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

Anonymous ACL submission

Abstract

Efficient tool invocation from large, hierarchical libraries remains a fundamental challenge for Large Language Models (LLMs), due to subtle semantic distinctions among tools and implicit conventions or hierarchical constraints that are rarely specified in formal documentation. Existing retriever-aware supervised fine-tuning methods are brittle when tools are textually similar yet functionally distinct, while reinforcement learning based approaches suffer from coarse credit assignment and unstable optimization due to sparse, delayed rewards and error propagation in long-horizon, hierarchical tool routing. To address these challenges, we propose Causally-decoupled Advantage Policy Optimization (CAPO), a 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 and structured feedback along the invocation trajectory. Additionally, we introduce FHTC-26K, a large-scale financial-domain benchmark with 26,314 professionally curated, hierarchically organized tools and expert-annotated invocation traces. Extensive experiments show that CAPO improves tool-invocation accuracy by 4.18%–7.94% over strong baselines while exhibiting substantially more stable optimization. Our code and data is available 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 models 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., 2025b; Zhang et al., 2025; Zeng et al., 2025a). However, most existing methods for tool invocation are trained and evaluated under *candidate-set* assumptions, where tool selection is performed over a small, pre-filtered shortlist. In contrast, industrial deployments require *open-ended full-library routing* over large, deeply hierarchical tool repositories, in which tools often exhibit substantial semantic overlap but differ in fine-grained scope, argument conventions, and implicit domain constraints. As a result, methods optimized under candidate-set assumptions often transfer poorly to industrial-scale tool ecosystems, revealing a key misalignment between current paradigms and real-world tool libraries.

Prior work on tool invocation largely 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., 2023); 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 cover the combinatorial diversity of tool usage in large libraries, leading to weak generalization and error amplification along long-horizon tool chains. To scale to large tool libraries, these methods often rely on retrieval to rank and shortlist tools given the query. However, conventional retrievers often fail when tools are *textually similar yet functionally distinct*, a pattern that is particularly prevalent the financial domain. For example, tools for valuation multiples may share near-identical surface descrip-

tions but differ in definition (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 developed and validated primarily under candidate-set assumptions, 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, which obscures credit assignment for early, high-impact routing decisions along the hierarchy and destabilizes learning in 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 where tools are easily separable at the surface level (Patil et al., 2023; Qin et al., 2024; Li et al., 2023). In such settings, simple semantic matching often suffices to identify the right tool. Other benchmarks adopt an *oracle candidate* protocol that supplies a prefiltered shortlist for each query (Chen et al., 2025; Patil et al., 2025), thereby removing the need for models to perform open ended selection over the full library. Both evaluation regimes differ from real professional settings, where LLMs must autonomously choose from tens of thousands of hierarchically organized tools under subtle conventions and hierarchy constraints. As a result, current evaluations fail to faithfully assess the reliability of long horizon hierarchical tool routing at industrial scale.

Taken together, existing tool invocation approaches fail to reflect how professional tool libraries are actually navigated: not by a single best-match decision, but via a sequence of hierarchy-conditioned choices. In deeply hierarchical libraries, a substantial fraction of errors originate 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 reflecting real user query distributions. Building upon this data, we propose Causally-decoupled Advantage Policy Optimization (CAPO), a hierarchy-constrained reinforcement learning 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. Concretely, our contributions are three-fold:

- We introduce FHTC-26K, a large-scale financial-domain benchmark of 26,314 hierarchical 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 Causally-decoupled Advantage Policy Optimization (CAPO), which improves token-level credit assignment for intermediate routing decisions via hierarchy-aligned advantages and a hybrid reward combining lexical overlap with explicit hierarchy-consistency feedback.
- We conduct extensive experiments over two open-ended tool-invocation benchmarks. Results demonstrate consistent and substantial improvements of CAPO over all baselines.

