ToolSearcher: Optimizing Tool Selection at Scale via Reinforcement Learning

Zhenlong Dai
Zhejiang University
zhenlongdai@zju.edu.cn

Xujie Song
Zhejiang University
songxujie@zju.edu.cn

Zitong Wang
Ant Group
yesi.wzt@antgroup.com

Tong Niu
Ant Group
niutong.niu@antgroup.com

Jian Liu*
Ant Group
rex.lj@antgroup.com

Weiqiang Wang
Ant Group
weiqiang.wwq@antgroup.com

Xiu Tang
Zhejiang University
tangxiu@zju.edu.cn

Sai Wu
Zhejiang University
wusai@zju.edu.cn

Chang Yao*
Zhejiang University
changy@zju.edu.cn

Jingyuan Chen
Zhejiang University
jingyuanchen@zju.edu.cn

Abstract

Large language models (LLMs) excel at natural language processing but struggle to interact with external environments. Tool learning provides a promising way to extend LLMs into actionable agents, where tool selection is a critical prerequisite for successful tool use. Existing work often assumes a small or predefined set of tools, leaving large-scale tool selection underexplored. Real-world repositories contain a vast and diverse array of tools, making it difficult for LLMs to effectively search, distinguish, and compose tools under context-length constraints. We identify large-scale tool selection as a new challenge for agentic reinforcement learning, highlighting that existing RL methods for knowledge-based question answering are inadequate for selecting tools while considering compatibility. To address this challenge, we propose ToolSearcher, a novel RL framework for effective multi-turn search and fine-grained optimization in large-scale tool selection. Specifically, we introduce category-constrained tool discrimination to improve the model's ability to distinguish functionally similar tools, event-level search modeling to explicitly optimize the discovery of target tools during multi-turn search, and trajectory-aligned credit allocation to provide fine-grained reward signals for different stages of the search-selection process. Extensive experiments on large-scale tool selection benchmarks demonstrate that ToolSearcher consistently outperforms a set of strong baselines in challenging settings involving iterative search and complex tool composition. The code and the dataset are available at https://github.com/zhenlongDai/ToolSearcher.

*Co-corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Large language models (LLMs) have recently demonstrated remarkable capabilities in natural language processing (NLP) tasks [22, 7, 6]. Despite their impressive performance, pretrained LLMs often struggle to actively perceive and interact with external environments [8, 9, 29], as their abilities are intrinsically constrained by NL interaction. Given the essential role of tools in enabling LLMs to interact with external environments and to act as user agents for solving real-world tasks, tool learning has attracted increasing attention [52]. The typical process of tool learning comprises three key stages: tool selection, tool calling, and response generation [37]. Existing studies mainly focus on tool calling (e.g., specifying function parameters, tool execution), while overlooking a crucial yet underexplored issue of tool selection: *how to effectively search and select appropriate candidate tools from a large-scale and diverse tool repository according to specific task requirements.*

Recent studies [35, 51, 44] typically assume a predefined or small set of candidate tools, which creates a substantial gap between these settings and real-world scenarios. Statistics from mainstream software and model ecosystems (e.g., PyPI, npm, and HuggingFace) show that both the number and functional diversity of available tools are already massive [62, 15]. Due to context-length limitations, it is impractical to provide LLMs with documentation for all tools, making the search a necessary step for effectively using existing tool resources. Moreover, large-scale tool repositories further exacerbate both functional and semantic similarities among tools. On one hand, many tools have similar semantics but differ in functionality. On the other hand, some tools offer similar functionality but have different interface designs (*i.e.*, input constraints and output information). Under this background, tool selection is inherently a complex process involving planning, search, and decision-making, requiring the coordination of multiple capabilities.

LLMs remain limited in autonomously organizing the capabilities required for tool selection. Current research [40, 49] utilizes search engines as part of the reasoning process, allowing LLMs to conduct multi-turn searches. Given the difficulty of collecting high-quality annotated trajectories and improving the model's adaptability to dynamic search environments [4, 2], they optimize via agentic reinforcement learning (RL) [18, 47]. However, these methods are designed for knowledge-intensive questions that rely on iterative information supplementation, making them unsuitable for tool selection tasks that require reasoning about tool composability. For tool selection, each search step needs to plan around a hypothesized tool chain and consider the composition of tools, then adjust the plan based on the search results. Based on the above analysis, applying RL to tool selection in large-scale scenarios mainly faces two key challenges: (1) **How to effectively perform multi-turn search over a large-scale tool repository**. Tool selection requires the model to search relevant tools, distinguish between functionally or semantically similar candidates, and reason about interface compatibility over multiple search steps. However, existing RL-based methods do not explicitly model the tool search process. (2) **Tool selection involves multiple capabilities, making joint learning and reward design difficult**. RL methods typically rely on trajectory-level rewards that assign the same value to a trajectory, leading to sparse supervision and making it difficult to optimize fine-grained behaviors throughout the multi-turn process.

To tackle the above challenges, we propose a novel RL approach named ToolSearcher, which is designed to enable effective multi-turn search and fine-grained optimization for large-scale tool selection. To differentiate between functionally similar tools, we propose category-constrained tool discrimination, which creates a highly constrained challenge environment designed to enhance LLMs' ability to understand and distinguish tool functionalities. To explicitly model the tool search process, we propose event-level search modeling, which optimizes semantic search by concentrating on events that discover previously unsearched target tools, thereby enhancing the LLM's ability to plan and search for compatible tool compositions. To facilitate the joint learning of multiple capabilities in tool selection, we design trajectory-aligned credit allocation, a fine-grained reward mechanism that assigns objective and quantifiable credit based on each sample's progression within the search-selection trajectory. Unlike outcome-only rewards, this method evaluates samples at different stages, providing matched feedback that accurately reflects their progress for tool selection.

Our contributions are summarized as follows:

- We identify large-scale tool selection as a novel challenge for agentic RL and demonstrate that existing RL methods designed for knowledge-oriented search are unsuited to multi-turn tool search and reasoning about tool composability.

- We propose ToolSearcher, a novel RL framework that explicitly models the multi-turn search process over large-scale tool repositories and provides fine-grained optimization signals for the diverse capabilities involved in tool selection.
- We conduct extensive experiments on large-scale tool selection benchmarks, where ToolSearcher consistently outperforms other baselines in challenging scenarios requiring iterative search and complex tool composition.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) enhances language models by grounding their responses in external knowledge sources [20]. RAG typically follows a retrieval-then-generation pipeline [12], where a search engine fetches relevant information based on the query, which is then combined with the query and fed into the LLM. Early RAG approaches [23, 19] focused on retrieval tasks in which answers reside within individual documents. Recent advances [3, 16, 48] have extended RAG to multi-hop reasoning, where answers reside across multiple documents. Although these methods improve reasoning over small document sets, they remain constrained by top-k retrieval paradigms, which fundamentally limit their ability to achieve complete document coverage for a large corpus [31]. These paradigms can lead to challenges such as retrieving irrelevant information [17] and failing to provide sufficiently relevant context [16]. Tool-integrated agents [49, 40] have expanded the possibilities of RAG by treating the search engine as a tool, enabling LLMs to interleave reasoning and action with external tools [25]. However, these training-free agentic approaches show promise but are limited by the quality of prompting strategies and lack the ability to learn from task-specific feedback.

2.2 Agentic Reinforcement Learning

Agentic reinforcement learning [56] enables agents to make sequential decisions aimed at maximizing cumulative rewards. In LLM tuning, RL has become a highly effective paradigm through RLHF [21], implemented with PPO [41]. To address the complexity inherent in PPO, methods like DPO [38] and SimPO [33] have been proposed as more efficient alternatives. While these methods offer computational efficiency, they suffer from off-policy issues [34] and do not consistently match the performance of pure RL approaches [18]. An effective RL method is Group Relative Policy Optimization (GRPO) [42], which significantly improves LLMs' reasoning by estimating baselines from group scores. Building on this, GSPO [60] enables stable optimization in multi-turn RL by handling precision discrepancies with sequence-level likelihoods. GDPO [28] decouples reward normalization to preserve relative differences to enhance multi-reward optimization. More recent work [18, 26] has applied RL to multi-step retrieval, but they primarily optimize local retrieval over small document subsets [10, 31]. MARAG-R1 [32] expands in this direction by leveraging RLOO [1] to achieve global retrieval tasks that require comprehensive document coverage. Despite these advances, RL for tool selection in large-scale repositories remains largely unexplored, particularly in distinguishing between similar tools and learning compatible tool combinations.

3 Method

LLMs remain limited in autonomously organizing the capabilities required for tool selection. To address this, we propose an RL-based approach, ToolSearcher , that enhances tool selection within large-scale tool repositories. In this section, we first define the tool selection task (Section 3.1) and provide the preliminary information (Section 3.2). We then propose category-constrained tool discrimination (Section 3.3) to improve the differentiation of functionally similar tools in large-scale tool repositories. Moreover, We introduce our RL approach for tool selection (Section 3.4), which incorporates event-level search modeling (Section 3.4.1) and trajectory-aligned credit allocation (Section 3.4.2) to effectively guide the learning process in searching and identifying compatible tool combinations that meet task requirements.

3.1 Task Definition

In a large-scale tool repository $\mathcal{T}$, given a user requirement q and a search engine $\mathcal{R}$, the goal is to identify a tool set $\mathcal{T}_s$ that satisfies task requirements. For each tool $T \in \mathcal{T}$, The document $d_T = (f, \mathcal{I}, \mathcal{O}_t)$ contains a function description f and an interface specification, including its input constraints $\mathcal{I}$ and output schema $\mathcal{O}$. Formally, the objective of tool selection is to identify a set of tools $\mathcal{T}_s = \{T_1, T_2, \ldots, T_n\}$ by a model π_θ such that the conditional probability $P_\theta(\mathcal{T}_s|q, \mathcal{T}, \mathcal{R})$ is maximized.

3.2 Preliminary

To enable the LLM to use the search engine tool, the schema of the search engine $\mathcal{R}$ is included in its prompt. The schema defines the search engine's callable interface, including the tool identifier and the required query argument fields, enabling the LLM to generate well-formed tool invocations in a structured format. At each step, the LLM produces a tool call enclosed by the special tokens `<tool_call>` and `</tool_call>`, where the enclosed content is a JSON object whose fields conform to the predefined schema. The search engine then executes the corresponding query and returns the retrieved results to the LLM as external feedback. Based on the feedback, the LLM can proceed to the next search round by following the same process.

3.3 Category-Constrained Tool Discrimination

Large-scale tool repositories exacerbate the similarity among tools, resulting in search outcomes that often include tools with similar functionality or semantics. This requires the model to understand and distinguish between similar tools within documents at each search step. To address this challenge, we propose category-constrained tool discrimination, which leverages tool categories to construct a strongly constrained challenge environment for tool search. The core idea is to transform tool retrieval from a global search over the entire repository into a category-constrained search within a functionally similar subspace.

Specifically, as shown in Fig. 1, each tool $T \in \mathcal{T}$ is assigned to a specific category C based on its functionality (*e.g.*, social, sports, finance). $\mathcal{T}$ is represented as $\{\mathcal{T}_{C_1}, \mathcal{T}_{C_2}, \ldots, \mathcal{T}_{C_m}\}$, where each subset $\mathcal{T}_{C_i}$ corresponds to a category-specific set of tools. We equip the search engine $\mathcal{R}$ with a subcategory search function, which is integrated into its schema as shown in Appendix G. At each search, Each search step can be formalized as follows:

$$D = \mathcal{R}(C_*, Q \mid \mathcal{T}_{C_*}), \tag{1}$$

where D is the tool documents that are searched within a specific subset of tools $\mathcal{T}_{C_*}$ based on a given category C_* and query Q. Accessing tools by category in large-scale repositories creates a challenging scenario, which greatly increases the similarity between search tools. This forces LLMs to focus on distinguishing functionally similar tools, rather than optimizing the query for fine-grained semantic search. We employ the RL training approach for tool selection described in Section 3.4, applying it in a two-phase training process. In the first phase, the LLM is trained using 30% of the data in a subcategory-specific search setting, focusing on distinguishing tools with functional similarities. In the second phase, the LLM is trained in a global search setting, learning to perform fine-grained semantic searches across the entire tool repository. Based on the progressive learning strategy, we guide LLMs to discriminate among similar tools and search effectively within large-scale tool repositories.

