# Guided by Experts, Empowered by Self-Exploration: A Unified Post-Training Methodology for Agentic Tool Use

**Anonymous ACL submission**

## Abstract

Post-training is crucial for enhancing tool-use capabilities in large language models (LLMs). However, the dominant paradigms face key limitations: Supervised Fine-Tuning (SFT) exhibits poor generalization, Reinforcement Learning (RL) suffers from sparse or coarse-grained rewards, and SFT-then-RL pipeline fails to balance supervised imitation with self-exploration. We propose **ToolUPT**, a **Tool**-Use **U**nified **P**ost-**T**raining methodology that synergistically integrates SFT and RL through dynamic token-wise weighting. To address credit assignment problem in sequence-level RL formulations, we introduce **Fine-Grained Advantage Estimation** with token-level rewards integrating outcome and process signals. To stabilize training and improve generalization, we propose **Selective Preference Optimization**, which incorporates expert supervision on curated tokens while teaching the model to recognize and avoid rejected tool use through negative advantages. Extensive experiments on BFCL-v3 across four base models show that ToolUPT significantly outperforms pure RL by **8.43%**, SFT-then-RL by **3.58%**, and previous methods combining expert demonstrations with self-exploration by **4.83%**, achieving state-of-the-art performance in tool-use post-training.

## 1 Introduction

The evolution of Large Language Models (LLMs) has extended beyond text generation toward the development of autonomous agents capable of interacting with the external world through tool use and planning (Huang et al., 2024; Qin et al., 2025). By translating natural language instructions into executable API calls, LLMs can perform complex real-world tasks, from data retrieval to system control. This capability has driven substantial research into post-training methods—including Supervised Fine-Tuning (SFT), Reinforcement Learning (RL), and SFT-then-RL pipeline—for enhancing tool-use performance (Qin et al., 2024; Qu et al., 2025).

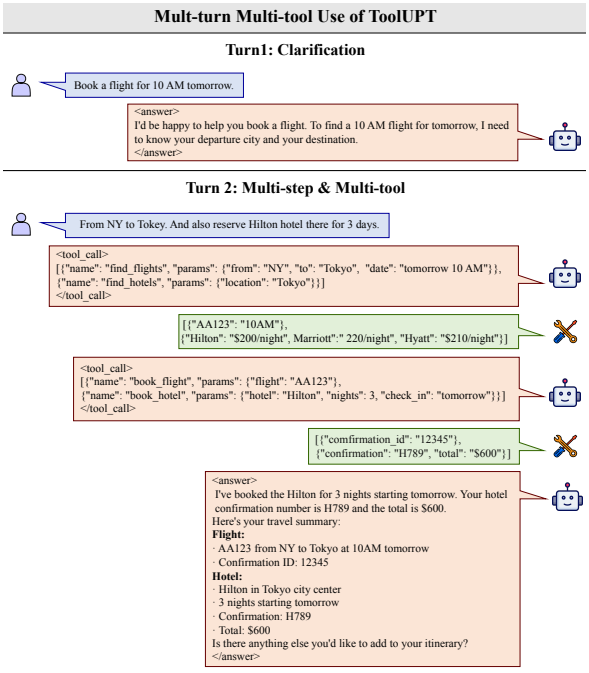


Figure 1: Overview of the agentic tool use task.

Despite this progress, current approaches face two critical limitations in tool-use scenarios. **First**, the dominant post-training paradigms face several challenges: (1) SFT tends to lock the trained models into imitative and rigid behavior modes (Chu et al., 2025); (2) existing Reinforcement Learning with Verifiable Rewards (RLVR) approaches suffer from coarse-grained credit assignment: assigning a single scalar reward to each complete response causes individual token errors (e.g., an incorrect parameter value) to incorrectly penalize all tokens within the same sequence, including correctly generated tool names and other parameters. This leads to noisy gradient updates that prevent the model from learning fine-grained distinctions critical for tool use (Tan and Pan, 2025; Zhang et al., 2025d); (3) determining the optimal SFT training duration is non-trivial in two-stage pipeline, as excessive supervised learning may constrain exploration in

the RL phase while insufficient training provides a poor initialization; and (4) the staged approach cannot dynamically adjust the balance between following expert guidance and exploring novel solutions (Zhang et al., 2025c; Ma et al., 2025; Fu et al., 2025). **Second**, while recent work has primarily focused on reasoning models that generate long Chain-of-Thought for detailed planning, their high inference latency renders them impractical for real-time applications (Sui et al., 2025; Zhang et al., 2025a). In contrast, instruction-tuned models that directly generate tool calls or answers offer significantly lower latency.

