

AUTO TOOL: DYNAMIC TOOL SELECTION AND INTEGRATION FOR AGENTIC REASONING

Jiaru Zou^{1,2*}, Ling Yang^{1*†}, Yunzhe Qi², Sirui Chen², Mengting Ai², Ke Shen,
Jingrui He^{2†}, Mengdi Wang^{1†}

¹Princeton University ²University of Illinois Urbana-Champaign

* Equal Contribution † Corresponding Authors

ABSTRACT

Agentic reinforcement learning has advanced large language models (LLMs) to reason through long chain-of-thought trajectories while interleaving external tool use. Existing approaches assume a fixed inventory of tools, limiting LLM agents’ adaptability to new or evolving toolsets. We present AutoTool, a framework that equips LLM agents with dynamic tool-selection capabilities throughout their reasoning trajectories. We first construct a 200k dataset with explicit tool-selection rationales across 1,000+ tools and 100+ tasks spanning mathematics, science, code generation, and multimodal reasoning. Building on this data foundation, AutoTool employs a dual-phase optimization pipeline: (i) supervised and RL-based trajectory stabilization for coherent reasoning, and (ii) KL-regularized Plackett–Luce ranking to refine consistent multi-step tool selection. Across ten diverse benchmarks, we train two base models, Qwen3-8B and Qwen2.5-VL-7B, with AutoTool. With fewer parameters, AutoTool consistently outperforms advanced LLM agents and tool-integration methods, yielding average gains of 6.4% in math & science reasoning, 4.5% in search-based QA, 7.7% in code generation, and 6.9% in multimodal understanding. In addition, AutoTool exhibits stronger generalization by dynamically leveraging unseen tools from evolving toolsets during inference.

1 INTRODUCTION

Recent advances in reinforcement learning (RL) (Hu et al., 2024; Shao et al., 2024; Yu et al., 2025a) have improved the reasoning capabilities of large language models (LLMs) (Lu et al., 2025; Guo et al., 2025; Team et al., 2025), enabling them to generate long chain-of-thought (CoT) trajectories for complex tasks across domains such as visual question answering (Antol et al., 2015; Pramanick et al., 2025), knowledge retrieval (Yang et al., 2018; Su et al., 2025), and mathematical reasoning (Maxwell-Jia, 2024; Rein et al., 2024). Building on this, a growing line of research on agentic RL (Singh et al., 2025; Lu et al., 2025; Qian et al., 2025; Chen et al., 2025) explores how LLMs, beyond purely text-based model internal reasoning, can operate as agents that interleave their reasoning with multi-turn interactions in external tool environments (Mai et al., 2025; Wang et al., 2025b), such as search engines (Kong et al., 2023; Jin et al., 2025), code interpreters (Yue et al., 2023; Feng et al., 2025a), and vision tools (Su et al., 2025; Peng et al., 2025; Wu & Xie, 2024). The tool integration process reduces compounding errors in long CoT trajectories and enables more precise and reliable computation, which in turn strengthens the model’s reasoning and leads to consistent task performance (Zhang et al., 2023; Feng et al., 2025b).

Despite these advances, existing approaches typically operate under the assumption of *single-domain fixed tools* (Gao et al., 2025a): the policy model learns when and how to invoke the designated tools, but the available tools are predefined and static for a specific task. In practice, these methods curate training datasets that explicitly cover the utilization of this fixed tool inventory, and then apply advanced supervised fine-tuning (SFT) or RL techniques to align the model’s behavior with the prescribed tool usage (Feng et al., 2025a; Jin et al., 2025). While effective within closed environments, such designs fail to capture realistic scenarios where (i) an agent must select the appropriate tool from a *complex, domain-diverse toolset*, and (ii) *new tools* unseen during training stages may later be introduced and required at inference time, as illustrated in Figure 1. Without an explicit mechanism for *dynamic tool selection*, LLM agents risk overfitting to a closed tool inventory and fail to generalize in evolving or previously unseen tool environments. Addressing this gap requires moving beyond

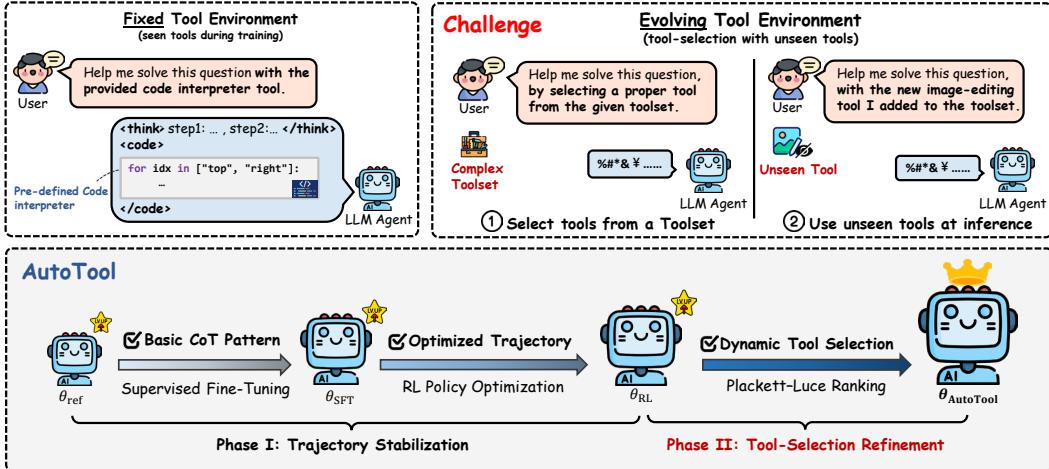


Figure 1: Illustration of fixed vs. (challenging) evolving tool environments. AutoTool enables LLM agents to dynamically select from complex unseen toolsets via a dual-phase optimization pipeline.

simple tool-use to explicitly training LLM agents to *adaptively choose tools from a dynamic and growing toolset during its trajectory generation process, while preserving strong reasoning ability*.

To this end, we introduce **AutoTool**, a framework that equips LLM agents with dynamic tool-selection capabilities throughout the reasoning process. AutoTool integrates reasoning and tool use into a unified trajectory, where the agent alternates between generating rationales and selecting tools from a large and evolving toolset. Specifically, we first construct a *200k dynamic tool-use dataset* through a dedicated curation pipeline, explicitly incorporating tool-selection rationales into the trajectory generation of LLM agents. Our curated dataset spans a diverse library of over 1,000 tools and more than 100 tasks across mathematics, science, code generation, and multimodal understanding.

Building on the data recipe, we further design a *dual-phase optimization pipeline* to train an LLM agent policy, as illustrated in Figure 1. In *Phase I*, we initialize the LLM agent with SFT followed by RL policy optimization, equipping the model with stable long CoT reasoning and trajectory generation abilities. In *Phase II*, we continue refining the LLM agent with a dedicated focus on tool selection, enabling the model to compose effective sequences of tool choices during its generation. To achieve this, we draw inspiration from the Plackett–Luce (PL) Ranking (Luce et al., 1959; Cheng et al., 2010) and frame tool-selection steps within model trajectories as a sequence ranking problem over the evolving toolset. We provide a theoretical bridge connecting the PL Ranking of tool-selection steps with the LLM agent’s policy optimization, and then optimize the LLM agent with a KL-regularized objective by minimizing a tractable policy-level Cross-Entropy (CE) loss. The optimization over tool preferences explicitly trains the LLM agent to favor more effective tool compositions over weaker alternatives, leading to effective and generalizable tool-selection strategies.

We conduct extensive experiments across ten diverse benchmarks, training both Qwen3-8B (Yang et al., 2025) and Qwen2.5-VL-7B (Bai et al., 2025) backbones with AutoTool. Incorporating AutoTool over previous training paradigms, such as SFT and GRPO, further boosts performance by an average of 6.4% in math & science, 4.5% in search, 7.7% in code generation, and 6.9% in multimodal understanding. Additionally, while advanced LLM agents or specialized tool-integration methods typically excel within fixed domains, they often fail to transfer their tool-usage abilities to other tasks. In contrast, AutoTool consistently demonstrates strong generalizability across all downstream tasks by leveraging dynamic tool-selection ability.