3.4 Reinforcement Learning For Tool Selection

3.4.1 Event-level Search Modeling.

Tools often exhibit dependencies and interface constraints, leading to non-executable tool chains when tool compositions are incompatible. To reliably identify compatible tool combinations, search and planning operate collaboratively during the multi-turn search process: The LLM needs to adjust its plan for a candidate tool chain by reasoning about tool composition based on current search results. However, existing RL-based approaches do not explicitly model the tool search process, limiting the ability to improve the LLM's search for compatible tool combinations. To address this challenge, we propose event-level search modeling, which integrates the search process with the RL objective by

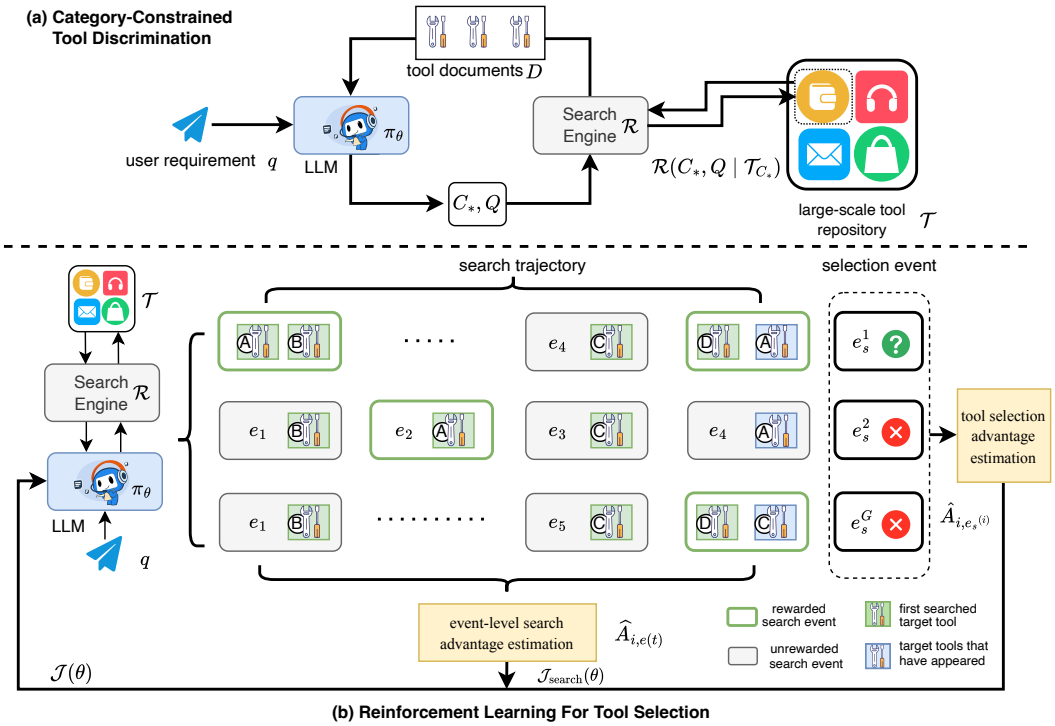

Figure 1: Overview of ToolSearcher. (a) Illustration of the category-constrained tool discrimination. (b) Illustration of RL for tool selection that contains event-level search modeling and trajectory-aligned credit allocation.

measuring the significance of individual search events for tool composition within each trajectory, thereby enhancing LLMs' ability to search and plan for compatible tool combinations.

Specifically, as shown in Fig. 1, we explicitly model the search trajectory $\mathcal{S}$ as a sequence of search events $\mathcal{E} = (e_1, e_2, \ldots, e_k)$, where e_k denotes the event at step k. Each event is defined as $e_k = (a_k, \mathcal{R}(a_k))$, where a_k is the model action at step k, which is a tool call to the search engine, and $\mathcal{R}(a_k)$ denotes the search tool sets. To prevent LLMs from learning and memorizing tokens from retrieved tool documents, we follow prior work [18] and apply loss masking to retrieved tokens, ensuring that the policy-gradient objective is computed only over LLM-generated tokens and excludes retrieved content $\mathcal{R}(a_k)$ from the optimization process. The set of events $\mathcal{E}^*$ that successfully search previously uncovered target tools for the first time, and the target tools $\mathcal{T}_{e_j}$ searched by specific events e_j, are defined as:

$$\begin{aligned} \mathcal{E}^* &= \{e_j \mid 1 \le j \le k,\ \mathcal{T}_{e_j} \neq \emptyset\}, \\ \mathcal{T}_{e_j} &= \mathcal{M}(e_j) \cap (\mathcal{T}_s \setminus \mathcal{M}(\mathcal{E}_{<j})), \end{aligned} \tag{2}$$

where $\mathcal{M}(\cdot)$ denotes the search tools obtained from the search events. Considering the potential inaccuracies caused by the search engine's performance and the necessity of enhancing the model's ability to recognize target tools, we optimize only search events $\mathcal{E}^*$ that involve previously unsearched target tools and avoid optimizing events that re-search tools already searched. To enhance LLMs' reasoning about tool interface compatibility within the current search process, we maximize the following event-level optimization objective:

$$\mathcal{J}_{\text{search}}(\theta) = \mathbb{E}_{\mathcal{S}_{\text{search}} \sim \pi_\theta} \left[\prod_{j=1}^{k} P_\theta(e_j | \mathcal{E}_{<j})^{\mathbb{1}_{e_j \in \mathcal{E}^*}} \right]. \tag{3}$$

Inspired by group relative policy optimization (GRPO) [42], we achieve the above objective by performing event-level advantage optimization across search events. Specifically, for each user requirement q that samples drawn from the dataset $\mathcal{D}$, the policy LLM $\pi_{\theta_{\text{old}}}$ generate G search

trajectories $\{\mathcal{S}_i\}_{i=1}^{G}$, the RL objective utilizing search engine $\mathcal{R}$ is formulated as follows:

$$\begin{aligned}&\mathcal{J}_{\text{search}}(\theta) = \mathbb{E}[q \sim \mathcal{D}, \{\mathcal{S}_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\mathcal{S}|q, \mathcal{T}, \mathcal{R})] \\ &\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathcal{S}_i|}\sum_{t=1}^{|\mathcal{S}_i|}\left\{\min\left[w_{i,t}(\theta,\mathcal{R})\hat{A}_{i,e(t)}, \text{clip}\left(w_{i,t}(\theta,\mathcal{R}), 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,e(t)}\right] - \beta\mathbb{D}_{\text{KL}}\left(\pi_\theta||\pi_{\text{ref}}\right)\right\},\end{aligned} \tag{4}$$

where ϵ and β are hyper-parameters, $\mathbb{D}_{\text{KL}}$ is a KL penalty term. The importance ratio $w_{i,t}(\theta, \mathcal{R})$ and event-level search advantage $\widehat{A}_{i,e(t)}$ of token $\mathcal{S}_{i,t}$ are:

$$\begin{aligned}&w_{i,t}(\theta,\mathcal{R}) = \frac{\pi_\theta(\mathcal{S}_{i,t} \mid q, \mathcal{R}, \mathcal{S}_{i,<t})}{\pi_{\theta_{\text{old}}}(\mathcal{S}_{i,t} \mid q, \mathcal{R}, \mathcal{S}_{i,<t})}, \\ &\widehat{A}_{i,e(t)} = \begin{cases}\max\limits_{T\in\mathcal{T}_{e(t)}}\left[\dfrac{r(T,\mathcal{S}_i) - \text{mean}\left(\{r(T,\mathcal{S}_j)\}_{j=1}^{G}\right)}{\text{std}\left(\{r(T,\mathcal{S}_j)\}_{j=1}^{G}\right)}\right], & e(t) \in \mathcal{E}_i^*, \\ 0, & \text{otherwise.}\end{cases}\end{aligned} \tag{5}$$

where $e(t)$ denotes the search event where the token $\mathcal{S}_{i,t}$ is located, r is a binary function that indicates whether the target tool T has been searched in the search trajectory $\mathcal{S}$ (1 if searched, 0 otherwise). The event-level search advantage $\widehat{A}_{i,e(t)}$ is defined as the maximum advantage within the set of target tools $\mathcal{T}_{e(t)}$, which are first searched by the event $e(t)$ in the trajectory $\mathcal{S}_i$. By comparing the advantage gap between a search event and other trajectories, the event-level search advantage measures the significance of each search event within its respective trajectory. Therefore, we explicitly model the impact of search events on the RL objective by adjusting the gradient of each event within the trajectory (see Appendix F for gradient analysis).

3.4.2 Trajectory-Aligned Credit Allocation.

Tool selection is a complex process involving planning, searching, and decision-making. Even among trajectories for the same task, progress may differ substantially: some complete the task, others gather sufficient tool information yet still struggle with tool selection, while some remain at an earlier stage of the search process. Therefore, a desirable credit assignment should account for the stage of each trajectory and allocate rewards that drive progress toward successful tool selection. To address this challenge, we propose trajectory-aligned credit allocation, which automatically focuses learning signals on unmastered steps by removing rewards for selection events associated with unfinished tool searches and rewards for search events that are mastered in trajectories.

Specifically, for trajectories $\{\mathcal{S}_i, e_{\text{s}}^{(i)}\}_{i=1}^{G}$, the final RL objective of i-th trajectory is formulated as follows:

$$\begin{aligned}&\mathcal{J}^{(i)}(\theta) = \mathcal{J}_{\text{search}}^{(i)}(\theta) + \mathbb{E}[e_{\text{s}}^{(i)} \sim \pi_{\theta_{\text{old}}}(e_{\text{s}}^{(i)}|q, \mathcal{S}_i)] \\ &\sum_{t=1}^{|e_{\text{s}}^{(i)}|}\min[\frac{\pi_\theta(e_{\text{s},t}^{(i)} \mid q, \mathcal{S}_i, e_{\text{s},<t}^{(i)})}{\pi_{\theta_{\text{old}}}(e_{\text{s},t}^{(i)} \mid q, \mathcal{S}_i, e_{\text{s},<t}^{(i)})}\hat{A}_{i,e_{\text{s}}^{(i)}}, \text{clip}(\frac{\pi_\theta(e_{\text{s},t}^{(i)} \mid q, \mathcal{S}_i, e_{\text{s},<t}^{(i)})}{\pi_{\theta_{\text{old}}}(e_{\text{s},t}^{(i)} \mid q, \mathcal{S}_i, e_{\text{s},<t}^{(i)})}, 1-\epsilon, 1+\epsilon)\hat{A}_{i,e_{\text{s}}^{(i)}}],\end{aligned} \tag{6}$$

where the tool selection advantage $\hat{A}_{i,e_{\text{s}}^{(i)}}$ is computed by the selection reward r_s if the search process $\mathcal{S}_i$ includes all the target tools $\mathcal{T}_s$:

$$\hat{A}_{i,e_{\text{s}}^{(i)}} = \begin{cases}\dfrac{r_s(\mathcal{T}_s, e_{\text{s}}^{(i)}|\mathcal{S}_i) - \text{mean}(\{r_s(\mathcal{T}_s, e_{\text{s}}^{(i)}|\mathcal{S}_i)\}_{j=1}^{G})}{\text{std}\left(\{r_s(\mathcal{T}_s, e_{\text{s}}^{(i)}|\mathcal{S}_i)\}_{j=1}^{G}\right)}, & \mathcal{M}(\mathcal{S}_i) \cap \mathcal{T}_s = \mathcal{T}_s, \\ 0, & \text{otherwise.}\end{cases} \tag{7}$$

The selection reward r_s is a binary function that equals 1 if the selection event $e_{\text{s}}^{(i)}$ matches the target tools $\mathcal{T}_s$, and 0 otherwise. For the selection stage, this RL objective prevents the model from making unrealistic guesses for tool selections based on past training outcomes or tools that were searched during the search trajectory. For the search stage, the event-level search advantage $\widehat{A}_{i,e(t)}$ is set to zero for search events involving corresponding target tools that have already been mastered within the group. As a result, as shown in Fig. 1, $\mathcal{J}_{\text{search}}^{(i)}(\theta)$ eliminates the reward for search capabilities related to tools that are mastered within the group (*i.e.*, colored in gray), while rewarding search events within trajectories that outperform other trajectories in searching for the target tools (*i.e.*, colored in green). Based on the above design, we allocate rewards for multiple search events and selection events in trajectories at different stages of progress, ensuring that the rewards correspond to the advancement made relative to overall progress.