To address these limitations, we propose **Tool** Use **U**nified **P**ost-**T**raining (**ToolUPT**), a unified methodology for instruction-tuned models that synergistically integrates SFT and RL through dynamic token-wise weighting in a single training stage. Within this unified framework, we introduce two key technical components: First, we propose **Fine-Grained Advantage Estimation** with token-level rewards that integrate outcome and process signals. This resolves the credit assignment problem in sequence-level RL formulations by enabling precise gradient updates and preventing correct tokens from being penalized by sequence-level errors. Second, we introduce **Selective Preference Optimization** that incorporates expert supervision on curated tokens while leveraging negative advantages to teach rejection patterns, thereby stabilizing training and improving generalization. By unifying SFT and RL through dynamic weighting, ToolUPT effectively balances expert guidance with self-exploration, achieving superior tool-use performance without the brittleness of staged training approaches.

In summary, our contribution are as follows:

- We propose ToolUPT, a novel post-training methodology that synergistically unifies offline supervised imitation and online self-exploration through dynamic token-wise weighting for agentic tool use.
- We introduce Fine-Grained Advantage Estimation and Selective Preference Optimization to enhance model robustness.
- We conduct extensive experiments on BFCL-v3 across four base models, confirming the superiority of ToolUPT over existing methods.

## 2 Related Work

### 2.1 Agentic Tool Use

Large language models have exhibited remarkable capabilities as autonomous agents for complex tasks through tool use and planning. While innovative prompting techniques—such as Chain-of-Thought (CoT)(Wei et al., 2022) and ReAct(Yao et al., 2023)—have yielded notable improvements, their efficacy remains fundamentally constrained by the models' intrinsic capabilities. To address this limitation, tool-augmented post-training has garnered increasing attention. Research in this domain can be taxonomized into two categories: (i) algorithm-centric methods, which focus on developing advanced training algorithms for sophisticated multi-turn multi-tool orchestration and parallel invocation (e.g., Toolformer(Schick et al., 2023), StepTool(Yu et al., 2025b), Hammer(Lin et al., 2025), ReTool(Feng et al., 2025), Search-R1(Jin et al., 2025), ToolRL(Qian et al., 2025), CodeTool(Lu et al., 2025), ToolZero(Zeng et al., 2025)), and (ii) data-centric methods, which prioritize the synthesis of high-quality, diverse training trajectories spanning real-world scenarios and simulated environments (e.g., ToolAlpaca(Tang et al., 2023), xLAM(Zhang et al., 2025b), APIGen-MT(Prabhakar et al., 2025), ToolACE(Liu et al., 2025b), MAGNET(Yin et al., 2025), FunReason-MT(Xu et al., 2025)).

### 2.2 Reinforcement Learning

Early LLM alignment relies on Proximal Policy Optimization (PPO) (Schulman et al., 2017), an Actor-Critic architecture that optimizes against a learned reward model. However, PPO incurs significant memory overhead and training complexity due to its value network and critic gradients. This limitation motivated off-policy methods like Direct Preference Optimization (DPO) (Rafailov et al., 2023), which reparameterizes the reward function directly into the policy loss, eliminating the explicit reward model. Subsequent variants such as IPO (Azar et al., 2023) and KTO (Ethayarajh et al., 2024) further address overfitting and relax paired data requirements, respectively. More recently, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) introduced group-based reward normalization across multiple sampled outputs per prompt, with refinements including DAPO (Yu et al., 2025a) for improved reproducibility and GSPO (Zheng et al., 2025) for stabilizing Mixture-of-Experts