2 DATA CURATION FOR TOOL-SELECTION

Since existing training data (Yu et al., 2025a; Feng et al., 2025a) and downstream tasks (Qian et al., 2025; Wang et al., 2025a) for LLM agents prescribe fixed tool usage tailored to specific tasks, they cannot be directly applied to train or evaluate tool selection setups. To address this limitation, we extend these data recipes to curate a 200k dataset that explicitly integrates tool selection into the reasoning trajectory. In particular, we design a data curation pipeline that simulates real-world scenarios of agentic tool selection and reasoning across diverse domains, including mathematical and

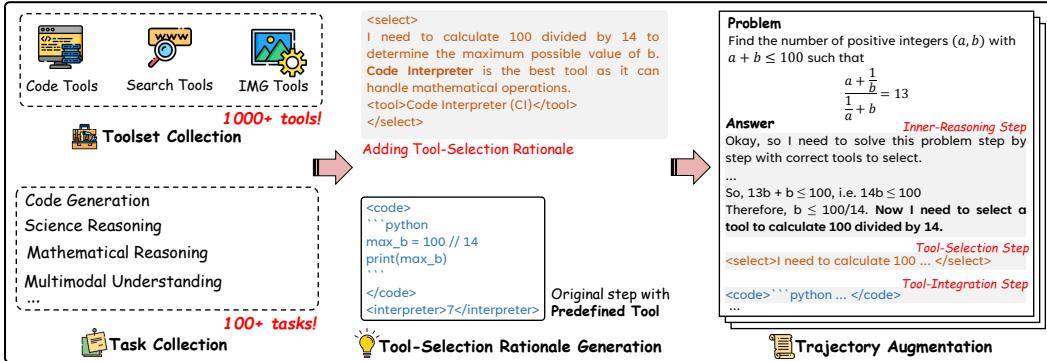


Figure 2: **Data curation pipeline for AutoTool** (Detailed in Section 2). The overall pipeline has three stages: (i) **Toolset & Task Collection**, assembling 1,000+ tools with metadata across 100+ tasks in math, science, code, and multimodal reasoning; (ii) **Tool-Selection Rationale Generation**, producing explicit justifications for tool choices; and (iii) **Trajectory Augmentation**, combining CoT reasoning, tool-selection, and tool-integration steps into complete trajectories for robust training.

scientific reasoning, code generation, and multimodal understanding. As illustrated in Figure 2, our pipeline consists of three key stages:

- 1 Toolset & Task Collection.** We begin by assembling a diverse library of candidate tools, drawing from prior tool integration studies (Su et al., 2025; Feng et al., 2025a; Jin et al., 2025). In particular, we collect three representative categories: (i) *Code Tools*, including code sandboxes and interpreters that allow the agent model to execute code and leverage outputs or error messages during generation; (ii) *Search Tools*, covering search engines and web browser APIs (e.g., Jina Reader) for retrieving external knowledge from various corpora; and (iii) *Image Tools*, comprising image processing and understanding modules such as OCR (Su et al., 2025) and GroundingDINO (Liu et al., 2024), which enable the agent model to extract and reason over textual content embedded in images. This results in a comprehensive toolset of over 1,000 tools. Each tool is annotated with a feature description specifying its toolset index, functionality, input-output schema, and usage constraints. In addition, we curate a diverse set of downstream tasks aligned with these tools, spanning mathematical and scientific reasoning, code generation, search-based QA, and related domains, covering over 100 task types. Together, the toolset and task set provide the foundation for enabling agents to operate in open-ended, multimodal environments.
- 2 Tool-Selection Rationale Generation.** In real-world scenarios, toolsets are dynamic and evolve with new incoming tools. Therefore, we avoid formulating tool selection as a simple classification problem over a fixed tool index. Instead, we cast it as a decision-making process in which the agent model should learn to reason step by step toward selecting the most appropriate tool for the current context. To support this, we first retrieve the original responses from the collected downstream tasks in the previous stage, which consist of reasoning trajectories involving both model reasoning and tool integration components. In each response chain, before a specific tool is invoked, we leverage an expert reasoning model (DeepSeek-R1) to generate explicit rationales that justify why a particular tool is preferred at each step, serving as the supervision signals that align tool selection with contextual reasoning. We illustrate our template to generate tool-selection rationale training data in Figure 5.
- 3 Trajectory Augmentation.** After collecting tool-selection rationales, we insert them back into the original response trajectories. To ensure quality, we first leverage LLM-as-a-judge (Zheng et al., 2023) to filter out invalid rationales that lead to incorrect tool choices. We then insert the valid rationales between the model’s internal reasoning and the subsequent tool invocation (see the example in Figure 2). Finally, we employ DeepSeek-R1 to review the entire trajectory, smoothing connections and eliminating logical inconsistencies. This process yields a corpus of 200k data instances, including various tasks with corresponding tool selection and integration trajectory responses. Additional details on the curated dataset statistics and our dynamic tool selection settings are provided in Appendix A.

3 AUTOTool

We introduce AutoTool, an agentic reasoning framework that equips LLM agents with dynamic tool-selection capabilities during generation. Our framework leverages a dual-phase optimization pipeline:

(i) trajectory stabilization through supervised fine-tuning (SFT) and RL-based policy optimization, which endows agents with stable long-form CoT reasoning patterns; and (ii) tool-selection refinement via reward-guided Plackett–Luce (PL) ranking optimization.

We begin by introducing the preliminaries and notation that formalize the tool-selection setting within LLM agents’ generation process (Section 3.1). We then describe how LLM agents interact with an evolving toolset through tool-embedding grounded trajectory generation (Section 3.2). Next, we present the overall pipeline for the dual-phase policy optimization strategy (Section 3.3). Finally, we elaborate on how we optimize tool-selection via PL ranking during policy training (Section 3.4).

3.1 PRELIMINARY

Toolset. We denote π_θ as an LLM agent, parameterized by θ . In our agentic reasoning setting, an LLM agent is endowed with an *evolving toolset*, i.e., $T = \{t_1, t_2, \dots, t_{|T|}\}$, where the size $|T|$ is not fixed. Each tool $t_k \in T$ is paired with a feature description $\mu(t_k)$, which serves as its “instruction manual,” specifying the tool’s name, functionality, input-output schema, and usage constraints.

Agentic Trajectory Generation. We conceptualize a complete reasoning trajectory generated by an LLM agent π_θ as an interplay among three complementary components:

$$\tau = \dots \tau_{i-1}^{\text{reason}} \oplus \tau_i^{\text{select}} \oplus \tau_{i+1}^{\text{integrate}} \dots, \quad (1)$$

where τ includes: (i) *inner-reasoning steps* τ^{reason} , which π_θ produces a CoT format internal reasoning process; (ii) *tool-selection steps* τ^{select} , which π_θ reasons over the toolset T and selects a tool $t_k \in T$ for use; and (iii) *tool-integration steps* $\tau^{\text{integrate}}$, which follow immediately after the tool-selection steps and involve invoking the selected tool, executing it, and feeding the feedback back into π_θ ’s middle generation.

Per-step Tool Selection. During trajectory generation, we consider the tool-selection step τ_i^{select} in particular and formulate π_θ ’s tool-selection process as a *per-step decision-making problem*. Given the input question x , the evolving toolset T , and π_θ ’s previous generated steps $\tau_{<i}$, we model π_θ ’s generation at step i as selecting τ_i^{select} from all potential tool-selection candidates $\{\tau_k^{\text{select}} \mid t_k \in T\}$ with each τ_k^{select} corresponds to the tool $t_k \in T$, i.e.,

$$\tau_i^{\text{select}} = \arg \max_{\tau_k^{\text{select}}} \pi_\theta(\tau_k^{\text{select}} \mid x, \tau_{<i}, T). \quad (2)$$

3.2 TOOL-AWARE TRAJECTORY GENERATION

We first describe how, in AutoTool, an LLM agent interacts with the external toolset T and makes tool selections during its trajectory generation. Our rollout procedure is designed to allow LLM agents to seamlessly interleave reasoning with tool-selection steps for more coherent and effective tool usage.

Tool Embedding. To incorporate rich and precise information from the external toolset T into the LLM agent π_θ , we first collect the embedding representation of each tool t_k together with its descriptive features $\mu(t_k)$. We leverage the internal embedding layer of π_θ to obtain a contextualized representation for each tool:

$$\mathbf{e}_{t_k} = \text{Emb}_{\pi_\theta}[t_k, \mu(t_k)], \quad \text{with } \mathcal{E}_T = \{\mathbf{e}_{t_k}\}_{k=1}^{|T|}, \quad (3)$$

where \mathcal{E}_T is the tool-embedding set for T . The latent embedding for each tool shares the same hidden state space as the LLM agent, ensuring natural alignment with its reasoning generation process.