4 Experimental Setup

4.1 Datasets and Metrics.

To evaluate the effectiveness of ToolSearcher in tool selection at scale, we use StableToolBench [14, 36] as the training and test dataset. StableToolBench contains the APIs of 16k tools across 49 diverse categories and covers various scenarios: **single-tool instructions (I1)**, **intra-category multi-tool instructions (I2)**, and **intra-collection multi-tool instructions (I3)**. To evaluate the robustness on out-of-distribution datasets, we conduct the evaluation on AppWorld [50], which is a high-quality execution environment featuring 9 day-to-day apps operable via 457 APIs and populated with realistic digital activities simulating the lives of approximately 100 fictitious users. To evaluate tool selection, we assess the results using several metrics: F1, Recall, Precision, and Match, which measure the accuracy of the selected tools compared to the ground truth. Additionally, we use SRecall to measure the effectiveness of retrieving the target tools during the search stage. For the AppWorld evaluation, we follow their predefined method and utilize FullCodeRefl with gpt-5-mini [2] as the agent to generate code and execute tasks based on the tool selection results. The key metrics reported include Task Goal Completion (TGC) and Scenario Goal Completion (SGC). For further details on the datasets and metrics, please refer to Appendix B.

4.2 Baselines.

We choose baselines from two paradigms of using search engines. For the retrieval-augmented generation paradigm, we adopt **RAG** [24] and $\textbf{RAG}_{\textbf{SFT}}$, where the latter is the supervised fine-tuned (SFT) [5] model. We further include **ToolEmbed** [30], which combines a fine-tuned Qwen3-Embedding-0.6B retriever with our supervised fine-tuned LLM. For the paradigm that treats the search engine as an external tool, we denote the untrained baseline as **Multi-turns**. We also include several mainstream RL methods for comparison, including **Search-R1** [18], **GSPO** [60], **GDPO** [28], **MARAG-R1** [32]. We additionally compare with two methods for tool-using agents: **ToolPO** [27], the RL-based training method in DeepAgent for end-to-end interactive tool use, and **ToolQP** [11], which relies on teacher-annotated plans to guide multi-step tool retrieval. To enable autonomous multi-turn tool calls with the search engine, we choose Qwen2.5-7B-Instruct [54] as the baseline, as it has strong and stable tool-calling capabilities for interacting with the search engine. To further verify the feasibility of our method on smaller-scale models, we also adopt Qwen3-4B-Instruct [53] as an additional baseline. We also compare with a range of leading general-purpose LLMs to assess their tool-selection capabilities. Further details are provided in Appendix C.

4.3 Implementation Details

Considering the instruction-following and tool-calling capabilities, we conduct experiments on Qwen2.5-7B-Instruct [54] and Qwen3-4B-Instruct [53]. For tool search, we use the tools from StableToolBench as the large-scale tool repository. Considering the limitation of the training context length, we set the number of retrieved tool documents to 5 per turn for all multi-turn methods. Based on the tool retrieval performance comparison, we finally adopt Qwen3-Embedding-0.6B [57] as the retriever for the search engine. During training, we set the maximum number of interaction turns to 8 and sample 5 rollouts for each prompt. Further implementation details are provided in Appendix D.

5 Experimental Analysis

5.1 Main Results

To evaluate the effectiveness of ToolSearcher in large-scale tool repositories, we conduct tool selection experiments on StableToolBench using Qwen2.5-7B-Instruct and Qwen3-4B-Instruct. Table 1 reports the experimental results of our model and the compared baselines. The experimental results show that: (1) ToolSearcher achieves substantial overall performance gains across backbone LLMs of different scales on all evaluation metrics, indicating both its effectiveness and its feasibility for smaller-scale LLMs. For example, on Qwen2.5-7B-Instruct, ToolSearcher improves the overall F1 score from 9.8% to 51.3% and the Match score from 4.6% to 27.8% compared with the multi-turn baseline. Similarly,

[2] https://developers.openai.com/api/docs/models/all

Table 1: Overall performance and per-scenario F1 scores on Stabletoolbench.

Method	Overall				I1			I2		I3
	F1	Recall	Precision	Match	Inst.	Tool	Cate.	Inst.	Cate.	Inst.
Gpt-5.5	**0.569**	**0.572**	**0.590**	0.336	**0.675**	**0.635**	**0.645**	0.474	0.454	0.316
Claude-Opus-4-7	0.560	0.556	0.581	0.369	0.645	0.580	0.597	**0.550**	**0.460**	**0.410**
Claude-Opus-4-6	0.552	0.555	0.565	**0.388**	0.623	0.549	0.637	0.517	0.457	0.408
Qwen3.5-397B-A17B	0.501	0.566	0.491	0.259	0.576	0.554	0.555	0.470	0.400	0.288
Qwen3-235B-A22B-Instruct	0.497	0.527	0.500	0.260	0.581	0.560	0.539	0.489	0.389	0.234
Gpt-4.1-mini	0.479	0.510	0.483	0.234	0.546	0.520	0.512	0.465	0.408	0.285
Qwen3.6-27B	0.461	0.490	0.466	0.247	0.537	0.514	0.529	0.418	0.374	0.198
Qwen3-14B	0.443	0.492	0.430	0.182	0.537	0.446	0.485	0.407	0.388	0.256
Qwen3-32B	0.423	0.475	0.410	0.178	0.496	0.459	0.476	0.374	0.360	0.213
Qwen3-8B	0.378	0.481	0.343	0.122	0.468	0.436	0.439	0.330	0.233	0.213
Qwen2.5-7B-Instruct										
RAG	0.194	0.285	0.174	0.075	0.257	0.226	0.230	0.139	0.105	0.126
RAG_{SFT}	0.441	0.433	0.470	0.250	0.536	0.546	0.522	0.316	0.322	0.170
ToolEmbed	0.337	0.348	0.347	0.177	0.376	0.295	0.408	0.340	0.338	0.152
Multi-turns	0.098	0.110	0.100	0.046	0.119	0.161	0.099	0.039	0.065	0.054
Search-R1	0.327	0.311	0.361	0.152	0.415	0.321	0.320	0.321	0.297	0.194
GSPO	0.444	0.417	0.505	0.221	0.535	0.512	0.565	0.331	0.278	0.196
GDPO	0.496	0.487	0.524	0.255	0.577	**0.567**	0.555	0.427	0.404	0.228
MARAG-R1	0.448	0.425	0.500	0.204	0.567	0.537	0.557	0.305	0.279	0.211
ToolPO	0.391	0.420	0.388	0.191	0.461	0.405	0.382	0.452	0.346	0.177
ToolQP	0.440	0.493	0.432	0.222	0.503	0.450	0.504	0.433	0.362	0.260
ToolSearcher	**0.513**	**0.511**	**0.534**	**0.278**	**0.595**	0.561	**0.577**	**0.485**	**0.405**	**0.294**
Qwen3-4B-Instruct										
RAG	0.387	**0.540**	0.384	0.220	0.438	0.426	0.432	0.324	0.356	0.221
RAG_{SFT}	0.427	0.416	0.463	0.235	0.558	0.460	0.447	0.391	0.352	0.167
ToolEmbed	0.449	0.447	0.472	0.284	**0.637**	**0.625**	**0.625**	0.413	0.364	0.271
Multi-turns	0.408	0.439	0.410	0.169	0.454	0.506	0.420	0.317	0.349	0.272
Search-R1	0.505	0.504	0.516	0.307	0.583	0.604	0.541	0.409	0.465	0.215
GSPO	0.473	0.453	0.518	0.258	0.576	0.572	0.576	0.305	0.362	0.221
GDPO	0.518	0.513	0.537	0.305	0.576	0.616	0.574	0.393	**0.505**	0.238
MARAG-R1	0.465	0.439	0.527	0.225	0.588	0.587	0.559	0.265	0.347	0.196
ToolPO	0.457	0.461	0.468	0.302	0.524	0.381	0.528	**0.465**	0.459	0.281
ToolQP	0.474	0.487	0.498	0.216	0.599	0.538	0.580	0.344	0.287	**0.314**
ToolSearcher	**0.531**	0.527	**0.546**	**0.316**	0.599	0.617	0.586	0.422	0.491	0.284

on Qwen3-4B-Instruct, ToolSearcher increases the overall F1 score from 40.8% to 53.1% and the Match score from 16.9% to 31.6%. (2) Compared with other training methods, ToolSearcher achieves substantial and stable improvements across both backbone models in overall performance, particularly in intra-collection multi-tool scenarios (I3) and unseen-instruction generalization (Inst.). For example, on Qwen2.5-7B-Instruct, ToolSearcher outperforms the second-best method (*i.e.*, GDPO) by 1.7% in overall F1. ToolSearcher also achieves the highest F1-Inst. scores across all three scenarios, reaching 59.5%, 48.5%, and 29.4% on I1, I2, and I3, respectively. This advantage is especially notable in the challenging I3 scenario, where ToolSearcher surpasses GDPO, MARAG-R1, GSPO, and Search-R1 by 6.6%, 8.3%, 9.8%, and 10.0%, respectively. A similar performance trend can be observed on Qwen3-4B-Instruct, further demonstrating the stable improvements achieved by our method. **Overall, ToolSearcher shows substantial and stable tool selection performance compared with other methods, especially on unseen instructions and difficult scenarios where complex tool search and compatible tool combinations are required.**

5.2 Tool-Selection Capabilities Across Diverse Scenarios

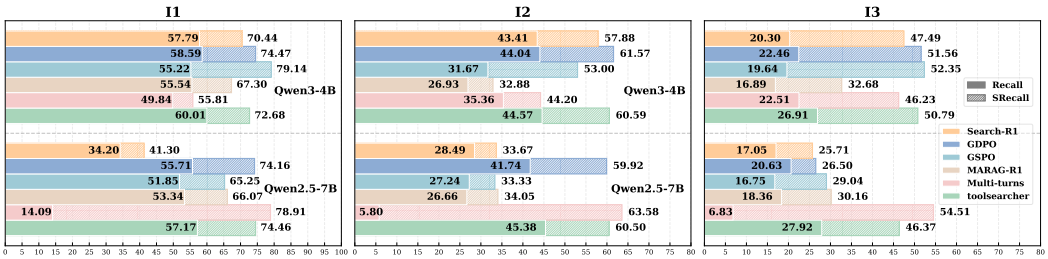

Figure 2: Tool-Selection Capabilities Across Diverse Scenarios.

Given that tool selection involves balancing search-stage retrieval and final-stage decision-making, we analyze model performance using SRecall and Recall across the three scenarios of StableToolBench.