2 Related works

Supervised and retrieval-augmented paradigms. 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., 2023). Their effectiveness, however, degrades as repositories scale and tool semantics become increasingly fine-grained (Schick et al., 2023), while general-purpose embedding models (Chen et al., 2024) often fail to capture subtle functional distinctions, leading to tool and argument mis-specification.

Reinforcement learning (RL) paradigms. RL-based methods can be further categorized into

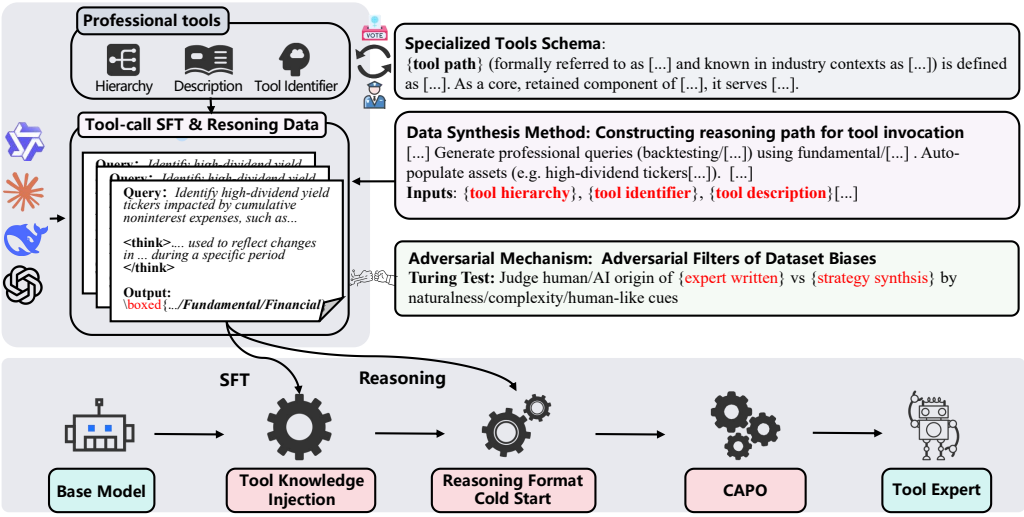

Figure 1: Open-ended tool invocation dataset construction and training pipeline for hierarchical tool invocation.

data-iterative and *exploration-driven* paradigms. Data-iterative approaches iteratively refine supervision via error-driven data synthesis or fragment-level correction, often coupled with preference optimization (Zeng et al., 2025b; Zhang et al., 2025). Exploration-driven approaches directly optimize tool-use policies via RL without SFT initialization, typically leveraging reward shaping to enforce format validity and semantic correctness (Zeng et al., 2025a). Despite their differing optimization strategies, both paradigms face substantial challenges in industrial-scale tool repositories, where the action space is extremely large and errors can accumulate over long-horizon, multi-tool compositions.

Tool Invocation Benchmark. Most existing tool-invocation benchmarks adopt a candidate-set setup, where each query comes with a short list of candidate tools and the model is evaluated on tool selection and argument filling within this restricted space, rather than open-ended tool discovery (Tang et al., 2023; team, 2023; Patil et al., 2025; Chen et al., 2025; Wu et al., 2024). Even few benchmarks that claim evaluation over a "full" toolset typically rely on relatively small repositories or simplified tool abstractions, limiting their ability to assess open-ended tool discovery under realistic hierarchical structures and fine-grained semantics (Li et al., 2023; Qin et al., 2024).

3 Method

We present an end-to-end framework designed for robust tool invocation in industrial-scale, deeply hierarchical repositories (as shown in Fig. 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 combining 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.

Semantic Foundation Construction. We construct a semantic foundation $\mathcal{K}_T$ for each tool T. Specifically, for a tool with hierarchical path $(H_1, \ldots, H_d)$, $\mathcal{K}_T$ aggregates three types of semantic metadata. First, the *tool descriptions* (i.e., Desc_{T_i}) provide LLM-distilled functional descriptions that goes beyond tool names and signatures, capturing fine-grained scope and intended usage. Second, the *expertise identifiers* (i.e., Identifier_{T_i}) collect domain-specific terminological aliases to

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

broaden lexical coverage for the underlying tool concept and better match real user phrasing. Third, the *hierarchy descriptions* (i.e., Desc_{H_i}) provide hierarchy-level semantic descriptions, explicitly encoding hierarchy-imposed constraints.

