Navigating Large Hierarchical Tool Spaces via Causally-Decoupled Advantage Policy Optimization

Anonymous ACL submission

Abstract

Efficient and accurate tool invocation from large, hierarchical libraries remains a fundamental challenge for large language models (LLMs) due to the subtle semantic distinctions among tools and implicit conventions or hierarchical constraints that are rarely specified in formal documentation. In this paper, we propose Causally-decoupled Advantage Policy Optimization (CAPO), a new reinforcement learning framework designed for reliable tool invocation in large hierarchical tool libraries. CAPO introduces *hierarchy-aligned advantage estimation* that causally decouples intermediate decision credit from downstream stochastic outcomes, enabling fine-grained learning even from partially successful or failed rollouts. To stabilize training, CAPO further incorporates a hybrid reward that combines dense semantic signals with topological hierarchy-consistency priors, providing continuous, structured feedback along the invocation trajectory. Additionally, we curate FHTC-26K, a large-scale financial benchmark with 26,314 hierarchically organized tools and expert-annotated invocation traces. Extensive experiments on both FHTC-26K and a standard benchmark, ToolBench, show that CAPO improves tool-invocation accuracy by 4.18%–7.94% over state-of-the-art baselines while exhibiting substantially more stable optimization. Our code and data are published anonymously at `https://anonymous.4open.science/r/CAPO-EF27`.

1 Introduction

While large language models (LLMs) have substantially advanced natural language processing (Comanici et al., 2025), many industrial applications increasingly require LLMs to act by invoking external tools rather than merely generating free-form text. This need has motivated a growing body of work on *tool-augmented LLMs* that learn to select and invoke tools as part of the reasoning process (Qin et al., 2024; Zeng et al., 2025a,b; Zhang et al., 2025b). However, most existing tool-invocation methods are trained and evaluated under the *fixed candidate set* assumption, where tool selection is performed over a small, pre-filtered shortlist. In practice, industrial deployments require *open-ended full-library routing* across large, deeply hierarchical tool repositories, where tools often substantially overlap semantically but differ in fine-grained scope, argument conventions, and implicit domain constraints. Consequently, methods optimized for small, pre-defined candidate sets often transfer poorly to industrial-scale tool ecosystems, revealing a key mismatch between current paradigms and real-world tool libraries.

Generally, prior work on tool invocation follows two broad paradigms: (i) *retriever-aware supervised fine-tuning (SFT)* methods that learn tool-use behaviors from demonstrations while using a retriever to narrow the candidate set during tool selection (Qin et al., 2024; Patil et al., 2024) and (ii) *reinforcement learning (RL)* methods that optimize tool-use policies through interaction (Zeng et al., 2025a,b). SFT-based methods train models to imitate demonstrated invocation trajectories. However, demonstrations are inherently limited and cannot capture the combinatorial diversity of tool usage across large libraries, leading to weak generalization and error amplification in long-horizon tool chains. To scale to large tool libraries, these methods rely on retrieval to rank and shortlist tools for a given query. However, conventional retrievers often fail when tools are *textually similar yet functionally distinct*, a pattern that is particularly prevalent in the financial domain. For example, several valuation tools may share near-identical surface descriptions but differ in their definitions (e.g., P/E vs. P/B), yielding ambiguous shortlists and brittle downstream decisions (Chen et al., 2024). Meanwhile, RL-based approaches optimize tool-use policies from interaction feedback (Zeng et al., 2025a,b). Most existing RL formulations are devel-

oped and validated primarily for pre-defined candidate sets, and their limitations become pronounced under full-library routing in hierarchical tool libraries. In this setting, the action space expands dramatically, and high-reward trajectories that are both semantically correct and schema-valid are exceedingly sparse, rendering exploration highly sample-inefficient (Zeng et al., 2025a,b). Moreover, existing group-wise policy optimization methods rely on trajectory-level terminal feedback and broadcast the same reward across steps, obscuring credit assignment for early, high-impact hierarchy-routing decisions and destabilizing long-horizon tool invocation (Yu et al., 2025; Shao et al., 2024).

From an evaluation perspective, a clear mismatch persists between existing benchmarks and industrial-scale tool libraries. Most benchmarks focus on small tool sets with surface-level separability (Patil et al., 2024; Qin et al., 2024; Li et al., 2023), while domain-specific financial search benchmarks emphasize expert-level reasoning rather than full-library hierarchical tool routing (Hu et al., 2025). In such settings, simple semantic matching often suffices to identify the right tool. Other benchmarks adopt an *oracle candidate* protocol that supplies a pre-filtered shortlist for each query (Chen et al., 2025; Patil et al., 2025), removing the need for open-ended selection over the library. Both regimes differ from professional settings, where LLMs must autonomously choose from tens of thousands of hierarchically organized tools under subtle conventions and constraints. As a result, existing benchmarks fail to accurately assess the reliability of long-horizon hierarchical tool routing at an industrial scale.

Collectively, existing tool invocation approaches cannot reflect how professional tool libraries are actually navigated: not by a single best-match decision, but by a sequence of hierarchy-conditioned choices. In deep hierarchies, a substantial fraction of errors originates early when the model routes the query to an inappropriate high-level hierarchy; once the trajectory enters the wrong branch, subsequent steps are forced to operate within an already mis-specified tool subspace, and later "local" corrections are often insufficient to recover. This motivates a hierarchy-centric formulation that treats the hierarchy as a first-class constraint during learning. To support reliable open-ended, full-library tool invocation at scale, we construct FHTC-26K, a large-scale dataset of tool invocation queries paired with explicit level-by-level hierarchy routing traces that reflect real user query distributions. Building upon the dataset, we propose Causally-decoupled Advantage Policy Optimization (CAPO), a hierarchy-constrained RL approach that integrates a hybrid reward combining dense lexical overlap with prefix-consistent hierarchy supervision, and performs hierarchy-aligned credit assignment by normalizing advantages within hierarchy routing segments rather than uniformly over the full trajectory, thereby directly locating early hierarchy routing errors that dominate failures in tool hierarchies. In summary, our main contributions are threefold:

- We introduce FHTC-26K, a large-scale financial-domain benchmark of 26,314 hierarchically organized tools with expert-annotated invocation traces for open-ended full-library routing, supporting open-set selection and schema-valid invocation without oracle candidate shortlists.
- We propose CAPO, which improves token-level credit assignment for intermediate routing decisions by leveraging hierarchy-aligned advantages and a hybrid reward that combines lexical overlap with explicit hierarchy-consistency feedback.
- We conduct extensive experiments on two open-ended tool-invocation benchmarks. The results demonstrate consistent and substantial improvements of CAPO over state-of-the-art baselines.

2 Related Work

SFT Paradigms for Tool Invocation: Retriever-aware supervised fine-tuning (SFT) methods learn tool invocation by imitating curated execution trajectories and, when applicable, reducing the candidate space via embedding-based retrieval (Qin et al., 2024; Patil et al., 2024). However, its effectiveness degrades as repositories scale and tool semantics become finer-grained (Schick et al., 2023); general-purpose embedding models (Chen et al., 2024) often miss subtle functional distinctions, leading to tool and argument mis-specification.

RL Paradigms for Tool Invocation: RL-based methods can be categorized into *data-iterative* and *exploration-driven* paradigms. Data-iterative approaches iteratively refine supervision via error-driven data synthesis or fragment-level correction, coupled with preference optimization (Rafailov et al., 2023; Zeng et al., 2025b; Zhang et al., 2025b). Exploration-driven approaches optimize tool-use policies via RL without SFT initialization, typically using reward shaping to enforce format valid-

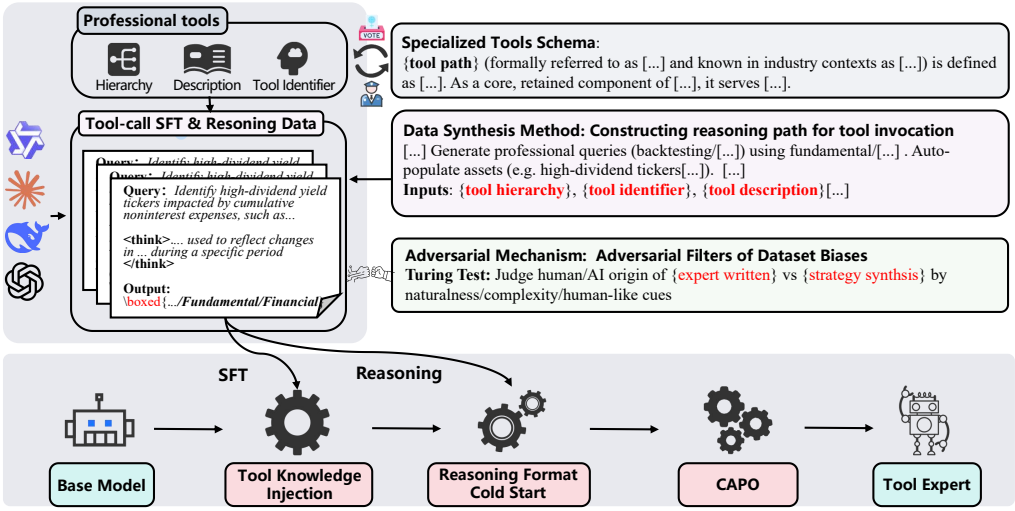

Figure 1: Pipeline for open-ended, hierarchical tool invocation dataset construction and training.

ity and semantic correctness (Zeng et al., 2025a). Recent reasoning and agentic RL studies further explore self-imitation, process-oriented feedback, and large-scale outcome-based optimization (Guo et al., 2025; Qin et al., 2026; Li et al., 2025). Despite these strategies, RL remains challenging in industrial-scale repositories, where the action space is extremely large and errors can accumulate over long-horizon multi-tool compositions.