As illustrated in Fig. 2, the experimental results indicate that **our method effectively balances these two capabilities across different backbone LLMs, thereby improving final tool-selection performance**: (1) For Qwen-4B-Instruct, our method improves both search-stage and final-selection recall over the base model, achieving consistent gains in final tool-selection performance compared with other methods across the three scenarios. Notably, in scenario I3, our method improves Recall to 26.91%, whereas other methods degrade the base model's final-selection performance while shifting their focus to the search stage. (2) Qwen-7B-Instruct shows high search-stage recall but weak final-selection ability, as it struggles to understand and choose among the retrieved tools. Our method slightly reduces SRecall but substantially improves the model's ability to interpret retrieved results and make final decisions, yielding Recall improvements of 43.08%, 39.58%, and 21.09% on I1/I2/I3, respectively. Compared with other RL methods, our method improves tool understanding and final selection, targeting the core missing ability and achieving a stronger overall trade-off.

5.3 Generalization Capability to Out-of-Distribution

We evaluate different RL methods on the out-of-distribution (OOD) benchmark AppWorld, where OOD refers to a distribution shift in task scenarios and tool usage patterns, particularly involving stateful app tools. As shown in Table 2, our method consistently achieves the best overall performance across both model scales. For tool selection, ToolSearcher obtains the highest F1, Recall, and SRecall on both LLMs, indicating more accurate and stable tool search and decision-making under distribution shifts. For tool calling performance, our method also achieves the best Task Goal Completion (TGC) and Scenario Goal Completion (SGC) on average. For example, ToolSearcher outperforms the second-best method (*i.e.*, GDPO) by 5.7% in both overall TGC and SGC on Qwen2.5-7B-Instruct. Similarly, on Qwen3-4B-Instruct, ToolSearcher outperforms the second-best method (*i.e.*, Search-R1) by 5.8% and 2.9% in TGC and SGC, respectively. It maintains consistent and better performance compared to all baseline methods across both Difficulty-1 and Difficulty-2 settings. In particular, in the highly challenging Difficulty-2 setting, which involves up to 8 unique APIs, our method consistently outperforms all baselines, while most baselines (*e.g.*, Search-R1, GSPO, and MARAG-R1) achieve near-zero SGC. Overall, compared with other methods, our method achieves more consistent performance in both tool selection and downstream execution under OOD scenarios. Further experiments are provided in Appendix I.

Table 2: Comparison of RL Methods Performance on the Out-of-Distribution Dataset AppWorld.

Method	Tool Selcetion				Tool Calling					
					Avg		Difficulty-1		Difficulty-2	
	F1	Recall	Precision	SRecall	TGC	SGC	TGC	SGC	TGC	SGC
Qwen2.5-7B-Instruct										
Multi-turns	0.327	0.263	0.483	0.428	0.133	0.029	0.140	0.053	0.125	0.00
Search-R1	0.455	0.381	0.640	0.591	0.181	0.086	0.298	0.158	0.042	0.00
GSPO	0.478	0.411	0.657	0.601	0.123	0.057	0.175	0.105	0.062	0.00
GDPO	0.483	0.425	0.626	0.693	0.277	0.171	0.351	0.263	**0.188**	**0.062**
MARAG-R1	0.463	0.402	0.614	0.602	0.104	0.029	0.140	0.053	0.062	0.00
ToolSearcher	**0.514**	**0.444**	**0.670**	**0.697**	**0.334**	**0.228**	**0.456**	**0.368**	**0.188**	**0.062**
Qwen3-4B-Instruct										
Multi-turns	0.436	0.355	0.615	0.435	0.152	0.086	0.228	0.158	0.062	0.00
Search-R1	0.555	0.529	0.617	0.635	0.314	0.228	0.456	0.368	0.146	0.062
GSPO	0.400	0.317	**0.635**	0.375	0.133	0.057	0.193	0.105	0.062	0.00
GDPO	0.475	0.421	0.611	0.568	0.296	0.229	0.404	0.316	0.167	**0.125**
MARAG-R1	0.395	0.314	0.598	0.369	0.134	0.057	0.211	0.105	0.042	0.00
ToolSearcher	**0.562**	**0.543**	0.612	**0.664**	**0.372**	**0.257**	**0.526**	**0.368**	**0.188**	**0.125**

5.4 Ablation Study and Method Analysis

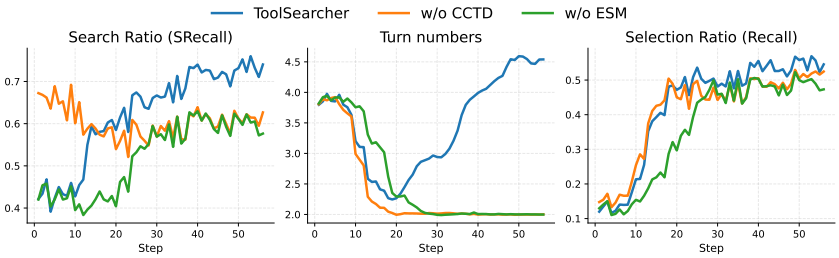

Figure 3: Training Curves of Metrics.

As illustrated in Table 3, we perform an ablation study to evaluate the contributions of individual components of our approach, namely Category-Constrained Tool Discrimination (CCTD), Event-level Search Modeling (ESM), and Trajectory-Aligned Credit Allocation (TCA), by removing each component separately on Qwen2.5-7B-Instruct. The experimental results show that: (1) No matter which component we drop, it hurts the overall performance of our model, which signals the

effectiveness of all three components in tool selection. (2) Removing TCA disables reward assignment based on trajectory-level progress, resulting in consistent performance drops across both datasets. On StableToolBench, the F1 score decreases from 51.3% to 46.1%, and SRecall drops from 68.0% to 53.8%. On AppWorld, the F1 score decreases from 51.4% to 46.9%, and SRecall drops from 69.7% to 56.5%. These results highlight that trajectory-aligned credit assignment provides more accurate and stable learning signals for tool selection at various stages of progress. (3) Removing ESM significantly decreases performance for tool searching, with the SRecall score dropping by 22.4% on StableToolBench and 39.4% on AppWorld, highlighting its importance in tool searching. In Fig. 3, the training curve for Qwen2.5-7B-Instruct illustrates that removing ESM reduces the average number of interaction rounds with the search engine from 4 to 2. In the absence of ESM, the model is constrained to result-based rewards, which discourage its active engagement with the search process. In contrast, our approach leverages event-level search rewards to facilitate continuous improvement, achieving superior performance in both the search phase and final selection. (4) As shown in Table 4, we evaluate the performance of the base model under different search settings. In the CCTD scenario, all metrics show a significant drop compared to the global search, demonstrating the effectiveness of CCTD in constructing challenging environments. The ablation results in Table 3 further confirm the importance of CCTD. For example, the F1 score, Recall, and Precision all exhibit substantial declines when CCTD is removed, highlighting its critical role in training for tool selection. Detailed ablation settings and gradient analysis of our method are provided in Appendix E and F.

Table 3: Ablation Study.

Datasets	Method	F1	Recall	Precision	SRecall
StableToolBench	**ToolSearcher**	**0.513**	**0.511**	**0.534**	**0.680**
	w/o CCTD	0.477	0.466	0.510	0.583
	w/o ESM	0.394	0.377	0.433	0.456
	w/o TCA	0.461	0.444	0.501	0.538
AppWorld	**ToolSearcher**	**0.514**	**0.444**	**0.669**	**0.697**
	w/o CCTD	0.433	0.375	0.576	0.452
	w/o ESM	0.328	0.280	0.447	0.303
	w/o TCA	0.469	0.431	0.567	0.565

Table 4: Comparsion under different Search Setting.

Model	Search Setting	F1	Recall	Precision	SRecall
Qwen2.5-7B-Instruct	CCTD	0.088	0.099	0.087	0.401
	Global search	**0.098**	**0.110**	**0.100**	**0.724**
Qwen3-4B-Instruct	CCTD	0.265	0.280	0.272	0.344
	Global search	**0.408**	**0.439**	**0.410**	**0.516**

6 Conclusion

This paper presents a novel reinforcement learning framework ToolSearcher for tool selection in large-scale repositories. By category-constrained tool discrimination and event-level search modeling, we significantly enhance the LLM's ability to distinguish similar tools and optimize tool compositions through multi-turn search. Additionally, the trajectory-aligned credit allocation enables fine-grained optimization during the selection process, ensuring that the rewards correspond to the advancement made relative to overall progress. Experimental results demonstrate that ToolSearcher outperforms several reinforcement learning baseline methods, both in large-scale tool selection scenarios StableToolBench and out-of-distribution scenarios AppWorld. These results confirm ToolSearcher's effectiveness in tasks requiring iterative search and complex tool composition, providing valuable insights for future applications of LLMs in large-scale tool selection.

7 Acknowledgements

This research was supported by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (2025C02022), the National Natural Science Foundation of China (No.62507040), the Ningbo "Yongjiang Talent Program" Youth Innovation Project (2024A-156-G), Ant Group, the National Natural Science Foundation of China (No.62293555).

References

[1] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, 2024.

[2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024.

[3] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2023.

[4] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

[5] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[6] Zhenlong Dai, Bingrui Chen, Zhuoluo Zhao, Xiu Tang, Sai Wu, Chang Yao, Zhipeng Gao, and Jingyuan Chen. Less is more: Adaptive program repair with bug localization and preference learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pages 128–136, 2025.

[7] Zhenlong Dai, Chang Yao, WenKang Han, Ying Yuan, Zhipeng Gao, and Jingyuan Chen. Mpcoder: Multi-user personalized code generator with explicit and implicit style representation learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3765–3780, 2024.

[8] Zhenlong Dai, Zhuoluo Zhao, Xiu Tang, Sai Wu, Chang Yao, Zhipeng Gao, and Jingyuan Chen. Enhancing repository-level code completion with reinforcement learning in real-world scenarios. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, pages 48512–48528, 2026.

[9] Zhenlong Dai, Zhuoluo Zhao, Hengning Wang, Xiu Tang, Sai Wu, Chang Yao, Zhipeng Gao, and Jingyuan Chen. Learner-tailored program repair: A solution generator with iterative edit-driven retrieval enhancement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 30467–30475, 2026.

[10] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[11] Wei Fang and James Glass. Beyond single-shot: Multi-step tool retrieval via query planning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 42119–42144, 2026.

[12] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32, 2023.

[13] Daya Guo, Shuai Lu, Nan Duan, Yanlin Wang, Ming Zhou, and Jian Yin. Unixcoder: Unified cross-modal pre-training for code representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7212–7225, 2022.

[14] Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11143–11156, 2024.

[15] Yanzhe Hu, Shenao Wang, Tianyuan Nie, Yanjie Zhao, and Haoyu Wang. Understanding large language model supply chain: Structure, domain, and vulnerabilities. *arXiv preprint arXiv:2504.20763*, 2025.

[16] Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 7969–7992, 2023.

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

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

[19] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, 2017.

[20] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 6769–6781, 2020.

[21] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback. *arXiv preprint arXiv:2312.14925*, 2023.

[22] Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025.

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

[24] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[25] Xiaonan Li, Changtai Zhu, Linyang Li, Zhangyue Yin, Tianxiang Sun, and Xipeng Qiu. Llatrieval: Llm-verified retrieval for verifiable generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5453–5471, 2024.

[26] Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438, 2025.

[27] Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, et al. Deepagent: A general reasoning agent with scalable toolsets. In *Proceedings of the ACM Web Conference 2026*, pages 2219–2230, 2026.

[28] Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.

[29] Yunqi Liu, Tong Niu, Zitong Wang, Zhenlong Dai, Yuqi Qing, Weiqiang Wang, and Jian Liu. Egobench: An interactive egocentric multimodal benchmark for tool-using agents. *arXiv preprint arXiv:2605.27820*, 2026.