Embedding-Anchored Tool Selection. For each tool-selection step τ_i^{select} , we ask π_θ to first generate a selection rationale s_i , followed by an explicit *anchor token* with predicted embedding representation \mathbf{e}'_i . The tool for τ_i^{select} is then selected by sampling from the softmax-normalized distance distribution (Dong et al., 2015) between the predicted \mathbf{e}'_i and the candidate tool embeddings $\mathbf{e}_{t_k} \in \mathcal{E}_T$:

$$\pi_\theta(t_k \mid x, \tau_{<i}, s_i, T) = \frac{\exp(-\gamma \|\mathbf{e}'_i - \mathbf{e}_{t_k}\|_F^2)}{\sum_{t_j \in T} \exp(-\gamma \|\mathbf{e}'_i - \mathbf{e}_{t_j}\|_F^2)}, \quad (4)$$

where $\|\cdot\|_F$ is the Frobenius norm and $\gamma > 0$ controls the distribution skewness. By sampling from an embedding-based distribution, the LLM agent dynamically selects tools that are most aligned with

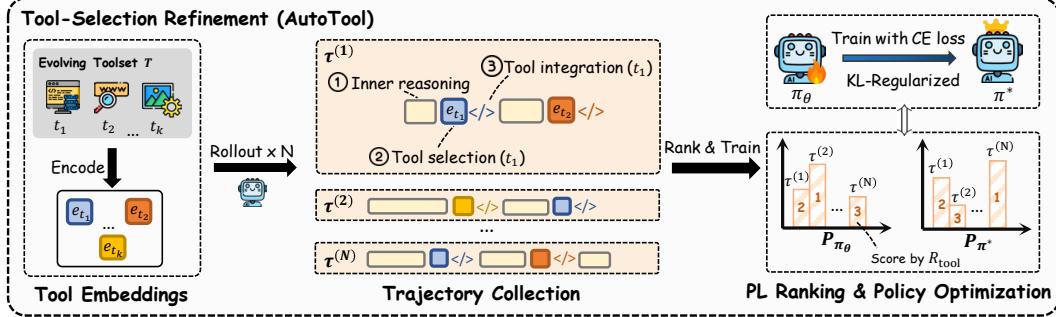


Figure 3: Illustration on AutoTool’s Tool-Selection Refinement Phase. We cast tool-selection as a Plackett–Luce (PL) ranking problem during trajectory generation, optimizing the LLM agent to align its policy distribution with reward-consistent tool preferences.

its previous reasoning context. After the tool selection step, the predicted tool is invoked and executed, with its outputs and feedback returned to the LLM agent for subsequent trajectory generation.

Takeaway: Why Embedding-Anchored Selection for Evolving Toolsets?

As the external toolset T evolves, directly generating tool names by the LLM agent risks failure on unseen tools. By anchoring selection in the embedding space which is generalizable (Mikolov et al., 2013; Ganguly et al., 2015), the agent can select new tools via *representation alignment*, avoiding reliance on memorized tool identifiers and remaining robust to evolving toolsets.

3.3 DUAL-PHASE POLICY TRAINING PIPELINE

Next, we describe how the LLM agent is trained as the policy model in AutoTool through a dual-phase optimization strategy. The overall training flow of π_θ is summarized as follows:

$$\underbrace{\theta_{\text{ref}} \xrightarrow{\text{SFT}} \theta_{\text{SFT}}}_{\text{Phase I - Trajectory Stabilization}} \xrightarrow{\text{RL}} \underbrace{\theta_{\text{RL}} \xrightarrow{\text{PL Ranking}} \theta_{\text{AutoTool}}}_{\text{Phase II - Tool-Selection Refinement}}$$

Phase I – Trajectory Stabilization. To enable the LLM agent to follow the reasoning patterns in Eq. 1 and produce valid tool-integrated trajectories, we initialize from a base reference model $\pi_{\theta_{\text{ref}}}$ (e.g., an off-the-shelf reasoning model). The LLM agent is then trained using agentic training strategies (Dong et al., 2025; Feng et al., 2025a), beginning with supervised fine-tuning and followed by RL-based policy optimization. The primary goal of Phase I is to equip the LLM agent with stable generation across inner-reasoning, tool-selection, and tool-integration steps, which serves as the basis for the subsequent tool-selection refinement in Phase II.

Phase II – Tool-Selection Refinement. After the agent acquires stable reasoning capabilities in Phase I, we further refine the model with a dedicated optimization focused on tool-selection steps. Unlike the full-trajectory optimization in Phase I, we mask out internal reasoning and tool-integration steps, restricting optimization to the tool-selection segments of trajectories. This masked formulation places special emphasis on improving the tool-selection process.

KL-Regularized Tool-Selection Objective. In Phase II, we train the LLM agent via a KL-regularized RL objective to refine its tool-selection process. Formally, given an input question x and the toolset T , our objective is to optimize the LLM agent policy from Phase I by:

$$\max_{\pi_\theta} \mathbb{E}_{\tau \sim \pi_\theta(\cdot | x, T)} [R_{\text{tool}}(\tau)] - \beta \cdot D_{\text{KL}}(\pi_\theta(\cdot | x, T) \| \pi_{\text{old}}(\cdot | x, T)), \quad (5)$$

where β controls the regularization intensity and $R_{\text{tool}}(\cdot)$ denotes the masked trajectory reward that evaluates solely on the correctness of tool-selection steps (defined later).

Motivation on Policy Optimization under PL Ranking. Previous agentic RL approaches (Shao et al., 2024; Yu et al., 2025a) directly optimize trajectory output rewards but do not explicitly capture the ordering relationships among candidate tools in the toolset. On the other hand, Plackett–Luce (PL) ranking (Luce et al., 1959; Cheng et al., 2010) converts trajectory rewards into a permutation

distribution, so that higher-reward tools are prioritized over weaker ones. This formulation naturally aligns with our embedding-anchored selection mechanism described in Eq. 4. Motivated by this, we optimize Eq. 5 under the PL ranking framework, aligning the policy-induced ranking with reward-guided tool preferences and providing a principled refinement of tool-selection behavior.

3.4 PL RANKING & OPTIMIZATION ON TOOL SELECTION

We detail how the LLM agent is optimized under the PL ranking framework. The key idea is to view trajectory rollouts for each query as a ranked list, where the relative order is induced by trajectory rewards derived from tool-selection steps. By mapping these rewards into a probabilistic ranking distribution, we connect the LLM agent policy optimization with ranking consistency and train the policy with a tractable loss that directly improves the LLM agent’s tool-selection accuracy.

Trajectory Collection & Reward Assignment. For each input question x and toolset T , we first sample N trajectory rollouts $\mathcal{T} = \{\tau^{(j)}\}_{j=1}^N$ from the LLM agent $\pi_\theta(\cdot | x, T)$. Within each trajectory, every tool-selection step τ_i^{select} is assigned a *step-level reward* r_{tool} that jointly evaluates the quality of the tool-selection rationale and the correctness of the final answer produced with the chosen tool:

$$r_{\text{tool}}(x, \tau_i^{\text{select}}) = \text{PRM}(\tau_i^{\text{select}}) + \text{Acc}(x, t_k), \quad t_k \in T, \quad (6)$$

where the process reward model, $\text{PRM}(\cdot)$, provides dense supervision by evaluating the reasoning quality of each tool-selection step τ_i^{select} , and $\text{Acc}(x, t_k)$ measures whether the final answer obtained using tool t_k is correct. We then aggregate all per-step selection rewards into a single masked *trajectory-level reward* $R_{\text{tool}}(\tau) = \frac{1}{|S(\tau)|} \sum_{i \in S(\tau)} r_{\text{tool}}(x, \tau_i^{\text{select}})$, where $S(\tau)$ is the (random) set of selection-step indices in each trajectory τ . We use R_{tool} for the KL-Regularized optimization in Eq. 5.

PL Ranking over Collected Rollouts. Given the collected set of trajectory rollouts \mathcal{T} with associated tool-selection rewards R_{tool} , for a permutation σ of N trajectories, we have the induced PL ranking model of the LLM agent π_θ :

$$P_{\pi_\theta}(\sigma | \mathcal{T}) = \prod_{j=1}^N \frac{\exp(R_{\text{tool}}(\tau^{\sigma(j)}))}{\sum_{l=j}^N \exp(R_{\text{tool}}(\tau^{\sigma(l)}))}, \quad (7)$$

where $\sigma(j)$ denotes the trajectory placed at rank j in the permutation σ . Eq. 7 shows that trajectory candidates with higher tool-selection reward appear earlier in the ranking order, thereby providing a tool-preference aware distribution over the N candidate rollouts.

Bridging PL Ranking with Policy Optimization. In Eq. 7, directly optimizing the PL distribution over the $N!$ possible trajectory permutations is computationally infeasible (Cheng et al., 2010). To obtain a tractable training objective, we establish a theoretical bridge that links the PL ranking to the LLM agent’s policy optimization. In Proposition 3.1 below, we establish that learning the optimal policy is equivalent to matching its induced PL ranking distribution over candidate trajectory rollouts.

Proposition 3.1 (Equivalence between Optimal Policy and PL Ranking). Consider the KL-regularized RL objective defined in Eq. 5 with the tool-selection reward R_{tool} defined in Eq. 6. Let π^* denote the corresponding optimal policy (Rafailov et al., 2023) for Eq. 5. Then, a trainable policy π_θ is equal to the optimal policy (i.e., $\pi_\theta = \pi^*$) if and only if their induced PL ranking distributions coincide for any input x and trajectory collection \mathcal{T} , i.e.

$$\pi_\theta = \pi^* \iff P_{\pi_\theta}(\sigma | \mathcal{T}) = P_{\pi^*}(\sigma | \mathcal{T}), \quad \forall \sigma.$$

Proposition 3.1 (proof in Appendix B) suggests that optimizing the LLM agent policy provides a tractable surrogate to direct PL ranking optimization. Accordingly, leveraging the closed-form solution of π^* (Schulman et al., 2017; Rafailov et al., 2023), we employ a practical Cross-Entropy (CE) loss to train the LLM agent policy π_θ towards the optimal policy π^* on a training set \mathcal{D} :

$$\begin{aligned} \mathcal{L}_{\text{CE}} &= -\mathbb{E}_{x \sim \mathcal{D}} \sum_{\tau \in \mathcal{T}} \pi^*(\tau | x, \mathcal{T}) \log \pi_\theta(\tau | x, \mathcal{T}) \\ &= -\mathbb{E}_{x \sim \mathcal{D}} \sum_{\tau \in \mathcal{T}} \frac{\pi_{\text{old}}(\tau | x, T) \exp\left(\frac{1}{\beta} R_{\text{tool}}(\tau)\right)}{\sum_{\tau' \in \mathcal{T}} \pi_{\text{old}}(\tau' | x, T) \exp\left(\frac{1}{\beta} R_{\text{tool}}(\tau')\right)} \log \frac{\pi_\theta(\tau | x, T)}{\sum_{\tau' \in \mathcal{T}} \pi_\theta(\tau' | x, T)}. \end{aligned} \quad (8)$$

Table 1: Comparison of AutoTool with both advanced LLM agents and tool-integration methods across math, search, and multimodal benchmarks. While some baselines excel in isolated domains (e.g., ReTool on math, Search-R1 on search, GPT-4o on multimodal), AutoTool delivers balanced and robust performance across all diverse tasks, demonstrating the generalization benefits of dynamic tool-selection and integration. The best results are **bold**, and the second-best results are underlined.