General-purpose Web and Research Agents: Recent web and deep-research agents improve open-ended information seeking through self-evolving search, world modeling, or RL in real-world environments (Fang et al., 2025; Zhang et al., 2025a; Zhai et al., 2025; Zheng et al., 2025b). They typically compose broad tools, such as search, browsing, and code execution, whereas our setting requires reliably invoking a fixed, professional library with a lot of hierarchical, well-defined tools.

Benchmarks for Tool Invocation: Most tool-invocation benchmarks use fixed candidate sets, evaluating tool selection and argument filling within a restricted shortlist rather than open-ended tool discovery (Tang et al., 2023; Srinivasan et al., 2023; Patil et al., 2025; Chen et al., 2025; Wu et al., 2024). Even benchmarks claiming to evaluate a "full" toolset typically rely on relatively small repositories or simplified tool abstractions, limiting their ability to assess open-ended tool discovery under realistic hierarchies and fine-grained semantics (Li et al., 2023; Qin et al., 2024).

3 Method

We present an end-to-end framework for robust tool invocation in industrial-scale, deeply hierarchical repositories, as shown in Figure 1. The framework comprises two primary stages: (i) data synthesis and refinement, which builds a semantic foundation and generates tool-aligned queries and invocation traces, with adversarial filtering to align the synthetic data with real user query patterns; and (ii) a two-stage hierarchy-aware training process, which performs SFT[1] followed by RL optimized via CAPO under a hybrid reward that combines lexical overlap and hierarchy consistency.

3.1 Tool-Invocation Data Synthesis

High-quality training data for professional tool invocation is scarce, and naive synthesis from tool documentation is brittle. In industrial repositories, documentation is often metadata-heavy yet semantically under-specified, lacking fine-grained functional scope, domain conventions, and hierarchy-imposed constraints. As a result, synthetic queries collapse to superficial name-based descriptions and deviate markedly from authentic user requests. We therefore develop a synthesis-and-curation pipeline and construct FHTC-26K for tool invocation in large hierarchical tool libraries (see Appendix A for details). The statistics of FHTC-26K are summarized in Appendix B.

Semantic Foundation Construction: We build a semantic foundation $\mathcal{K}_T$ for each tool T. Specifically, for a tool with a hierarchical path $(H_1, \ldots, H_d)$, $\mathcal{K}_T$ aggregates three types of semantic metadata. (1) The *tool description* Desc_T provides LLM-distilled functional descriptions that go beyond tool names and signatures, capturing fine-

[1]We adopt standard SFT cold-start practices and omit implementation details for brevity.

grained scope and intended usage. (2) The *expertise identifier* Identifier_T collects domain-specific terminological aliases to broaden lexical coverage for the underlying tool concept and better match real user phrasing. (3) The *hierarchy description* Desc_H provides semantic descriptions that explicitly encode hierarchy-imposed constraints.

To improve the reliability of the metadata, we apply an iterative refinement protocol. Different LLMs independently generate candidate entries, which are then consolidated through multi-round consensus voting, and we keep only high-agreement items. Domain experts subsequently audit a sampled subset to uncover systematic failure modes and update the generation prompts accordingly. We repeat this generate-vote-audit loop until the retained metadata no longer exhibits salient errors under expert review.

Context-Aware Query Generation: Given the constructed semantic foundation $\mathcal{K}_T$, we generate tool-aligned queries via in-context learning (ICL) using a structured prompt template π. For a target tool set T_i, we condition generation on the tools' functional description Desc_{T_i}, expertise identifier $\text{EIDs} \in \text{Identifier}_{T_i}$, and a curated seed set of real user queries $\mathcal{S}_{\text{seed}}$, i.e.,

$$q_i \sim \text{LLM}(\pi(T_i, \text{EIDs}, \text{Desc}_{T_i}, \mathcal{S}_{\text{seed}})).$$

Here, Desc_{T_i} enforces functional alignment, EIDs increases lexical diversity, and $\mathcal{S}_{\text{seed}}$ anchors the query style to real-user patterns.

For each generated query q_i, we additionally synthesize a hierarchy-structured reasoning trace ζ_i using a trace prompt π_{trace}, i.e.,

$$\zeta_i \sim \text{LLM}\big(\pi_{\text{trace}}(q_i, \text{Desc}_{T_i}, \{\text{Desc}_{H_k}\}_{k=1}^{d_i})\big).$$

We condition trace generation on the hierarchy descriptions $\{\text{Desc}_{H_k}\}_{k=1}^{d_i}$ along the ground-truth path of T_i, which encourages hierarchy-consistent reasoning rather than name-based matching. Crucially, π_{trace} enforces a hierarchy-by-hierarchy `<think>` trace with explicit hierarchy-aware markers, and we require one marked segment for each hierarchy level. We use this marker schema as a post-hoc validator and drop traces that do not conform to it. With expert-in-the-loop iterative refinement of both π and π_{trace}, we obtain the initial synthetic dataset $\mathcal{S}_{\text{syn}} = \{(q_i, \zeta_i, T_i)\}_{i=1}^{N}$.

Adversarial Filtering and Distributional Alignment: To mitigate distributional drift and reduce synthetic artifacts in $\mathcal{S}_{\text{syn}}$, we adopt a Turing-style adversarial filtering scheme with an LLM-as-a-Judge (Le Bras et al., 2020). For each synthetic query $q_i \in \mathcal{S}_{\text{syn}}$, we estimate an adversarial predictability score $P_{\text{adv}}(q_i)$ by pairing q_i with M real queries $\{q_{\text{real}}^{(j)}\}_{j=1}^{M}$ and measuring how often a discriminator identifies q_i as the synthetic one, i.e.,

$$P_{\text{adv}}(q_i) = \frac{1}{M} \sum_{j=1}^{M} \mathbb{I}\left(\mathcal{D}(q_i, q_{\text{real}}^{(j)}) = q_i\right).$$

Here, $\mathcal{D}$ is an LLM-based discriminator that selects the synthetic query from each pair. We construct FHTC-26K by retaining samples with $P_{\text{adv}}(q_i) \leq \tau$. The filtering process is performed offline.

3.2 Causally-Decoupled Advantage Policy Optimization (CAPO)

To enable effective hierarchical tool invocation, we first perform a lightweight SFT-based supervised warm-up on FHTC-26K, training the model on tool metadata and hierarchy-aware reasoning traces to provide a stable initialization for the subsequent RL phase (Qin et al., 2024; Patil et al., 2024).

Figure 2 illustrates CAPO, an RL framework that couples hybrid reward computation with hierarchy-aligned optimization for hierarchical tool invocation. Specifically, rewards are obtained by combining a lexical overlap signal with a hierarchical prefix reward, and group-relative normalization is performed within hierarchy-aligned trace segments to produce segment-aligned advantages. This design avoids the uniform credit assignment in GRPO-style optimization, where a rollout-level reward leads to an advantage that is broadcast across all tokens, and it enables finer-grained credit assignment under the causal dependencies of hierarchical decisions.

3.2.1 Hybrid Reward Design for CAPO

Following FHTC-26K, we turn to reward design for the second-stage RL training. Existing rewards typically rely on either terminal verification or learned preference/reward models (Shao et al., 2024; Yu et al., 2025; Whitehouse et al., 2025; Cheng et al., 2024; Liu et al., 2025b), which can be sparse or require extra model-based judgments that are unreliable for fine-grained tool invocation. We therefore propose a hybrid reward: a lexical-overlap term that provides dense feedback toward the reasoning tool trace and a hierarchical term that enforces hierarchy-consistent decisions.