[30] Xuan Lu, Haohang Huang, Rui Meng, Yaohui Jin, Wenjun Zeng, and Xiaoyu Shen. Tools are under-documented: Simple document expansion boosts tool retrieval. In *International Conference on Learning Representations*, volume 2026, pages 141631–141653, 2026.

[31] Haoran Luo, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, et al. Hypergraphrag: Retrieval-augmented generation via hypergraph-structured knowledge representation. *arXiv preprint arXiv:2503.21322*, 2025.

[32] Qi Luo, Xiaonan Li, Yuxin Wang, Tingshuo Fan, Yuan Li, Xinchi Chen, and Xipeng Qiu. Marag-r1: Beyond single retriever via reinforcement-learned multi-tool agentic retrieval. *arXiv preprint arXiv:2510.27569*, 2025.

[33] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

[34] Richard Y Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems*, 37:116617–116637, 2024.

[35] Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

[36] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

[37] Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343, 2025.

[38] 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.

[39] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc, 2009.

[40] 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.

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

[42] 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.

[43] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297, 2025.

[44] Zhengliang Shi, Shen Gao, Lingyong Yan, Yue Feng, Xiuyi Chen, Zhumin Chen, Dawei Yin, Suzan Verberne, and Zhaochun Ren. Tool learning in the wild: Empowering language models as automatic tool agents. In *Proceedings of the ACM on Web Conference 2025*, pages 2222–2237, 2025.

[45] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

[46] Aivin V Solatorio. Gistembed: Guided in-sample selection of training negatives for text embedding fine-tuning. *arXiv preprint arXiv:2402.16829*, 2024.

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

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

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

[50] Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076, 2024.

[51] Boshi Wang, Hao Fang, Jason Eisner, Benjamin Van Durme, and Yu Su. Llms in the imaginarium: tool learning through simulated trial and error. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10583–10604, 2024.

[52] Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*, 2023.

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

[54] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[55] Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel Campos. Arctic-embed 2.0: Multilingual retrieval without compromise. *arXiv preprint arXiv:2412.04506*, 2024.

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

[57] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

[58] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

[59] Chujie Zheng, Pei Ke, Zheng Zhang, and Minlie Huang. Click: Controllable text generation with sequence likelihood contrastive learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1022–1040, 2023.

[60] 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.

[61] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

[62] Xiaoyan Zhou, Feiran Liang, Zhaojie Xie, Yang Lan, Wenjia Niu, Jiqiang Liu, Haining Wang, and Qiang Li. A large-scale fine-grained analysis of packages in open-source software ecosystems. *arXiv preprint arXiv:2404.11467*, 2024.

Appendix

A Limitations

Although our approach demonstrates improved performance over baseline models on the StableToolBench training set and out-of-distribution settings like AppWorld, there are several limitations to consider. First, the training data used in our method is synthesized based on LLMs (from Stabletoobench's training data), where the majority of tool combinations are parallel rather than sequential, even when cross-category tool compositions are involved. This results in a weak progression in tool combination usage, which is further exacerbated in complex scenarios. In contrast, AppWorld's APIs are mostly stateful, meaning they alter user-related information and impact database states. Due to the lack of state-awareness in the synthesized training data, our model's performance improvement in AppWorld remains limited.

Beyond large-scale tool selection, the challenge of effectively combining stateful data and APIs with tight cross-category links remains an open problem, and we consider this as future work. Additionally, despite achieving superior performance over baseline models in AppWorld, our approach is constrained by the context window during training, limiting the interaction rounds to 8 (equivalent to a total of 5,000 tokens). However, the average number of APIs involved in an AppWorld scenario exceeds 8, which presents a challenge even though our method outperforms other RL methods. However, the average number of APIs involved in the AppWorld scenario can exceed 8. We believe a feasible strategy to address this is to summarize interaction context information appropriately, thereby shortening the context window. Future work will focus on exploring how to effectively implement this during both training and inference to handle scenarios that involve more complex tool combinations.

B Dataset and Metric details

B.1 StableToolbench Dataset details.

StableToolbench consists of 765 samples, with the maximum and mean API call counts required to complete the tasks being 6 and 2.35, respectively. StableToolbench gather 16,464 representational state transfer (REST) APIs from RapidAPI. RapidAPI is a leading API marketplace that connects developers with thousands of real-world APIs, streamlining the process of integrating diverse services into applications. Developers All APIs in RapidAPI can be classified into 49 coarse-grained categories, such as sports, finance, and weather. The categories associate an API with the most relevant topic. Additionally, the hub also provides 500+ fine-grained categorization called **collections** (*e.g.*, Chinese APIs and database APIs.) APIs in the same collection share a common characteristic and often have similar functionalities or goals. It evaluates the generalization ability across different scenarios: (1) **Inst.**: unseen instructions for the same set of tools, (2) **Tool**: unseen tools from the same category, and (3) **Cate.**: unseen tools from a different category than those in the training data.

StableToolbench perform experiments on three scenarios: **single-tool instructions (I1)**, **intra-category multi-tool instructions (I2)**, and **intra-collection multi-tool instructions (I3)**. For I1, it conduct the evaluation for the aforementioned three levels (I1-Inst., I1-Tool, and I1-Cat.); for I2, since the training instructions already involve different tools of the same category, it only perform level 1 and level 3 for the generalization evaluation (I2-Inst. and I2-Cat.); similarly, it only perform level 1

generalization for I3 (I3-Inst.) since it already covers instructions that involve various combinations of tools from different categories (the tools in a RapidAPI collection may come from different RapidAPI categories).

For the training data, we utilize their dataset, which is automatically constructed by sampling different combinations of APIs and crafting various instructions involving them. Specifically, they focus on two key aspects in the instruction generation process: (1) diversity: to train LLMs to handle a wide range of API usage scenarios, thereby boosting their generalizability and robustness; and (2) multi-tool usage: to mirror real-world situations that often demand the interplay of multiple tools, improving the practical applicability and flexibility of LLMs. We filtered out non-English data and unnatural instructions, particularly those containing API names, as they inadvertently revealed the target tool's API, leading to a subsequent decrease in the quality of the training data. The training dataset consists of 14,418 samples, which contains 11,329/2,253/836 samples for I1/I2/I3. Overall statistics of the dataset are given in Table 5.

Table 5: Core Information of Training Data.

Data Source	Number of Samples	API Count (Mean/Max)
I1	11,329	2.13/5
I2	2,253	2.30/5
I3	836	2.56/5
Total	14,418	2.18/5

B.2 Appworld Dataset details.

AppWorld Benchmark consists of complex tasks that cover everyday scenarios using the applications in the AppWorld Engine. These tasks are natural, challenging, and diverse, and are carefully designed with distractors and hurdles to require thorough reasoning. Each task typically involves multiple applications and requires the use of many APIs in an intricate workflow, with an average of 9.5 APIs and a maximum of 26 APIs per task. Since AppWorld tasks can often be completed through multiple valid paths, the required tool sequence is not unique. Therefore, for tool selection evaluation, we use the AppWorld training set as test set, which provides ground-truth annotations for the involved tools, while not using it for training. In contrast, the original AppWorld Test-N set does not provide tool-selection labels due to the multi-solution nature of the tasks Following the original AppWorld evaluation protocol, we evaluate tool selection on Test-N through execution outcomes, where the selected tools are used to generate and execute code, and performance is measured based on whether the final task goal is successfully completed.

AppWorld is adopted as an out-of-distribution robustness benchmark to evaluate ToolSearcher in an unseen tool environment. Notably, AppWorld places greater emphasis on stateful API interactions: it simulates rich user states, and API executions may modify user data. For tool selection evaluation, we include 147 tasks covering 49 distinct scenarios. For downstream evaluation "tool calling", we use a subset consisting of Difficulty (D)-1 and D-2, which contains 105 tasks across 34 scenarios. We exclude D-3 because it involves substantially more APIs, making code generation and execution by the downstream agent significantly more challenging; consequently, performance on D-3 may not accurately reflect the quality of tool selection.

B.3 Metric details.

For evaluating tool selection, we assess the accuracy of the results using several metrics: **F1**, which balances precision and recall to provide a comprehensive evaluation; **Recall**, which measures the proportion of relevant tools that were selected; **Precision**, which evaluates the proportion of selected tools that are actually relevant; and **Match**, which measures the exact match between the selected tools and the ideal set. Additionally, we calculate **SRecall** to determine the recall rate during the search phase, assessing the effectiveness of finding the target tools.

For the AppWorld evaluation, we follow their predefined method and utilize FullCodeRefl with gpt-5-mini [3] as the agent to generate code and execute tasks based on the tool selection result. FullCodeRefl

[3] https://developers.openai.com/api/docs/models/all

generates the entire code in one go. If execution fails, it is shown the error stack trace and is asked to reflect on its mistake (like Reflexion [45]) and retry. This repeats in a loop if it fails again. The key metrics reported include **Task Goal Completion (TGC)**, which represents the percentage of tasks for which all evaluation tests from the scenario agent passed successfully, and **Scenario Goal Completion (SGC)**, which reflects the percentage of task scenarios for which the agent passed all evaluation tests.

C Baselines.

We choose several mainstream RL methods for comparison:

(1) **SearchR1** [18]: an RL framework that supports LLM rollouts and direct optimization with a search engine. It incorporates retrieved-token masking to stabilize RL training, multi-turn interleaved reasoning and search for complex task solving, and an effective outcome reward function based on GRPO [42].

(2) **GSPO** [60]: Group Sequence Policy Optimization (GSPO), a RL algorithm for training large language models. The key innovation of GSPO lies in its theoretically grounded definition of importance ratio based on sequence likelihood [59]. Additionally, GSPO computes the normalized rewards as the advantages of multiple responses to a query, ensuring the alignment between sequence-level rewarding and optimization. In multi-turn RL scenarios, a finer-grained advantage adjustment than the sequence level may is desired. GSPO makes it possible to directly use the likelihoods returned by the inference engine for optimization, thereby avoiding the need for recomputation with the training engine. Following the token-level objective variant of GSPO, GSPO can be especially beneficial in scenarios like partial rollout and multi-turn RL.

(3) **GDPO** [28]: Group reward-Decoupled Normalization Policy Optimization (GDPO) is a policy optimization method designed for multi-reward RL in tool-calling scenarios. It addresses the limitation of directly applying GRPO, where different reward combinations may collapse into identical advantage values after normalization, weakening the training signal. GDPO decouples the normalization of individual rewards, better preserving their relative differences and enabling more accurate optimization over multiple tool-calling rewards, thereby improving training stability and generalization. In our setting, we design two rewards for tool calling: the recall of the search stage and the exact match of the final results, where the latter follows the practice in Search-R1.

(4) **MARAG-R1** [32]: MARAG-R1, a reinforcement-learned multi-tool framework that enables LLMs to dynamically coordinate multiple retrieval mechanisms for broader and more precise information access. MARAG-R1 equips the model with multiple retrieval tools and learns both how and when to use them through a two-stage training process: supervised fine-tuning followed by reinforcement learning by RLOO. This design allows the model to interleave reasoning and retrieval, progressively gathering sufficient evidence for corpus-level synthesis. In our large-scale tool retrieval setting, we find that keyword search and the use of multiple heterogeneous retrieval tools in MARAG-R1 perform poorly, making it difficult to retrieve the target tools. For a fair comparison, we therefore use the same search engine tool as in our method. Following the original MARAG-R1 reward design, we adopt the token-level F1 score to evaluate the tool selection results, the Document Coverage Reward to assess the completeness and precision of retrieved evidence, and the Tool Exploration Reward to encourage sufficient but not excessive tool usage.

(5) **ToolPO** [27]: the RL-based training method introduced in DeepAgent for end-to-end interactive tool use. We adapt it to multi-turn tool search and final API selection, following its token-level policy optimization with tool-related reward signals.