Method	Size	Math ↑		Search ↑		Multimodal ↑		
		AIME24	AIME25	HotpotQA	2Wiki	V-Chart	V-Math	V-Code
<i>Advanced LLM Agents</i>								
GPT4o	–	18.2	15.8	38.7	29.8	<u>27.3</u>	41.4	51.2
Qwen2.5-VL-72B-Instruct	72B	6.3	4.1	13.9	10.4	22.5	24.5	18.2
Qwen2.5-Math-72B-Instruct	72B	31.3	27.4	23.7	16.7	–	–	–
QwQ-32B	32B	47.3	31.8	32.6	22.3	–	–	–
DeepSeek-R1-Distill-Qwen-14B	14B	65.9	44.1	24.2	14.1	–	–	–
<i>Tool-Integration Methods</i>								
v-ToolRL	2B	–	–	–	–	21.6	19.5	13.3
Search-R1	7B	12.1	7.3	<u>43.3</u>	<u>44.5</u>	–	–	–
ReTool	32B	70.1	<u>47.4</u>	21.4	19.7	–	–	–
<i>Dynamic Tool-Selection & Integration</i>								
AutoTool (Qwen2.5-VL-7B)	7B	45.3	38.9	33.2	36.5	13.2	<u>44.3</u>	<u>52.5</u>
AutoTool (Qwen3-8B)	8B	<u>68.8</u>	51.2	45.1	48.8	24.7	53.0	56.1

4 EXPERIMENTS

Datasets. To assess the effectiveness and generalizability of AutoTool, we conduct evaluations on a broad range of downstream tasks. For mathematical and scientific reasoning, we evaluate on AIME24 (Maxwell-Jia, 2024), AIME25 (math ai, 2025), and GPQA-Diamond (Rein et al., 2024). For search-based reasoning, we use multi-hop question answering datasets including HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (2Wiki) (Ho et al., 2020), and Bamboogle (Press et al., 2023), which together cover a broad spectrum of search and reasoning challenges. For multimodal understanding, we first apply MMSearch (Jiang et al., 2024), a comprehensive benchmark for evaluating multimodal search performance. In addition, to assess AutoTool performance in multi-turn interaction scenarios, we curate 500 image-based questions spanning three domains: (i) V-Chart, covering chart reasoning with questions sourced from the ChartGemma dataset (Masry et al., 2024); (ii) V-Math, consisting of math problems derived from GSM8K (Cobbe et al., 2021) and AIME24 (Maxwell-Jia, 2024); and (iii) V-Code, focusing on code generation with questions drawn from MBPP (Austin et al., 2021), HumanEval (Chen et al., 2021), and LiveCodeBench (Jain et al., 2024). All three domains are presented to the LLM agents in image form, requiring them to parse visual inputs before reasoning.

Baselines and Models. We compare AutoTool with a broad set of baselines, including advanced LLM agents, fixed tool-integration approaches, and agentic training paradigms. (i) For advanced LLM agents, we compare against strong reasoning models including GPT-4o (Hurst et al., 2024), Qwen2.5-VL-72B (Bai et al., 2025), Qwen2.5-Math-72B (Yang et al., 2025), QwQ-32B (Team, 2025), and DeepSeek-R1-Distill-Qwen-14B (Guo et al., 2025). (ii) For tool-integration methods, we compare with ReTool (Feng et al., 2025a) for strategic tool use on math and science reasoning, Search-R1 (Jin et al., 2025) for search-based question answering, and v-ToolRL (Su et al., 2025) for multimodal understanding. (iii) For training-stage based comparisons, we compare with two common agent training paradigms, including SFT where the policy model is directly fine-tuned on curated trajectories that interleave reasoning with tool invocations, and GRPO (Shao et al., 2024), which trains the SFT checkpoint to refine reasoning trajectories.

Implementation Details. We incorporate AutoTool into two off-the-shelf models Qwen3-8B (think mode) (Yang et al., 2025) and Qwen2.5-VL-7B (Bai et al., 2025), and train them on our curated 200k dataset described in Section 2. For Phase I training, we leverage LLaMa-Factory (Zheng et al., 2024) for SFT and VeRL (Sheng et al., 2024) for GRPO training. For Phase II training, we train

Table 2: Comparison of AutoTool with agentic training baselines (SFT and GRPO). AutoTool consistently outperforms the training-based baselines, demonstrating that the Phase II PL-ranking stage provides complementary gains by explicitly optimizing dynamic tool selection.

Method	Math & Science ↑			Search ↑			Multimodal ↑			
	AIME24	AIME25	GQPA	HotpotQA	2Wiki	Bamboogl	V-Chart	V-Math	V-Code	MMSearch
Qwen3-8B	46.7	40.0	58.1	26.2	31.1	37.4	6.0	39.4	43.2	–
SFT	53.3	40.0	63.6	34.3	37.9	40.2	16.3	44.6	48.6	–
GRPO (with SFT)	60.0	46.7	70.7	38.6	45.2	52.8	22.8	51.8	55.2	–
AutoTool	66.7	53.3	73.7	45.1	48.8	56.8	24.7	53.0	56.1	–
Qwen2.5-VL-7B	6.7	13.2	7.6	14.0	18.2	10.6	2.6	22.9	26.6	27.8
SFT	30.0	20.0	30.8	24.9	22.0	24.5	6.8	28.4	43.2	39.7
GRPO (with SFT)	36.7	33.3	33.3	28.5	32.5	44.0	10.7	42.7	49.6	45.2
AutoTool	46.7	40.0	38.4	33.2	36.5	48.2	13.2	44.3	52.5	49.3

the induced PL ranking framework by setting the rollout size N to 8 per question and training for 3 epochs. All training and inference experiments are conducted on 8xA100-80G GPUs. We leave additional experimental setups in Appendix C.

4.1 AUTO TOOL BALANCES PERFORMANCE ACROSS VARIOUS DOMAINS

As shown in Table 1, we first compare AutoTool against both advanced LLM agents and tool-integration methods. Large LLM agents such as GPT-4o and Qwen2.5-VL-72B rely primarily on internal chain-of-thought reasoning and lack the ability to adaptively invoke external tools. As a result, their performance is uneven. For example, GPT-4o shows strong multimodal ability (27.3% on V-Chart, 51.2% on V-Code) but lags behind on math-intensive tasks (18.2% on AIME24, 15.8% on AIME25). Qwen2.5-Math-72B, while stronger in math (31.3% on AIME24, 27.4% on AIME25), fails to generalize to search or multimodal tasks. In contrast, AutoTool extends reasoning beyond the model’s internal capacity through dynamic tool selection, achieving 68.8%/51.2% on AIME24/25, 45.1%/48.8% on HotpotQA/2Wiki, and 53.0%/56.1% on V-Math/V-Code.

For tool-integration methods, we observe a limitation in their inability to maintain consistent reasoning across different domains due to their fixed tool-usage abilities. For example, ReTool excels in math (e.g., 70.1% on AIME24, 47.4% on AIME25) but drops sharply on search tasks (21.4% on HotpotQA, 19.7% on 2Wiki). Such brittleness highlights the limitations of fixed-task optimization. In contrast, AutoTool achieves balanced and robust performance across all domains, with AutoTool (Qwen3-8B) reaching the highest performance on 6 out of 7 benchmarks. These results demonstrate that explicit tool-selection optimization not only adapts tool use to task demands but also preserves stable reasoning across diverse modalities.

4.2 AUTO TOOL ADVANCES BEYOND STANDARD AGENTIC TRAINING PARADIGMS

Table 2 compares AutoTool with two commonly adopted agent training paradigms. For Qwen3-8B, we equipped the model with vision-extraction tools such as OCR to retrieve text from image inputs. AutoTool consistently surpasses both SFT and GRPO by explicitly modeling and optimizing tool selection within reasoning trajectories. For instance, on Qwen3-8B, AutoTool delivers a 6.7% improvement on AIME24 and 6.5% on HotpotQA over GRPO, while on Qwen2.5-VL-7B, AutoTool boosts multimodal performance to 49.3% (MMSearch) and 52.5% (V-Code), outperforming the GRPO baseline by over 7%. Our results show that, beyond stabilizing reasoning via fine-tuning and improving trajectory coherence with GRPO, Phase II PL-ranking provides complementary benefits by explicitly modeling dynamic tool selection, thereby enabling tool-aware optimization.

