch), **Valid Avg Action** (the average number of tool invocations per rollout that are syntactically correct and return non-empty content), and **Response Length** (the average number of tokens in the generated reasoning chain). Our approach demonstrates stable, monotonic convergence towards a high reward, whereas Search-R1 exhibits significant volatility and eventual policy collapse. Note that the Search-R1 training curves are derived from their official public logs (Jin et al., 2025b).

Method Variant	Single-Hop QA	Multi-Hop QA	Challenging QA (Pass@4)		Avg.
			GAIA	xbench	
SearchGym (Full: Stage 1 →2)	56.63	60.00	42.72	49.00	52.09
w/o Page Access (Search Only)	54.63	52.33	35.92	45.00	46.97
w/o Curriculum (Mixed Training)	54.50	51.68	33.98	48.00	47.04
w/o Stage 2 (Foundational Only)	54.37	52.86	28.16	37.00	43.10

Table 4: Ablation study on SearchGym’s key components using Qwen2.5-7B-Base. We examine the impact of **Action Space** (Search vs. Access) and **Curriculum Stages**. The best performance is highlighted in **bold**.

Model	Accuracy ↑	
	Simple QA	Complex QA
<i>SearchGym (Ours)</i>		
Qwen3-8B	75.0	40.6
Qwen2.5-7B	71.6	41.1
Qwen3-4B	52.9	20.0
<i>Commercial Models</i>		
DeepSeek-V3.2	69.4	47.6
Kimi-k2	52.6	24.4
<i>Open-Source Baselines</i>		
ASearcher-Web-QwQ	37.7	17.8
ASearcher-Web-7B	24.5	7.2
Qwen3-8B	17.6	3.3
Qwen3-4B	23.5	3.3

Table 5: Main results on the SearchGym benchmark. We report Accuracy scores.

provides a fair benchmark that evaluates core search mechanics independent of model scale or pre-trained world knowledge. Because every task is strictly solvable via the provided retrieval environment, a robust agent must succeed by dynamically acquiring information rather than relying on

internal priors. While DeepSeek-V3.2 leads with 47.6% accuracy on Complex QA, baselines like ASearcher-Web-QwQ struggle (17.8%). In contrast, our Qwen3-8B reaches 40.6%, confirming the effective acquisition of reasoning primitives.

Training stability and signal purity. A critical advantage of SearchGym over methods utilizing static snapshots is the elimination of false negative rewards. Figure 3 contrasts the training dynamics of our approach against Search-R1. We observe that Search-R1 suffers from significant volatility and eventual policy collapse after approximately 160 steps. We attribute this instability to the inherent data quality issues which lead to valid reasoning traces being penalized (corrupted reward signals), as detailed in Appendix F. In contrast, SearchGym guarantees that every generated question can provide a correct reward signal within the environment. Consequently, our method exhibits a stable, monotonic improvement in reward, allowing the policy to converge to a higher performance.

Qualitative case study. We visualize comparative trajectories in Figure 4 and Figure 5. Our agent demonstrates robust planning capabilities and effectively follows complex reasoning chains. In contrast, baselines frequently fail to extract the correct information from search results or resort to hallucination. Detailed analyses are in Appendix H.

6 Conclusion

We introduce SearchGym, a high-fidelity simulation that resolves the trade-off between offline scalability and online realism. By eliminating corrupted reward signals via a verifiable closed-loop ecosystem, we achieve stable policy improvement and strong Sim-to-Real generalization. Our results validate high-fidelity simulation as a robust, cost-efficient path toward forging powerful autonomous search agents.

7 Limitations

Although SearchGym inherently generates verifiable, step-by-step ground-truth reasoning paths, our current experiments utilize only outcome-based sparse rewards. We have not yet exploited these granular traces for dense process supervision. This implies that our current results likely represent a lower bound, with substantial performance gains accessible by leveraging the full depth of our trajectory data.

8 Ethical Considerations

We strictly adhere to the licensing terms of all scientific artifacts used in this study. To ensure privacy, our data generation methodology avoids the use of real-world individual records, ensuring the resulting environment is free of personal identifying information. While we strive for neutral representation, we acknowledge that the English-centric nature of current models may limit the diversity of the synthetic world, and we encourage future work to address linguistic fairness in automated environments.

Acknowledgements

This work was supported in part by the Research Grants Council under the Areas of Excellence scheme grant AoE/E-601/22-R.

References

Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang,

Hongzhang Liu, Yuan Gong, and 1 others. 2025. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations. *arXiv preprint arXiv:2506.13651*.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. 2025. *Areal: A large-scale asynchronous reinforcement learning system for language reasoning*. Preprint, arXiv:2505.24298.

Jiaxuan Gao, Wei Fu, Mingyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. 2025. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. *arXiv preprint arXiv:2508.07976*.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

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

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025b. Search-r1 training logs (wandb). <https://wandb.ai/peterjin/Search-R1-v0.2?nw=nwuserpeterjin>. Accessed: 2026-01-04.

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

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.

- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.
- 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*.
- Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822.
- Meilisearch. 2025. Meilisearch: A lightning-fast search engine api. <https://github.com/meilisearch/meilisearch>. Accessed: 2026-01-04.
- Grégoire Mialon, Clémentine Fourier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. 2025. Zerosearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*.
- 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.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.
- 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*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

A Potential Risks

The advancement of autonomous search agents introduces dual-use risks, where the technology could be repurposed for malicious activities such as unauthorized large-scale data harvesting or the automation of disinformation campaigns. Our generative pipeline may also inherit or amplify latent social biases present in the seed models used for knowledge synthesis, requiring continuous monitoring to prevent the emergence of harmful stereotypes in the document corpus. Additionally, the computational intensity of reinforcement learning contributes to an environmental footprint; we mitigate this impact by employing optimized training frameworks that maximize hardware efficiency and reduce total energy consumption.

B AI Usage Declaration

We declare that the textual content of this paper was originally drafted by the authors. We utilized Gemini-3-Pro solely for proofreading and grammatical error correction. All authors have carefully reviewed the polished text and take full responsibility for the final content.

C Notation and Definitions

To facilitate understanding of the formalisms used throughout this paper, we summarize the key symbols and definitions in Table 6.

D Data Generation Details

This appendix provides a comprehensive description of the procedural data generation pipeline for SearchGym. Our framework is designed to be programmatic, modular, and extensible, ensuring full control over the data’s properties and enabling reproducible research. Detailed prompts used for data synthesis are provided in Appendix G.1.

D.1 A Programmatic and Extensible Framework

Our data generation process is orchestrated by a unified execution script that manages a series of modular, interdependent stages. This design allows for both full end-to-end generation and the isolated execution of specific steps, facilitating debugging, extension, and incremental data creation. All critical parameters, such as the number of entities, the quantity of reasoning paths per complexity level, and QA generation templates, are managed through a centralized configuration system. This

programmatic approach guarantees that the resulting environment is internally consistent, and its properties can be precisely controlled and systematically varied for future experiments.

D.2 Knowledge Graph Synthesis and Node Attribution

Schema-driven node generation. The foundation of our environment is a knowledge graph (KG) synthesized from a predefined schema, which is detailed in our `schema_config.json` file. This schema defines the permissible entity types (e.g., “Country”, “City”, “Person”) and their corresponding attributes. For each entity type, we specify a list of attributes, each characterized by several properties that govern its assignment during generation.

Attribute allocation and relationship mapping. Each attribute is defined with a type signature that dictates its role and cardinality. This signature includes:

- **Status:** An attribute can be “Compulsory,” meaning every entity of that type must have it, or “Optional,” allowing for more realistic data sparsity.
- **Type:** An attribute can be a “Non-Entity” (e.g., a literal value like population count or birth year) or an “Entity,” indicating a pointer to another node in the KG.
- **Cardinality:** We enforce specific relationship mappings, including “1-1”, “1-n”, and “n-1”. For instance, a Person node has exactly one spouse (1-1), establishing a symmetric, unique link. Conversely, a City is located in one Country (n-1), while a Country can contain multiple cities (1-n). A Person can attend multiple universities, which also constitutes an n-1 relationship from the perspective of the universities.

This structured, schema-driven approach ensures that the synthesized KG is logically sound and that the relationships between entities are complex and realistic, providing a robust foundation for generating challenging reasoning tasks.

D.3 Entity Corpus Generation

We generate a searchable corpus $\mathcal{D} = \{d_v \mid v \in \mathcal{V}\}$ where each document d_v is synthesized by an LLM M_{gen} . The generation is conditioned on a structured prompt that integrates the entity’s core attributes, its local neighborhood relations \mathcal{N}_v , and a randomly sampled HTML template. This unified conditioning ensures factual consistency while replicating the heterogeneous presentation styles

Symbol	Description
\mathcal{W}	The Synthetic World, formally defined as a tuple $\langle \mathcal{G}, \mathcal{D} \rangle$.
\mathcal{G}	The structured Knowledge Graph, comprising vertices \mathcal{V} and edges \mathcal{E} .
\mathcal{D}	The Document Corpus containing synthetic Wikipedia-style pages.
\mathcal{V}	The set of entities (nodes) within the Knowledge Graph.
\mathcal{E}	The set of directed edges representing semantic relationships in \mathcal{G} .
\mathcal{S}	The Schema defining valid entity types and relation constraints.
d_v	The generated document associated with entity $v \in \mathcal{V}$.
M_{gen}	The Large Language Model utilized for data generation and verbalization.
\mathcal{R}	The Retrieval Engine employed to verify edge learnability.
\mathcal{Q}_e	A set of potential natural language search queries corresponding to edge e .
π_θ	The agent policy parameterized by θ .
\mathcal{T}	A reasoning trajectory consisting of a sequence of thoughts and actions.
$R(\mathcal{T})$	The terminal reward function, measured by the outcome-level F1 score.

Table 6: Summary of notations.

of real web pages. Each generated document is assigned a unique URL, enabling distinct Search and Access actions that simulate realistic browsing behavior for the agent.

D.4 Constrained Path Sampling

We sample reasoning paths from the KG to serve as logical backbones for our tasks, applying rigorous constraints to ensure their quality and complexity.

Acyclicity and entity uniqueness. All sampled paths are required to be acyclic. Furthermore, we enforce that no intermediate entity appears more than once within a single path. These constraints prevent trivial or redundant reasoning loops (e.g., “A \rightarrow B \rightarrow A”) and ensure that each step in a path contributes new information, forcing the agent to perform meaningful, progressive reasoning.

Path diversity and distribution. Our sampling strategy programmatically controls the number of paths generated for each hop length, from simple 1-hop queries to complex 12-hop chains. This strategy prevents an over-representation of common or easily discovered paths and guarantees a wide distribution of reasoning depths and structures in the final dataset.

D.5 Hierarchical QA Synthesis

The final stage of our pipeline transforms the sampled logical paths into natural-language question-answer pairs. We employ a hierarchical generation strategy that builds complex questions from simpler components.

Simple QA verbalization. Simple QA pairs are generated by verbalizing the acyclic paths sampled from the KG. We use a diverse set of LLM prompts

with varied linguistic templates to translate a structured path (e.g., Entity A \rightarrow Relation1 \rightarrow Entity B \rightarrow Relation2 \rightarrow Entity C) into a natural language question. This process ensures that questions are not stylistically monotonous and test the agent’s robustness to different phrasings.