(6) **ToolQP** [11]: a query-planning-based method that decomposes user requests into sub-tasks to guide multi-round tool retrieval. Following its original setting, we use gpt-4.1-mini to annotate training plans and incorporate plan-level similarity scores to evaluate generated plans during training.

Training and Inference Setup. In our experiment, we carefully control the training setup to make the experimental comparison as fair as possible. For the RL method, all baselines and our method use the same training configuration (e.g., same retrieval model, the number of epochs, batch size, the maximum number of documents per retrieval, the number of interaction rounds, and rollout size for each prompt). For supervised fine-tuning, we ensure the same overall amount of training data

is observed by adjusting the number of epochs, since the RL method samples multiple answers for each prompt. The model is trained until convergence, and we report the best checkpoint according to validation performance. For the RAG paradigm, we use the top-100 retrieved documents as the context for both SFT and testing. This choice is motivated by the observation that a smaller top-k in single-round retrieval is often insufficient to recall the majority of target tools.

D Implementation Details

Following the instruction and tool-calling format of the search engine, we conduct experiments on Qwen2.5-7B-Instruct. To further examine the robustness of our method on smaller-scale models, we adopt Qwen3-4B-Instruct as an additional baseline. Evaluation is conducted separately on StableToolBench and AppWorld to assess in-domain and out-of-domain performance, respectively.

Tool Retriever for Search Engine. We evaluated the performance of different retrievers on StableToolBench, including BM25 [39], UniXcoder [13], GIST-Large [46], Arctic-Embed 2.0 [55], and the Qwen-Embedding series (0.6B, 4B, 8B) [57]. Among these models, Qwen-Embedding-0.6B achieved the best performance while maintaining a relatively small parameter size, reducing the computational load during RL training. Due to its superior performance, we adopt it as the retriever to implement the search engine functionality during both training and testing.

Training and Inference Settings. In training stage, for tool search, we use the tools from StableToolBench as the large-scale tool repository. To ensure a fair comparison while considering the training context length, we set the number of retrieved tool documents to 5 per turn for all multi-turn methods. Training is performed on a single node with 8 A100 (80G) GPUs. We use a total batch size of 256, and train for 1 epoch consisting of 56 steps on the training set. We train under the Category-Constrained Tool Discrimination setting for 30% of the data (17 steps), after which we switch back to global search. The maximum response sequence length is set to 5,000 tokens, with retrieved content capped at 768 tokens per turn. To optimize GPU memory usage, we enable gradient checkpointing and use Fully Sharded Data Parallel (FSDP) [58] with CPU offloading. For efficient LLM rollouts, we adopt sglang [61] with a tensor parallel size of 2 and GPU memory utilization ratio of 0.5. The rollout sampling uses a temperature of 1.0 and a top-p value of 1.0. The KL divergence regularization coefficient β and clip ratio ϵ are set to 0.001 and 0.2. We set the policy LLM learning rate to 1e-6 and sample 5 responses per prompt, following the GRPO implementation in Verl [43]. we set the maximum number of interaction turns to 8 for each prompt. The model is trained with a learning rate warm-up ratio of 0.285. During inference, we keep the number of retrieved documents at 5 per turn and set the maximum response sequence length to 30,000 tokens. The maximum number of interaction turns remains the same as in training (*i.e.*, 8 turns). To support sufficient reasoning and avoiding truncation, we increase the tokens of retrieved content to 4,096 tokens per turn. For all RL-based training baselines, we adopt the same hyperparameter settings as described above to ensure a fair comparison.

E Detailed Settings of Ablation Study

We perform an ablation study to evaluate the contributions of individual components of our approach, namely Category-Constrained Tool Discrimination (CCTD), Event-level Search Modeling (ESM), and Trajectory-Aligned Credit Allocation (TCA), by removing each component separately. (1) For Category-Constrained Tool Discrimination (CCTD), we remove category-level constraints and instead allow the model to perform global search over the entire tool space during the whole training process, rather than restricting search within sub-categories. (2) For Event-level Search Modeling (ESM), we remove event-level reward modeling and replace it with trajectory-level supervision, where rewards are assigned based only on the final outcome of the entire trajectory instead of intermediate search events. (3) For Trajectory-Aligned Credit Allocation, we do not constrain the reward for search events that discover the same target tool at the trajectory level and remove reward correction for tool selection in trajectories where the target tool has not been fully identified.

F Gradient Analysis

ToolSearcher. The RL objective of tool seraching using the search engine $\mathcal{R}$ is formulated as follows:

$$\begin{aligned}&\mathcal{J}_{\text{search}}(\theta) = \mathbb{E}[q \sim \mathcal{D}, \{\mathcal{S}_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\mathcal{S}|q, \mathcal{T}, \mathcal{R})] \\ &\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathcal{S}_i|}\sum_{t=1}^{|\mathcal{S}_i|}\left\{\min\left[w_{i,t}(\theta,\mathcal{R})\hat{A}_{i,e(t)}, \text{clip}\left(w_{i,t}(\theta,\mathcal{R}), 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,e(t)}\right] - \beta\mathbb{D}_{\text{KL}}\left(\pi_\theta||\pi_{\text{ref}}\right)\right\},\end{aligned} \tag{8}$$

Following the GRPO [42]'s Gradient Analysis (assume $\pi_\theta = \pi_{\theta_{\text{old}}}$ for simplified analysis), the gradient of $\mathcal{J}_{\text{search}}(\theta)$ is:

$$\begin{aligned}&\nabla_\theta \mathcal{J}_{Search}(\theta) = \mathbb{E}[q \sim \mathcal{D}, \{\mathcal{S}_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\mathcal{S}|q, \mathcal{T}, \mathcal{R})] \\ &\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathcal{S}_i|}\sum_{t=1}^{|\mathcal{S}_i|}\left[\hat{A}_{i,e(t)} + \beta\left(\frac{\pi_{ref}(\mathcal{S}_{i,t} \mid q, \mathcal{R}, \mathcal{S}_{i,<t})}{\pi_\theta(\mathcal{S}_{i,t} \mid q, \mathcal{R}, \mathcal{S}_{i,<t})} - 1\right)\right]\nabla_\theta \log \pi_\theta(\mathcal{S}_{i,t} \mid q, \mathcal{R}, \mathcal{S}_{i,<t}).\end{aligned} \tag{9}$$

The gradient coefficient for the token $\mathcal{S}_{i,t}$ from search-event $e(t)$ is :

$$GC_{search}(q, \mathcal{R}, t, \pi_\theta) = \hat{A}_{i,e(t)} + \beta\left(\frac{\pi_{ref}(o_{i,t}|o_{i,<t})}{\pi_\theta(o_{i,t}|o_{i,<t})} - 1\right), \tag{10}$$

where $\hat{A}_{i,e(t)}$ is computed based on the event-level reward scores. The second term in the gradient expression comes from the KL divergence, which quantifies the difference between the reference policy π_{ref} and the current policy π_θ. We simplify this term to simplify the analysis. For different search events, the gradient contribution is computed based on the event-specific advantage $\hat{A}_{i,e(t)}$.

From the event-level perspective, for the group of trajectory $\{\mathcal{S}_i, e_{\text{s}}^{(i)}\}_{i=1}^{G}$, the gradient coefficient for the search event is defined as:

$$GC_{search}(q, \mathcal{R}, e(t), \mathcal{S}_i, \pi_\theta) = \begin{cases}\hat{A}_{i,e(t)}, & e(t) \in \mathcal{E}_i^*, \\ 0, & \text{otherwise.}\end{cases} \tag{11}$$

The gradient optimization for different search events $e(t) \in \mathcal{S}$ is controlled through the gradient coefficient $GC_{search}(q, \mathcal{R}, e(t), \mathcal{S}_i, \pi_\theta)$.

Similarly, for selection event e_{s}, the gradient of $\mathcal{J}_{\text{select}}(\theta)$ is:

$$\begin{aligned}&\nabla_\theta \mathcal{J}_{select}(\theta) = \mathbb{E}[e_{\text{s}}^{(i)} \sim \pi_{\theta_{\text{old}}}(e_{\text{s}}^{(i)}|q, \mathcal{S}_i)] \\ &\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|e_{\text{s}}^{(i)}|}\sum_{t=1}^{|e_{\text{s}}^{(i)}|}\left[\hat{A}_{i,e_{\text{s}}^{(i)}} + \beta\left(\frac{\pi_{ref}(e_{\text{s},t}^{(i)} \mid q, \mathcal{S}_i, e_{\text{s},<t}^{(i)})}{\pi_\theta(e_{\text{s},t}^{(i)} \mid q, \mathcal{S}_i, e_{\text{s},<t}^{(i)})} - 1\right)\right]\nabla_\theta \log \pi_\theta(e_{\text{s},t}^{(i)} \mid q, \mathcal{S}_i, e_{\text{s},<t}^{(i)}).\end{aligned} \tag{12}$$

The gradient coefficient for the selection event $e_{\text{s}}^{(i)}$ is defined as:

$$GC_{select}(q, \mathcal{S}_i, e_{\text{s}}^{(i)}, \pi_\theta) = \begin{cases}\hat{A}_{i,e_{\text{s}}^{(i)}}, & \mathcal{M}(\mathcal{S}_i) \cap \mathcal{T}_s = \mathcal{T}_s, \\ 0, & \text{otherwise.}\end{cases} \tag{13}$$

By assigning event-specific gradient coefficients and applying targeted optimization signals, the model is encouraged to learn and improve the corresponding capabilities across different stages of the overall process.

In the following, we provide a brief analysis of the gradients for the remaining methods, based on the same assumptions and simplifications.

GRPO. The gradient coefficient for the trajectory $\{\mathcal{S}_i, e_{\text{s}}^{(i)}\}$ in GRPO is:

$$GC_{\text{GRPO}}(q, \mathcal{R}, \mathcal{S}_i, e_{\text{s}}^{(i)}, \pi_\theta) = \hat{A}_{select}^{i},, \tag{14}$$

where $\hat{A}_{select}$ is the normalized advantage of the final selection, computed by subtracting the mean and dividing by the standard deviation of the final selection rewards. The final selection reward is a binary function that equals 1 if the selection outcome matches the target tools $\mathcal{T}_s$, and 0 otherwise. The gradient coefficient GC_{GRPO} over the entire trajectory is governed by the group-wise relative advantage derived from the final selection outcome, meaning that all actions along the trajectory share the same optimization signal based on the selection result.

GDPO. The gradient coefficient for the trajectory $\{\mathcal{S}_i, e_{\text{s}}^{(i)}\}$ in GDPO is:

$$GC_{\text{GDPO}}(q, \mathcal{R}, \mathcal{S}_i, e_{\text{s}}^{(i)}, \pi_\theta) = \hat{A}^i_{SRecall}, + \hat{A}^i_{select}, \tag{15}$$

where $\hat{A}_{SRecall}$ is the normalized advantage computed from the search recall for target tools $\mathcal{T}_s$, reflecting the contribution of search stages to retrieving relevant tools, and $\hat{A}_{select}$ is the normalized advantage of the final selection as in GRPO. In GDPO, the trajectory gradient is influenced by both intermediate search performance and the final selection outcome. This means that each action along the trajectory receives a combined optimization signal. However, this also implies that gradients at different positions along the trajectory may not accurately reflect their true contribution. For example, two trajectories could have the same $\hat{A}_{SRecall}$ value but retrieve different target tools. Some search steps during the trajectory may be more critical than others, yet all steps are assigned the same reward signal, potentially leading to suboptimal credit assignment.