4.3 IN-DEPTH ANALYSES ON AUTO TOOL

Case Study. In Figure 4, we present an example of AutoTool (Qwen3-8B) solving an image-based cryptarithmic puzzle from V-Math. The agent first invokes OCR to extract textual content from the input image, and then adaptively switches to the Code Interpreter to verify digit assignments and compute the final solution. This case study highlights how AutoTool enables dynamic tool selection for the agent model to seamlessly integrate visual perception with symbolic reasoning.

Case Study on V-Math

Solve the following problem:

$$\begin{array}{r} \text{W H I T E} \\ + \text{W A T E R} \\ \hline \text{P I C N I C} \end{array}$$

Each distinct letter represents a different digit.
Find the number represented by the answer **PICNIC**.

problem.png

Input Prompt: You are given a problem and a set of tools. Solve the problem step by step by selecting the proper tools to use during your thinking.

Here is the name and usage instructions for the toolset, where you will select the appropriate tools to use.

Tool 1: Code Interpreter (CI) ...

Tool 2: OCR ...

Tool 3: SearchAPI ...

...

Now, given the input image `<problem.png>` about the cryptarithmetic puzzle, solve the problem and put the final answer into `\boxed{ }`

AutoTool Response:

Okay, let's tackle this cryptarithmetic puzzle step by step. First, given the input image, I need to extract the information about the puzzle for later calculation.

<select> We need to extract the information from the input image and then reason programmatically about the cryptarithmetic problem. OCR is suitable because it converts the visual image into structured text for further automated reasoning. **<tool>OCR</tool>** **</select>**

<ocr>

```
{"name": "OCR", "action": "extract_text", "input": "problem.png"}
```

</ocr>

<interpreter> Extracted Text: Solve the following problem ... by the answer **PICNIC**. **</interpreter>**

From the texts, each letter represents a unique digit. We need to find the number represented by PICNIC. First, since adding two 5-digit numbers results in a 6-digit number, the leading digit P must be 1 due to the maximum carryover. So, $P = 1$. Next, we analyze the columns from right to left. The key steps involve checking carryovers and ensuring unique digits. Let's verify possible values for W and I based on the equation $2W + c_4 = 10 + I$.

<select> We need to systematically check possible values for W and I based on mathematical constraints. This requires iterating through multiple possibilities and performing calculations. A code execution environment is ideal for this task. **<tool>Code Interpreter (CI)</tool>** **</select>**

<code>

```
possible_w_i=[]
for c4 in [0,1]:
    for W in range(2,10):
        I=2*W+c4-10
        if 0<=I<=9 and I!=1:
            possible_w_i.append((W,I,c4))
print(possible_w_i)
```

</code>

<interpreter>

```
[(5, 0, 0), (6, 2, 0), (7, 4, 0), (8, 6, 0), (9, 8, 0), (6, 3, 1), (7, 5, 1), (8, 7, 1), (9, 9, 1)]
```

</interpreter>

From the output, valid (W, I) pairs are filtered. We then explore each case. After several attempts, we find a valid solution when $W = 8$, $I = 6$, $c_4 = 0$, and other constraints align. Thus, WHITE is 83,642 and WATER is 85,427. Adding together, the PICNIC is 169,069. All digits are unique and satisfy the constraints. Thus, the answer is `\boxed{169069}`.

Figure 4: Case Study of AutoTool on V-Math.

AutoTool vs. Oracle Tool Assignment. To examine the effectiveness of AutoTool’s dynamic tool selection abilities, we further conduct experiments to compare AutoTool with a ground-truth tool assignment baseline where the correct tool is directly provided before each invocation. As shown in Table 3, we find that AutoTool achieves performance highly competitive with this “oracle” setup, with only marginal differences across benchmarks (e.g., 49.3 vs. 49.8 on MMSearch and 44.3 vs. 43.7 on V-Math). This minimized gap demonstrates that AutoTool’s Phase II optimization enables agents to autonomously select tools nearly as effectively as ground-truth assignment, validating the robustness of explicit tool-selection training.

5 RELATED WORK

Agentic Reasoning in RL. A prominent line of research on agentic reasoning in LLMs builds upon reinforcement learning to optimize reasoning behaviors directly. The introduction of GRPO and its variants (Shao et al., 2024; Zheng et al., 2025; Zhao et al., 2025; Li et al., 2025; Zhang et al., 2025; Liu et al., 2025; Yu et al., 2025a; Lu et al., 2025) marked a turning point, proposing a critic-free formulation that estimates group-relative advantages across sampled responses. Building on these foundations, recent work has explored agentic reasoning with tool use, such as ARTIST (Singh et al., 2025) and rStar2 (Shang et al., 2025), which employ outcome-based RL in settings with noisy or uncertain tool feedback. In parallel, recent research on self-evolving agents (Fang et al., 2025; Yu et al., 2025b; Gao et al., 2025a) shifts the view from static agents toward agents designed for continual adaptation. This line of work emphasizes mechanisms such as continual learning, structural modification, and autonomous self-improvement to extend an agent’s reasoning capacity over time.

Tool Use and Integration in LLMs. Early work on tool use in LLMs relied on prompting strategies that interleaved reasoning with external queries or API calls (Press et al., 2023; Yao et al., 2023). Additional existing works (Nakano et al., 2021; Schick et al., 2023) proposed training-based approaches that explicitly empower LLMs with tool-use capabilities, as well as extended tool-usage to large-scale API integration and planning (Patil et al., 2024; Song et al., 2023; Zou et al., 2025), multi-tool orchestration (Shen et al., 2023), and adaptive selection (Qin et al., 2024). More recent work integrates tool use tightly into training and system design, such as multimodal agent tuning (Gao et al., 2025b) and infrastructure for dynamic discovery and synchronization (Ding, 2025; Mastouri et al., 2025). However, existing tool-integration work setups generally assume a fixed set of designated tools for limited tasks, whereas realistic scenarios usually demand dynamic selection from large and evolving tool pools. We address this by explicitly modeling tool selection within reasoning trajectories.

Tool Selection in LLM Agents. Prior work on enabling LLM agents to interact with external tools has explored both retrieval-based selection and tool-generation paradigms. Retrieval-based approaches, such as AnyTool (Du et al., 2024), employ external hierarchical retrievers to map user queries to candidate tools. These retrievers are trained independently of the LLM agent and operate as standalone indexing systems, resulting in a two-stage pipeline in which the LLM does not directly optimize its internal representations for tool selection. Another line of research focuses on tool generation, including CRAFT (Yuan et al., 2024), ToolMaker (Cai et al., 2023), and CREATOR (Qian et al., 2023), which synthesize new tools or executable code when suitable tools are unavailable. While effective, these methods emphasize tool construction and often incur substantial execution, validation, and debugging overhead. In contrast, AutoTool focuses on the complementary challenge of tool selection by jointly aligning the LLM’s internal representations with tool embeddings via PL-ranking, enabling efficient and robust generalization to large and evolving toolsets.

6 CONCLUSION

We introduced AutoTool, a framework that equips LLM agents with dynamic tool-selection capabilities. First, we curate a large-scale dataset with explicit tool-selection rationales to ground reasoning in realistic tool-use scenarios. Then, we train the agent policy through a dual-phase optimization pipeline, enabling AutoTool to move beyond static tool use and achieve adaptive, generalizable

Table 3: Comparison with a ground-truth tool assignment baseline where the correct tool is directly provided to the LLM agent before invocation (i.e., no requirements for explicit tool-selection actions).

Qwen2.5-VL-7B	Pre-given (No Tool-Selection)	AutoTool
AIME24	46.7	46.7
HotpotQA	35.2	33.2
2Wiki	38.1	36.5
MMSearch	49.8	49.3
V-Math	43.7	44.3

reasoning. Extensive experiments across mathematics, science, retrieval, and multimodal tasks show that AutoTool consistently outperforms training-only baselines and existing tool-enhanced methods. These results highlight the importance of explicit tool-selection optimization in building extensible agents that can effectively operate under evolving tool environments.

REFERENCES

- Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pp. 2425–2433, 2015.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL <https://arxiv.org/abs/2502.13923>.
- Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. *ArXiv preprint*, abs/2305.17126, 2023. URL <https://arxiv.org/abs/2305.17126>.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. Research: Learning to reason with search for llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2503.19470>.
- Weiwei Cheng, Eyke Hüllermeier, and Krzysztof J Dembczynski. Label ranking methods based on the plackett-luce model. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 215–222, 2010.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Peng Ding. Toolregistry: A protocol-agnostic tool management library for function-calling llms. *arXiv preprint arXiv:2507.10593*, 2025.
- Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2): 295–307, 2015.
- Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazheng Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.
- Yu Du, Fangyun Wei, and Hongyang Zhang. Anytool: Self-reflective, hierarchical agents for large-scale api calls. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6757–6775. PMLR, 2024. URL <https://arxiv.org/abs/2402.04253>. ArXiv preprint arXiv:2402.04253.
- Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025.

- Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms, 2025a. URL <https://arxiv.org/abs/2504.11536>.
- Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025b.
- Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth JF Jones. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp. 795–798, 2015.
- Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025a.
- Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaojian Ma, Tao Yuan, Yue Fan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. Multi-modal agent tuning: Building a vlm-driven agent for efficient tool usage. In *International Conference on Learning Representations (ICLR)*, 2025b.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.
- Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. *Advances in Neural Information Processing Systems*, 37:139348–139379, 2024.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Os-trow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.
- Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanmin Wu, Jiayi Lei, Pengshuo Qiu, Pan Lu, Zehui Chen, Chaoyou Fu, Guanglu Song, et al. Mmsearch: Benchmarking the potential of large models as multi-modal search engines. *arXiv preprint arXiv:2409.12959*, 2024.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL <https://arxiv.org/abs/2503.09516>.
- Yilun Kong, Jingqing Ruan, Yihong Chen, Bin Zhang, Tianpeng Bao, Shiwei Shi, Guoqing Du, Xiaoru Hu, Hangyu Mao, Ziyue Li, et al. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world systems. *arXiv preprint arXiv:2311.11315*, 2023.
- Siheng Li, Zhanhui Zhou, Wai Lam, Chao Yang, and Chaochao Lu. Repo: Replay-enhanced policy optimization. *arXiv preprint arXiv:2506.09340*, 2025.
- Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pp. 38–55. Springer, 2024.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.
- Fanbin Lu, Zhisheng Zhong, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Arpo: End-to-end policy optimization for gui agents with experience replay. *arXiv preprint arXiv:2505.16282*, 2025.

- R Duncan Luce et al. *Individual choice behavior*, volume 4. Wiley New York, 1959.
- Xinji Mai, Haotian Xu, Weinong Wang, Jian Hu, Yingying Zhang, Wenqiang Zhang, et al. Agent rl scaling law: Agent rl with spontaneous code execution for mathematical problem solving. *arXiv preprint arXiv:2505.07773*, 2025.
- Ahmed Masry, Megh Thakkar, Aayush Bajaj, Aaryaman Kartha, Enamul Hoque, and Shafiq Joty. Chartgemma: Visual instruction-tuning for chart reasoning in the wild. *arXiv preprint arXiv:2407.04172*, 2024.
- Meriem Mastouri, Emna Ksontini, and Wael Kessentini. Making rest apis agent-ready: From openapi to mcp servers for tool-augmented llms. *arXiv preprint arXiv:2507.16044*, 2025.
- math ai. AIME 2025 dataset. <https://huggingface.co/datasets/math-ai/aime25>, 2025. Accessed: 2025-05-15.
- Maxwell-Jia. AIME 2024 dataset. https://huggingface.co/datasets/Maxwell-Jia/AIME_2024, 2024. Accessed: 2025-05-15.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.
- Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.
- Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.
- Yingzhe Peng, Gongrui Zhang, Miaozen Zhang, Zhiyuan You, Jie Liu, Qipeng Zhu, Kai Yang, Xingzhong Xu, Xin Geng, and Xu Yang. Lmm-r1: Empowering 3b lmms with strong reasoning abilities through two-stage rule-based rl. *arXiv preprint arXiv:2503.07536*, 2025.
- Shraman Pramanick, Rama Chellappa, and Subhashini Venugopalan. Spiqa: A dataset for multimodal question answering on scientific papers, 2025. URL <https://arxiv.org/abs/2407.09413>.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.
- Cheng Qian, Chi Han, Yi R. Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 462–477, 2023. URL <https://aclanthology.org/2023.findings-emnlp.462/>. EMNLP Findings 2023.
- Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=dHng200Jjr>.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.

Ning Shang, Yifei Liu, Yi Zhu, Li Lyra Zhang, Weijiang Xu, Xinyu Guan, Buze Zhang, Bingcheng Dong, Xudong Zhou, Bowen Zhang, et al. rstar2-agent: Agentic reasoning technical report. *arXiv preprint arXiv:2508.20722*, 2025.

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

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, et al. Restgpt: Connecting large language models with real-world restful apis. *arXiv preprint arXiv:2306.06624*, 2023.

Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, et al. Openthinkimg: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*, 2025.

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

Qwen Team. QwQ-32B: Embracing the power of reinforcement learning. <https://qwenlm.github.io/blog/qwq-32b/>, March 2025.

Chenyu Wang, Weixin Luo, Sixun Dong, Xiaohua Xuan, Zhengxin Li, Lin Ma, and Shenghua Gao. Mllm-tool: A multimodal large language model for tool agent learning. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6678–6687. IEEE, 2025a.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

Penghao Wu and Saining Xie. V?: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13084–13094, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengan Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025a.
- Zhaochen Yu, Ling Yang, Jiaru Zou, Shuicheng Yan, and Mengdi Wang. Demystifying reinforcement learning in agentic reasoning. *arXiv preprint arXiv:2510.11701*, 2025b.
- Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R. Fung, Hao Peng, and Heng Ji. Craft: Customizing llms by creating and retrieving from specialized toolsets. In *12th International Conference on Learning Representations (ICLR 2024)*, 2024. URL <https://arxiv.org/abs/2309.17428>. ArXiv preprint arXiv:2309.17428.
- Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhui Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.
- Kaichen Zhang, Yuzhong Hong, Junwei Bao, Hongfei Jiang, Yang Song, Dingqian Hong, and Hui Xiong. Gvpo: Group variance policy optimization for large language model post-training. *arXiv preprint arXiv:2504.19599*, 2025.
- Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*, 2023.
- Yuzhong Zhao, Yue Liu, Junpeng Liu, Jingye Chen, Xun Wu, Yaru Hao, Tengchao Lv, Shao-han Huang, Lei Cui, Qixiang Ye, et al. Geometric-mean policy optimization. *arXiv preprint arXiv:2507.20673*, 2025.
- Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.
- Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyuan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*, 2024.
- Jiaru Zou, Soumya Roy, Vinay Kumar Verma, Ziyi Wang, David Wipf, Pan Lu, Sumit Negi, James Zou, and Jingrui He. Tattoo: Tool-grounded thinking prm for test-time scaling in tabular reasoning, 2025. URL <https://arxiv.org/abs/2510.06217>.

A ADDITIONAL DETAILS ON AUTOOL DATA CURATION

A.1 DATA CURATION STATISTICS

Through Stage 1: Toolset & Task Collection, we curate a comprehensive toolset of 1,346 tools drawn from diverse domains (e.g., online APIs, open-source libraries, multimodal utilities). Among them, 460 tools are used during training, while the remaining 886 tools are kept completely unseen and reserved exclusively for inference-time evaluation. During training, the candidate toolset is fixed to the 460 predefined tools, because all tool-selection rationales and ground-truth trajectories in our source datasets involve only these tools. This fixed training pool ensures stable tool-embedding learning and consistent PL-ranking optimization. Note that Stage 1 is decoupled from Stages 2 and 3, which focus on collecting high-quality reasoning trajectories strictly over the 460 seen tools. This separation prevents leakage of unseen tools into training and enables a clean evaluation of generalization. During inference, we expand the available toolset to the full 1,346 tools, simulating an *evolving* tool environment in which many tools have never been observed during training. This design allows us to systematically evaluate AutoTool’s ability to perform dynamic tool selection under distribution shift, where the candidate tools may differ across tasks and include entirely new unseen tools. The overall dataset statistics are summarized in Table 4. We next describe how experiments are conducted under the evolving toolset setting.

A.2 EVOLVING TOOLSET SETTINGS

Training-Time Toolset. During training, the toolset is fixed to the 460 “seen” tools collected through Stage 2 and Stage 3 of our data curation pipeline. All training-time tool-selection rationales and ground-truth tool trajectories reference only these 460 tools because the underlying source datasets provide demonstrations exclusively for this subset. As a result, every training instance uses the same 460-tool candidate pool. This design ensures stable tool-embedding learning and consistent PL-ranking optimization across all training samples. As AutoTool learns dynamic tool-selection behavior through its embedding-based matching mechanism (Eq. 4), rather than through dynamic changes in the training-time toolset itself. Therefore, an evolving toolset is not required during training for the model to develop dynamic selection capabilities.

Inference-Time Evolving Toolset. During inference, AutoTool does not assume a static or closed tool inventory. Instead, we simulate realistic tool evolution by evaluating the model over the full toolset that includes both the 460 tools seen during training and an additional 886 unseen tools from our toolset collection. This allows us to evaluate AutoTool under realistic, large-scale tool environments while keeping comparisons fair across baselines.