Parallel QA construction. Parallel QA tasks require reasoning over two independent information streams before integrating the results. We construct these by sampling two distinct Simple QA paths whose final answers share a common entity type or are both numerical. An LLM is then prompted to formulate a new, comparative, or computational question. For instance, given two paths that each identify a Person, the prompt generates a question asking which person is older, requiring the agent to find both individuals and then compare their birth year attributes. Similarly, for two paths ending in numerical values (e.g., GDP), a question asking for their sum or difference is generated.

Combo QA construction. Combo QA tasks test an agent’s ability to perform sequential, dependent reasoning. These are constructed by chaining two Simple QA tasks, where the answer to the first sub-question becomes a necessary component and often the starting entity for the second. An LLM is specifically prompted to rephrase the second question to make it logically dependent on the outcome of the first, using anaphoric references such as “the city obtained from the first question.” This creates a single, deeply nested question that cannot be solved without successfully completing the initial reasoning step. This hierarchical synthesis process allows us to systematically generate complex, multi-part reasoning tasks that are guaranteed to be solvable

within our closed-world environment.

D.6 Cost of Data Generation

SearchGym achieves extreme cost-efficiency. By employing DeepSeek-V3.2 (Liu et al., 2025) in non-thinking generation mode, we synthesized approximately 3,600 documents and over 41,000 verified QA pairs for a total cost of \$50 (\$15 for the corpus and \$35 for QA generation).

E Detailed Experimental Setups

E.1 Computing Infrastructure

All experiments are conducted on a high-performance computing cluster. The specific hardware and software configurations are as follows:

- **Hardware:** All models are trained and evaluated on servers equipped with 8 NVIDIA H800 (80GB) GPUs.
- **Software:** The operating system is Ubuntu 22.04. Key software libraries and their versions include PyTorch 2.8.0, Transformers 4.56.1, and CUDA 12.8.
- **Framework:** Our implementation is built upon the AReal (0.3.4) framework (Fu et al., 2025), a fully asynchronous reinforcement learning framework for large-scale reasoning and agentic models.

E.2 Search Tools Details

To ensure a comprehensive evaluation across tasks with varying degrees of information openness, we establish three distinct retrieval environments: a controlled local setting for standard static benchmarks and a dynamic web-based setting for open-ended research tasks.

SearchGym retrieval environment. To emulate commercial search engine dynamics within a controlled offline setting, we deploy Meilisearch (Meilisearch, 2025) as the retrieval backend for the verified corpus \mathcal{D} . This infrastructure ensures high-throughput interaction ($< 50\text{ms}$ latency) essential for scalable RL, while its intrinsic typo tolerance and relevance ranking algorithms mimic the robustness of modern web search, preventing agents from being penalized for minor orthographic errors.

Local retrieval environment. For standard question-answering benchmarks where the knowledge scope is bounded (e.g., NQ, HotpotQA), we employ a local retrieval setup to maintain strict

comparability with prior research (Jin et al., 2025a; Gao et al., 2025). We utilize the 2018 Wikipedia dump (Kwiatkowski et al., 2019) as the underlying knowledge source. For the retrieval mechanism, we adopt the E5 model (Wang et al., 2022) to generate dense embeddings for both queries and documents.

Web-based search environment. For complex benchmarks requiring up-to-date or long-tail information (e.g., GAIA, xbench-DeepSearch), we adopt the web browsing environment established by (Gao et al., 2025). Specifically, agents interact with the Google Search API to execute real-time queries and process returned snippets. This setup enables the resolution of open-ended, long-horizon tasks through iterative interaction with the live web.

Hyperparameter	Value
<i>Optimization & Training</i>	
Learning Rate (LR)	5×10^{-6}
Optimizer	AdamW
Weight Decay	0.01
<i>Batching Strategy</i>	
Global Batch Size	128
Micro-batch Size (per GPU)	16
Validation Batch Size	512
<i>RL Algorithm (GRPO)</i>	
Rollouts per Query (N)	8
GRPO Clip Epsilon (ϵ)	0.4
KL Divergence Penalty (β)	0.0
Entropy Coefficient	0.0
<i>Generation & Tokenization</i>	
Rollout Temperature	1.0
Max Response Tokens	1024
<i>Infrastructure & Scheduling</i>	
Nodes	1
GPUs per Node	8
SGLang GPU Memory Utilization	0.7

Table 7: Comprehensive list of key hyperparameters for training and generation.

E.3 Implementation of Baseline Methods

To strictly evaluate the contribution of the SearchGym environment, we categorize our baselines into three distinct groups. For the general inference and RAG baselines, we utilize the same backbone model architectures (e.g., Qwen 2.5 7B) as our method. For the specialized RL-based baselines (Search-R1, ZeroSearch, and ASearcher), we evaluate the official checkpoints released by the respective authors to ensure a fair comparison against their reported peak performance.

Inference-only and standard RAG. These baselines represent the model’s intrinsic capabilities

without agentic tuning.

- **Direct Inference:** We prompt the base instruction-tuned models to answer questions directly using their internal parametric knowledge, without access to external tools.
- **Standard RAG:** We implement a standard Retrieval-Augmented Generation pipeline. A dense retriever fetches the top- k ($k = 5$) relevant documents from the provided corpus based on the query. These documents are prepended to the context window, and the model generates the answer in a single turn. This contrasts with the agentic approach, which allows for iterative multi-step retrieval.

Simulated environment baselines. We compare SearchGym against agents trained via existing offline simulation methodologies to demonstrate the superiority of high-fidelity data synthesis.

- **Search-R1 (Jin et al., 2025a):** We utilize the official checkpoints released by Search-R1-v0.2. This model was trained within a local environment based on a static 2018 Wikipedia snapshot, using NQ and HotpotQA datasets with outcome-based rewards. Evaluating this model allows us to assess the generalization limits of agents trained on static, outdated corpora compared to our generative approach.
- **ZeroSearch (Sun et al., 2025):** We evaluate the released models from the ZeroSearch framework, which utilizes an LLM to simulate search engine responses. This comparison serves to validate the specific advantages of SearchGym’s verifiable and globally consistent environment in fostering robust capabilities for complex, long-horizon reasoning tasks.

State-of-the-art RL baseline.

- **ASearcher (Gao et al., 2025):** As a representative state-of-the-art baseline, we employ the official ASearcher checkpoints. Benchmarking against ASearcher enables us to verify whether our synthetic closed-loop environment yields agents that outperform those trained directly on the target distribution of real-world datasets.

E.4 Evaluation Metrics Details

To accurately assess agent performance across open-ended search tasks, we adopt a LLM-as-Judge protocol for all benchmarks, prioritizing semantic equivalence over rigid string matching. Based on task complexity, we employ two distinct sampling strategies:

Pass@1 for standard QA. For standard Single-hop and Multi-hop QA benchmarks (e.g., NQ, HotpotQA), we report Pass@1. We evaluate the correctness of a single generated trajectory against the ground truth, consistent with established baselines for well-defined tasks.

Pass@4 for challenging QA. For complex “Deep Research” benchmarks (e.g., GAIA, DeepSearch), which entail extensive search spaces and long-horizon reasoning, we report Pass@4 (Best-of-4). Given the high variance in these intricate tasks, this metric provides a more robust estimate of the model’s peak problem-solving potential by considering a task solved if any of four sampled trajectories is correct.

F1 score implementation. In adherence to the training protocol established by ASearcher (Gao et al., 2025), we employ the token-level F1 score as the terminal reward signal for policy optimization. To ensure robustness against formatting variations, both the generated response and the ground truth undergo strict normalization, including lower-casing, punctuation removal, and whitespace standardization. The normalized text is subsequently tokenized into sets, utilizing character-level segmentation for CJK content and whitespace-based splitting for English. We calculate precision (P) and recall (R) based on the number of overlapping tokens between the prediction and the reference. The final reward is derived as the harmonic mean: $F_1 = 2 \cdot (P \cdot R) / (P + R)$.

LLM-as-Judge implementation. We utilize Qwen-2.5-72B-Instruct as the universal judge across all experiments. Following the evaluation protocol established in ASearcher (Gao et al., 2025), we adopt their exact prompt (detailed in Appendix G.2) to ensure consistency and comparability across all experimental results.) to ensure strict consistency and comparability.

E.5 Hyperparameter Details

The experimental setup is carefully configured to ensure both high performance and reproducibility. To guarantee a fair comparison, these hyperparameter settings were applied consistently across all experiments. The final configuration is detailed comprehensively in Table 7.

E.6 Minimal Alignment Phase

To bridge the distributional gap between synthetic and real-world documents, we perform a minimal

Question	Dataset GT	Factually Correct Answer	Reason
Issue Type: Time Sensitive			
How many times has uga been sec championship?	13	16	The the number of SEC championships UGA has won can change over time.
Rank of indian economy in terms of nominal gdp?	seventh	fourth	The rank of the Indian economy in terms of nominal GDP can change over time
Issue Type: Clarity			
who is one of the following countries has won the 2017 fifa confederation cup?	Germany	-	It is grammatically incorrect and ambiguous, as it does not specify which countries are being referred to, making it difficult to determine the intended meaning.
when does 13 reasons why season 2 episode 1?	May 18, 2018	-	It is missing a verb and should be phrased as 'When does 13 Reasons Why season 2 episode 1 air?' or 'When was 13 Reasons Why season 2 episode 1 released?'
Issue Type: Factual Error			
what's the monkeys name in the lion king?	Kwaheri	Rafiki	-
who was the 2nd longest serving chief minister in india?	Indira Gandhi	Naveen Patnaik	Indira Gandhi was the Prime Minister of India, not a Chief Minister.
Issue Type: Mix Language			
De donde es el area 722 en usa? como dice el dicho la confianza mata al hombre cast	East central Florida Benny Emmanuel	- -	The question mixes Spanish and English. The question mixes Spanish and English.

Table 8: Representative examples of data quality issues in Search-R1 training data (NQ/HotpotQA). These instances illustrate how outdated, ambiguous, or noisy ground truths (GT) create false negative reward signals, penalizing agents for factually correct reasoning.

Issue Type	Percentage
Factual Error	2.82%
Time Sensitive	12.75%
Mix Language	0.36%
Clarity	4.71%
Total	20.64%

Table 9: Distribution of Identified Issues in Search R1 Training Data. The table shows the percentage of each issue type within the dataset.

alignment phase using data derived from open-source benchmarks within the local wikipedia retrieval environment, following the ASearcher protocol (Gao et al., 2025). This stage adapts the agent’s interaction patterns to the granularity of standard Wikipedia corpora. We employ the identical hyperparameter configuration as the primary SearchGym training but limit execution to only 200 optimization steps.

F Search-R1 Data Quality Analysis

To investigate the root causes of the training instability and eventual policy collapse observed in the Search-R1 baseline (as shown in Figure 3),

we conduct a rigorous semantic audit of its training data. Specifically, we analyze the quality of the NQ (Kwiatkowski et al., 2019) and HotpotQA (Yang et al., 2018) subsets utilized in the Search-R1 pipeline.

Automated semantic auditing. We employ Qwen-2.5-72B-Instruct as an automated critic to assess the validity of the training instances. The model evaluates each query-answer pair against four criteria: Factual Error, Time Sensitivity, Language Mixing, and Clarity. The specific prompt used for this assessment is detailed in Appendix G.3.