To ensure the reliability of the metadata, we adopt an iterative refinement protocol. Multiple 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) with a structured prompt template π. For 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 $\mathcal{S}_{\text{seed}}$:

$$q_i \sim \text{LLM}(\pi(T_i, \text{EIDs}, \text{Desc}_{T_i}, \mathcal{S}_{\text{seed}})). \quad (1)$$

Here, Desc_{T_i} enforces functional alignment, EID 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}:

$$\zeta_i \sim \text{LLM}\Big(\pi_{\text{trace}}(q_i, \text{Desc}_{T_i}, \{\text{Desc}_{H_k}\}_{k=1}^{d_i})\Big). \quad (2)$$

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 $\mathcal{S}_{\text{synth}} = \{(q_i, \zeta_i, T_i)\}_{i=1}^{N}$.

Adversarial Filtering and Distributional Alignment. To mitigate distributional drift and reduce synthetic artifacts in $\mathcal{S}_{\text{synth}}$, we adopt a Turing-style adversarial filtering scheme with an LLM-as-a-Judge (Le Bras et al., 2020). For each synthetic query $q_i \in \mathcal{S}_{\text{synth}}$, 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:

$$P_{\text{adv}}(q_i) = \frac{1}{M} \sum_{j=1}^{M} \mathbb{I}\Big(\mathcal{D}(q_i, q_{\text{real}}^{(j)}) = q_i\Big) \quad (3)$$

Here $\mathcal{D}$ is an LLM-based discriminator that selects the synthetic query from each pair. We construct FHTC-26K by retaining samples with $P_{\text{adv}}(q_i) \leq \tau$. This filtering 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., 2023).

Figure 2 overviews CAPO, a novel reinforcement learning framework which 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 broadcast across all tokens, and it enables finer-grained credit assignment under the causal dependency of hierarchy 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 models (Shao et al., 2024; Yu et al., 2025; Whitehouse et al., 2025), which can be respectively sparse or unreliable for 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.

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,

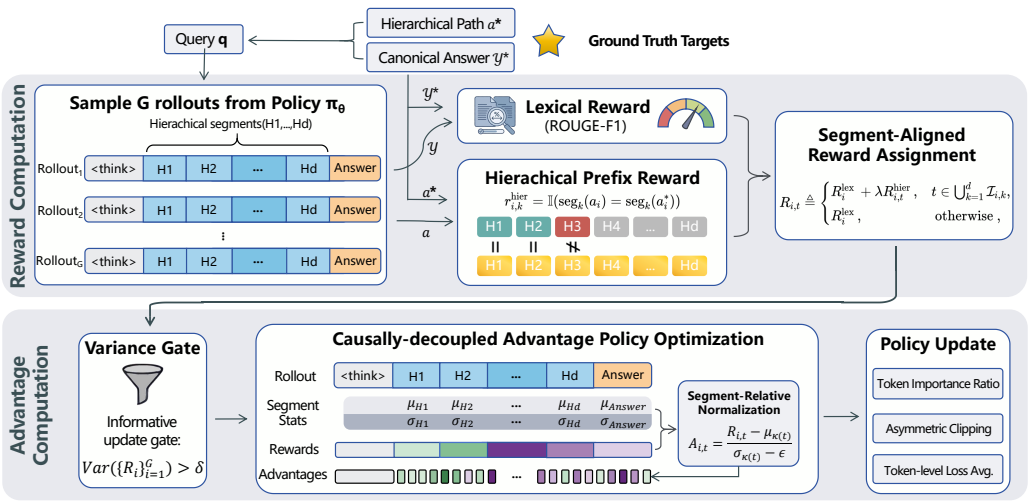

Figure 2: Overview of CAPO for hierarchical tool invocation.

lexical overlap with the reference trace becomes a reliable proxy for correctness in this setting. Although ROUGE is often criticized for open-ended generation with high paraphrastic diversity, it is well-suited here as an efficient reward for structured tool-invocation traces (Lin, 2004). 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 the aligned components and provides informative feedback, especially early in training when fully correct traces are rare.