A.3 TEMPLATE ON GENERATING TOOL-SELECTION RATIONALES

Template for Tool-Selection Rationale
<p>You are a Tool Use Expert in selecting the most appropriate tool(s) to answer user questions. You are provided with three inputs:</p> <p># INPUT QUESTION: ...</p> <p># MODEL ANSWER: ...</p> <p># TOOLSET (including different types of tools and capabilities)</p> <p><tool1 name> + <description></p> <p><tool2 name> + <description></p> <p>...</p> <p>Your task is to provide the rationale and the final selected tool from the toolset before the model invokes a tool. Your response:</p>

Figure 5: Template to generate tool-selection rationales during AutoTool’s data curation process.

B THEORETICAL BRIDGE BETWEEN OPTIMAL POLICY AND PL RANKING

B.1 PROOF OF PROPOSITION 3.1

In the main body, we denote $P_\pi(\sigma|\mathcal{T})$ by omitting the query x for notation brevity. For the derivations below, we reinstate this dependence. In this case, for a collection of trajectories $\mathcal{T} = \{\tau^{(j)}\}_{j=1}^{|\mathcal{T}|}$ and query x , the Plackett-Luce (PL) ranking model induced by a policy π , relative to a previous policy π_{old} , will be defined as:

$$P_\pi(\sigma|\mathcal{T}, x) = \prod_{i=1}^{|\mathcal{T}|} \frac{\exp\left(\beta \log \frac{\pi(\tau^{\sigma(i)}|x)}{\pi_{\text{old}}(\tau^{\sigma(i)}|x)}\right)}{\sum_{j=i}^{|\mathcal{T}|} \exp\left(\beta \log \frac{\pi(\tau^{\sigma(j)}|x)}{\pi_{\text{old}}(\tau^{\sigma(j)}|x)}\right)}, \quad (9)$$

where we have σ being a permutation (or ranking) of the indices $\{1, 2, \dots, |\mathcal{T}|\}$, and denote $\tau^{\sigma(i)}$ being the trajectory ranked at position i in permutation σ . $\frac{\pi(\tau|x)}{\pi_{\text{old}}(\tau|x)}$ represents the relative preference of policy π over the previous checkpoint before updating.

We consider two policy models: π_θ is a trainable policy model with parameters θ , and π^* is the optimal policy model. Subsequently, let us denote the two induced PL ranking models by

$$\begin{aligned} P_{\pi_\theta}(\sigma|\mathcal{T}, x) &= \prod_{i=1}^{|\mathcal{T}|} \frac{\exp\left(\beta \log \frac{\pi_\theta(\tau^{\sigma(i)}|x)}{\pi_{\text{old}}(\tau^{\sigma(i)}|x)}\right)}{\sum_{j=i}^{|\mathcal{T}|} \exp\left(\beta \log \frac{\pi_\theta(\tau^{\sigma(j)}|x)}{\pi_{\text{old}}(\tau^{\sigma(j)}|x)}\right)}, \\ P_{\pi^*}(\sigma|\mathcal{T}, x) &= \prod_{i=1}^{|\mathcal{T}|} \frac{\exp\left(\beta \log \frac{\pi^*(\tau^{\sigma(i)}|x)}{\pi_{\text{old}}(\tau^{\sigma(i)}|x)}\right)}{\sum_{j=i}^{|\mathcal{T}|} \exp\left(\beta \log \frac{\pi^*(\tau^{\sigma(j)}|x)}{\pi_{\text{old}}(\tau^{\sigma(j)}|x)}\right)}. \end{aligned}$$

We will then have the following result on the equivalence between the optimal policy and the optimal PL ranking model. Results from the following Proposition B.1 supports the conclusion from Proposition 3.1.

Proposition B.1 (Equivalence between Optimal Policy and PL Ranking). Consider the KL-regularized RL objective defined in Eq. 5 with the tool-selection reward R_{tool} defined in Eq. 6. Let π^* denote the corresponding optimal policy (Rafailov et al., 2023) for Eq. 5. Then, a trainable policy π_θ is equal to the optimal policy (i.e., $\pi_\theta = \pi^*$) if and only if their induced PL ranking distributions coincide for any input x and trajectory collection \mathcal{T} , i.e.

$$\pi_\theta = \pi^* \iff P_{\pi_\theta}(\sigma | \mathcal{T}) = P_{\pi^*}(\sigma | \mathcal{T}), \quad \forall \sigma.$$

Proof. We need to prove both directions of the equivalence, and we will start with the forward direction that optimal policy indicates the optimal PL ranking.

Forward Direction. Suppose $\pi_\theta(\tau|x) = \pi^*(\tau|x)$ for all τ and x . Then, since the policies are identical, their ratios with respect to the old policy are also identical, leading to

$$\frac{\pi_\theta(\tau|x)}{\pi_{\text{old}}(\tau|x)} = \frac{\pi^*(\tau|x)}{\pi_{\text{old}}(\tau|x)} \quad (10)$$

Therefore, with $\beta > 0$, for any permutation σ :

$$\begin{aligned} P_{\pi_\theta}(\sigma|\mathcal{T}, x) &= \prod_{i=1}^{|\mathcal{T}|} \frac{\exp\left(\beta \log \frac{\pi_\theta(\tau^{\sigma(i)}|x)}{\pi_{\text{old}}(\tau^{\sigma(i)}|x)}\right)}{\sum_{j=i}^{|\mathcal{T}|} \exp\left(\beta \log \frac{\pi_\theta(\tau^{\sigma(j)}|x)}{\pi_{\text{old}}(\tau^{\sigma(j)}|x)}\right)} = \prod_{i=1}^{|\mathcal{T}|} \frac{\exp\left(\beta \log \frac{\pi^*(\tau^{\sigma(i)}|x)}{\pi_{\text{old}}(\tau^{\sigma(i)}|x)}\right)}{\sum_{j=i}^{|\mathcal{T}|} \exp\left(\beta \log \frac{\pi^*(\tau^{\sigma(j)}|x)}{\pi_{\text{old}}(\tau^{\sigma(j)}|x)}\right)} \\ &= P_{\pi^*}(\sigma|\mathcal{T}, x) \end{aligned}$$

Thus, if the policies are identical, then their induced PL ranking models are also identical.

Reverse Direction. Suppose $P_{\pi_\theta}(\sigma \mid \mathcal{T}, x) = P_{\pi^*}(\sigma \mid \mathcal{T}, x)$ for possible permutations σ , finite trajectory sets \mathcal{T} , and queries x (with $\beta > 0$ and $\pi_{\text{old}}(\cdot \mid x) > 0$ on the support). First, define the scores

$$s_\pi(\tau \mid x) := \beta \log \frac{\pi(\tau \mid x)}{\pi_{\text{old}}(\tau \mid x)}.$$

Next, consider any two trajectories τ, τ' and the two-trajectory collection $\mathcal{T} = \{\tau, \tau'\}$. The PL probabilities for the two possible permutations satisfy

$$P_{\pi_\theta}((\tau, \tau') \mid \mathcal{T}, x) = P_{\pi^*}((\tau, \tau') \mid \mathcal{T}, x), \quad P_{\pi_\theta}((\tau', \tau) \mid \mathcal{T}, x) = P_{\pi^*}((\tau', \tau) \mid \mathcal{T}, x).$$

By the PL definition on a two-item set $\mathcal{T} = \{\tau, \tau'\}$, the probability of placing τ first under π will be

$$P_\pi((\tau, \tau') \mid \mathcal{T}, x) = \frac{e^{s_\pi(\tau \mid x)}}{e^{s_\pi(\tau \mid x)} + e^{s_\pi(\tau' \mid x)}} = \frac{1}{1 + \exp(s_\pi(\tau' \mid x) - s_\pi(\tau \mid x))}.$$

Hence equality of the two top-1 probabilities for π_θ and π^* implies equality

$$\frac{P_{\pi_\theta}((\tau, \tau') \mid \mathcal{T}, x)}{P_{\pi_\theta}((\tau', \tau) \mid \mathcal{T}, x)} = \frac{P_{\pi^*}((\tau, \tau') \mid \mathcal{T}, x)}{P_{\pi^*}((\tau', \tau) \mid \mathcal{T}, x)} \iff e^{s_{\pi_\theta}(\tau \mid x) - s_{\pi_\theta}(\tau' \mid x)} = e^{s_{\pi^*}(\tau \mid x) - s_{\pi^*}(\tau' \mid x)}.$$

Taking logarithms yields

$$s_{\pi_\theta}(\tau \mid x) - s_{\pi_\theta}(\tau' \mid x) = s_{\pi^*}(\tau \mid x) - s_{\pi^*}(\tau' \mid x).$$

Fix an arbitrary reference τ_0 and set $\tau' = \tau_0$. The above identity then gives, for every τ ,

$$s_{\pi_\theta}(\tau \mid x) - s_{\pi_\theta}(\tau_0 \mid x) = s_{\pi^*}(\tau \mid x) - s_{\pi^*}(\tau_0 \mid x).$$

Based on the invariance property of the exponential scoring and ranking mechanism (Lemma B.2), this is equivalent to the existence of a constant $C(x) := s_{\pi_\theta}(\tau_0 \mid x) - s_{\pi^*}(\tau_0 \mid x)$ (independent of τ) such that

$$s_{\pi_\theta}(\tau \mid x) = s_{\pi^*}(\tau \mid x) + C(x), \quad \forall \tau.$$

Equivalently,

$$\beta \log \frac{\pi_\theta(\tau \mid x)}{\pi_{\text{old}}(\tau \mid x)} = \beta \log \frac{\pi^*(\tau \mid x)}{\pi_{\text{old}}(\tau \mid x)} + C(x),$$

so dividing by β and exponentiating gives

$$\frac{\pi_\theta(\tau \mid x)}{\pi_{\text{old}}(\tau \mid x)} = \frac{\pi^*(\tau \mid x)}{\pi_{\text{old}}(\tau \mid x)} e^{C(x)/\beta} \implies \pi_\theta(\tau \mid x) = e^{C(x)/\beta} \pi^*(\tau \mid x).$$

Since both π_θ and π^* are probability distributions and will sum to 1 over all possible trajectories, $e^{C(x)/\beta} = 1$, which implies $\pi_\theta(\tau \mid x) = \pi^*(\tau \mid x)$. Therefore, if the induced PL ranking models are identical, then the underlying policy models will also be identical. Combining the results from both directions will complete the proof. \square

B.2 PROOF OF LEMMA B.2

In this section, we provide the proof for Lemma B.2 previously used in the proof of Proposition B.1.