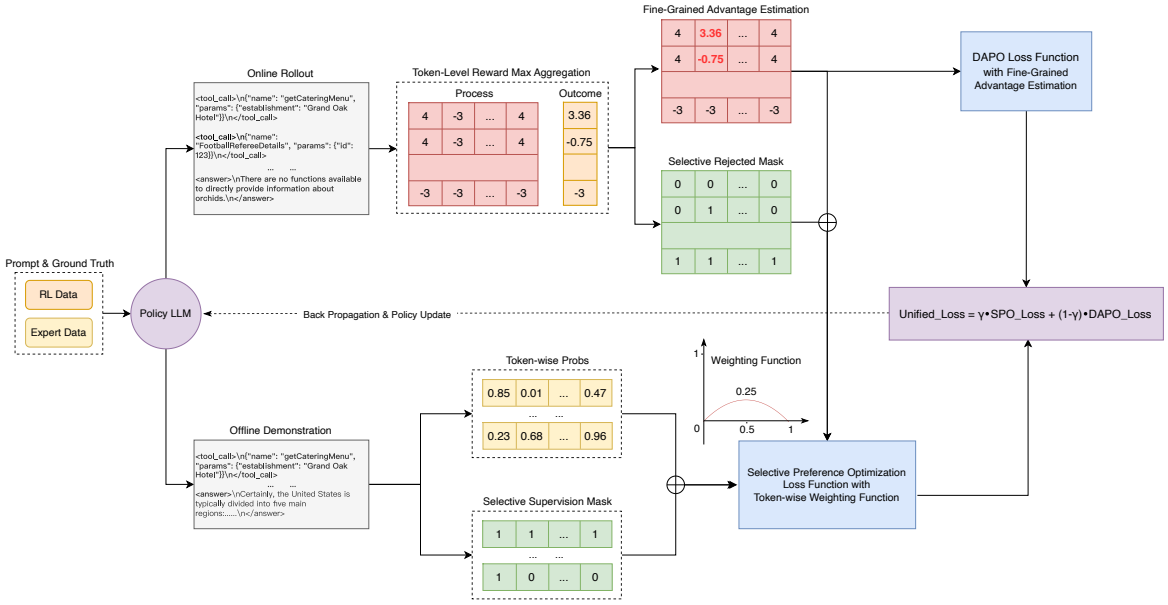


Figure 2: The architecture of our ToolUPT methodology, which synergistically unifies Fine-Grained DAPO and Selective Preference Optimization via adaptive dynamic weighting.

(MoE) off-policy training.

### 2.3 Combining SFT with RL

Recently, some post-training works seek to concurrently perform both SFT and RL, to enable the model to explore solutions effectively while preserving the structural guidance of expert supervision. LUFFY (Yan et al., 2025), SRFT (Fu et al., 2025), and CHORD (Zhang et al., 2025c) introduce a single-stage dynamically weighted objective that simultaneously leverages expert demonstrations and self-exploration rollouts. ReLIFT (Ma et al., 2025) induces capabilities that exceed the limitations of the base model by interleaving RL updates with targeted SFT on challenging examples. Similarly, UFT (Liu et al., 2025a) applies SFT loss to "hint" tokens or partial solutions while applying RL loss to the subsequent completion.

## 3 Methodology

In this section, we propose the **Tool** Use **U**nified **P**ost-**T**raining (**ToolUPT**) methodology that unifies SFT and RL via a dynamically weighted loss, illustrated in Figure 2. Our framework is built upon three key components: (1) Fine-Grained Advantage Estimation that addresses the limitations of sequence-level RL formulations, (2) Selective Preference Optimization that leverages expert supervision while enhancing rejected token identification, (3) a unified dynamic weighted loss function that combines the advantages of the above two components, balancing self-exploration with expert-guided preference optimization.

### 3.1 Preliminaries

Addressing real-world user queries with LLM's tool-use capabilities can be conceptualized as a step-wise planning process. Formally, let $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ represent the set of available tools. For each tool $t_i$, there exists a calling protocol $d_i \in \mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ that provides the tool's functional description and parameter specifications. For a given user query $Q$, the LLM agent engages in a multi-step decision-making process to solve $Q$. At each step $k$, the agent decides between two types of actions:

- **Tool Invocation** $\mathcal{A}_{\text{tool}}$: The agent selects one or more tools $\mathcal{T}_k \subseteq \mathcal{T}$ and instantiates their parameters according to the respective protocols. The tool system executes these calls and returns observations, which are appended to the interaction history.
- **Answer Generation** $\mathcal{A}_{\text{answer}}$: The agent produces a direct response or clarification request, terminating the planning process.