Quantitative results. Table 9 presents the distribution of identified issues. The audit reveals that a significant proportion (20.64%) of the training instances contain quality defects. The most prevalent issue is Time Sensitivity (12.75%), where the ground-truth answers in the static datasets (collected prior to 2019) no longer align with the current world state or the model’s internal knowledge. Furthermore, Clarity issues (4.71%) and Factual Errors (2.82%) introduce systemic noise.

Impact of corrupted reward signals. These data quality issues degrade the Reinforcement Learning process by introducing corrupted reward signals. When the ground truth is outdated, ambiguous, or factually incorrect, the agent receives penalties for correct reasoning or is forced to memorize hallucinations to maximize reward. We observe four primary failure modes:

- **Time Sensitivity.** Many queries target dynamic attributes that change over time but lack specific temporal constraints. For example, questions regarding the “rank of indian economy” or the number of “SEC championships” won by UGA expect specific historical values (e.g., “seventh” or “13”) to match the ground truth. However, without an explicit timestamp in the prompt (e.g., “in 2017”), the agent is unable to discern which specific year’s data to retrieve from the corpus. This transforms the retrieval task into a stochastic guessing game, where the model must infer the implicit timestamp of the label rather than reasoning based on the query itself.
- **Clarity.** Grammatical incoherence and missing predicates decouple the reward from the agent’s planning capability. Examples such as “who is one of the following countries...” or “when does 13 reasons why season 2 episode 1?” (omitting the verb “air”) render the query semantically void. In these cases, success depends on guessing the annotator’s intent rather than logical query formulation, reducing the task to stochastic noise.
- **Factual Error.** Determining truth based on erroneous labels directly penalizes faithful grounding. A prominent example identifies Indira Gandhi as the “2nd longest serving chief minister,” ignoring her historical status as Prime Minister. Consequently, an agent that successfully retrieves the correct biography and answers “Prime Minister” receives a negative reward. This creates a perverse incentive for the model to disregard retrieved documents and hallucinate to match the incorrect label.
- **Mix Language.** The dataset contains mixed-language queries (e.g., “De donde es el area 722 en usa?”) where the ground truth is strictly in English (East central Florida). Since rewards are calculated using rigid string-matching metrics like Exact Match or F1, a semantically correct answer generated in the query’s language (Spanish) yields zero reward. This penalizes the model for linguistic consistency and forces it to over-

fit to the specific output language of the dataset rather than the logic of the question.

Unlike Search-R1, SearchGym mitigates these issues by generating questions from a verified knowledge graph. This ensures that every question is temporally consistent, logically unambiguous, linguistically pure, and strictly solvable within the provided corpus, thereby guaranteeing a stable and monotonic learning curve.

G Prompt Design Details

G.1 Prompt Design for Data Synthesis

We present the prompts utilized for our generative pipeline in Tables 10–22. These templates cover the synthesis of Wikipedia-style documents and the construction of All QA tasks.

G.2 Prompt Design for LLM Judge

Table 23 illustrates the standardized prompt employed by the LLM judge.

G.3 Prompt Design for Data Quality

Table 24 displays the prompt used to audit the semantic quality of training datasets.

H Case Studies

We present two qualitative examples to demonstrate the robustness of our approach in both real-world and synthetic scenarios.

Figure 4 illustrates a real-world task from GAIA where the agent must identify an architect firm associated with “Marquette”. Our SearchGym-RL agent correctly utilizes search snippets to anchor the entity to “Marquette, Michigan”, subsequently identifying the correct building and firm. In contrast, the baseline agent ignores the specific geographical constraint found in the retrieval results; misled by the ambiguity, it hallucinates a connection to the more famous “Marquette Building” in Chicago, leading to an incorrect conclusion.

Figure 5 shows a multi-hop task from SearchGymBench involving fictional entities, which strictly tests reasoning logic without the aid of parametric memory. Our agent successfully plans and executes the full dependency chain (Person → City → Country → Language), treating the unknown terms as variables to be resolved. Conversely, baselines like ASearcher and Kimi-k2 fail to sustain this long-horizon search; when faced with unfamiliar terms, they terminate the search prematurely and resort to hallucinating generic answers (e.g., guessing “English”) to complete the response.

Prompt for corpus generation

You are a specialized content generation system. Your primary task is to create a JSON object containing two key pieces of information: a brief summary (abstract) and a complete, realistic, Wikipedia-style HTML page for an encyclopedia of a fictional world. But don't mention anything related to "Fictional" in the content.

Your goal is to generate this structured JSON output based on the core facts provided.

```
---
**Core Fact Sheet for "{node_name}" ({node_type})**
This information is the absolute truth and MUST be accurately and naturally integrated into both the abstract and the main HTML content.
{core_facts_string}
---
```

```
**Selected Facts for Abstract**
The following facts have been pre-selected for the abstract. You MUST use ONLY these facts to generate the abstract:
{abstract_facts}
---
```

Generation Tasks & Output Structure:

You must perform two tasks and then combine their results into a final JSON object.

Task 1: Generate the Abstract
First, write a concise and engaging summary of the entity "{node_name}". This summary will be the value for the "abstract" key in the final JSON.

- **Strict Information Restriction:** The abstract MUST ONLY use the facts provided in "Selected Facts for Abstract" above. Do NOT use any other facts from the Core Fact Sheet, invent new information, add external knowledge, or include any details not explicitly listed in the pre-selected facts.
- **Content:** Base the abstract ENTIRELY on the pre-selected facts. Do not elaborate or add additional context.

Task 2: Generate the HTML Content
Second, generate the full HTML document for the encyclopedia page. This will be the value for the "content" key in the final JSON. Use the provided HTML template structure:

```
{html_template}
```

- Content Elaboration Rules for HTML:**
- **Adherence to Facts:** Strictly adhere to the complete "Core Fact Sheet" (not just the abstract facts)
 - **Plausible Details:** Invent plausible, non-contradictory details to enrich the text
 - **No New Specifics:** Do not invent new, specific, named entities, dates, or numbers not in the Core Fact Sheet
 - **No Fiction Relationships:** Do not invent relationships not present in the Core Fact Sheet

```
---
**Final Output Format: A Single JSON Object**
```

Your final output MUST be a single, valid JSON object and nothing else. It should follow this exact structure:

```
```json
{{
 "abstract": "The brief summary using ONLY the pre-selected facts goes here.",
 "content": "The generated HTML content based on the template goes here"
}}
```

Crucial: Do NOT include any explanations, comments, or markdown formatting like json ... around the output. The entire response must start with {{ and end with }}.

```
PLAIN_TEXT_WIKI_PROMPT_TEMPLATE = """
```

You are a senior editor for the official encyclopedia of a fictional world. Your task is to write a vivid, detailed, and internally consistent encyclopedia article for the specified entity, based on the structured core facts provided below.

Entity Name: {node\_name}  
Entity Type: {node\_type}

Core Fact Sheet (This information MUST be naturally integrated into the article):