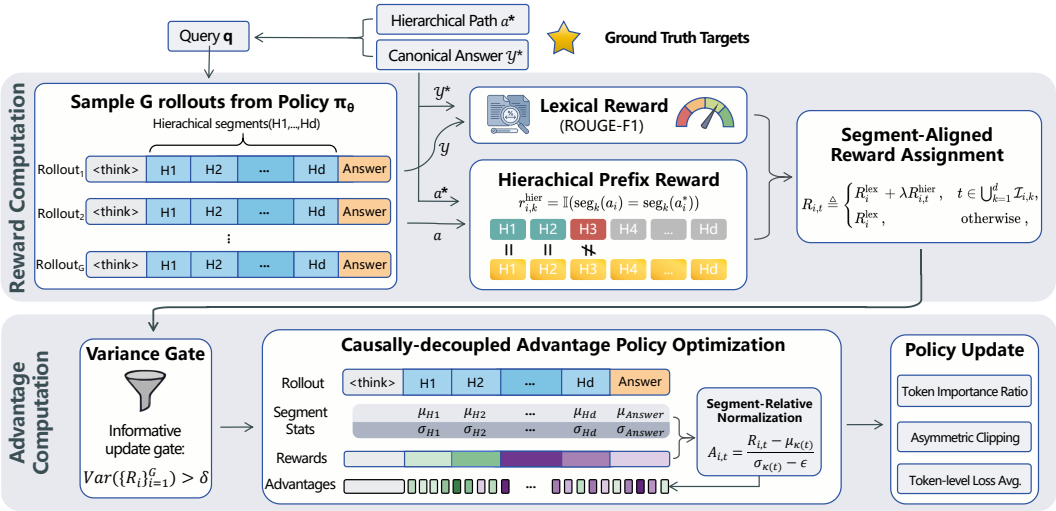

Figure 2: Overview of CAPO for hierarchical tool invocation.

Rewards Based on Lexical Overlap: In hierarchical tool invocation, outputs are largely canonicalized: tool choices are constrained by the hierarchy, and call schemas follow fixed conventions, leaving limited semantically valid surface variation. Hence, lexical overlap with the reference trace is expected to track ground-truth invocation correctness more closely than in open-ended text generation, while still providing graded feedback before an entire invocation becomes exactly correct. Although ROUGE (Lin, 2004) is often criticized for open-ended generation with high paraphrastic diversity (Grusky, 2023), it is well-suited here as an efficient reward for structured tool-invocation traces. Crucially, ROUGE provides a *dense* and *continuous* signal by assigning partial credit according to lexical overlap with the reference trace, instead of reducing all deviations to a binary correctness label. For instance, when a prediction matches the correct hierarchy path but differs in the final tool or call fields, ROUGE still captures aligned components and provides informative feedback, especially early in training when fully correct traces are rare.

Since tool invocations may be interdependent, we match invocations by position. Let $\hat{\mathcal{Y}}_i = \mathrm{Ans}(o_i) = (\hat{y}_{i,1}, \ldots, \hat{y}_{i,m})$ be the ordered list of tool invocations in the <answer> span of rollout o_i, and let $\mathcal{Y}^* = (y_1^*, \ldots, y_n^*)$ be the reference list. We define the lexical reward $\rho^{\mathrm{lex}}(\cdot,\cdot)$ as ROUGE-F1. The rollout lexical reward is

$$R^{\mathrm{lex}}\left(\hat{\mathcal{Y}}_i, \mathcal{Y}^*\right) = \frac{1}{L}\sum_{j=1}^{L} \rho^{\mathrm{lex}}\left(\hat{y}_{i,j}, y_j^*\right), \quad (1)$$

where $L = \max(m, n)$. If an invocation is missing or extra at position j on either side, its corresponding term is set to 0.

Hierarchical Reward: While lexical-overlap rewards provide dense feedback, they do not verify hierarchy-correct tool selection. A rollout can overlap with the reference on surface tokens (e.g., call fields or partial paths) yet *diverge* at an upper-level hierarchy node, rendering the resulting tool path invalid under hierarchical constraints. Moreover, tool invocations in a rollout may be interdependent and are naturally ordered. We therefore define the hierarchical reward by matching predicted and ground-truth invocations *position-wise* in this order. The same reward is applied to both single-tool and multi-tool rollouts, and we present the formulation using a single-tool example for clarity.

As discussed above, hierarchy decisions are prefix-constrained, and an incorrect choice at hierarchy k invalidates all lower-hierarchy selections. We define hierarchical rewards using prefix consistency. Level-k credit is assigned only when the predicted path matches the ground-truth path up to level k. Under the trace format enforced in SFT, we segment each rollout o_i into d hierarchy-level <think> segments $\{\mathcal{S}_{i,k}\}_{k=1}^{d}$ and one <answer> segment ($K = d + 1$), and extract one predicted hierarchy node per level to obtain a hierarchy-path prediction a_i. Formally, let $a_i = (H_{i,1}, \ldots, H_{i,d})$ be the hierarchy path from rollout o_i, and let $a_i^* = (H_{i,1}^*, \ldots, H_{i,d}^*)$ be the ground-truth path. Hierarchical segment reward is defined as

$$r_{i,k}^{\mathrm{hier}} = \mathbb{I}(\mathrm{seg}_k(a_i) = \mathrm{seg}_k(a_i^*)), \quad (2)$$

where $\mathrm{seg}_k(\cdot)$ extracts the prefix of length k. This design provides explicit supervision for early, high-

impact decisions near the root and ensures that downstream segments receive no reward once an upstream decision becomes invalid.

Let $\mathcal{I}_{i,k}$ denote the token index set of segment $\mathcal{S}_{i,k}$. We assign each token position $t \in \mathcal{I}_{i,k}$ the cumulative hierarchical reward from level k onward, i.e., $R^{\text{hier}}_{i,t} = \sum_{k'=k}^{d} r^{\text{hier}}_{i,k'}$, $t \in \mathcal{I}_{i,k}$. The cumulative form gives earlier segments larger potential credit, reflecting the stronger causal leverage of high-hierarchy choices.

Reward Composition: Let $R_{i,t}$ denote the reward assigned to the token at position t in rollout o_i. For brevity, define $R^{\text{lex}}_i = R^{\text{lex}}\left(\hat{\mathcal{Y}}_i, \mathcal{Y}^*\right)$. We add hierarchical rewards only to tokens in the hierarchy-reasoning segments of the <think> trace, i.e., tokens indexed by $\bigcup_{k=1}^{d} \mathcal{I}_{i,k}$. Formally,

$$R_{i,t} \triangleq \begin{cases} R^{\text{lex}}_i + \lambda\, R^{\text{hier}}_{i,t}, & t \in \bigcup_{k=1}^{d} \mathcal{I}_{i,k}, \\ R^{\text{lex}}_i, & \text{otherwise}, \end{cases} \tag{3}$$

where λ is the weight of the hierarchical reward.

3.2.2 Hierarchy-Aligned Group-Relative Advantage and Optimization

We optimize the policy using Causally-decoupled Advantage Policy Optimization (CAPO), tailored to hierarchical tool invocation with our hybrid reward. The core idea is to decouple advantage normalization along the hierarchy by using segment-conditioned group statistics instead of a single rollout-level normalization broadcast to all tokens. Concretely, CAPO computes a baseline for each hierarchy-aligned trace segment and normalizes token rewards with the statistics of their corresponding segment, yielding segment-aligned advantages for finer-grained credit assignment.