To enable the model to autonomously determine when to invoke tools or answer directly, we introduce two special token pairs: `<tool_call>` $\cdots$ `</tool_call>` for tool invocations and `<answer>` $\cdots$ `</answer>` for direct answers or clarifications (prompt shown in Appendix C).

The post-training of LLMs aims to optimize the model policy $\pi_\theta(a_t|s_t)$ to produce desirable responses on specific tasks. SFT and RL constitute two prominent paradigms in this domain. Given a dataset of $N$ expert demonstrations, $D = \{(x_i, y_i^*)\}_{i=1}^N$, where $x_i$ is a prompt and $y_i^* = (y_{i,1}^*, y_{i,2}^*, ..., y_{i,|y_i^*|}^*)$ is the corresponding expert response of $|y_i^*|$ tokens. The SFT objective minimizes the negative log-likelihood of expert responses, averaged over a mini-batch of size $B$:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\frac{1}{\sum_{i=1}^{B}|y_i^*|}\sum_{i=1}^{B}\sum_{t=1}^{|y_i^*|} log\ \pi_\theta(y_{i,t}^*|x_i, y_{i,<t}^*) \tag{1}$$

GRPO serves as a prominent policy gradient algorithm for RL. For each prompt $x_i$, GRPO samples a group of $K$ candidate responses $\{\tau_{i,1}, \tau_{i,2}, ..., \tau_{i,K}\}$ from policy model $\pi_\theta$. These responses are evaluated by the reward function to obtain its reward $R(\tau_{i,k})$. The policy model $\pi_\theta$ is optimized using the following PPO-style objective function:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{\hat{B}K}\sum_{i=1}^{\hat{B}}\sum_{k=1}^{K}\frac{1}{|\tau_{i,k}|}\sum_{t=1}^{|\tau_{i,k}|}\min(r_{i,k,t}(\theta)A_{i,k}, \text{clip}(r_{i,k,t}(\theta),\, 1-\epsilon,\, 1+\epsilon)A_{i,k}) \tag{2}$$

where $\hat{B}$ is the number of prompts in the mini-batch. The group-wise advantage $A_{i,k}$ for each token is computed by $\frac{R(\tau_{i,k})-\mu_i}{\sigma_i+\epsilon}$, where $\mu_i$ and $\sigma_i$ are the mean and standard deviation of rewards $\{R(\tau_{i,k})\}_{k=1}^K$ within group $i$, and $\epsilon$ is a small constant for numerical stability. The $r_{i,k,t}(\theta) = \frac{\pi_\theta(\tau_{i,k,t}|x,\tau_{i,k,<t})}{\pi_{sample}(\tau_{i,k,t}|x,\tau_{i,k,<t})}$ is the token-wise importance sampling (IS) ratio and the clipping mechanism constrains the policy update to a trusted region. Consistent with recent advancements, our formulation omits the KL divergence term to avoid restricting policy performance.

## 3.2 Fine-Grained Advantage Estimation

As illustrated in Figure 3, a key limitation of RLVR lies in its sequence-level reward formulation, which assigns a single scalar reward to each complete response. This coarse granularity causes individual token errors to incorrectly penalize correct tokens within the same response, resulting in suboptimal policy gradient updates. To address this, we introduce a Fine-Grained Advantage Estimation (**FGAE**) method with token-level rewards that integrate both outcome and process supervision.

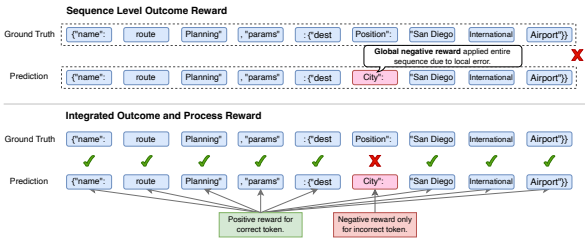


Figure 3: Token-level integrated (bottom) vs. sequence-level outcome reward (top).