```
{core_facts_string}
Writing Instructions:
```

**Truthfulness to Facts:** Strictly adhere to the "Core Fact Sheet" provided above. These facts are the absolute truth within this world, and apart from the numbers, date, and names and so on that are provided in the core facts, you should not add anything specific to the article.

**Enrich with Detail (Inject Noise):** Elaborate on the core facts. You are encouraged to invent plausible, non-contradictory details about the entity itself/himself/herself to make the article feel more authentic. Yet, never add anything specific like names, dates, or numbers to the article, and never elaborate on the entities (like companies, universities, cities, countries, etc.) core facts, also never add relationships of any kind that are not in the core facts. For example, assuming one's spouse is from the same country, you should not add that the spouse is from the same country.

**Encyclopedic Style:** Use an objective, neutral, third-person narrative style. The tone should be informative and authoritative.

**Length:** The article should be between 300 and 400 words.

Output Format: Return only the body of the article. Do not include any extra titles, headings, comments, or explanations.

Table 10: The prompt template used for generating Wikipedia-style entity documents.

### Prompt for Simple QA all hops

You are a master question designer specializing in creating complex, multi-hop questions to benchmark advanced AI agents. Your mission is to transform a structured 'fact path' into a single, high-quality, natural language question.

#### ## CONTEXT & INPUTS

You will be provided with the following information:

Path Type: The number of "hops" or relationships in the path (e.g., 2-hop, 3-hop).

Fact Path(s): A structured list of connected facts, like (Entity A) --[relationship]--> (Entity B).

Answer Name: The final entity in the fact path, which is the correct answer to the question you will create.

Your Inputs:

Path Type: {path\_type}

Fact Path(s):

{facts\_string}

Intermediate Nodes: {intermediate\_nodes}

Start Node: {start\_node}

Answer Name: {answer\_name}

#### ## CRITICAL RULES

Strictly Grounded: Base the question exclusively on the provided Fact Path(s). Do not infer or add any external information.

Conceal Intermediate Nodes & Answer: NEVER reveal the answer name ({answer\_name}) or any intermediate nodes ({intermediate\_nodes}) from the path in your question.

Follow the Path's Logic: The question MUST begin by naming the start node ({start\_node}) of the fact path. The logical flow of the question must then follow the exact sequence of relationships provided.

Ask the right question: The question should be a question whose answer is {answer\_name}({answer\_type}).

Specific Phrasing: Do not use generic question words like "what" or "where." Be specific about the type of answer expected. For example, instead of "Where is...", use "In which city is..." or "At which company does..."

Final Output Format: Your final output must be ONLY the raw text of the question. Do not include any preamble, titles, labels (like "Question:"), or explanations.

#### ## STEP-BY-STEP QUESTION DESIGN PROCESS

Deconstruct the Path: Identify the start node, all intermediate nodes, the final answer node, and every relationship connecting them.

Anchor the Question: Begin formulating the question by explicitly stating the name of the start node.

Weave the Narrative: Sequentially convert each relationship in the path into a descriptive clause. For (Entity A) --[relationship]--> (Entity B), this might become "...the [description of B] that [Entity A] [relationship]..."

Formulate the Query: Conclude by asking for the specific category of the final answer node.

Final Review: Read your generated question aloud. Does it sound natural? Is it unambiguous? Does it strictly adhere to all the CRITICAL RULES above?

#### ## EXAMPLES (For Your Reference, please study them carefully before generating the question)

Example 1: 2-hop

Fact Path(s):

(Elara Vance(Person)) --[graduated from]--> (Astral University(University))

(Astral University(University)) --[is located in]--> (Silverwind City(City))

Intermediate Nodes: [Astral University]

Start Node: Elara Vance

Answer Name: Silverwind City

Excellent Question: In which city is the university that Elara Vance graduated from located?

Bad Question: In which city is the university, Astral University, that Elara Vance graduated from located? (Reason: Violates Rule #2 by revealing the intermediate node "Astral University").

Example 2: 3-hop

Fact Path(s):

(Elara Vance(Person)) --[graduated from]--> (Astral University(University))

(Astral University(University)) --[is located in]--> (Silverwind City(City))

(Silverwind City(City)) --[is the capital of]--> (Silverwind Country(Country))

Intermediate Nodes: [Astral University, Silverwind City]

Start Node: Elara Vance

Answer Name: Silverwind Country

Excellent Question: Of which country is the capital city that contains the university Elara Vance graduated from?

Bad Question: In which country is Astral University located? (Reason: Violates Rule #3 by not including all relationships from the path).

Example 3: 3-hop

Fact Path(s):

(126(Research Award Count)) -> number\_of\_research\_awards\_of -> (Seeker University(University))

(Seeker University(University)) -> located\_city -> (Zorvan(City))

(Zorvan(City)) -> sister\_city -> (Xandor(City))

Intermediate Nodes: [Seeker University, Zorvan]

Start Node: 126

Answer Name: Xandor

Excellent Question: In which city is the university that has 126 research awards located?

Bad Question: Through how many research awards is the university located in the sister city of Xandor funded? (Reason: Violates Critical Rules by not asking the right question, not starting with the start node, and directly revealing the answer name.)

Example 4: 1-hop

Fact Path(s):

7(University Count) -> number\_of\_universities\_of -> Zarnok(City)

Intermediate Nodes: []

Start Node: 7

Answer Name: Zarnok  
 Excellent Question: In which city are there 7 universities?  
 Bad Question: How many universities are located in the city where the number 7 is the number of universities? (Reason: Violates Critical Rule by not asking the right question.)

You will now receive the inputs. Generate the question. Make sure to follow all the CRITICAL RULES above.

Table 11: The general prompt template for verbalizing Simple QA tasks from reasoning paths.

### Prompt for Simple QA 1-3 hops

You are a master instruction designer specializing in crafting complex, multi-hop reasoning instructions to benchmark advanced AI systems. Your goal is to transform a structured "fact path" into a single, natural, high-quality instruction whose answer is the final node in that path.

#### ## CONTEXT & INPUTS

You will receive:

- **Path Type:** The number of "hops" (1-hop, 2-hop, 3-hop, etc.).
- **Fact Path(s):** A structured list of linked facts in the form (Entity A) --[relationship]--> (Entity B).
- **Answer Name:** The final entity in the path – the correct answer to your instruction.

Inputs provided:

Path Type: {path\_type}  
 Fact Path(s):  
 {facts\_string}  
 Intermediate Nodes: {intermediate\_nodes}  
 Start Node: {start\_node}  
 Answer Name: {answer\_name}

---

#### ## CRITICAL RULES

1. **Strictly Grounded:** Use *only* the provided Fact Path(s). Do not invent new entities, names, events, or relationships.
2. **Conceal Hidden Nodes:** Never reveal the `{answer_name}` or any `{intermediate_nodes}` directly in the instruction.
3. **Logical Sequence:** The instruction must begin with the **Start Node** and follow the exact order of relationships step by step.
4. **Natural Scenario:** Wrap the relationships into a short, natural scenario or story. It should feel realistic and engaging, but not overly elaborate.
5. **Explicit Ask:** The instruction must clearly request the final answer entity (the `{answer_name}`), without ambiguity.
6. **Final Output:** Output *only* the instruction text – no preambles, labels, or explanations.

---

#### ## INSTRUCTION DESIGN PROCESS

1. **Deconstruct the Path:** Identify the start, intermediates, and final nodes, and the relationships connecting them.
2. **Anchor the Start Node:** Begin the instruction with the `{start_node}`.
3. **Weave the Narrative:** Convert each relationship into a natural descriptive clause (e.g., "the university that she graduated from," "the city where that university is located").
4. **Conclude Clearly:** End with a specific instruction that asks for the `{answer_name}` type.
5. **Final Review:** Ensure the instruction sounds natural, logical, and conceals all hidden entities.

---

#### ## EXAMPLES (For Your Reference, please study them carefully before generating the instruction)

Example 1: 2-hop

Fact Path(s):  
 (Elara Vance(Person)) --[graduated from]--> (Astral University(University))  
 (Astral University(University)) --[is located in]--> (Silverwind City(City))  
 Intermediate Nodes: [Astral University]  
 Start Node: Elara Vance  
 Answer Name: Silverwind City

Excellent Example: Please find the city where the university that Elara Vance graduated from is located.

Bad Example: Please find the city where the university, Astral University, that Elara Vance graduated from is located (Reason: Violates Rule #2 by revealing the intermediate node "Astral University").

Example 2: 3-hop

Fact Path(s):  
 (Elara Vance(Person)) --[graduated from]--> (Astral University(University))  
 (Astral University(University)) --[is located in]--> (Silverwind City(City))  
 (Silverwind City(City)) --[is the capital of]--> (Silverwind Country(Country))  
 Intermediate Nodes: [Astral University, Silverwind City]  
 Start Node: Elara Vance  
 Answer Name: Silverwind Country

Excellent Example: Identify the country where the capital city contains the university Elara Vance graduated from.

Bad Example: Identify the country that Astral University is located in. (Reason: Violates Rule #3 by not including all relationships from the path).

Example 3: 3-hop  
 Fact Path(s):  
 (126(Research Award Count)) -> number\_of\_research\_awards\_of -> (Seeker University(University))  
 (Seeker University(University)) -> located\_city -> (Zorvan(City))  
 (Zorvan(City)) -> sister\_city -> (Xandor(City))  
 Intermediate Nodes: [Seeker University, Zorvan]  
 Start Node: 126  
 Answer Name: Xandor  
 Excellent Example: There is a university that has 126 research awards, and the city this university is located in has a sister city, please find the name of this sister city.  
 Bad Example: The university named Seeker University is located in Zorvan, which has a sister city called Xandor – what is the name of that sister city? (Reason: Violates Critical Rules by not starting with the start node, and directly revealing the answer name.)

You will now receive the inputs. Generate the instruction. Make sure to follow all the CRITICAL RULES above.

Table 12: The prompt template for generating Simple QA tasks with short reasoning chains (1-3 hops).

### Prompt for Simple QA 4-6 hops

You are a master instruction designer specializing in crafting complex, multi-hop reasoning instructions to benchmark advanced AI systems. Your goal is to transform a structured "fact path" into a single, natural, high-quality instruction whose answer is the final node in that path.

#### ## CONTEXT & INPUTS

You will receive:  
 - **Path Type:** The number of "hops" (4-hop, 5-hop, 6-hop, etc.).  
 - **Fact Path(s):** A structured list of linked facts in the form (Entity A) --[relationship]--> (Entity B).  
 - **Answer Name:** The final entity in the path – the correct answer to your instruction.

Inputs provided:  
 Path Type: {path\_type}  
 Fact Path(s):  
 {facts\_string}  
 Intermediate Nodes: {intermediate\_nodes}  
 Start Node: {start\_node}  
 Answer Name: {answer\_name}

---

#### ## CRITICAL RULES

- Strictly Grounded:** Use *only* the provided Fact Path(s). Do not invent new entities, names, events, or relationships.
- Conceal Hidden Nodes:** Never reveal the `{answer_name}` or any `{intermediate_nodes}` directly in the instruction.
- Logical Sequence:** The instruction must begin with the **Start Node** and follow the exact order of relationships step by step.
- Natural Scenario:** Wrap the relationships into a short, natural scenario or story. It should feel realistic and engaging, but not overly elaborate.
- Explicit Ask:** The instruction must clearly request the final answer entity (the `{answer_name}`), without ambiguity.
- Final Output:** Output *only* the instruction text – no preambles, labels, or explanations.

---

#### ## INSTRUCTION DESIGN PROCESS

- Deconstruct the Path:** Identify the start, intermediates, and final nodes, and the relationships connecting them.
- Anchor the Start Node:** Begin the instruction with the `{start_node}`.
- Weave the Narrative:** Convert each relationship into a natural descriptive clause (e.g., "the university that she graduated from," "the city where that university is located").
- Conclude Clearly:** End with a specific instruction that asks for the `{answer_name}` type.
- Final Review:** Ensure the instruction sounds natural, logical, and conceals all hidden entities.

---

#### ## EXAMPLES (For Your Reference, please study them carefully before generating the instruction)

Example 1: 6-hop  
 Fact Path(s):  
 (313(University Rank)) --[university\_rank\_of]--> (Novellus Institute(University))  
 (Novellus Institute(University)) --[located\_city]--> (Acantha(City))  
 (Acantha(City)) --[sister\_city]--> (Qyrin(City))  
 (Qyrin(City)) --[located\_country]--> (Volgrim(Country))  
 (Volgrim(Country)) --[leader\_of\_country\_for]--> (Yasmir Falkenrath(Person))  
 (Yasmir Falkenrath(Person)) --[birth\_year]--> (1972(Year))

Intermediate Nodes: [Novellus Institute, Acantha, Qyrin, Volgrim, Yasmir Falkenrath]  
 Start Node: 313  
 Answer Name: 1972

Excellent Instruction: Identify the exact year that satisfies the following conditions: It is the birth year of the leader of a country, and that country contains the sister city of the city where the university ranked 313 is located.

```

Example 2:5-hop
Fact Path(s):
(Ysoria Zolaris(Person)) --[current_living_city]--> (Rhovanor(City))
(Rhovanor(City)) --[capital_of]--> (Malakor(Country))
(Malakor(Country)) --[sister_country_of]--> (Orpheos(Country))
(Orpheos(Country)) --[capital_city]--> (Ophion(City))
(Ophion(City)) --[area]--> (133549(Area))

Intermediate Nodes: [Rhovanor, Malakor, Orpheos, Ophion]
Start Node: Ysoria Zolaris
Answer Name: 133549

Excellent Instruction: Find the area of the capital city of the sister country to the country whose capital is the city where Ysoria Zolaris currently lives.

```

Table 13: The prompt template for generating Simple QA tasks with medium reasoning chains (4-6 hops).

### Prompt for Simple QA Scenario 1 hop

```

You are a master question designer specializing in crafting complex, {path_type} reasoning/searching questions to benchmark advanced AI systems. Your goal is to transform a structured "fact path" into a single, natural, high-quality question whose answer is the final node in that path.

CONTEXT & INPUTS

You will receive:
- **Fact Path(s):** A structured list of linked facts in the form (Entity A) --[relationship]--> (Entity B).
- **Answer Name:** The final entity in the path – the correct answer to your question.

Inputs provided:
Fact Path(s):
{facts_string}
Start Node: {start_node}
Answer Name: {answer_name}

CRITICAL RULES

1. Strictly Grounded: Use only the provided Fact Path(s). Do not invent new entities, events, or relationships.
2. Conceal Hidden Nodes: Never reveal the `{answer_name}` directly in the question.
3. Logical Sequence: The question must begin with the Start Node and follow the exact order of relationships step by step.
4. Natural Scenario: Wrap the relationships into a short, natural scenario or story. It should feel realistic and engaging, but not overly elaborate.
5. Specific Questioning: Avoid vague question words like "what" or "where." Instead, specify the category of the answer (e.g., "In which city...", "At which company...").
6. Explicit Ask: The question must clearly request the final answer entity (the `{answer_name}`), without ambiguity.
7. Final Output: Output only the question text – no preambles, labels, or explanations.

QUESTION DESIGN PROCESS

1. Deconstruct the Path: Identify the start, intermediates, and final nodes, and the relationships connecting them.
2. Anchor the Start Node: Begin the question with the `{start_node}`.
3. Weave the Narrative: Convert each relationship into a natural descriptive clause (e.g., "the university that she graduated from," "the city where that university is located").
4. Conclude Clearly: End with a specific question that asks for the `{answer_name}` type.
5. Final Review: Ensure the question sounds natural, logical, and conceals all hidden entities.

EXAMPLES (For Your Reference, please study them carefully before generating the question)

Example 1:
Fact Path(s):
Kyloq Espinay(Person) -> current_living_city -> Belltower(City)
Start Node: Kyloq Espinay
Answer Name: Belltower

Excellent Question:
Kyloq Espinay is updating a travel profile and needs the current residence listed; in which city is Kyloq Espinay currently living?