Since tool invocations may be inter-dependent, 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:

$$R^{\mathrm{lex}}\left(\hat{\mathcal{Y}}_i, \mathcal{Y}^*\right) = \tfrac{1}{L}\sum_{j=1}^{L} \rho^{\mathrm{lex}}\left(\hat{y}_{i,j}, y_j^*\right) \tag{4}$$

where $L = \max(m, n)$. If an invocation is missing or extra at position j on either side, the 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, which makes the resulting tool path invalid under hierarchical constraints. Moreover, tool invocations in a rollout may be inter-dependent 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:

$$r_{i,k}^{\mathrm{hier}} = \mathbb{I}(\mathrm{seg}_k(a_i) = \mathrm{seg}_k(a_i^*)), \tag{5}$$

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 it also 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:

$$R_{i,t}^{\mathrm{hier}} = \sum_{k'=k}^{d} r_{i,k'}^{\mathrm{hier}}, \qquad t \in \mathcal{I}_{i,k}. \tag{6}$$

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 $R_i^{\text{lex}} = R^{\text{lex}}\left(\hat{\mathcal{Y}}_i, \mathcal{Y}^*\right)$. We add hierarchical reward 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,

$$R_{i,t} \triangleq \begin{cases} R_i^{\text{lex}} + \lambda\, R_{i,t}^{\text{hier}}, & t \in \bigcup_{k=1}^{d} \mathcal{I}_{i,k}, \\ R_i^{\text{lex}}, & \text{otherwise}, \end{cases} \tag{7}$$

where λ is the weight of 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, 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:

$$\mu_k = \operatorname*{mean}_{j\in[G]}\left(\widetilde{R}_{j,k}\right), \quad \sigma_k = \operatorname*{std}_{j\in[G]}\left(\widetilde{R}_{j,k}\right). \tag{8}$$

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 $\kappa_i(t)$:

$$\hat{A}_{i,t} = \frac{R_{i,t} - \mu_{\kappa_i(t)}}{\sigma_{\kappa_i(t)} + \epsilon} \tag{9}$$

where $\epsilon > 0$ is a small constant for numerical stability. We optimize the policy using the hierarchy-aligned advantages:

$$\begin{aligned} \mathcal{J}_{\text{CAPO}}(\theta) = \mathbb{E}_{(q,\mathcal{Y}^*)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \Bigg[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \\ \min\Big(r_{i,t}(\theta)\,\hat{A}_{i,t},\ \text{clip}\big(r_{i,t}(\theta), 1-\varepsilon_{\text{L}}, 1+\varepsilon_{\text{H}}\big)\,\hat{A}_{i,t}\Big)\Bigg] \\ \text{s.t.} \quad \text{Var}\Big(\{\bar{R}_i\}_{i=1}^{G}\Big) > \delta, \qquad \bar{R}_i = R^{\text{lex}}\Big(\hat{\mathcal{Y}}_i, \mathcal{Y}^*\Big) \end{aligned} \tag{10}$$

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:

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}, \tag{11}$$

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 Experiment

4.1 Experiment Setup

The experiments cover a diverse set of LLMs, including open-source baselines 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 results from strong proprietary model families (GPT-series, Claude-series, and Gemini-series).

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

RL Setup. All RL algorithms incorporated a format-specific reward. For our *hybrid reward* in Sec. 3.2.1, we set the lexical-overlap reward R^{lex} weight to 1.0 and the hierarchical reward weight to $\lambda = 0.5$ in Eq. 7. *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[5] (Dou et al., 2025).