For each rollout o_i, the tokens are assigned to hierarchy-aligned segments using $\kappa_i(t)$, where $\kappa_i(t) = k$ iff $t \in \mathcal{I}_{i,k}$. We use a segment-shared reward $\widetilde{R}_{j,k}$ and assign $R_{j,t} = \widetilde{R}_{j,k}$ for all $t \in \mathcal{I}_{j,k}$. Based on $\widetilde{R}_{j,k}$, we compute the segment-wise mean and standard deviation over the G rollouts as

$$\mu_k = \operatorname*{mean}_{j\in[G]}\left(\widetilde{R}_{j,k}\right), \quad \sigma_k = \operatorname*{std}_{j\in[G]}\left(\widetilde{R}_{j,k}\right). \tag{4}$$

Given these segment-wise statistics, CAPO computes the segment-aligned token advantage by normalizing $R_{i,t}$ with the mean and standard deviation of the segment indexed by $\kappa_i(t)$, i.e.,

$$\hat{A}_{i,t} = \frac{R_{i,t} - \mu_{\kappa_i(t)}}{\sigma_{\kappa_i(t)} + \epsilon}, \tag{5}$$

where $\epsilon > 0$ is a small constant for numerical stability. We optimize the policy with a PPO-style clipped objective (Schulman et al., 2017) using the hierarchy-aligned advantages, i.e.,

$$\begin{aligned} \mathcal{J}_{\text{CAPO}}(\theta) = \mathbb{E}_{(q,\mathcal{Y}^*)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \Bigg[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \\ \min\Big(r_{i,t}(\theta)\,\hat{A}_{i,t},\ \text{clip}\big(r_{i,t}(\theta), 1-\varepsilon_{\text{L}}, 1+\varepsilon_{\text{H}}\big)\,\hat{A}_{i,t}\Big)\Bigg] \\ \text{s.t.} \quad \text{Var}\Big(\{\bar{R}_i\}_{i=1}^{G}\Big) > \delta, \qquad \bar{R}_i = R^{\text{lex}}\Big(\hat{\mathcal{Y}}_i, \mathcal{Y}^*\Big) \end{aligned} \tag{6}$$

Following DAPO (Yu et al., 2025), we adopt asymmetric clipping with hyper-parameters ε_{L} and ε_{H}. Here, $r_{i,t}(\theta)$ is the standard importance ratio, i.e.,

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})},$$

where π_θ is the policy being optimized and $\pi_{\theta_{\text{old}}}$ is the fixed reference policy. As training progresses, sampled rollout groups for a fixed query may exhibit low-variance rewards, making group-relative normalization unstable. We therefore drop groups with $\text{Var}(\{\bar{R}_i\}_{i=1}^{G}) < \delta$.

4 Experiments

4.1 Experimental Setup

The experiments cover a diverse set of LLMs, including open-source models from the Qwen3 family (Qwen3-8B/14B/32B and Qwen3-30B-MoE)[2], a domain-specific financial model (Fin-R1)[3], and LLaMA-3.1-8B-Instruct[4]. We also report the results of several proprietary models (GPT-series, Claude-series, and Gemini-series).

Datasets: We trained on the curated FHTC-26K dataset (§ 3.1). Each tool is associated with up to four instances. Evaluation employed the test set of 5,251 real-world user queries. We additionally trained and evaluated on ToolBench (Qin et al., 2024) to assess generalization to open-ended tool invocation beyond FHTC-26K.

Metrics: For FHTC-26K, we report the LLM-judged tool-invocation accuracy score (%). For each generated invocation, a multi-vote judge determines whether the tool path and call are semantically consistent with the user query; the reported score is the average judged correctness over the test set. For ToolBench, we report Macro Precision (%) following the protocol in Appendix E.

[2] https://huggingface.co/collections/Qwen/qwen3
[3] https://huggingface.co/SUFE-AIFLM-Lab/Fin-R1
[4] https://huggingface.co/meta-llama

Method	Qwen3-8B	Qwen3-14B	Qwen3-32B	Qwen3-30B-MoE	Fin-R1	LLaMA-3.1-8B
ToolLLM (Qin et al., 2024)	51.04	54.96	57.35	61.57	54.52	53.48
GRPO (Shao et al., 2024)	59.66	64.69	66.43	70.83	65.24	64.37
DAPO (Yu et al., 2025)	61.52	67.11	72.89	76.31	64.68	64.22
CAPO (w/o H.)	68.27	73.80	75.59	80.74	69.50	69.58
CAPO (ours)	**69.44**	**75.05**	**77.07**	**81.87**	**72.81**	**70.26**

Table 1: Comparison of CAPO against ToolLLM, DAPO, and GRPO across different LLMs on FHTC-26K, as measured by LLM-judged tool-invocation accuracy score (%). The best and second-best results are highlighted in **bold** and underlined fonts, respectively. "H." denotes hierarchical reward.

RL Setup: All RL algorithms incorporated a format-specific reward. For our *hybrid reward* in § 3.2.1, we set $\lambda = 0.5$ in Eq. (3). *Accuracy-based* rewards were only used by RL-based baselines, with the reward weight fixed to 1.0. Models were trained for one epoch on the prepared dataset. For the ablation study on RM-based RL, we compute RL rewards using a human-preference-aligned reward model, specifically PoLAR-7B (Dou et al., 2026). All experiments were conducted on the VERL platform (Sheng et al., 2025), with a cluster comprising two nodes, each equipped with 8 NVIDIA H100 GPUs.

4.2 Experimental Results

Overall Performance: On FHTC-26K, Table 1 compares CAPO with a *SFT-based* baseline for open-ended tool invocation, ToolLLM (Qin et al., 2024), and two *RL-based* methods, GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025), which further optimize cold-start/SFT-initialized models using an accuracy reward. We evaluate across diverse LLM backbones and parameter scales, including Qwen3 models (8B/14B/32B/30B-MoE), Fin-R1 (Liu et al., 2025a), and LLaMA-3.1-8B.

CAPO achieves the best performance among all evaluated models, demonstrating robust gains across dense and MoE architectures and parameter scales. Notably, CAPO without hierarchical reward already outperforms GRPO and DAPO. This indicates that replacing sparse accuracy rewards with lexical-overlap reward yields denser optimization feedback and helps the model track incremental improvements in tool-use reasoning. Adding hierarchical reward further improves performance by assigning credit to high-level routing decisions through hierarchy-consistent feedback and segment-aligned advantages. All RL-based methods outperform the SFT-only baseline, suggesting that supervised imitation alone is insufficient for correcting fine-grained reasoning errors in long-horizon, hierarchy-constrained decisions.

Method	Score (%)
RAG + GPT-4o	52.66
RAG + DeepSeek-V3.2	54.56
RAG + Claude4.5-sonnet	55.74
RAG + Qwen3-max	55.89
RAG + Gemini3-pro	57.70
RAG + GPT-5.2	58.12
CAPO w Qwen3-8B	**69.44**

Table 2: Comparison of CAPO with Qwen3-8B against RAG on different open-source and proprietary LLMs.

Across the Qwen series, performance improves with model scale, indicating that stronger backbones better capture functional distinctions. Qwen3-30B-MoE outperforms the similarly sized dense model, suggesting that specialized experts help with heterogeneous tool categories, while Fin-R1 exceeds Qwen3-8B and demonstrates the value of high-quality domain data for specialized tool invocation.

CAPO vs. RAG: Table 2 compares CAPO with no-finetuning retrieval-augmented prompting baselines. Directly prompting a frontier model with the full tool library is impractical: even a compact schema representation of 26,314 tools at roughly 100 tokens per tool would require approximately 2.6M schema tokens before the user query and output instructions. We therefore use a practical RAG pipeline, where an embedding retriever selects candidate tools from the full library and an LLM performs the final tool selection and invocation from the retrieved path, description, and schema fields. Across representative LLMs, RAG remains far below CAPO even with frontier-scale models, suggesting that retrieval alone is insufficient for reliable invocation in large hierarchical libraries. This gap indicates three key limitations: (1) embedding models struggle with fine-grained tool semantics; (2) LLM answer identification deteriorates in excessively long contexts despite strong needle-in-

Group	ToolLLM (Qin et al., 2024)	CAPO	Δ
G1	70.32	**82.05**	+11.73
G2	52.13	**53.30**	+1.17
G3	55.09	**58.21**	+3.12
Avg.	59.18	**64.52**	+5.34

Table 3: CAPO vs. ToolLLM on ToolBench (Macro Precision, %). G1, G2, and G3 for task groups; Δ for the absolute improvement of CAPO over ToolLLM.

Reward Function	**Score (%)**
Accuracy Reward (Shao et al., 2024)	61.52
PoLAR (Dou et al., 2026)	66.87
Lexical Reward	68.27
Hybrid Reward (Lexical & Hierarchical)	**69.44**

Table 4: Ablation results on reward functions for RL.

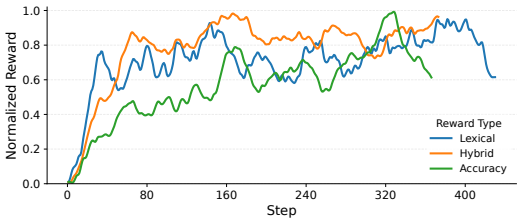

Figure 3: Reward trajectories over training steps.

Reward	**Advantage Function**	**Score (%)**
Hybrid	Group-Level (Shao et al., 2024)	68.30
	RLOO (Kool et al., 2019)	67.72
	GSPO (Zheng et al., 2025a)	68.61
	Causally-Decoupled	**69.44**

Table 5: Ablation results for advantage functions.

haystack reasoning; and (3) LLMs trained on general Web corpora lack domain-specific adaptation, yielding suboptimal performance.