MARAG-R1. The gradient coefficient for the trajectory $\{\mathcal{S}_i, e_{\text{s}}^{(i)}\}$ in MARAG-R1 is:

$$GC_{\text{MARAG-R1}}(q, \mathcal{R}, \mathcal{S}_i, e_{\text{s}}^{(i)}, \pi_\theta) = \hat{A}^i_{R(\mathcal{T})}, , \tag{16}$$

where $R(\mathcal{T})$ is defined as:

$$R(\mathcal{T}) = R_A + R_E + R_T, \tag{17}$$

Answer Reward R_A evaluates the correctness of the final answer using a token-level F1 score, which provides partial credit for answers that are semantically close to the reference:

$$R_A = \text{F1}(A, A^*), \tag{18}$$

where A and A^* denote the predicted and ground-truth answers, respectively.

Document Coverage Reward R_E assess the completeness and precision of retrieved evidence, computes an F1 score over document identifiers. Let $\mathcal{D}_{\text{pred}}$ denote the set of retrieved document IDs and $\mathcal{D}^*$ the ground-truth supporting documents. The reward is defined as:

$$R_E = \text{F1}(\mathcal{D}_{\text{pred}}, \mathcal{D}^*) = \frac{2 \cdot P \cdot \text{Rec}}{P + \text{Rec}}, \tag{19}$$

where precision $P = \frac{|\mathcal{D}_{\text{pred}} \cap \mathcal{D}^*|}{|\mathcal{D}_{\text{pred}}|}$ and recall $\text{Rec} = \frac{|\mathcal{D}_{\text{pred}} \cap \mathcal{D}^*|}{|\mathcal{D}^*|}$. This reward encourages retrieving all necessary documents while minimizing irrelevant retrievals.

Tool Exploration Reward R_T promotes strategic exploration by rewarding sufficient but not excessive tool usage. Let N_{call} and N^*_{call} denote the number of tool calls in the predicted and expert trajectories (*i.e.*, the number of rounds for invoking the search engine, the numbers of target tools), respectively:

$$R_T = \begin{cases} 1, & \text{if } N_{\text{call}} \leq N^*_{\text{call}}, \\ \max\left(0,\, 1 - \frac{N_{\text{call}} - N^*_{\text{call}}}{N^*_{\text{call}}}\right), & \text{otherwise.} \end{cases} \tag{20}$$

These three reward components are directly summed to form the trajectory-level reward. However, simply adding them together does not highlight the model's current weaknesses in specific capabilities, especially after summation and normalization. Moreover, applying the same combined reward to all tokens along the trajectory introduces inaccurate and noisy learning signals, since different tokens and steps may contribute differently to the overall performance.

GSPO. In scenarios like multi-turn RL, GSPO use a finer-grained advantage adjustment than the sequence level to allow token-wise advantage customization:

$$\begin{aligned} &\mathcal{J}_{\text{GSPO-token}}(\theta) = \\ &\mathbb{E}_{q \sim \mathcal{D},\, \{y_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot|q, \mathcal{T}, \mathcal{R})} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min\left(s_{i,t}(\theta) \widehat{A}_{i,t},\, \text{clip}\left(s_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right) \widehat{A}_{i,t} \right) \right], \end{aligned} \tag{21}$$

where $y_i = \{\mathcal{S}_i, e_{\text{s}}^{(i)}\}$, $s_{i,t}(\theta)$ is defined as:

$$s_{i,t}(\theta) = \text{sg}\left[s_i(\theta)\right] \cdot \frac{\pi_\theta(y_{i,t}|q, \mathcal{T}, \mathcal{R}, y_{i,<t})}{\text{sg}\left[\pi_\theta(y_{i,t}|q, \mathcal{T}, \mathcal{R}, y_{i,<t})\right]}, \tag{22}$$

and $\mathrm{sg}[\cdot]$ denotes only taking the numerical value but stopping the gradient, corresponding to the `detach` operation in PyTorch. where the importance ratio $s_i(\theta)$ based on sequence likelihood:

$$s_i(\theta) = \left(\frac{\pi_\theta(y_i|q,\mathcal{T},\mathcal{R})}{\pi_{\theta_{\text{old}}}(y_i|q,\mathcal{T},\mathcal{R})} \right)^{\frac{1}{|y_i|}} = \exp\left(\frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \log \frac{\pi_\theta(y_{i,t}|q,\mathcal{T},\mathcal{R},y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|q,\mathcal{T},\mathcal{R},y_{i,<t})} \right). \tag{23}$$

The gradient of GSPO can be derived as:

$$\begin{aligned}
\nabla_\theta \mathcal{J}_{\text{GSPO-token}}(\theta) &= \nabla_\theta \mathbb{E}_{q\sim\mathcal{D},\,\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q,\mathcal{T},\mathcal{R})} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} s_{i,t}(\theta)\widehat{A}_{i,t} \right] \\
&= \mathbb{E}_{x\sim\mathcal{D},\,\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q,\mathcal{T},\mathcal{R})} \left[\frac{1}{G}\sum_{i=1}^{G} s_i(\theta) \cdot \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \widehat{A}_{i,t} \frac{\nabla_\theta \pi_\theta(y_{i,t}|q,\mathcal{T},\mathcal{R},y_{i,<t})}{\pi_\theta(y_{i,t}|q,\mathcal{T},\mathcal{R},y_{i,<t})} \right] \\
&= \mathbb{E}_{x\sim\mathcal{D},\,\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q,\mathcal{T},\mathcal{R})} \left[\frac{1}{G}\sum_{i=1}^{G} \left(\frac{\pi_\theta(y_i|q,\mathcal{T},\mathcal{R})}{\pi_{\theta_{\text{old}}}(y_i|q,\mathcal{T},\mathcal{R})} \right)^{\frac{1}{|y_i|}} \cdot \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \widehat{A}_{i,t} \nabla_\theta \log \pi_\theta(y_{i,t}|q,\mathcal{T},\mathcal{R},y_{i,<t}) \right].
\end{aligned} \tag{24}$$

Note that the term $\frac{\pi_\theta(y_{i,t}|q,\mathcal{T},\mathcal{R},y_{i,<t})}{\mathrm{sg}[\pi_\theta(y_{i,t}|q,\mathcal{T},\mathcal{R},y_{i,<t})]}$ has a numerical value of 1, so $s_{i,t}(\theta)$ is numerically equal to $s_i(\theta)$. GSPO set the advantages of all the tokens in the response y_i to the same value (i.e., $\widehat{A}_{i,t} = \widehat{A}^i_{select}$,).

The gradient coefficient for the trajectory $\{\mathcal{S}_i, e_{\text{s}}^{(i)}\}$ in GSPO is:

$$GC_{\text{GSPO}}(q,\mathcal{R},\mathcal{S}_i,e_{\text{s}}^{(i)},\pi_\theta) = \left(\frac{\pi_\theta(y_i|q,\mathcal{T},\mathcal{R})}{\pi_{\theta_{\text{old}}}(y_i|q,\mathcal{T},\mathcal{R})} \right)^{\frac{1}{|y_i|}} * \widehat{A}_{i,t}, \tag{25}$$

where GSPO adjusts the advantages for each token through the token-wise scaling factor $s_{i,t}(\theta)$. Although GSPO performs fine-grained optimization at the token level from the perspective of the entire trajectory, its optimization objective is still evaluated based on trajectory-level rewards. As a result, non-critical parts of the trajectory may still receive similar reward signals, even if they contribute little to the final outcome, which can introduce redundant or noisy optimization signals.

G Prompt

In this section, we detail the prompts used in our experiments.

Prompt for LLM

System Prompt
You are a super intelligent AI assistant that achieves my day-to-day tasks completely autonomously by interacting with apps/tools using their associated APIs on my behalf.

Tool Schema
{tool_schema}

User Prompt
Your job is to provide a list of tool API names that can be used to solve the given task, using a search engine to look up the necessary API documentation like an experienced programmer. You can call the search tool "api_doc_search_tool", which will return the top searched results from the tools' API documentation, enclosed between < tool_response> and </tool_response>. You can search as many times as you want, but include at most one search per turn. If you find no further external tool APIs needed, you can provide the final list of tool API names inside <tool_list> and </tool_list>. Format each tool API name as [category_name].[tool_name].[api_name], and separate multiple names with commas.
Example:
<tool_list>
Media.open_library.osearch_author, Media.movie_db.get_movie_details
</tool_list>
Please return API names based only on the search results, and do not fabricate any.

Description of Tool Categories:
Here are the tool categories available for search:
{tool_categories}

Task Question:
{question}

Search Engine's Schema (Global Search)

```
tool_schema:
  type: function
  function:
    name: api_doc_search_tool
    description: Searches relevant tools' API documents
      based on the given query.
    parameters:
      type: object
      properties:
        query:
          type: string
          description: A fully-formed semantic query.
            The tool will return the top searched
            results from the tools' API documentation
            for the query.
      required:
        - query
```

Search Engine's Schema (Category-based Search)

```
tool_schema:
  type: function
  function:
    name: api_doc_search_tool
    description: Searches relevant tools' API documents based on the
    given query and optional category filter.
    parameters:
      type: object
      properties:
        category:
          type: string
          description: Optional category to filter the tool's API
          documents. If set to "all", searches across all categories.
        query:
          type: string
          description: A fully-formed semantic query. The tool will
          return the top searched results from the tools' API
          documentation for the query.
      required:
        - query
```

H Broader Impact

This work has the potential to positively impact the development of reliable LLM-based agents by improving their ability to search for, distinguish, and select appropriate tools from large-scale tool repositories. Since real-world tool ecosystems contain a vast number of diverse and functionally similar tools, more effective tool selection can help agents better solve complex tasks, reduce failures caused by incompatible tool composition, and make external tool use more scalable and practical. Our proposed framework may therefore benefit task that require accurate tool orchestration. At the same time, improving the tool-selection capability of LLM agents may also introduce negative societal risks. More capable agents could be misused to automate harmful actions, invoke inappropriate or unsafe tools, or interact with external environments without sufficient human oversight. In addition, errors in tool selection may still lead to unreliable outcomes, privacy leakage, or unintended execution behaviors when deployed in real-world systems. Therefore, we emphasize that such methods should be used with proper safeguards, including tool-access control, sandboxed execution, security and privacy filtering, and human supervision in high-stakes scenarios. In conclusion, we posit that the benefits of our approach surpass its drawbacks concerning social impact.

I Further Experiment

I.1 Performance under Different Tool Library Sizes

To evaluate the effect of tool library size, we conduct additional experiments with 4k and 8k tools using Qwen3-4B-Instruct, reporting F1, Recall, Precision, and Match. The full 16k setting is reported in Table 1 and is not repeated here. We exclude the 1k setting because the benchmark contains 1,125 unique ground-truth tools, so a library of this size would necessarily omit some target tools.

As shown in Table 6 and Table 1, the overall F1 scores of all compared methods decrease as the tool library grows from 4k to 8k and then to 16k, indicating the increased difficulty of tool selection in larger repositories. Nevertheless, ToolSearcher achieves the highest F1 and Match scores at all three scales and consistently improves over the Multi-turns baseline, demonstrating its effectiveness across different tool library sizes.

Table 6: Performance under different tool library sizes on StableToolBench using Qwen3-4B-Instruct.

Method	4k				8k			
	F1	Recall	Precision	Match	F1	Recall	Precision	Match
RAG	0.496	0.491	0.517	0.331	0.415	0.454	0.413	0.248
RAG_{SFT}	0.553	0.565	0.561	0.340	0.440	0.493	0.432	0.275
Multi-turns	0.573	0.601	0.579	0.299	0.520	0.542	0.528	0.256
Search-R1	0.696	**0.695**	0.711	0.472	0.639	0.638	0.653	0.420
GSPO	0.690	0.692	0.703	0.462	0.643	0.644	0.655	0.417
GDPO	0.692	0.683	0.717	0.458	0.644	0.637	0.664	0.412
MARAG-R1	0.632	0.599	0.705	0.349	0.577	0.543	0.649	0.301
ToolSearcher	**0.700**	0.693	**0.721**	**0.478**	**0.651**	**0.646**	**0.672**	**0.429**

I.2 Effect of Per-Round Retrieval Size

We evaluate the effect of the number of tool documents retrieved per search round using Qwen3-4B-Instruct, reporting F1 and Match for $k \in \{5, 10, 20, 40\}$. Our model is trained with $k = 5$, while the per-round retrieval size is varied during evaluation.