All RL experiments were conducted on the verl platform[6] (Sheng et al., 2024), utilizing a computational infrastructure comprising two nodes, each equipped with 8 NVIDIA H100 GPUs.

[2] https://huggingface.co/collections/Qwen/qwen3
[3] https://huggingface.co/SUFE-AIFLM-Lab/Fin-R1
[4] https://huggingface.co/meta-llama
[5] https://huggingface.co/internlm/POLAR-7B
[6] https://github.com/volcengine/verl

Method	Qwen3-8B	Qwen3-14B	Qwen3-32B	Qwen3-30B-MoE	Fin-R1	LLaMA3.1-8B
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: Comparative of CAPO against DAPO and GRPO across multiple LLMs.

4.2 Experimental Result

Main Results. On FHTC-26K, Tab. 1 compares CAPO with a state-of-the-art *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 backbones and scales, including Qwen3 models (8B/14B/32B/30B-MoE), Fin-R1 (Liu et al., 2025), and LLaMA3.1-8B.

CAPO achieves the best performance across all evaluated models, demonstrating robust generalization across both dense and MoE architectures and multiple parameter scales. Notably, CAPO (w/o H.) already surpasses GRPO and DAPO on every backbone. This indicates that replacing sparse accuracy rewards with the lexical-overlap reward provides a denser training signal, which continuously assigns effective credit during optimization and helps the model reliably track incremental improvements in tool-use reasoning. Adding the hierarchical reward further improves performance, showing that hierarchy-consistent feedback and causally-decoupled, segment-aligned advantage together enable more accurate credit assignment for high-hierarchy routing decisions, which translates into consistent gains over the lexical-only variant. Overall, RL-based methods (GRPO, DAPO, and CAPO) consistently outperform the SFT-only baseline. This gap suggests that supervised imitation alone is insufficient for hierarchical tool invocation. While SFT effectively fits demonstration distributions, it provides limited learning signals for correcting fine-grained reasoning errors and for improving long-horizon, hierarchy-constrained decisions. In contrast, RL directly optimizes task-aligned rewards and can iteratively refine both tool routing and call specification, leading to substantial gains in open-ended tool-use accuracy.

Across the Qwen series, performance improves with model scale, indicating that larger backbones can better capture functional distinctions. Qwen3-30B-MoE further outperforms the similarly-sized Qwen3-32B dense model, which is consistent with the intuition that MoE architectures can allocate capacity to specialized experts for heterogeneous tool categories in large hierarchical libraries. In addition, Fin-R1, a Qwen-7B variant instruction-tuned with curated financial data, exceeds the standard Qwen3-8B baseline, highlighting the value of high-quality domain data for strengthening tool invocation in specialized settings.

Method	Score
RAG + GPT-4o (Achiam et al., 2023)	52.66
RAG + DeepSeek-V3.2 (Liu et al., 2024)	54.56
RAG + Claude4.5-sonnet (Anthropic, 2025)	55.74
RAG + Qwen3-max (Yang et al., 2025)	55.89
RAG + Gemini3-pro (Comanici et al., 2025)	57.70
RAG + GPT-5.2 (OpenAI, 2025)	58.12
CAPO w Qwen3-8B (Yang et al., 2025)	**69.44**

Table 2: Comparative of the CAPO against RAG.

Effectiveness of CAPO compared with RAG. Tab. 2 compares CAPO with a standard RAG pipeline, where an embedding retriever selects candidate tools from tool sets and a LLM performs final tool selection and invocation. We evaluate RAG with representative closed-source and open-source model families, including GPT, DeepSeek, Claude, Gemini, and Qwen-Max. Across all backbones, RAG remains far below CAPO, even with frontier-scale LLMs, suggesting that retrieval alone is insufficient for reliable open-ended tool invocation in large hierarchical libraries. This performance gap indicates three key limitations: (a) existing embedding models struggle to capture fine-grained semantic distinctions among tools; (b) LLMs' ability to identify correct answers deteriorates in excessively long contexts, despite strong needle-in-haystack reasoning; and (c) LLMs trained on general web corpora lack domain-specific adaptation, resulting in suboptimal performance.