Performance of CAPO on ToolBench: We further evaluate CAPO on the open-source ToolBench benchmark using Qwen3-14B to assess generalization beyond FHTC-26K in open-ended, multi-tool invocation settings. We report *Macro Precision* under a standardized comparison protocol that aligns ToolBench evaluation with our open-ended setting and defer full metric and protocol details to Appendix E. As shown in Table 3, CAPO consistently outperforms the ToolBench SFT baseline (ToolLLM) across all three task groups, yielding an average absolute improvement of 5.34% in Macro Precision. These gains indicate that CAPO remains effective beyond curated traces and that the hybrid reward, together with hierarchy-aligned, causally-decoupled optimization, improves tool routing in broad-coverage, general-domain tool-use settings with large tool libraries.

Reward Function Ablation: Table 4 shows how reward design affects RL performance on Qwen3-8B. The accuracy reward performs worst due to sparsity, while PoLAR improves over it but remains below lexical reward for fine-grained tool distinctions. Lexical reward provides denser partial credit for canonical tool traces. Appendix G further shows that it correlates with ground-truth routing correctness and improves leaf-level accuracy, rather than merely rewarding surface overlap. Adding hierarchical reward further improves performance by assigning learning signals to intermediate routing steps, enhancing prefix consistency and final leaf-tool accuracy.

Effect of Reward Design on Training: Figure 3 shows normalized reward trajectories over training steps for different reward designs; rewards are normalized because their scales differ. The accuracy reward is sparse, yielding few effective early learning signals and slowing optimization. The lexical reward accelerates early convergence through dense partial credit, but has higher step-to-step variance and becomes less discriminative among near-correct traces. The hybrid reward combines lexical density with hierarchy-consistent supervision, yielding the fastest and most stable convergence.

Advantage Function Ablation: Table 5 compares advantage estimators under the same hybrid reward on Qwen3-8B. Causally-Decoupled (C-D) advantage consistently outperforms group-wise variants (group-level normalization, RLOO, and GSPO), indicating that segment-aligned normalization improves credit assignment in hierarchical tool invocation. By normalizing rewards within hierarchy-aligned trace segments, C-D advantage better attributes learning signals to upstream routing decisions in the reasoning trace.

5 Conclusion

To enable reliable open-ended full-library tool invocation at scale, we introduce FHTC-26K, a dataset of 26,314 hierarchical tools with reasoning traces for robust evaluation, and propose the Causally-decoupled Advantage Policy Optimization (CAPO) framework, which addresses sparse rewards and improves training efficiency through a hybrid reward combining lexical overlap with explicit hierarchy-consistency feedback. Extensive experiments show that CAPO substantially enhances LLM performance on real-world tool-invocation tasks.

Limitations

This work addresses the problem of open-ended tool invocation at scale, yet several limitations remain. First, FHTC-26K is partially synthesized; despite being adversarially filtered, it may still contain generator- and judge-induced biases. Second, CAPO relies on fixed-depth, hierarchy-aligned segmentation, leaving extensions to dynamic depths, multiple routing paths, and multi-turn interactions as open challenges. Finally, our primary evaluation focuses on a professional financial hierarchy; extending CAPO to dynamic, multi-domain API ecosystems is left for future work.

Ethical Considerations

Our approach still shares risks common to tool-augmented LLM assistants: biased data or incorrect tool use may produce plausible but misleading results, particularly in high-stakes settings. CAPO should therefore be used with human oversight for decision support, not for automated decisions or direct real-world trading.

References

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Yuefeng Huang, Xiangcheng Liu, Wang Xinzhi, and Wu Liu. 2025. ACEBench: A comprehensive evaluation of LLM tool usage. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12970–12998.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multilinguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Lei Cheng, Yekun Chai, Shuohuan Wang, Yu Sun, Hao Tian, Ningyu Zhang, and Hua Wu. 2024. Tool-augmented reward modeling. In *The Twelfth International Conference on Learning Representations*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Shihan Dou, Shichun Liu, Yuming Yang, Yicheng Zou, Yunhua Zhou, Shuhao Xing, Chenhao Huang, Qiming Ge, Demin Song, Haijun Lv, Songyang Gao, Chengqi Lv, Enyu Zhou, Honglin Guo, Zhiheng Xi, Wenwei Zhang, Qipeng Guo, Qi Zhang, Xipeng Qiu, and 3 others. 2026. Pre-trained policy discriminators are general reward models. *Preprint*, arXiv:2507.05197.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. 2025. WebEvolver: Enhancing web agent self-improvement with co-evolving world model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8970–8986.

Max Grusky. 2023. Rogue scores. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1914–1934.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638.

Liang Hu, Jianpeng Jiao, Jiashuo Liu, Yanle Ren, Zhoufutu Wen, Kaiyuan Zhang, Xuanliang Zhang, Xiang Gao, Tianci He, Fei Hu, Yali Liao, Zaiyuan Wang, Chenghao Yang, Qianyu Yang, Mingren Yin, Zhiyuan Zeng, Ge Zhang, Xinyi Zhang, Xiying Zhao, and 4 others. 2025. FinSearchComp: Towards a realistic, expert-level evaluation of financial search and reasoning. *Preprint*, arXiv:2509.13160.

Wouter Kool, Herke van Hoof, and Max Welling. 2019. Buy 4 REINFORCE samples, get a baseline for free! In *ICLR 2019 Workshop on Deep Reinforcement Learning Meets Structured Prediction*.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1078–1088.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116.

Zhiwei Li, Yong Hu, and Wenqing Wang. 2025. Encouraging good processes without the need for good answers: Reinforcement learning for LLM agent planning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1654–1666.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Zhaowei Liu, Xin Guo, Zhi Yang, Fangqi Lou, Lingfeng Zeng, Jinyi Niu, Mengping Li, Qi Qi, Zhiqiang Liu, Yiyang Han, Dongpo Cheng, Ronghao Chen, Huacan Wang, Xingdong Feng, Huixia Judy Wang, Chengchun Shi, and Liwen Zhang. 2025a. Fin-R1: A large language model for financial reasoning through reinforcement learning. *Preprint*, arXiv:2503.16252.

Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. 2025b. Inference-time scaling for generalist reward modeling. *Preprint*, arXiv:2504.02495.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 48371–48392.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. *Advances in Neural Information Processing Systems*, 37:126544–126565.

Yujia Qin, Sheng-Chen Cai, Yuxiang Geng, Jiaqi Chen, Zexuan Zhou, Sheng Zhao, Yan-Fan Li, Ganqu Niu, Zhiyuan Liu, Juanzi Li, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Yulei Qin, Xiaoyu Tan, Zhengbao He, Gang Li, Haojia Lin, Zongyi Li, Zihan Xu, Yuchen Shi, Siqi Cai, Renting Rui, Shaofei Cai, Yuzheng Cai, Xuan Zhang, Sheng Ye, Ke Li, and Xing Sun. 2026. Learn the ropes, then trust the wins: Self-imitation with progressive exploration for agentic reinforcement learning. In *The Fourteenth International Conference on Learning Representations*.

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

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

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

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

Venkat Krishna Srinivasan, Zhen Dong, Banghua Zhu, Brian Yu, Hanzi Mao, Damon Mosk-Aoyama, Kurt Keutzer, Jiantao Jiao, and Jian Zhang. 2023. Nexus-Raven: a commercially-permissive language model for function calling. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023. ToolAlpaca: Generalized tool learning for language models with 3000 simulated cases. *Preprint*, arXiv:2306.05301.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. 2025. J1: Incentivizing thinking in LLM-as-a-judge via reinforcement learning. *Preprint*, arXiv:2505.10320.

Mengsong Wu, Tong Zhu, Han Han, Chuanyuan Tan, Xiang Zhang, and Wenliang Chen. 2024. Seal-Tools: Self-instruct tool learning dataset for agent tuning and detailed benchmark. In *Natural Language Processing and Chinese Computing - 13th National CCF Conference, NLPCC 2024, Hangzhou, China, November 1-3, 2024, Proceedings, Part II*, pages 372–384.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. DAPO: An open-source LLM reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Yirong Zeng, Xiao Ding, Yutai Hou, Yuxian Wang, Li Du, Juyi Dai, Qiuyang Ding, Duyu Tang, Dandan Tu, Weiwen Liu, Bing Qin, and Ting Liu. 2025a. Tool zero: Training tool-augmented LLMs via pure RL from scratch. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 9135–9147.