Lemma B.2 (Softmax Shift-invariance Property). *Let $z, w \in \mathbb{R}^d$, $d \geq 2$ be two vectors of the same dimension. The softmax outputs are identical, $\text{softmax}(z) = \text{softmax}(w)$, if and only if there exists a scalar constant $C \in \mathbb{R}$ such that the inputs differ by a constant shift, i.e., $w = z + C \cdot \mathbf{1}$, where $\mathbf{1}$ is the vector of ones. The softmax function is defined component-wise as $\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^K \exp(x_j)}$ for a position i .*

Proof. (**Forward** \Rightarrow) Suppose vector $w = z + C \cdot \mathbf{1}$ for some constant C . Then, for any component i :

$$\begin{aligned}\text{softmax}(w)_i &= \frac{\exp(w_i)}{\sum_{j=1}^K \exp(w_j)} = \frac{\exp(z_i + C)}{\sum_{j=1}^K \exp(z_j + C)} \\ &= \frac{\exp(z_i)e^C}{e^C \sum_{j=1}^K \exp(z_j)} = \frac{\exp(z_i)}{\sum_{j=1}^K \exp(z_j)} = \text{softmax}(z)_i.\end{aligned}$$

Since this holds for all i , $\text{softmax}(w) = \text{softmax}(z)$.

(**Backward** \Leftarrow) Suppose $\text{softmax}(z) = \text{softmax}(w)$. Let $S_z = \sum_j \exp(z_j)$ and $S_w = \sum_j \exp(w_j)$. This condition implies $\frac{\exp(z_i)}{S_z} = \frac{\exp(w_i)}{S_w}$ for all i . Since $S_z, S_w > 0$, we rearrange to get

$$\exp(w_i) = \exp(z_i) \cdot (S_w/S_z).$$

Let the positive constant $K = S_w/S_z$. Taking the natural logarithm yields $w_i = \ln(\exp(z_i)K) = z_i + \ln(K)$. Setting the constant $C = \ln(K)$, we have $w_i = z_i + C$ for all i . Thus, we will have $w = z + C \cdot \mathbf{1}$, which completes the proof.

□

C EXPERIMENT SETUPS

We evaluate our model on a comprehensive suite of benchmarks to assess its capabilities across various domains. The datasets used are detailed below, categorized by the primary task they are designed to evaluate.

Mathematical and Scientific Reasoning

- **AIME24 and AIME25** ([Maxwell-Jia, 2024](#); [math ai, 2025](#)): The American Invitational Mathematics Examination (AIME) datasets, specifically from the years 2024 and 2025, consist of challenging mathematical problems from this prestigious high school competition. These datasets are used to evaluate the mathematical reasoning and problem-solving abilities of large language models. The problems cover various domains like algebra, geometry, and number theory and require multi-step reasoning.
- **GPQA / GPQA-Diamond** ([Rein et al., 2024](#)): GPQA is a benchmark of 448 graduate-level multiple-choice questions in biology, chemistry, and physics, crafted by domain experts and validated to be “Google-proof.” Expert accuracy is 65% (or 74% after correction), while skilled non-experts (with unlimited web access) achieve 34%. The “Diamond” subset is often used to select the hardest subset of these questions.

Search-Based Reasoning

- **HotpotQA** ([Yang et al., 2018](#)): A multi-hop QA dataset built over Wikipedia, containing 113k questions. Each question often spans multiple documents, and supporting fact annotations are provided to encourage explainable reasoning.
- **2WikiMultiHopQA (2Wiki)** ([Ho et al., 2020](#)): Similar to HotpotQA, this is a multi-hop question-answering dataset created from Wikipedia. It’s designed to evaluate the reasoning steps of a model and includes evidence to support the answers. It has different question types, including comparison, inference, and compositional questions.
- **Bamboogle** ([Press et al., 2023](#)): This dataset is a collection of questions that are designed to be difficult for Google’s search engine to answer correctly. It is used to test the compositional reasoning abilities of language models, pushing them beyond simple information retrieval.

Multimodal Understanding

- **MMSearch** ([Jiang et al., 2024](#)): This is a benchmark for evaluating the multimodal search performance of large language models. The dataset consists of instances that are not present in the training data of current models, so the correct answer can only be found by searching. It evaluates models on tasks like re-querying, reranking, and summarization in a multimodal context.

Data Example of V-Chart

Solve the following problem: How many bars are there in the graph?

problem.png

Input Prompt: You are given a problem and a set of tools. Solve the problem step by step by selecting the proper tools to use during your thinking.

Here is the name and usage instructions for the toolset, where you will select the appropriate tools to use.

Tool 1: Code Interpreter (CI) ...

Tool 2: OCR ...

Tool 3: SearchAPI ...

...

Now, given the input image <problem.png> about the cryptarithmetic puzzle, solve the problem and put the final answer into \boxed{ }

Answer: 3

• V-Chart

- **ChartGemma** ([Masry et al., 2024](#)): The ChartGemma dataset is used for chart understanding and reasoning. It contains a diverse set of charts and is used to train and evaluate models on their ability to interpret and answer questions about the visual information presented in charts.

• V-Math

- **GSM8K** (Cobbe et al., 2021): This dataset, which stands for "Grade School Math 8K," contains 8,500 high-quality and linguistically diverse grade school math word problems. It is designed to measure the multi-step reasoning capabilities of language models.
- **AIME24**: Using data from AIME24, we reframe the text data into an image form.

Data Example of V-Math

Solve the following problem:

The 27 cells of a 3×9 grid are filled in using the numbers 1 through 9 so that each row contains 9 different numbers, and each of the three 3×3 blocks heavily outlined in the example below contains 9 different numbers, as in the first three rows of a Sudoku puzzle.

4	2	8	9	6	3	1	7	5
3	7	9	5	2	1	6	8	4
5	6	1	8	4	7	9	2	3

The number of different ways to fill such a grid can be written as $p^a \cdot q^b \cdot r^c \cdot s^d$, where p, q, r , and s are distinct prime numbers and a, b, c , and d are positive integers.

problem.png

Answer: 81

Input Prompt: You are given a problem and a set of tools. Solve the problem step by step by selecting the proper tools to use during your thinking.

Here is the name and usage instructions for the toolset, where you will select the appropriate tools to use.

Tool 1: Code Interpreter (CI) ...

Tool 2: OCR ...

Tool 3: SearchAPI ...

...

Now, given the input image <problem.png> about the cryptarithmetic puzzle, solve the problem and put the final answer into \boxed{}

• V-Code

- **MBPP** (Austin et al., 2021): The "Mostly Basic Python Problems" dataset consists of around 1,000 entry-level Python programming problems. It's used to evaluate the ability of language models to generate code from natural language descriptions.
- **HumanEval** (Chen et al., 2021): A dataset created by OpenAI, HumanEval consists of 164 handwritten programming problems to evaluate the functional correctness of code generated by language models. The problems are designed to be novel and not easily found on the web, to prevent models from simply recalling existing code.
- **LiveCodeBench** (Jain et al., 2024): A dynamic benchmark collecting new programming problems from competitive programming platforms (LeetCode, AtCoder, Codeforces), designed to provide “contamination-free” code evaluation by only testing on problems released after model training cutoffs.

Data Example of V-Code

Solve the following problem:

Write a function to multiply two lists using map and lambda function.

problem.png

Answer:

```
def mul_list(nums1, nums2):
    result = map(lambda x,
                 y: x * y, nums1, nums2)
    return list(result)
```

Input Prompt: You are given a problem and a set of tools. Solve the problem step by step by selecting the proper tools to use during your thinking.

Here is the name and usage instructions for the toolset, where you will select the appropriate tools to use.

Tool 1: Code Interpreter (CI) ...

Tool 2: OCR ...

Tool 3: SearchAPI ...

...

Now, given the input image <problem.png> about the cryptarithmetic puzzle, solve the problem and put the final answer into \boxed{}