Effectiveness of CAPO on ToolBench. We further evaluate CAPO on the open-source ToolBench

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: ToolBench results of CAPO and ToolLLM(Macro Precision, %). G1,G2,G3 denote task groups. Δ denotes absolute improvement.

Reward Function	**Score**
Accuracy Reward (Shao et al., 2024)	61.52
PoLAR (Dou et al., 2025)	66.87
Lexical Reward	68.27
Hybrid Reward (Lexical. & Hierarchical.)	**69.44**

Table 4: Reward ablations for RL on Qwen3-8B.

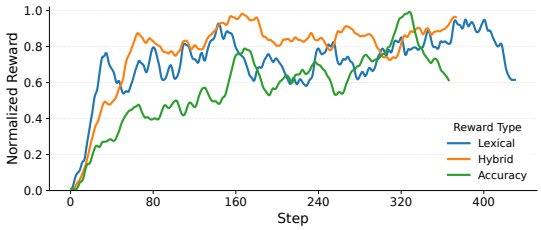

Figure 3: Rewards trajectories over training steps.

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

Table 5: Ablation of advantage function on Qwen3-8B.

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 D. As shown in Tab. 3, CAPO consistently outperforms the ToolBench SFT baseline (ToolLLM) across all three task groups (G1–G3), yielding an average absolute improvement of 5.34 macro-precision points. 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-Scale Ablation. Tab. 4 shows that reward design strongly affects RL performance. The accuracy reward performs worst, consistent with its sparsity. PoLAR improves over accuracy reward but remains below lexical reward, suggesting limited reliability for fine-grained tool distinctions. In contrast, the lexical reward provides a denser signal and yields substantial gains. Adding the hierarchical reward further improves results, indicating that hierarchy-consistent feedback complements lexical overlap and leads to more robust optimization.

Effectiveness of Reward Design on Training. Fig. 3 plots normalized reward trajectories over training steps for different reward designs. Because rewards have different scales, we normalize them for visualization. The accuracy reward is sparse, producing few effective learning signals in early training steps, which slows optimization. The lexical reward accelerates early convergence by providing dense partial credit, but exhibits higher variance across steps and becomes less discriminative among near-correct traces, which weakens the late-stage optimization signal. The hybrid reward maintains both a strong early learning signal and stable late-stage separation by combining lexical overlap with hierarchy-consistent supervision, resulting in the fastest and most stable convergence.

Advantage-Scale Study. Tab. 5 compares advantage estimators under the same hybrid reward on Qwen3-8B. Causally-Decoupled (C-D) advantage consistently outperforms group-wise variants (e.g., 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 hierarchy routing decisions in reasoning trace, leading to improved overall performance.

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. Experiments demonstrate that CAPO substantially enhances LLM performance on real-world tool-driven tasks.

Limitations

This work addresses the challenging problem of open-ended tool invocation at scale, but several limitations remain. First, FHTC-26K is partially synthesized; despite adversarial filtering, 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.

Potential Risks

Our approach shares risks common to tool-augmented financial 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 as decision support, not for automated decisions or direct real-world trading.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2025. Claude 4.5 sonnet. https://www.anthropic.com/news/claude-sonnet-4-5. Accessed: 2025-12-23.

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, Suzhou, China. Association for Computational Linguistics.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Shihan Dou, Shichun Liu, Yuming Yang, Yicheng Zou, Yunhua Zhou, Shuhao Xing, Chenhao Huang, Qiming Ge, Demin Song, Haijun Lv, and 1 others. 2025. Pre-trained policy discriminators are general reward models. *arXiv preprint arXiv:2507.05197*.

Wouter Kool, Herke van Hoof, and Max Welling. 2019. Buy 4 reinforce samples, get a baseline for free!