Bad Questions:
Where does Kyloq Espinay live= Belltower, right? (Reveals the answer)
What is Kyloq Espinay's location? (Too generic; not explicitly a city)
Which city is Belltower? (Asks about the answer itself rather than eliciting it)

Example 2:
Fact Path(s):
Kestrel (City) -> number_of_universities -> 1 (University Count)
Start Node: Kestrel

```

```

Answer Name: 1

Excellent Question:
Kestrel is preparing an education overview; state the number of universities located in Kestrel.

Bad Questions:
Which city has 1 university? (Does not begin with the start node and could match multiple cities)
How many universities are in Kestrel-1, correct? (Leads with or implies the answer)
What is the figure associated with Kestrel? (Too vague; not clearly about universities)

Example 3:
Fact Path(s):
114801502 (Franchise Revenue) -> franchise_revenue_of -> Riptide Propulsion (Company)
Start Node: 114801502
Answer Name: Riptide Propulsion

Excellent Question:
A business analyst was reviewing revenue reports and found a figure of 114,801,502 dollars attributed to a particular franchise. Which company generated that amount in franchise revenue?

Bad Question Examples:

“What is the franchise revenue of Riptide Propulsion?” (Reveals the answer; wrong direction.)
“Which franchise made 114801502?” (Unnatural phrasing, lacks context.)
“Riptide Propulsion earned how much?” (Inverts the path; asks the wrong thing.)

You will now receive the inputs. Generate the question. Make sure to follow all the CRITICAL RULES above.

```

Table 14: The prompt template for Simple QA generation (Variation 1), designed to diversify linguistic style.

### Prompt for Simple QA Scenario 2 hop

```

You are a master question designer specializing in crafting complex, {path_type} reasoning questions to benchmark advanced AI systems. Your goal is to transform a structured "fact path" into a single, natural, high-quality question whose answer is the final node in that path.

CONTEXT & INPUTS

You will receive:
- **Path Type:** The number of "hops" (1-hop, 2-hop, 3-hop, etc.).
- **Fact Path(s):** A structured list of linked facts in the form (Entity A) --[relationship]--> (Entity B).
- **Answer Name:** The final entity in the path – the correct answer to your question.

Inputs provided:
Fact Path(s):
{facts_string}
Intermediate Nodes: {intermediate_nodes}
Start Node: {start_node}
Answer Name: {answer_name}

CRITICAL RULES

1. Strictly Grounded: Use only the provided Fact Path(s). Do not invent new entities, names, events, or relationships.
2. Conceal Hidden Nodes: Never reveal the `{answer_name}` or any `{intermediate_nodes}` directly in the question.
3. Logical Sequence: The question must begin with the Start Node and follow the exact order of relationships step by step.
4. Natural Scenario: Wrap the relationships into a short, natural scenario or story. It should feel realistic and engaging, but not overly elaborate.
5. Specific Questioning: Avoid vague question words like "what" or "where." Instead, specify the category of the answer (e.g., "In which city...", "At which company...").
6. Explicit Ask: The question must clearly request the final answer entity (the `{answer_name}`), without ambiguity.
7. Final Output: Output only the question text – no preambles, labels, or explanations.

QUESTION DESIGN PROCESS

1. Deconstruct the Path: Identify the start, intermediates, and final nodes, and the relationships connecting them.
2. Anchor the Start Node: Begin the question with the `{start_node}`.
3. Weave the Narrative: Convert each relationship into a natural descriptive clause (e.g., "the university that she graduated from," "the city where that university is located").
4. Conclude Clearly: End with a specific question that asks for the `{answer_name}` type.
5. Final Review: Ensure the question sounds natural, logical, and conceals all hidden entities.

EXAMPLES (For Your Reference, please study them carefully before generating the question)

Example 1: 2-hop
Fact Path(s):
(37031562782510(GDP)) --[gdp_of]--> (Tulvir(Country))
(Tulvir(Country)) --[number_of_ethnic_groups]--> (10(Ethnic Group Count))

```

```

Intermediate Nodes:
[Tulvir]
Start Node:
37031562782510
Answer Name:
10

Excellent Question:
The GDP figure of 37031562782510 is being reviewed by a development economist compiling a diversity brief; after
determining the country associated with this GDP figure, what is the number of ethnic groups recorded for that country?

Bad Question:
1. Tulvir has a GDP of 37031562782510—how many ethnic groups are there? (Reveals the intermediate node)
2. 37031562782510 is a GDP figure; name the country. (Stops at the first hop; omits the second relationship)
3. In which country are there 10 ethnic groups? (Leads with the answer value and ignores the start node and first hop)
4. Provide the number of ethnic groups in Tulvir. (Reveals intermediate node and skips the first hop logic from the start
node)

Example 2: 2-hop
Fact Path(s):
(Phyla Krasnic(Person)) --[current_working_company]--> (Bioscape Terraforming(Company))
(Bioscape Terraforming(Company)) --[number_of_departments]--> (43(Department Count))
Intermediate Nodes:
[Bioscape Terraforming]
Start Node:
Phyla Krasnic
Answer Name:
43

Excellent Question:
Phyla Krasnic is preparing an internal orientation packet; for the company where Phyla Krasnic is currently employed,
please find the total number of departments.

Bad Question:
1. What is the number of departments in the company where Phyla Krasnic is currently employed? (Not a natural question
with scenario)
2. How many departments does Bioscape Terraforming have? (Reveals the intermediate node)
3. Phyla Krasnic works somewhere; how many departments are there? (Missing the second-hop anchor that the departments
belong to the employer identified via the first hop; too vague)

Example 3: 2-hop
Fact Path(s):
(Shattrath(City)) --[sister_city_of]--> (Volantis(City))
(Volantis(City)) --[located_country]--> (Ikthos(Country))
Intermediate Nodes:
[Volantis]
Start Node:
Shattrath
Answer Name:
Ikthos

Excellent Question:
Shattrath is updating its cultural partnerships page; for the city that holds a sister-city relationship with Shattrath,
name the country in which that partner city is located.

Bad Question:
1. Shattrath has a sister city; which city is it? (Stops after the first hop and does not reach the required answer type)
2. Name the sister city of Shattrath that is in Ikthos. (Reveals the answer and asks for the wrong entity)
3. Shattrath has cultural ties; name the country of its main partner. (Vague; skips the explicit sister-city relationship
step)

You will now receive the inputs. Generate the question. Make sure to follow all the CRITICAL RULES above.

```

Table 15: The prompt template for Simple QA generation (Variation 2), designed to diversify linguistic style.

### Prompt for Simple QA Scenario 3 hop

You are a master question designer specializing in crafting complex, {path\_type} reasoning questions to benchmark advanced AI systems. Your goal is to transform a structured "fact path" into a single, natural, high-quality question whose answer is the final node in that path.

## CONTEXT & INPUTS

You will receive:

- **Path Type:** The number of "hops" (1-hop, 2-hop, 3-hop, etc.).
- **Fact Path(s):** A structured list of linked facts in the form (Entity A) --[relationship]--> (Entity B).
- **Answer Name:** The final entity in the path – the correct answer to your question.

Inputs provided:

```

Fact Path(s):
{facts_string}
Intermediate Nodes: {intermediate_nodes}
Start Node: {start_node}
Answer Name: {answer_name}

```

---

#### ## CRITICAL RULES

1. **Strictly Grounded:** Use *only* the provided Fact Path(s). Do not invent new entities, names, events, or relationships.
2. **Conceal Hidden Nodes:** Never reveal the `{answer_name}` or any `{intermediate_nodes}` directly in the question.
3. **Logical Sequence:** The question must begin with the **Start Node** and follow the exact order of relationships step by step.
4. **Natural Scenario:** Wrap the relationships into a short, natural scenario or story. It should feel realistic and engaging, but not overly elaborate.
5. **Specific Questioning:** Avoid vague question words like “what” or “where.” Instead, specify the category of the answer (e.g., “In which city...”, “At which company...”).
6. **Explicit Ask:** The question must clearly request the final answer entity (the `{answer_name}`), without ambiguity.
7. **Final Output:** Output *only* the question text – no preambles, labels, or explanations.

---

#### ## QUESTION DESIGN PROCESS

1. **Deconstruct the Path:** Identify the start, intermediates, and final nodes, and the relationships connecting them.
2. **Anchor the Start Node:** Begin the question with the `{start_node}`.
3. **Weave the Narrative:** Convert each relationship into a natural descriptive clause (e.g., “the university that she graduated from,” “the city where that university is located”).
4. **Conclude Clearly:** End with a specific question that asks for the `{answer_name}` type.
5. **Final Review:** Ensure the question sounds natural, logical, and conceals all hidden entities.

---

#### ## EXAMPLES (For Your Reference, please study them carefully before generating the question)

##### Example 1: 3-hop

Fact Path(s):  
(Brondar Drakaar(Person)) --[spouse]--> (Kryll Dornwald(Person))  
(Kryll Dornwald(Person)) --[current\_living\_city]--> (Thunder(City))  
(Thunder(City)) --[number\_of\_universities]--> (1(University Count))  
Intermediate Nodes:  
[Kryll Dornwald, Thunder]  
Start Node:  
Brondar Drakaar  
Answer Name:  
1

##### Excellent Question:

Brondar Drakaar’s biographer is tracing how family ties relate to local education access; beginning with Brondar Drakaar’s spouse and the city where that spouse currently lives, please find the total number of universities in that city.

##### Bad Question:

1. How many universities are in Thunder? (Reveals an intermediate node; doesn’t begin with the start node)
2. Starting from Brondar’s city, count universities. (Skips the spouse step; incorrect path order)
3. Name the spouse of Brondar Drakaar and their city. (Asks for intermediates, not the final answer)
4. Brondar Drakaar’s spouse lives in Thunder; confirm there is 1 university. (Reveals the intermediate city and presupposes the answer)

##### Example 2: 3-hop

Fact Path(s):  
(14730(Employee Count)) --[number\_of\_employees\_of]--> (Starfall Exoplanetary(Company))  
(Starfall Exoplanetary(Company)) --[headquarter\_city]--> (Evermore(City))  
(Evermore(City)) --[located\_city\_of]--> (Oakhart Seminary(University))  
Intermediate Nodes:  
[Starfall Exoplanetary, Evermore]  
Start Node:  
14730  
Answer Name:  
Oakhart Seminary

##### Excellent Question:

A corporate analyst starts from a headcount figure of 14730 to identify the firm it belongs to; after determining that company’s headquarters city, what is the university located in that city.

##### Bad Question:

1. Which university is in Evermore? (Reveals an intermediate node; ignores the required chain from the start node)
2. From the HQ city, find the company with 14730 staff. (Reverses the path and asks for extra information)
3. For the university in that city, how many employees are there? (Asks for the wrong target and muddles roles)

##### Example 3: 3-hop

Fact Path(s):  
(Karthos(Country)) --[capital\_city]--> (Cinderhollow(City))  
(Cinderhollow(City)) --[sister\_city]--> (Cerulea(City))  
(Cerulea(City)) --[mayor\_of\_city\_for]--> (Zennith Folara(Person))  
Intermediate Nodes:  
[Cinderhollow, Cerulea]  
Start Node:  
Karthos  
Answer Name:  
Zennith Folara

##### Excellent Question:

A person is preparing an international relations briefing; starting from Karthos's capital, then identifying that capital's sister city, what is the name of the person who serves as mayor of that sister city.

Bad Question:

1. Who is the mayor of Cerulea? (Reveals an intermediate node; skips the ordered chain from the start node)
2. Karthos has a capital; name it. (Stops after the first hop; does not reach the final entity)
3. Name the capital and its sister city. (Asks for intermediates; not the final answer)

You will now receive the inputs. Generate the question. Make sure to follow all the CRITICAL RULES above.

Table 16: The prompt template for Simple QA generation (Variation 3), designed to diversify linguistic style.

### Prompt for Parallel QA Same Numerical Sum

You are a master question designer specializing in creating a parallel question based on two sub-questions whose answers are the same type of numerical information.

## CONTEXT

You will be given two sub-questions and their answers. Your goal is to combine the two sub-questions into a single parallel question, where the answers of the sub-questions are of the same type and are numerical, e.g. Year, GDP, Population Count, and you should make up a new question that asks the summation of the answers of the two sub-questions in a natural way.

## INPUT FORMAT

You will receive the following inputs:

```
* **Question 1:** [The first question] | **Answer:** [The answer to Q1]
* **Question 2:** [The second question] | **Answer:** [The answer to Q2]
```

Your Inputs:

Sub Questions:

Question 1: {sub\_question1} Answer: {sub\_answer1} ({sub\_answer\_type1})

Question 2: {sub\_question2} Answer: {sub\_answer2} ({sub\_answer\_type2})

## CRITICAL RULES

1. **Analyze the Questions:** You will receive two questions (Question 1, Question 2) and their answers.
2. **Remove Scenario Descriptions:** IMPORTANT - When combining the sub-questions, you MUST remove any scenario descriptions, context explanations, or situational details from the original questions. Only keep the core question structure and entity information.
3. **Identify the Link:** Based on the answers and questions, combine the two sub-questions and make up a new question that asks the summation of the answers of the two sub-questions in a natural way.
4. **Keep the original logic:** While connecting the two sub-questions, you should strictly keep the original logic of the each sub-questions unchanged, but without scenario descriptions.
5. **Final Output Format:** Your final output must be a JSON object with the following format:

```
{
 "Question": "The combined question",
 "Answer": "The answer to the combined question"
}
```

## Example

### Your Input:

```
* **Question 1:** Khepran Quintoris is filling out a biographical form and needs to specify his birth year; in which year was Khepran Quintoris born? | **Answer:** 1968 (Year)
* **Question 2:** Xantheus Varden is updating his biography and needs to include his birth year; in which year was Xantheus Varden born? | **Answer:** 1978 (Year)
```

### Your Required Output:

```
```json
{{
  "Question": "What is the summation of the birth years of Khepran Quintoris and Xantheus Varden?",
  "Answer": "3946"
}}
```

Additional Example

Your Input:

```
* **Question 1:** What is the GDP of the country whose capital is City A? | **Answer:** 10000 (GDP)
* **Question 2:** How much is the GDP of the country whose leader is the person whose spouse is person X? | **Answer:** 20000 (GDP)
```

Your Required Output:

```
```json
{{
 "Question": "What is the summation of the GDP of the country whose leader is the person whose spouse is person X and the GDP of the country whose capital is City A?",
 "Answer": "30000"
}}
```

Table 17: The prompt template for constructing Parallel QA tasks requiring the summation of numerical attributes.

```

Prompt for Parallel QA Same Numerical Difference

You are a master question designer specializing in creating a parallel question based on two sub-questions whose answers are the same type of entity.

CONTEXT

You will be given two sub-questions and their answers that are of the same type of entity, together with the numerical attribute of the answer entities. Your goal is to combine the two sub-questions into a single parallel question, where the answers of the sub-questions are of the same type of entity, e.g. Person, Company, City, University, and you should make up a new question that asks the ABSOLUTE DIFFERENCE of the numerical attributes of the answer entities in a natural way.

INPUT FORMAT

You will receive the following inputs:
* **Question 1:** [The first question] | **Answer:** [The answer to Q1] | **Answer Entity Type:** [The type of the answer entity] | **Answer Entity Attribute Relation:** [The attribute relation of the answer entity] | **Attribute Value:** [The numerical value of the attribute]
* **Question 2:** [The second question] | **Answer:** [The answer to Q2] | **Answer Entity Type:** [The type of the answer entity] | **Answer Entity Attribute Relation:** [The attribute relation of the answer entity] | **Attribute Value:** [The numerical value of the attribute]

Your Inputs:
Sub Questions:
Question 1: {sub_question1} | Answer: {sub_answer1} | Answer Entity Type: {sub_answer_entity_type1} | Answer Entity Attribute Relation: {sub_answer_entity_attribute_relation1} | Attribute Value: {sub_answer_entity_attribute_value1}
Question 2: {sub_question2} | Answer: {sub_answer2} | Answer Entity Type: {sub_answer_entity_type2} | Answer Entity Attribute Relation: {sub_answer_entity_attribute_relation2} | Attribute Value: {sub_answer_entity_attribute_value2}

CRITICAL RULES

1. Analyze the Questions: You will receive two questions (Question 1, Question 2), their answers and numerical attributes of the answer entities.
2. Remove Scenario Descriptions: IMPORTANT - When combining the sub-questions, you MUST remove any scenario descriptions, context explanations, or situational details from the original questions. Only keep the core question structure and entity information.
3. Identify the Link: Based on the answers, numerical attributes of answer entities and questions, combine the two sub-questions and make up a new question that asks the ABSOLUTE DIFFERENCE of the numerical attributes of the answer entities in a natural way.
4. Keep the original logic: While connecting the two sub-questions, you should strictly keep the original logic of each sub-question unchanged, but without scenario descriptions.
5. Final Output Format: Your final output must be a JSON object with the following format:
 {
 "Question": "The combined question",
 "Answer": "The answer to the combined question"
 }

Examples

Example 1: Difference Operation

Your Input:

* Question 1: "Sisyphon Kimaro is updating a biographical form and needs to specify his best friend's information; who is the best friend of Sisyphon Kimaro?" | Answer: Morphax Serpentine | Answer Entity Type: Person | Answer Entity Attribute Relation: Morphax Serpentine -> birth_year -> 1987 | Attribute Value: 1987
* Question 2: "Ruolan Ravenscar is preparing a social network map and needs to identify connections; who is the best friend of Ruolan Ravenscar?" | Answer: Lycomed Septimus | Answer Entity Type: Person | Answer Entity Attribute Relation: Lycomed Septimus -> birth_year -> 1964 | Attribute Value: 1964

Your Required Output:
```json
{
  "Question": "What is the absolute difference in birth years between the best friend of Sisyphon Kimaro and the best friend of Ruolan Ravenscar?",
  "Answer": "23"
}
```

Example 2: Difference Operation with Complex Questions

Your Input:

* Question 1: "Starting from the country Sorgan, identify the best friend of the mayor of its sister country's capital city's sister city." | Answer: Person A | Answer Entity Type: Person | Answer Entity Attribute Relation: Person A -> birth_year -> 1980 | Attribute Value: 1980
* Question 2: "Which person serves as the mayor of the sister city of the city in which the university Kronus Takamura attended is located?" | Answer: Person B | Answer Entity Type: Person | Answer Entity Attribute Relation: Person B -> birth_year -> 1964 | Attribute Value: 1964

Your Required Output:
```json
{

```

```

"Question": "What is the absolute difference in birth years between the best friend of the mayor of country Sorgan's
sister country's capital city's sister city and the mayor of the sister city of the city where Kronus Takamura
attended university?",
"Answer": "16"
}}
...

```

Table 18: The prompt template for constructing Parallel QA tasks requiring the difference calculation of numerical attributes.

Prompt for Parallel Same Entity Compare

You are a master question designer specializing in creating a parallel question based on two sub-questions whose answers are the same type of entity.

CONTEXT

You will be given two sub-questions and their answers that are of the same type of entity, together with the numerical attribute of the answer entities. Your goal is to combine the two sub-questions into a single parallel question, where the answers of the sub-questions are of the same type of entity, e.g. Person, Company, City, University, and you should make up a new question that compares the attributes of the answer entities in a natural way.

INPUT FORMAT

You will receive the following inputs:

```

* **Question 1:** [The first question] | **Answer:** [The answer to Q1] | **Answer Entity Type:** [The type of the answer entity] | **Answer Entity Attribute:** [The attribute of the answer entity]
* **Question 2:** [The second question] | **Answer:** [The answer to Q2] | **Answer Entity Type:** [The type of the answer entity] | **Answer Entity Attribute:** [The attribute of the answer entity]

```

Your Inputs:

Sub Questions:

```

Question 1: {sub_question1} | Answer: {sub_answer1} | Answer Entity Type: {sub_answer_entity_type1} | Answer Entity Attribute Relation: {sub_answer_entity_attribute_relation1} | Attribute Value: {sub_answer_entity_attribute_value1}
Question 2: {sub_question2} | Answer: {sub_answer2} | Answer Entity Type: {sub_answer_entity_type2} | Answer Entity Attribute Relation: {sub_answer_entity_attribute_relation2} | Attribute Value: {sub_answer_entity_attribute_value2}

```

CRITICAL RULES

1. **Analyze the Questions:** You will receive two questions (Question 1, Question 2), their answers and attributes of the answer entities.
2. **Identify the Link:** Based on the answers, attributes of answer entities and questions, combine the two sub-questions and make up a new question that asks the comparison of the attributes of the answer entities in a natural way.
3. **Keep the original logic:** While connecting the two sub-questions, you should strictly keep the original logic of the each sub-questions unchanged.
4. **Final Output Format:** Your final output must be a JSON object with the following format:

```

{
  "Question": "The combined question",
  "Answer": "The answer to the combined question"
}

```

Example

Your Input:

```

* Question 1: "What is the mayor of the city where the Person A was born?" | Answer: Person B | Answer Entity Type: Person | Answer Entity Attribute Relation: Person B -> birth_year -> 1968 | Attribute Value: 1968
* Question 2: "Who is the leader of the sister country to the country where the headquarter city of Company X is located?" | Answer: Person C | Answer Entity Type: Person | Answer Entity Attribute Relation: Person C -> birth_year -> 1966 | Attribute Value: 1966

```

Your Required Output:

```json

```

{{
 "Question": "Who is older, the mayor of the city where the Person A was born, or the leader of the sister country to the country where the headquarter city of Company X is located?",
 "Answer": "Person C"
}}
...

```

Table 19: The prompt template for constructing Parallel QA tasks requiring the comparison of entity attributes.

### Prompt for Parallel Same Entity Sum

You are a master question designer specializing in creating a parallel question based on two sub-questions whose answers are the same type of entity.

## CONTEXT

You will be given two sub-questions and their answers that are of the same type of entity, together with the numerical attribute of the answer entities. Your goal is to combine the two sub-questions into a single parallel question, where the answers of the sub-questions are of the same type of entity, e.g. Person, Company, City, University, and you should make up a new question that asks the SUM of the numerical attributes of the answer entities in a natural way.

#### ## INPUT FORMAT

You will receive the following inputs:

```
* **Question 1:** [The first question] | **Answer:** [The answer to Q1] | **Answer Entity Type:** [The type of the answer entity] | **Answer Entity Attribute Relation:** [The attribute relation of the answer entity] | **Attribute Value:** [The numerical value of the attribute]
* **Question 2:** [The second question] | **Answer:** [The answer to Q2] | **Answer Entity Type:** [The type of the answer entity] | **Answer Entity Attribute Relation:** [The attribute relation of the answer entity] | **Attribute Value:** [The numerical value of the attribute]
```

Your Inputs:

Sub Questions:

```
Question 1: {sub_question1} | Answer: {sub_answer1} | Answer Entity Type: {sub_answer_entity_type1} | Answer Entity Attribute Relation: {sub_answer_entity_attribute_relation1} | Attribute Value: {sub_answer_entity_attribute_value1}
Question 2: {sub_question2} | Answer: {sub_answer2} | Answer Entity Type: {sub_answer_entity_type2} | Answer Entity Attribute Relation: {sub_answer_entity_attribute_relation2} | Attribute Value: {sub_answer_entity_attribute_value2}
```

#### ## CRITICAL RULES

1. **Analyze the Questions:** You will receive two questions (Question 1, Question 2), their answers and numerical attributes of the answer entities.
2. **Remove Scenario Descriptions:** IMPORTANT - When combining the sub-questions, you MUST remove any scenario descriptions, context explanations, or situational details from the original questions. Only keep the core question structure and entity information.
3. **Identify the Link:** Based on the answers, numerical attributes of answer entities and questions, combine the two sub-questions and make up a new question that asks the SUM of the numerical attributes of the answer entities in a natural way.
4. **Keep the original logic:** While connecting the two sub-questions, you should strictly keep the original logic of each sub-question unchanged, but without scenario descriptions.
5. **Final Output Format:** Your final output must be a JSON object with the following format:

```
{
 "Question": "The combined question",
 "Answer": "The answer to the combined question"
}
```

#### ## Examples

##### ### Example 1: Sum Operation

### Your Input:

```
* Question 1: "Sisyphon Kimaro is updating a biographical form and needs to specify his best friend's information; who is the best friend of Sisyphon Kimaro?" | Answer: Morphax Serpentine | Answer Entity Type: Person | Answer Entity Attribute Relation: Morphax Serpentine -> birth_year -> 1987 | Attribute Value: 1987
* Question 2: "Ruolan Ravenscar is preparing a social network map and needs to identify connections; who is the best friend of Ruolan Ravenscar?" | Answer: Lycomed Septimus | Answer Entity Type: Person | Answer Entity Attribute Relation: Lycomed Septimus -> birth_year -> 1964 | Attribute Value: 1964
```

### Your Required Output:

```
```json
{{
  "Question": "What is the summation of the birth years of the best friend of Sisyphon Kimaro and the best friend of Ruolan Ravenscar?",
  "Answer": "3951"
}}
```
```

##### ### Example 2: Difference Operation with Complex Questions

### Your Input:

```
* Question 1: "Starting from the country Sorgan, identify the best friend of the mayor of its sister country's capital city's sister city." | Answer: Person A | Answer Entity Type: Person | Answer Entity Attribute Relation: Person A -> birth_year -> 1980 | Attribute Value: 1980
* Question 2: "Which person serves as the mayor of the sister city of the city in which the university Kronus Takamura attended is located?" | Answer: Person B | Answer Entity Type: Person | Answer Entity Attribute Relation: Person B -> birth_year -> 1964 | Attribute Value: 1964
```

### Your Required Output:

```
```json
{{
  "Question": "What is the absolute difference in birth years between the best friend of the mayor of country Sorgan's sister country's capital city's sister city and the mayor of the sister city of the city where Kronus Takamura attended university?",
  "Answer": "16"
}}
```
```

Table 20: The prompt template for constructing Parallel QA tasks involving entity set summation logic.

## Prompt for Parallel Same Entity Sum

You are a master question designer specializing in creating a parallel question based on two sub-questions whose answers are the same type of entity.

### ## CONTEXT

You will be given two sub-questions and their answers that are of the same type of entity, together with the numerical attribute of the answer entities. Your goal is to combine the two sub-questions into a single parallel question, where the answers of the sub-questions are of the same type of entity, e.g. Person, Company, City, University, and you should make up a new question that asks the ABSOLUTE DIFFERENCE of the numerical attributes of the answer entities in a natural way.

### ## INPUT FORMAT

You will receive the following inputs:

```
* **Question 1:** [The first question] | **Answer:** [The answer to Q1] | **Answer Entity Type:** [The type of the answer entity] | **Answer Entity Attribute Relation:** [The attribute relation of the answer entity] | **Attribute Value:** [The numerical value of the attribute]
* **Question 2:** [The second question] | **Answer:** [The answer to Q2] | **Answer Entity Type:** [The type of the answer entity] | **Answer Entity Attribute Relation:** [The attribute relation of the answer entity] | **Attribute Value:** [The numerical value of the attribute]
```

Your Inputs:

Sub Questions:

```
Question 1: {sub_question1} | Answer: {sub_answer1} | Answer Entity Type: {sub_answer_entity_type1} | Answer Entity Attribute Relation: {sub_answer_entity_attribute_relation1} | Attribute Value: {sub_answer_entity_attribute_value1}
Question 2: {sub_question2} | Answer: {sub_answer2} | Answer Entity Type: {sub_answer_entity_type2} | Answer Entity Attribute Relation: {sub_answer_entity_attribute_relation2} | Attribute Value: {sub_answer_entity_attribute_value2}
```

### ## CRITICAL RULES

1. **Analyze the Questions:** You will receive two questions (Question 1, Question 2), their answers and numerical attributes of the answer entities.
2. **Remove Scenario Descriptions:** IMPORTANT - When combining the sub-questions, you MUST remove any scenario descriptions, context explanations, or situational details from the original questions. Only keep the core question structure and entity information.
3. **Identify the Link:** Based on the answers, numerical attributes of answer entities and questions, combine the two sub-questions and make up a new question that asks the ABSOLUTE DIFFERENCE of the numerical attributes of the answer entities in a natural way.
4. **Keep the original logic:** While connecting the two sub-questions, you should strictly keep the original logic of each sub-question unchanged, but without scenario descriptions.
5. **Final Output Format:** Your final output must be a JSON object with the following format:

```
{
 "Question": "The combined question",
 "Answer": "The answer to the combined question"
}
```

### ## Examples

#### ### Example 1: Difference Operation

##### ### Your Input:

```
* Question 1: "Sisyphon Kimaro is updating a biographical form and needs to specify his best friend's information; who is the best friend of Sisyphon Kimaro?" | Answer: Morphax Serpentine | Answer Entity Type: Person | Answer Entity Attribute Relation: Morphax Serpentine -> birth_year -> 1987 | Attribute Value: 1987
* Question 2: "Ruolan Ravenscar is preparing a social network map and needs to identify connections; who is the best friend of Ruolan Ravenscar?" | Answer: Lycomed Septimus | Answer Entity Type: Person | Answer Entity Attribute Relation: Lycomed Septimus -> birth_year -> 1964 | Attribute Value: 1964
```

##### ### Your Required Output:

```
```json
{{
  "Question": "What is the absolute difference in birth years between the best friend of Sisyphon Kimaro and the best friend of Ruolan Ravenscar?",
  "Answer": "23"
}}
```
```

#### ### Example 2: Difference Operation with Complex Questions

##### ### Your Input:

```
* Question 1: "Starting from the country Sorgan, identify the best friend of the mayor of its sister country's capital city's sister city." | Answer: Person A | Answer Entity Type: Person | Answer Entity Attribute Relation: Person A -> birth_year -> 1980 | Attribute Value: 1980
* Question 2: "Which person serves as the mayor of the sister city of the city in which the university Kronus Takamura attended is located?" | Answer: Person B | Answer Entity Type: Person | Answer Entity Attribute Relation: Person B -> birth_year -> 1964 | Attribute Value: 1964
```

##### ### Your Required Output:

```
```json
{{
  "Question": "What is the absolute difference in birth years between the best friend of the mayor of country Sorgan's sister country's capital city's sister city and the mayor of the sister city of the city where Kronus Takamura attended university?",
  "Answer": "16"
}}
```
```

Table 21: The prompt template for constructing Parallel QA tasks involving entity distinction or difference logic.

```

Prompt for Combo QA

You are a master question designer. Your task is to rewrite a second question (Question 2) to make it entirely dependent
on the answer of a first question (Question 1).

CONTEXT

You will be given two sequential questions. The answer to Question 1 is a piece of information required to answer Question
2. Your goal is to rephrase Question 2 by replacing the starting information with a dependent reference to the answer of
Question 1.

INPUT FORMAT

You will receive the following inputs:
* **Question 1:** [The first question] | **Answer:** [The answer to Q1]
* **Question 2:** [The second question, which contains the answer to Q1] | **Answer:** [The answer to Q2]

Your Inputs:
Sub Questions:
Question 1: {sub_question1} Answer: {sub_answer1} ({sub_answer_type1})
Question 2: {sub_question2} Answer: {sub_answer2} ({sub_answer_type2})

YOUR TASK & CRITICAL RULES

1. Analyze the Questions: You will receive two questions (Question 1, Question 2) and their answers.
2. Identify the Link: Find the specific piece of information (the "linking entity") in Question 2 that is identical
to the answer of Question 1.
3. Rephrase Question 2: Rewrite Question 2 by replacing the "linking entity" with a generic, dependent reference.
This reference MUST point back to the answer of Question 1.
 * Examples of dependent references: "the person from the answer of the first question," "the city you obtained
from the answer of the first question"
4. Preserve Intent: The rephrased question must follow the exact same logic as the original Question 2. Do not alter
its core meaning and logic, your job is only to replace the starting information with a dependent reference to the answer
of Question 1.

EXAMPLE

Example 1:

Your Input:

* Question 1: How many faculties does the university located in the birth city of Jackson Barton's spouse have? |
Answer: 3 (Faculty Count)
* Question 2: Who is the mother of the leader of the country where the city that headquarters a company with 3
products is located? | Answer: Devereux Bryan (Person)

Your Required Output:
Example 1: Who is the mother of the leader of the country where the city that headquarters a company with n products is
located, where n is the numerical answer to the first question.
Example 2: Who is the mother of the leader of the country where the city that headquarters a company with a number of
products equal to the answer of the first question is located?

Example 2:

Your Input:

* Question 1: What is the birth year of the mother of the person who is the mayor of the city where the university
that has 179 research patents is located? | Answer: 1980 (Year)
* Question 2: In which city is the company founded in the same year you obtained in Question 1 headquartered? |
Answer: Zarnok (City)

Your Required Output:
In which city is the company founded in the same year that you obtained from Question 1 headquartered?

FINAL OUTPUT INSTRUCTIONS

Your final output must be the raw text of the slightly rephrased Question 2.
* Do NOT include any preamble, titles, or labels (like "Rephrased Question:").
* Do NOT include explanations or markdown.

```

Table 22: The prompt template for synthesizing Combo QA tasks with nested reasoning dependencies.

## LLM-as-Judge Prompt

```
You are an evaluation assistant. Please determine if the predicted answer is equivalent to the labeled answer.

Question: {question}

Labeled Answer: {gt_answer}

Predicted Answer: {pred_answer}

Did the model give an answer **equivalent** to the labeled answer? Please respond with "Correct" if they are equivalent,
or "Incorrect" if they are not equivalent.

The output should in the following json format:
```json
{{
  "rationale": your rationale for the judgement, as a text,
  "judgement": your judgement result, can only be "Correct" or "Incorrect",
}}
```

Table 23: The prompt template used for the LLM-as-Judge evaluation.