Table 7: Effect of per-round retrieval size on StableToolBench using Qwen3-4B-Instruct.

Method	Top-5		Top-10		Top-20		Top-40	
	F1	Match	F1	Match	F1	Match	F1	Match
Qwen3-4B-Instruct	0.387	0.220	0.393	0.167	0.382	0.171	0.189	0.065
Search-R1	0.505	0.307	0.519	0.327	0.526	0.348	0.391	0.187
GSPO	0.473	0.258	0.493	0.325	0.524	0.328	0.393	0.208
GDPO	0.518	0.305	0.525	0.323	0.529	0.331	0.382	0.199
MARAG-R1	0.465	0.225	0.475	0.244	0.482	0.242	0.352	0.195
ToolSearcher	**0.531**	**0.316**	**0.535**	**0.336**	**0.540**	**0.353**	**0.412**	**0.231**

As shown in Table 7, ToolSearcher achieves the highest F1 and Match scores across all tested retrieval sizes. Increasing k from 5 to 20 improves its F1 from 0.531 to 0.540 and Match from 0.316 to 0.353, suggesting that moderately expanding the candidate set can benefit tool selection. However, increasing k to 40 substantially reduces performance across all methods. A larger candidate set may introduce more irrelevant or distracting tools, increasing the difficulty of distinguishing and retaining relevant tool information for final selection. Despite this decline, ToolSearcher maintains its advantage over the compared baselines.

I.3 Effect of Explicit Planning and Thinking

We investigate whether explicit planning or thinking improves tool selection through two complementary comparisons: adding a reasoning step to the retrieval policy and enabling the native thinking mode of general-purpose models.

Explicit planning before retrieval. We compare ToolSearcher with ToolQP [11], which decomposes requests into sub-tasks to guide multi-round retrieval using teacher-annotated plans and plan-level similarity feedback during training. We also evaluate a per-turn reasoning variant of ToolSearcher, which generates a reasoning/planning step before each retrieval turn during both training and inference.

As shown in Table 8, ToolSearcher achieves higher overall F1 than both ToolQP and the per-turn reasoning variant on both backbones. However, explicit reasoning benefits some metrics: on Qwen3-4B-Instruct, the per-turn reasoning variant improves Match from 0.316 to 0.331, despite reducing F1 from 0.531 to 0.496. We observe that without dedicated supervision or rewards for intermediate reasoning quality, generated thoughts often paraphrase the request rather than provide useful planning. The effectiveness of teacher-guided decomposition can also depend on the completeness and granularity of the initial plan. These results suggest that adding explicit planning does not consistently

Table 8: Comparison of explicit planning and per-turn reasoning on StableToolBench. Best and second-best values are marked within each backbone.

Method	Overall				I1			I2		I3
	F1	Recall	Precision	Match	Inst.	Tool	Cate.	Inst.	Cate.	Inst.
Qwen2.5-7B-Instruct										
ToolQP	0.440	0.493	0.432	0.222	0.503	0.450	0.504	0.433	0.362	0.260
ToolSearcher (per-turn reasoning)	0.462	**0.526**	0.446	0.226	0.525	0.493	0.494	0.457	0.384	**0.302**
ToolSearcher	**0.513**	0.511	**0.534**	**0.278**	**0.595**	**0.561**	**0.577**	**0.485**	**0.405**	0.294
Qwen3-4B-Instruct										
ToolQP	0.474	0.487	0.498	0.216	0.599	0.538	0.580	0.344	0.287	0.314
ToolSearcher (per-turn reasoning)	0.496	0.491	0.517	**0.331**	**0.615**	0.434	0.538	**0.498**	0.451	**0.327**
ToolSearcher	**0.531**	**0.527**	**0.546**	0.316	0.599	**0.617**	**0.586**	0.422	**0.491**	0.284

improve tool selection, while event-level feedback can guide useful retrieval actions without requiring teacher-annotated plans.

Native thinking mode. We further compare three general-purpose models with their native thinking mode enabled and disabled. Table 9 shows that enabling thinking reduces overall F1 for all three models: from 0.443 to 0.392 for Qwen3-14B, from 0.461 to 0.268 for Qwen3.6-27B, and from 0.423 to 0.388 for Qwen3-32B. Although some scenario-specific scores improve, native thinking does not yield better overall tool-selection performance in these experiments. Without task-specific training or feedback, additional reasoning may introduce unnecessary semantic expansion or divert search away from the required tool functionalities. Together, these comparisons suggest that effective reasoning should be aligned with the tool-selection objective; simply adding explicit reasoning steps or enabling native thinking does not guarantee better retrieval and selection.

Table 9: Effect of native thinking mode on StableToolBench. Best and second-best values are marked within each model.

Model	Thinking	Overall				I1			I2		I3
		F1	Recall	Precision	Match	Inst.	Tool	Cate.	Inst.	Cate.	Inst.
Qwen3-14B	Off	**0.443**	**0.492**	**0.430**	**0.182**	**0.537**	**0.446**	**0.485**	**0.407**	**0.388**	**0.256**
Qwen3-14B	On	0.392	0.400	0.408	0.161	0.473	0.414	0.435	0.364	0.335	0.174
Qwen3.6-27B	Off	**0.461**	**0.490**	**0.466**	**0.247**	**0.537**	**0.514**	**0.529**	**0.418**	**0.374**	**0.198**
Qwen3.6-27B	On	0.268	0.275	0.280	0.126	0.265	0.307	0.237	0.324	0.282	0.129
Qwen3-32B	Off	**0.423**	**0.475**	0.410	**0.178**	**0.496**	**0.459**	**0.476**	0.374	**0.360**	0.213
Qwen3-32B	On	0.388	0.400	**0.411**	0.152	0.471	0.362	0.422	**0.388**	0.349	**0.231**

I.4 Task Diversity and Multi-Aspect Tool Requirements

Large-scale tool selection must accommodate requests from diverse domains rather than a small, fixed set of applications. Moreover, an individual request can require multiple tools serving different aspects of the user's goal. Table 10 illustrates these properties with examples from the StableToolBench test set. The examples span travel, weather, food, and entertainment, illustrating the breadth of functionality a general-purpose tool repository must cover. Within a single request, complementary requirements can also map to different tool families: finding diving sites and accommodation, for example, requires both activity-related and lodging-related tools. A single query emphasizing one aspect may fail to retrieve tools needed for another. These properties motivate both a broad tool repository and multi-turn search that can adjust its direction to cover the different requirements.

I.5 Case Study of Multi-Turn Search

We examine the travel request in Table 10: "Planning a Thailand vacation with family, interested in scuba diving. Recommend dive sites in Thailand, and find top-tier stays in that area for accommodation." The two target APIs serve distinct purposes: `world_scuba_diving_sites_api.query_divesites_by_a_country_or_a_region` finds diving sites by country or region, while `airbnb_v2.get_top_tier_stays_filter` provides filters for top-tier accommodation. Table 11 presents the trajectory produced by ToolSearcher, comprising three search rounds followed by final tool selection. The first query focuses on diving sites and retrieves the corresponding target API, but does not retrieve the lodging target. The second query shifts toward accommodation and discovers

Table 10: Examples of diverse, multi-aspect requests from StableToolBench (abbreviated). Tool names identify the API families required for each request.

Domain	User request	Required tool families	Distinct aspects
Travel	Planning a Thailand vacation with family, interested in scuba diving. Recommend dive sites in Thailand, and find top-tier stays in that area for accommodation.	`world_scuba_diving_sites` (dive sites) `airbnb_v2` (stays)	activity venue lodging
Weather	Planning a hiking trip to the Swiss Alps. Fetch the climate data for the location by lat/lon, and the moonrise/moonset times during our trip.	`climate_data` (2 APIs) `stormglass_complete` (astronomy)	climate astronomy
Food	Planning a surprise dinner. Recommend low-carb chicken recipes, and give the nutritional info of milk — calories, protein, fat per 100g.	`recipe_search_and_diet` (recipes) `food_nutrional_data` (nutrition)	recipe nutrition
Movies / Entertainment	Find a new TV show to watch with family. Suggest a high-IMDb family-friendly series, check its streaming availability on Netflix/Disney+/Apple TV, and provide cast & crew info.	`ott_details` (show details) `streaming_availability` (streaming) `watchmode` (cast & crew)	details streaming cast

Table 11: Multi-turn search with direction adjustment for the travel request. Each search returns five API candidates; the final row shows the selected APIs.

Step	Search query and observation	Retrieved / selected APIs
1	**Aspect A: dive sites.** API to find scuba diving sites in Thailand *Observation:* The dive-site target API is retrieved; the lodging target API is still missing.	1. `world_scuba_diving_sites_api.query_divesites_by_a_country_or_a_region` 2. `world_scuba_diving_sites_api.query_divesites_by_gps_boundaries_for_use_with_maps` 3. `provinces_of_thailand.subdistrict` 4. `oil_thai_price.get_image_result_today_price` 5. `oil_thai_price.get_latest_price`
2	**Switch to Aspect B: lodging.** API to find top-tier accommodation options in Thailand for scuba diving sites *Observation:* The query shifts toward lodging and retrieves its target API. Both target APIs have now been found.	1. `world_scuba_diving_sites_api.query_divesites_by_a_country_or_a_region` 2. `world_scuba_diving_sites_api.query_divesites_by_gps_boundaries_for_use_with_maps` 3. `airbnb_v2.get_top_tier_stays_filter` 4. `airbnb_search.properties_search` 5. `airbnb_search.get_top_tier_stays_filter`
3	**Confirm lodging-side API family.** how to combine scuba diving site search with accommodation API for Thailand *Observation:* The query explores a combined requirement but retrieves no additional target API.	1. `world_scuba_diving_sites_api.query_divesites_by_a_country_or_a_region` 2. `world_scuba_diving_sites_api.query_divesites_by_gps_boundaries_for_use_with_maps` 3. `airbnb_v2.get_accessibility_filters` 4. `airbnb_v2.test_api` 5. `airbnb_v2.get_amenities_filters`
4	**Final selection.** The final selection combines the two target APIs found in earlier rounds.	1. `world_scuba_diving_sites_api.query_divesites_by_a_country_or_a_region` 2. `airbnb_v2.get_top_tier_stays_filter`

the remaining target API, covering both aspects of the request. The third search does not discover a new target API; the final selection retains the two relevant APIs from the accumulated search results. This case illustrates how changing search direction across turns can recover a complementary tool that was absent from the initial results.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contribution, which can be found in Section 1.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We provide it in Appendix A.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: We provide the full set of assumptions in Section 3 and gradient analysis in Appendix F.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We elaborate it in Section 4 and Appendix B and D.

   Guidelines:

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

5. **Open access to data and code**

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

Answer: [Yes]

Justification: The datasets we use are publicly available, and the link to the released code is presented in the abstract.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We elaborate it in Section 4 and Appendix B and D.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: Error bars are not reported because it would be too computationally expensive. We fix the seed for all our experiments, which ensures the reliability of our experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provide our training GPU device and training setup in Appendix B and D.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have made sure to preserve the paper's anonymity and its conformation with NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss that in Appendix H.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The RL training on tool selection does not impose such risk.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We use only publicly available datasets and models, and we tag and cite all corresponding sources in the paper.

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

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

    Guidelines:

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

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

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

    Answer: [N/A]

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

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: LLMs are an important component of our core method. We use an LLM as the policy model for multi-turn tool search and tool selection, and optimize it with reinforcement learning. The paper describes how the LLM is used in the proposed framework, as well as the corresponding training and inference settings.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.