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. In *International conference on machine learning*, pages 1078–1088. Pmlr.

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. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Zhaowei Liu, Xin Guo, Fangqi Lou, Lingfeng Zeng, Jinyi Niu, Zixuan Wang, Jiajie Xu, Weige Cai, Ziwei Yang, Xueqian Zhao, and 1 others. 2025. Fin-r1: A large language model for financial reasoning through reinforcement learning. *arXiv preprint arXiv:2503.16252*.

OpenAI. 2025. Gpt-5 system card. OpenAI blog / publication page. Accessed: 2025-12-23.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, 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 *Forty-second International Conference on Machine Learning*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*, volume 36, pages 72059–72076.

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 (ICLR)*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

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. *arXiv preprint arXiv:2402.03300*.

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

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.

Nexusflow.ai team. 2023. Nexusraven: Surpassing the state-of-the-art in open-source function calling llms.

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. *arXiv 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 fornbsp;agent tuning andnbsp;detailed benchmark. In *Natural Language Processing and Chinese Computing: 13th National CCF Conference, NLPCC 2024, Hangzhou, China, November 1–3, 2024, Proceedings, Part II*, page 372–384, Berlin, Heidelberg. Springer-Verlag.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv 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, Suzhou, China. Association for Computational Linguistics.

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, Suzhou, China. Association for Computational Linguistics.

Kangning Zhang, Wenxiang Jiao, Kounianhua Du, Yuan Lu, Weiwen Liu, Weinan Zhang, Lei Zhang, and Yong Yu. 2025. Looptool: Closing the data-training loop for robust llm tool calls. *arXiv 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, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

A Computational Efficiency

We assess the computational overhead of our proposed method, CAPO, compared to the baseline DAPO. Tab. 6 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 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 context of the overall training loop. Crucially, it is important to note that 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 demonstrated in our ablation studies (see Section 4.2), this statistical approach not only maintains high computational efficiency but also yields superior performance compared to RM-based baselines.

Table 6: Comparison of computational time per training step.

Component	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

B Data-Scale Study

As evidenced by Table 7, performance improvements for both SFT and CAPO remain negligible with increased data scale from 26K to 96K