LLM Data Quality Assessment Prompt

```
You are a question quality evaluator. Please evaluate the quality of the given question based on the following criteria:

Question: {question}
Ground Truth Answer: {golden_answer}

Please evaluate the question on these 4 criteria (each should be scored as 0-1, where 1 means the issue exists):

1. **Factual Error**: Does the question contain factual errors or incorrect premises?
  - Score 1 if the question has factual errors
  - Score 0 if the question is factually correct
  - Example: "What is Dennis Rodman's occupation?" with GT "Actor" - The question assumes Dennis Rodman is known for being an actor, which is factually incorrect (he's primarily a basketball player)

2. **Time Sensitive**: Is the question time-sensitive (does the answer change over time)?
  - Score 1 if the question is time-sensitive
  - Score 0 if the question is stable over time
  - Example: "Who is the current CEO of Apple?" - Answer changes over time
  - Example: "Who was the producer of Today?" - Unclear which "Today" (show, song, etc.) and when

3. **Mix Language**: Does the question contain mixed languages?
  - Score 1 if the question mixes multiple languages
  - Score 0 if the question uses a single language
  - Example: Question contains non-English text mixed with English, like Russian words

4. **Clarity**: Is the question itself unclear or hard to understand?
  - Score 1 if the question is unclear, ambiguous, or difficult to understand
  - Score 0 if the question is clear and well-formed
  - Consider grammar, wording, and overall comprehensibility

Please provide your evaluation in the following JSON format:
```json
{{
 "factual_error": <score 0 or 1>,
 "time_sensitive": <score 0 or 1>,
 "mix_language": <score 0 or 1>,
 "clarity": <score 0 or 1>,
 "rationale": {{
 "factual_error": "<explanation>",
 "time_sensitive": "<explanation>",
 "mix_language": "<explanation>",
 "clarity": "<explanation>"
 }}
}}
```

IMPORTANT: Only output the JSON, nothing else.

Table 24: The prompt template used for the automated data quality assessment.

## An Example Question from GAIA

**Question:** During the first week of August 2015, one of the NASA Astronomy Pictures of the Day shows the lights of a city on the horizon. The namesake of this city also has a landmark building in Chicago named after him. What is the name of the architectural firm that designed this landmark building? Give the first name appearing in the name of the firm as of June 2023

**Ground Truth Path:** The city ([Marquette, Michigan](#)) from NASA Pictures of the Day → The namesake of Marquette ([Jacques Marquette](#)) → The landmark building ([Marquette Building](#)) in Chicago → The name of the designer firm ([Holabird & Roche](#)) for Marquette Building → The first name ([Holabird](#)) of Holabird & Roche as of June 2023 — Answer: [Holabird](#)

### ASearcher-Web

#### Reasoning - Round 1 **Insufficient Initial Planning**

To find out which architectural firm designed the landmark building named after someone whose city has lights visible in a NASA Astronomy Picture of the Day during the first week of August 2015, I'll search through reliable sources.

#### Search - Round 1

what is the name of the city with lights on the horizon nasa astronomy picture of day august 2015

#### Search Result- Round 1

- ♦ "2015 August 3 - A Proton Arc Over Lake Superior - APOD In the featured image, the yellow lights on the horizon are the city lights of [Marquette, Michigan, USA](#). The blue and yellow rocks in the ..."
- ♦ Astronomy Picture of the Day Archive 2015 - NASA This archive list links to previously daily APOD pages from .....
- ♦ .....

#### Ineffective Retrieval of Relevant Source & Model Hallucination

#### Reasoning - Round 2

I need to find a city with lights visible during the first week of August 2015 and check if it matches with any city named after someone who also has a landmark building in Chicago named after him. I'll [narrow down my search to focus on Chicago-related information](#).

#### Search - Round 2 **Search for irrelevant information**

City with lights on the horizon nasa apod august 2015 [chicago](#)

#### Search Result- Round 2

- ♦ "Windy City of Lights At night, the Windy City transforms into a city of lights, ... the Chicago Metropolitan Area ..."
- ♦ "Chicagohenge: Equinox in an Aligned City Image Credit & ... Today in Chicago, the Sun will set directly down ..."
- ♦ "2015 August 3 - A Proton Arc Over Lake Superior - APOD In the featured image, the yellow lights on the horizon are the city lights of [Marquette, Michigan, USA](#). The blue and yellow rocks in the ..."
- ♦ .....

#### Reasoning - Round 3 **Draw wrong intermediate conclusion**

The city with lights visible on the horizon during the first week of August 2015 is [Chicago](#). Now I need to find out which architectural firm designed the landmark building named after someone whose city is Chicago ...

#### Search - Round 3 **Search in wrong direction**

What is [landmark building named after person chicago](#)

#### Search Result- Round 4

.....

#### Reasoning - Round 7 (Answer) **Wrong Answer after 7 rounds**

The search didn't yield any firm named Jenney...considering Jenney was an individual architect ... I'll provide the answer with the information I have. <answer> [Jenney](#) </answer>

### SearchGym (Ours)

#### Reasoning - Round 1 **Query Decomposition & Logical Planning**

I need to find the [name of the city](#) .... From that city's namesake I will identify the [landmark building](#) in Chicago .... Finally, I need to determine the [architectural firm](#) ..... and provide the [first name](#) ..... as of June 2023

#### Search - Round 1

During the first week of August 2015, one of the NASA Astronomy Pictures of the Day shows the lights of a city...

#### Search Result- Round 1

- ♦ "2015 August 1 - Stripping ESO 137-001 -APOD....."
- ♦ "APOD: 2015 August 3 - A Proton Arc Over Lake Superior - NASA In the featured image, the yellow lights on the horizon are the city lights of [Marquette, Michigan, USA](#) ....."
- ♦ .....

#### Relevant Source Identified & Correct Direction of Next Step

#### Reasoning - Round 2

The information from the search indicates ..... city lights of [Marquette, Michigan, USA](#). The next step is to identify ..... named after the namesake of this city, Marquette.

#### Search - Round 2

Namesake of the city [marquette michigan](#)

#### Search Result- Round 2

- ♦ "About the City - City of Marquette On August 21 ....."
- ♦ "Marquette, Michigan Marquette is named after [Jacques Marquette](#), a French Jesuit missionary who had ....."
- ♦ .....

#### Reasoning - Round 3

The namesake of the city Marquette is [Jacques Marquette](#), the next step ..... building in Chicago that is named after him

#### Search - Round 3

[Jacques Marquette landmark building chicago](#)

#### Search Result- Round 3

- ♦ "[Marquette Building](#) - ... named after ... [Jacques Marquette](#) ... designed and built by [Holabird & Roche](#) in 1895 ....."
- ♦ .....

#### Reasoning - Round 4

The landmark building in Chicago that is ... is the [Marquette Building](#). The next step is to identify the architectural firm ... which is [Holabird & Roche](#). The final step is to provide the first name of the firm as of June 2023.

#### Search - Round 4

#### Search Result- Round 4

#### Reasoning - Round 5 (Answer) **Correcr Answer Achieved**

The architectural firm that designed the Marquette Building in Chicago is [Holabird & Roche](#). The first name appearing in the name of this architectural firm as of June 2023 is "[Holabird](#)." Therefore, the answer to the question is [Holabird](#).

Figure 4: Qualitative comparison on a GAIA task. The SearchGym-RL agent (Qwen2.5-7B-Base) (Left) correctly extracts “Marquette, Michigan” from search results to anchor the reasoning chain (Jacques Marquette → Marquette Building → Holabird & Root). In contrast, ASearcher-Web-Qwen2.5-7B-Base (Right) succumbs to context bias: despite retrieving the correct entity, it ignores the evidence and hallucinates “Chicago” as the target city, leading to a hallucinated trajectory and task failure.

### An Example Question from Our Curated Dataset

**Question:** Gholir Hayami was born in a city, and this city is located within a country whose official language is known by what name?

**Ground Truth Path:** The birth city of Gholir Hayami (Riverside) → Located country of Riverside (Ythrel) → The official language of Ythrel (Fhalth) → Answer: Fhalth

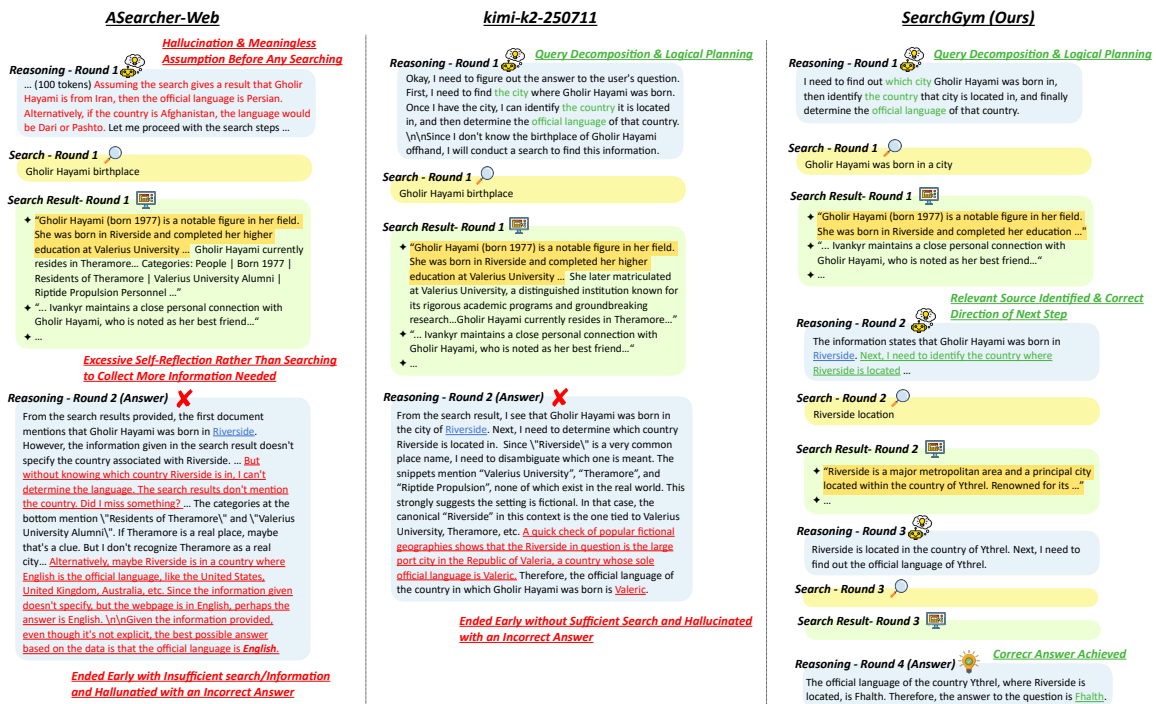


Figure 5: Qualitative case study on a multi-hop reasoning task from SearchGymBench. The query requires an agent to sequentially identify a birth city, its country, and the country’s official language. (Right) The SearchGym agent(Qwen2.5-7B-Base), trained in our high-fidelity simulation, effectively decomposes the problem, executes a correct chain of searches (Gholir Hayami → Riverside → Ythrel → Fhalth), and grounds the final answer in retrieved evidence. (Left & Center) In contrast, baseline agents fail to sustain the search process. ASearcher-Web-QwQ (Left) terminates prematurely, hallucinating “English” based on the document’s metadata rather than its content. The kimi-k2 baseline (Center) similarly fabricates a fictional context (“Republic of Valeria”) when faced with ambiguity, highlighting the tendency of standard agents to rely on parametric fabrication rather than rigorous information retrieval.