Yirong Zeng, Xiao Ding, Yuxian Wang, Weiwen Liu, Yutai Hou, Wu Ning, Xu Huang, Duyu Tang, Dandan Tu, Bing Qin, and Ting Liu. 2025b. iTool: Reinforced fine-tuning with dynamic deficiency calibration for advanced tool use. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13901–13916.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin

Ding, and Jingren Zhou. 2025. AgentEvolver: Towards efficient self-evolving agent system. *Preprint*, arXiv:2511.10395.

Ding-Chu Zhang, Yida Zhao, Jialong Wu, Liwen Zhang, Baixuan Li, Wenbiao Yin, Yong Jiang, Yu-Feng Li, Kewei Tu, Pengjun Xie, and Fei Huang. 2025a. EvolveSearch: An iterative self-evolving search agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13134–13147.

Kangning Zhang, Wenxiang Jiao, Kounianhua Du, Yuan Lu, Weiwen Liu, Weinan Zhang, and Yong Yu. 2025b. LoopTool: Closing the data-training loop for robust LLM tool calls. *Preprint*, arXiv:2511.09148.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025a. Group sequence policy optimization. *Preprint*, arXiv:2507.18071.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025b. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 414–431.

A Tool Library Hierarchy

Figure 4 illustrates the hierarchical organization of the tool library used in FHTC-26K, where tools are grouped into a multi-hierarchy taxonomy that supports step-wise routing from coarse domains to fine-grained functionalities. In the illustrated branch, the root node corresponds to *U.S. Equities*, which then branches into major functional groups such as *Valuation Metrics*, *Basic Information*, and *Financial Analysis*. The *Financial Analysis* subtree then expands into finer categories, including *Solvency Metrics*, *Growth Metrics*, and *Capital Structure*, etc. In addition, the hierarchy includes statement-centric branches and sector-specific structures, which together create a high branching factor and a long-horizon decision space.

For presentation purposes, Figure 4 visualizes only the U.S. market branch of the full library, and we merge semantically similar intermediate hierarchies while pruning small or less frequently used subtrees, so the figure highlights the dominant routing structure rather than exhaustively listing all nodes. Importantly, it depicts only the hierarchy levels used for routing, whereas concrete tools typically reside two to three levels deeper. For example, a node such as *U.S. Equities / Financial Analysis / Solvency Metrics / Interest Coverage (EBIT)* can further expand into accounting variants (e.g., *TTM* vs. *MRQ*) and then into a specific callable endpoint (e.g., a tool that returns the interest-coverage series under a chosen reporting convention). These deeper expansions yield a large set of distinct, closely related tools, contributing to the overall scale and complexity of FHTC-26K, which contains 26,314 hierarchically organized tools. Using the Qwen tokenizer, each FHTC-26K training example contains on average 23.86 query tokens and 67.03 final-answer tokens. The held-out test set contains 12.19 query tokens and 60.69 final-answer tokens per sample on average. Final-answer counts exclude `<think>` reasoning traces.

Item	Description
Tool library	26,314 financial tools
Training data	115,564 synthetic instances, up to 4 per tool
Test data	5,251 real-world user queries
Query tokens	Train avg. 23.86; test avg. 12.19
Answer tokens	Train avg. 67.03; test avg. 60.69
Tool metadata	Description, aliases, hierarchy path, schema
Trace format	Hierarchy-aware `<think>` + `<answer>`
Domain	Professional financial data terminals

Table 6: Statistics of FHTC-26K, where training examples are synthesized from curated tool metadata and real-query-style seeds and evaluation uses held-out real-world user queries without oracle candidate shortlists. Token statistics are computed with the Qwen tokenizer and exclude reasoning traces from final-answer counts.

B FHTC-26K Dataset Statistics

Table 6 summarizes the statistics of FHTC-26K. The benchmark is sourced from the financial domain but spans deeply nested subdomains, including equities, funds, valuation and solvency metrics, technical indicators, and financial statement fields. Each tool is denoted by a callable schema, functional description, domain-specific aliases, and hierarchy path, enabling both schema-valid invocation and level-by-level routing supervision.

C Results for Computational Efficiency

We assess the computational overhead of our proposed method, CAPO, compared to the baseline DAPO. Table 7 presents a detailed breakdown of the time consumption per training step, averaged over the training process. The total training time per step for CAPO (18.61 s) is highly comparable to that of DAPO (18.19 s), with only a marginal increase of approximately 2.3%. While the *Reward* and *Advantage (Adv.)* calculation stages in

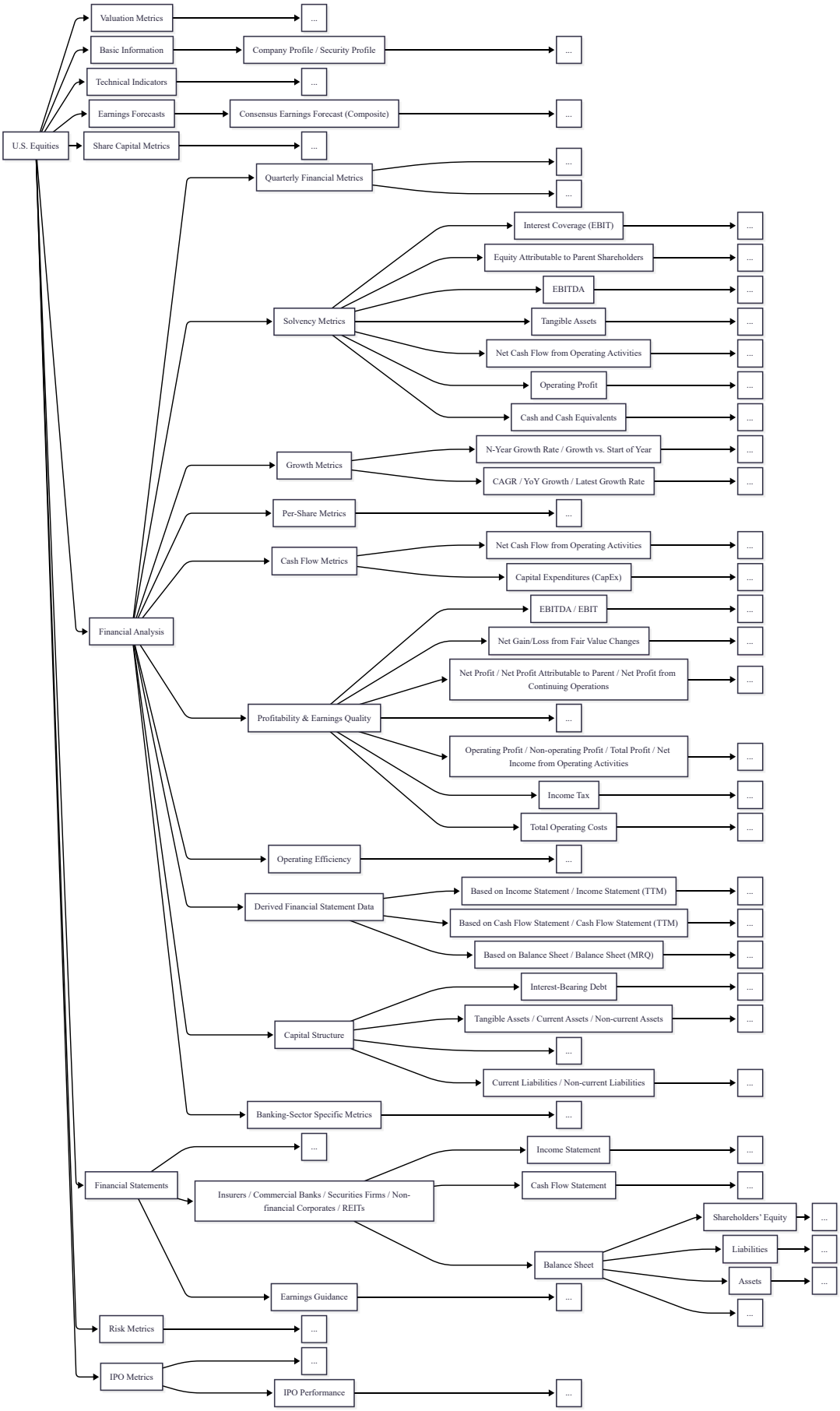

Figure 4: Tool library hierarchy in FHTC-26K (U.S. equities branch).

Stage	DAPO	CAPO (Ours)
Rollout Generation	13.08	12.84
Reward Calculation	0.18	0.76
Advantage Calculation	0.00	0.20
Update Policy	3.22	3.25
Reference LogProb	1.71	1.57
Total	18.19	18.61

Table 7: Computational time (s) per training step.