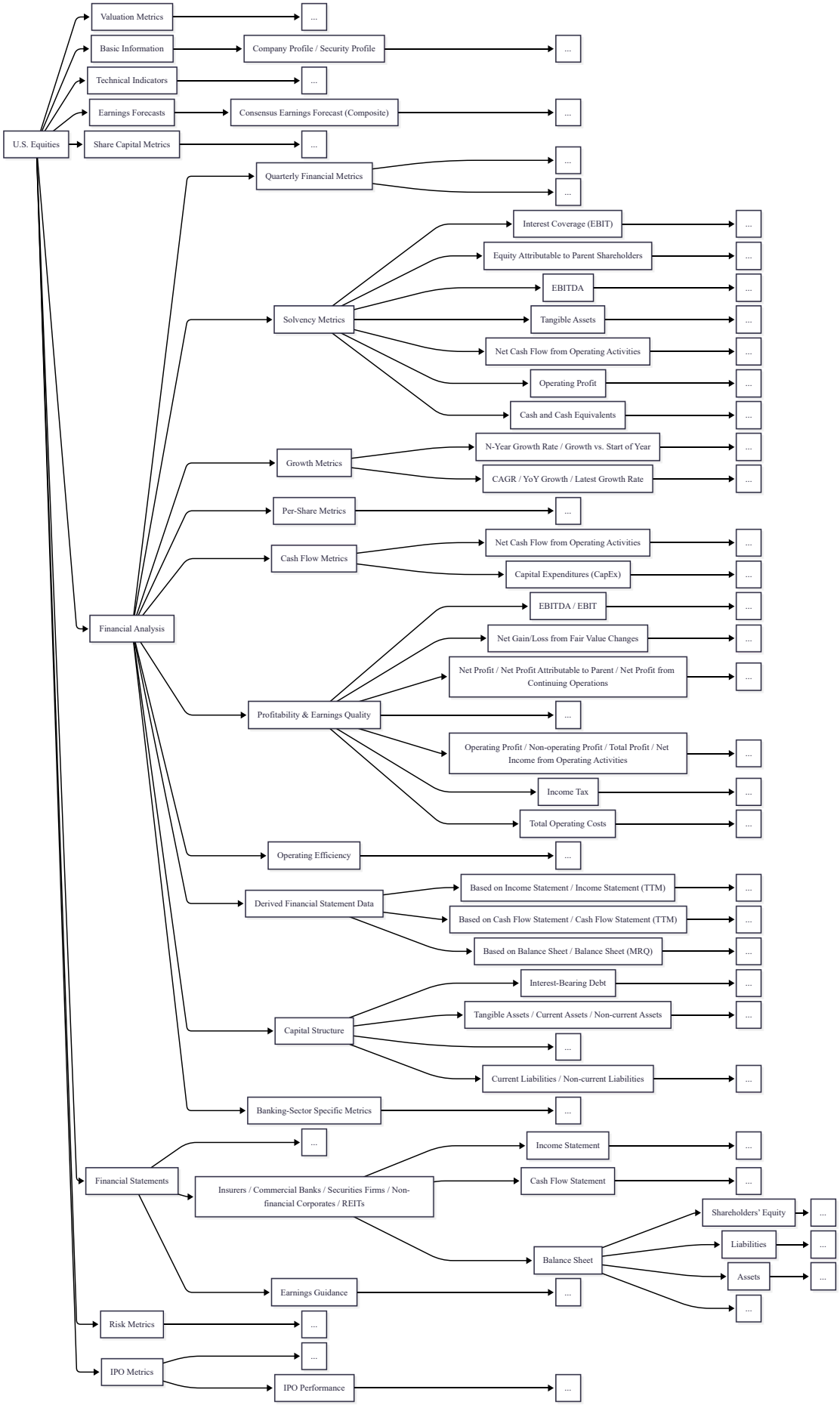

Figure 4: Tool library hierarchy in FHTC-26K (U.S. equities branch), shown as a pruned and merged tree for readability.

Method	Datascale	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 7: Ablation Study on the Impact of Dataset Scale.

instances. This limitation manifests more prominently in SFT models, where larger datasets correlate with 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 CAPO models trained on 26K versus 96K instances (e.g., 69.87 vs. 69.44 for Qwen-7B) further 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 for domain-specific tool usage: Data quality substantially outweighs scale, with high-diversity examples providing significantly greater value than volumetrically larger datasets containing superficial variations

C 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 further branches into major functional groups such as *Valuation Metrics*, *Basic Information*, and *Financial Analysis*, etc. 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 as well as sector-specific structures, which together create a large branching factor and 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, the figure 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.

D ToolBench Evaluation Protocol

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

$$\mathrm{Prec}(\hat{Y}, Y^*) = \frac{|\hat{Y} \cap Y^*|}{|\hat{Y}|}. \tag{12}$$

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 for open-ended multi-tool prediction. ToolBench baselines (e.g., ToolLLM) typically retrieve tools via similarity ranking, while CAPO generates an ordered multi-tool invocation trace, which may have variable length under 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.

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 Dataset Construction Details

This section describes 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.

Concretely, Fig. 5 presents the expert-guided iterative refinement template for curating tool/indicator semantics, Fig. 6 shows the expertise-aware seeded template for diverse query synthesis in FHTC-26K, Fig. 7 provides the hierarchy-aware chain-of-thought template for producing decompositional reasoning traces, and Fig. 8 describes the Turing-style judge template used to filter out synthetic artifacts. Due to page constraints, we only include the essential prompt structures and key constraints in these figures, omitting auxiliary instructions and boilerplate that do not affect the core logic.

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 Prompting for Decompositional Reasoning

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