Method	Size	Qwen-7B	Qwen-14B	Qwen-32B
SFT	26K	58.45	60.46	57.95
	96K	51.04	54.96	55.15
CAPO	26K	**69.87**	**75.14**	76.67
	96K	69.44	75.05	**77.07**

Table 8: Effect of training set size on the accuracy score (%) of SFT and CAPO.

CAPO take slightly longer (0.76 s and 0.20 s, respectively) due to the computation of causal dependencies and the hybrid structural constraints, this overhead is negligible in the overall training loop. Crucially, CAPO achieves this efficiency without relying on external Reward Models (RMs) or LLM-based judges. Instead, it utilizes a carefully designed statistical reward function. This design choice avoids the substantial computational cost and latency associated with performing additional forward passes on large reward models during training. As shown in our ablation studies (see § 4.2), this statistical approach not only maintains high computational efficiency but also achieves superior performance over RM-based baselines.

D Effect of Training Data Scale

As evidenced by Table 8, performance improvements for SFT and CAPO remain negligible as the training data size increases from 26K to 96K instances. This manifests more prominently in SFT models, where larger datasets lead to performance degradation (e.g., Qwen-7B drops from 58.45% to 51.04%). Such deterioration suggests that SFT primarily acquires superficial semantic mappings between tool names and contexts, failing to develop the complex reasoning capabilities essential for robust tool exploration. Scaling limited-diversity data appears to exacerbate this fundamental constraint.

The marginal performance gap between the two CAPO models trained on 26K and 96K instances (e.g., 69.87% vs. 69.44% for Qwen-7B) indicates that data diversity constitutes a critical bottleneck. As training iterations increase with larger datasets, models receive insufficiently diverse reward signals, hindering knowledge acquisition beyond initial patterns. These findings collectively demonstrate that, in domain-specific tool usage scenarios, data quality substantially outweighs scale, with high-diversity examples providing significantly greater value than larger datasets that contain merely superficial variations.

Expert-Guided Iterative Refinement Prompt

Role Setting:
You are a professional financial analyst tasked with generating standardized aliases for financial indicators.
Input Format Specifications:
- "**answer**": The official full name of the indicator, must be treated in its entirety, including any suffixes (e.g., TTM, FY, etc.).
- "**type**": The type of indicator, for reference.

Core Objective:
Your goal is to consolidate the input information and output a clean, comprehensive, and professionally consistent list of aliases. This requires combining extraction, generation, and refinement processes.
Thought Process:
1. Parse Input:
- Extract the official full name of the indicator, including any suffixes, from the `"answer"` field of the input JSON. [...]
2. Define Metric & Suffix:
- Based on the indicator name extracted from the `"answer"` field, understand and summarize its core financial meaning.
- Identify and explain suffixes: Recognize statistical suffixes (e.g., TTM, LFY, MRQ, etc.) in the indicator name. [...]
3. Initial Alias Generation:
Based on your definition of the indicator's core concept, generate a series of possible English aliases.
a. Formal & Regulatory Terms: Refer to standard expressions under International Financial Reporting Standards (IFRS) and **[...]**
b. Common Industry Aliases: Refer to widely used names in mainstream financial data terminals **...**
[...]
Output Format Requirements:
- Your final response must be a correctly JSON format object.[...] **Example:**

{
"metric_name": "The indicator name from the 'answer' field in the input",
"definition": "A concise, professional description of the indicator's core meaning, including an explanation of the suffix's importance and handling.",
"aliases": [
"The final refined, deduplicated, suffix-consistent, and lowercase-converted list of English aliases"]
}
Given financial indicator:
{indicator}

Figure 5: Expert-guided iterative refinement prompting for knowledge curation.

E ToolBench Evaluation Protocol

Metric We report *Macro Precision* (%) on different ToolBench task groups. For an instance with ground-truth tool set Y^* and predicted tool set $\hat{Y}$, precision is defined as

$$\mathrm{Prec}(\hat{Y}, Y^*) = \frac{|\hat{Y} \cap Y^*|}{|\hat{Y}|}. \quad (7)$$

The macro precision is computed by averaging $\mathrm{Prec}(\hat{Y}, Y^*)$ over instances within each task group and then reporting group-level scores.

Cardinality-Matched Evaluation ToolBench baselines (e.g., ToolLLM) typically retrieve tools

Expertise-Aware Seeded Prompt

Role Setting
You are a Financial Query Expert. Your core tasks are:
1. Logic Validation: First, evaluate whether the input indicator combinations contain logical conflicts or contradictions, ensuring that the query object is an asset.
2. Query Generation: Construct professional, diverse, and logically rigorous analytical, backtesting, forecasting queries using given indicators [...]
3. Auto-Completion: The corresponding queries should include asset information.
Input Fields
- Fundamental Indicators (fund_indicators): {fund_indicators}
- Macroeconomic Indicators (macro_indicators): {macro_indicators}

[...]
- Example of Queries: {seed_sample}

Output Requirements
If the input indicators are logically consistent: Generate 5 logically complete queries at once, strictly adhering to the JSON structure below: [...]
Core Generation Rules
- Logic Validation Principle: Before generation, evaluate all indicators. [...]
- Indicator Exclusivity Principle: Generated query must and only use the indicators explicitly provided in the input fields. Introducing any financial indicators not mentioned in the input is strictly prohibited. [...]
- Natural Expression Principle: Generated queries must align with natural human expression:
  - Integrate indicators naturally into the sentence context[...]
  - Use common financial terminology rather than raw indicator labels.
  - Maintain a professional yet fluent financial analysis style.
- Strict Format Alignment: The output must be a list of multiple JSON objects. Each object contains a `query` string and an `indicators` array.
  - query: A natural language query.
  - indicators: An array where each object represents an indicator used in the input query.
- Query Diversity: Generated queries should reflect different intents (e.g., backtesting,screening, correlation analysis) and cater to both professional quant analysts and general investors.
- General Rules: Geographic labeling, time ranges, special identifier handling, single-asset restrictions, etc., still apply and should be incorporated into the `query` string construction. [...]
- Expression Standards:
  - Use professional yet natural English expressions.
  - Imply indicators through context rather than direct listing (e.g., "high-ROE stocks" instead of "screening using ROE indicator"). [...]
- Parentheses `()` Handling Rules: When an economic indicator's `name` or `aliases_name` includes parentheses `()`, Do not directly copy the parentheses `()` and their contents (e.g., `(MoM)`). [...]

[...]
Output Format Standards:
- Strictly adhere to JSON syntax rules.
- Enclose strings with English double quotes `"..."`.

[...]

Figure 6: Expertise-aware seeded prompting for diverse query generation.

Hierarchy-Aware Chain-of-Thought Prompt

Role Setting
Provide clear and concise financial analysis based on given financial queries and their corresponding indicators. The core objective is to explain the meaning of the query and elucidate the role and significance of relevant indicators in financial analysis.
Analysis Requirements
1. Query Intent Explanation:
   Clearly explain the core intent of the user's query - what the user seeks to understand.
2. Indicator Meaning Explanation:
   Explain why the provided financial indicators correspond to the query and their role/significance in the financial analysis.
3. Logical Coherence:
   Ensure clear and smooth logic from query to indicators to analysis, explaining the relevance at each step.
4. Conciseness:
   Avoid redundant information, expressing core content with minimal text.
5. Emphasis on Key Points:
   Highlight the query's core objective and analysis focus for quick comprehension.
6. Direct Output:
   Output only the financial analysis results without repeating the query or indicators.

Examples
- Financial Query : "What is the 2023 forecasted EPS for 210002.BI?"
- Financial Indicators : /Index All Indicators/Earnings Forecast/Forecast EPS
- Analysis:

1. Asset Class: Index
   - The query keyword "210002.BI" indicates an index asset class. "BI" represents Bloomberg Index, while "210002" is the specific index code. Thus, the user is querying index assets.
2. Indicator Category: Earnings Forecast
   - The query "2023 forecasted EPS" clearly points to future financial performance estimates, placing this under earnings forecasts.
3. Specific Indicator: Forecast EPS
   - The final indicator "Forecast EPS" perfectly matches the user's "2023 forecasted EPS" request, accurately reflecting their need for future EPS projections.

Summary:
This query targets the Earnings Forecast indicator for Index assets, specifically Forecast EPS. So the answer is \boxed{/Index All Indicators/Earnings Forecast/Forecast EPS}
Input
- Financial Query (query): {query}
- Financial Indicators (indicators): {indicators}
- Financial Indicator Hierarchy Description: {hierarchies}

Output
Financial Analysis:

Figure 7: Hierarchy-aware chain-of-thought (CoT) prompting for decompositional reasoning.

via similarity ranking, whereas CAPO generates an ordered multi-tool invocation trace that may vary in length in open-ended settings. To standardize evaluation and isolate tool identification quality under multi-tool scenarios, we match the predicted tool-set size to the ground-truth cardinality for each instance. Concretely, for an instance whose ground-truth tool set has size $k = |Y^*|$, we evaluate retrieval-based baselines using their top-k retrieved tools as $\hat{Y}$. For CAPO, we parse the generated tool sequence in the `<answer>` span and truncate it to the first k tools to form $\hat{Y}$ before computing precision. This protocol controls for tool-set cardinality across methods and avoids confounding effects due to predicting too many or too few tools in open-ended generation.

F Prompt Templates for FHTC-26K Construction

We describe the prompt templates used to construct the CAPO training data. It covers how we format tool metadata, synthesize the tool exploration dataset (FHTC-26K), generate the QA reasoning cold-start data, and perform adversarial filtering for data curation. Figure 5 presents the expert-guided iterative refinement template for curating tool/indicator semantics; Figure 6 shows the expertise-aware seeded template for diverse query synthesis in FHTC-26K; Figure 7 provides the hierarchy-aware chain-of-thought template for producing decompositional reasoning traces; and Figure 8 describes the Turing-style judge template used to filter out synthetic artifacts. Due to space limits, we include only the essential prompt structures and key constraints in these figures, omitting

Adversarial Turing Judge Prompt

Role Setting
Analyze the following two text samples and determine which one is more likely to be human-written based on these criteria:
1. Naturalness:
- Evaluate flow, conversational rhythm, and intuitive phrasing
- Identify awkward constructions or unnatural word choices

2. Complexity:
- Assess sentence structure variety (e.g., mixing simple/complex sentences)
- Examine logical coherence between ideas
- Note inappropriate complexity (e.g., overly technical terms in casual contexts)

3. Human-like Qualities:
- Detect imperfections like colloquialisms, reasonable digressions, or contextual pauses
- Identify emotional resonance and subjective perspective
- Evaluate contextual awareness (e.g., cultural references, implied knowledge)

Input Texts:
Text A: {expert written/strategy synthsis}
Text B: {expert written/strategy synthsis}
Output Format:
Analysis:
1. Naturalness:
- Text A: [Observation]
- Text B: [Observation]

2. Complexity:
- Text A: [Observation]
- Text B: [Observation]

3. Human-like Qualities:
- Text A: [Observation]
- Text B: [Observation]

Conclusion: [Text A/B] is more likely human-written because [Concise synthesis of key evidence]"

Figure 8: Adversarial Turing judge prompting for synthetic data filtration.

auxiliary instructions and boilerplate that do not affect the core logic.

G Additional Diagnostic Experiments

We include additional diagnostics to complement the accuracy results in Table 1. These experiments focus on two questions raised by large-scale hierarchical tool invocation: whether model improvements correspond to better hierarchy-level routing rather than only better final matching, and whether the auxiliary reward signals are informative enough to guide learning before an entire tool invocation becomes exactly correct. All results in this section are intended as diagnostic analyses over the available judged-output pool, rather than replacements for the main held-out benchmark comparison.

Hierarchy-aware Metrics Since the FHTC-26K dataset contains variable-depth tool paths, fixed-level accuracy cannot faithfully compare predictions across all examples. We therefore report normalized prefix accuracy (NPA), defined as the length of the longest common prefix between the predicted and reference tool paths, normalized by the reference depth. We also report Prefix@1,2,3, which measures whether the prediction preserves the first, second, or third hierarchy decisions, and LeafExact, which measures exact agreement at the final tool node. These metrics separate upstream routing errors from leaf-level confusions among sibling tools.

Diagnostic Protocol We parse each model output into a canonical tool path and call specification, then compare it with the reference hierarchy path and invocation schema. Table 9 reports hierarchy-routing diagnostics across baseline and CAPO variants under the default Qwen3-8B setting. It provides a reference-based routing breakdown across methods. Table 10 further examines whether reward signals correlate with ground-truth path correctness, using path depth, query length, and random scores as negative controls. It directly tests whether the training rewards track ground-truth hierarchy correctness rather than merely the lexical surface form.

Case Analysis Table 11 presents several representative Qwen3-8B examples comparing CAPO with CAPO w/o H. Across technical indicators, recommendation/ranking queries, asset-domain selection, and financial-statement fields, CAPO more consistently preserves the intended semantic hierarchy and avoids drifting to nearby but incorrect sibling indicators. We also include two remaining error cases to show that CAPO is not immune to fine-grained market-domain or ratio-family confusions, although these failures are not the dominant pattern in the paired evaluation. In the current-ratio case, CAPO routes the query to a leverage-oriented capital-structure branch and selects Debt-to-Asset Ratio, whereas the request targets a liquidity ratio under solvency analysis. In the SPY moving-average case, CAPO identifies the correct indicator family but selects the wrong market hierarchy, since SPY is a U.S. ETF rather than a China fund.

Method	NPA	Prefix@1	Prefix@2	Prefix@3	LeafExact
SFT	83.62	94.29	88.38	78.57	74.02
GRPO	84.73	94.93	90.07	83.01	78.73
DAPO	85.73	94.89	91.19	84.04	78.93
CAPO w/o H.	86.54	95.23	91.64	81.07	79.51
CAPO	88.80	96.12	92.78	84.75	81.76

Table 9: Qwen3-8B hierarchy-routing diagnostics against ground-truth reference paths. Metrics are computed from parsed model outputs and compare prefix-level and leaf-level correctness.

Signal	Non-zero (%)	ρ w/ NPA	ρ w/ LeafExact	ρ w/ Prefix@2	ρ w/ Prefix@3
Soft hierarchical reward	95.2	0.996	0.988	0.718	0.912
Soft leaf reward	80.7	0.988	0.996	0.642	0.881
Path-depth control	100.0	-0.032	-0.036	0.062	0.000
Query-length control	100.0	-0.011	-0.018	0.021	0.010
Random control	100.0	-0.006	-0.007	0.003	-0.004

Table 10: Reward-signal diagnostics on the full judged evaluation set. Soft rewards use the Qwen3-235B vote probabilities before thresholding; path depth, query length, and random scores are negative controls. Lexical reward is omitted for this recovered judged pool because gold canonical invocations are unavailable.

Type	Query	CAPO predicted path	CAPO w/o H predicted path
Technical indicator	Find stocks where the 300-day moving average is below the 500-day moving average	Stocks → Technical Patterns → Trend Indicators → Simple Moving Average	Stocks → Technical Patterns → Trend Indicators → Mass Index
Technical indicator	Find stocks whose 21-day EMA is above 50-day EMA	Stocks → Technical Patterns → Trend Indicators → Exponential Moving Average	Stocks → Technical Indicators → Trend Indicators → EMD Indicator
Recommendation/ranking	Return stock codes ranked by quantitative recommendation score in the top ten	Stocks → Earnings Forecasts → Investment Rating Consensus → Quantitative Recommendation Ranking	Stocks → Earnings Forecasts → Forecast Details → Buy Recommendation Count
Recommendation/rating	Find securities that are rated as Strong Buy by TipRanks	Stocks → Earnings Forecasts → Investment Rating Consensus → Rating	Stocks → Earnings Forecasts → Forecast Details → Sell Rating Count
Asset/domain	Find financial securities with debt-to-equity ratio <0.5	Stocks → Financial Analysis → Capital Structure → Debt-to-Equity Ratio	Bonds → Analytics → Z-score Warning → Debt-to-Equity Ratio Below 0.5
Fund-market domain	Find the holding codes of ARKG	US Fund Indicators → U.S. ETF → Holdings → Holding Code	China Fund Indicators → Portfolio → Holdings → Top Holding Security Code
Financial line item	Find semiconductor securities with R&D investment >1B	Stocks → Notes → R&D Expense Details → R&D Investment	Stocks → Notes → R&D Expense Details → Lease Expense
Statement subtype	ONCY stock fiscal-year net income	Global Stocks → Financial Statements → General Corporates → Income Statement → Net Income	U.S. Stocks → Financial Statements → Commercial Banks → Income Statement → Net Income
Remaining error	Find healthcare stocks with current ratio greater than 1.5	Stocks → Financial Analysis → Capital Structure → Debt-to-Asset Ratio	Stocks → Financial Analysis → Solvency → Current Ratio
Remaining error	ETF SPY 9-day simple moving average	China Fund Indicators → Technical Indicators → Simple Moving Average	US Fund Indicators → U.S. ETF → Technical Indicators → Simple Moving Average

Table 11: Qwen3-8B cases with Qwen3-235B as a judge. Red text marks the erroneous path segments. The final two rows are the CAPO error cases.