### 3.2.1 Integrated Outcome and Process Reward Shaping

As our research focuses on instruction-tuned models rather than reasoning models, we leverage ground truth to compute precise process reward supervision. Our reward formulation comprises two components: a sequence-level outcome reward and a token-wise process reward refinement. The outcome reward establishes the reward lower bound for each token, and the process refinement mechanism identifies completely correct tokens and assigns them the reward upper bound. Formally, let $R(\tau_k) = \{R(\tau_{k,1}), R(\tau_{k,2}), ..., R(\tau_{k,|\tau_k|})\}$ represent final token-wise rewards of $k$-th response within group $i$, calculated as follows:

**Sequence Outcome Reward** In line with prior work (DeepSeek-AI, 2025), we adopt a reward formulation combining format and correctness components, denoted as $R_{\text{format}}(\tau_k)$ and $R_{\text{correct}}(\tau_k)$, respectively.

**Format Reward** The format reward verifies the presence of required special tokens at the start and end of the model output as specified by ground truth:

$$R_{\text{format}}(\tau_k) = \begin{cases} 1, & \text{if format requirements are met,} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

**Correctness Reward** For prompts with ground truth in the `<answer>` category, verification is straightforward: a response is correct if no `<tool_call>` or `</tool_call>` appears between `<answer>` and `</answer>`. By not constraining answer content, we allow the model to explore diverse responses. For prompts with ground truth in the `<tool_call>` category, the correctness reward evaluates predicted tool calls $P = \{P_1, P_2, \ldots, P_m\}$ against ground-truth tool calls $G = \{G_1, G_2, \ldots, G_n\}$, where $P$ is parsed from the model's tool call trajectory. It comprises three components:

- Tool Syntax Correctness:
  Checks whether the tool use instruction conforms to the defined syntax and contains the required keys "name" and "params".

$$r_{\text{syntax}}(\tau_k) = \begin{cases} 1, & \text{if syntax requirements are met,} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

- Tool Name Correctness:

$$r_{\text{name}}(\tau_k) = \frac{|N_P \cap N_G|}{|N_P \cup N_G|} \in [0, 1] \tag{5}$$

  where $N_P$ and $N_G$ are the sets of tool names extracted from the predicted and ground-truth tool calls, respectively.

- Tool Parameter Correctness:
  Let $U \subseteq \{1, 2, \ldots, n\}$ be the set of indices of already matched predicted tools, we define a predicted tool $P_j$ as a candidate for matching a ground truth tool $G_i$ only if their names match $P_j$ and has not been used yet.

$$\text{Cand}(G_i, P_j) = \begin{cases} \text{true} & \text{if } (\text{name}_{G_i} = \text{name}_{P_j}) \\ & \wedge (j \notin U), \\ \text{false} & \text{otherwise.} \end{cases} \tag{6}$$

  The parameter key matching score measures the overlap of parameter keys, normalized by the the union of parameter keys.

$$s_{\text{key}}(G_i, P_j) = \frac{|\text{key}_{G_i} \cap \text{key}_{P_j}|}{|\text{key}_{G_i} \cup \text{key}_{P_j}|} \tag{7}$$

  The parameter value matching score measures the proportion of common keys that have identical values.

$$s_{\text{value}}(G_i, P_j) = \frac{\sum_{k \in \text{key}_{G_i} \cap \text{key}_{P_j}} \mathbf{1}(\text{value}_{G_i,k} = \text{value}_{P_j,k})}{\max(|\text{value}_{G_i}|, |\text{value}_{P_j}|)} \tag{8}$$

  The best-match score for a ground truth tool $G_i \in G$ among all candidate predicted tools is determined as follows:

$$s_{\text{Best}}(G_i) = \max_{\{j | \text{Cand}(G_i, P_j)\}} \frac{s_{\text{key}}(G_i, P_j) + s_{\text{value}}(G_i, P_j)}{2} \tag{9}$$

  Then the tool parameter correctness reward is obtained by summing the best match scores for all ground truth tools and normalizing by the size of the larger set (either $G$ or $P$).

$$r_{\text{param}}(\tau_k) = \sum_{i=1}^{m} \frac{s_{\text{Best}}(G_i)}{\max(m, n)} \in [0, 1] \tag{10}$$

The final correctness reward combines the above three components through a weighted sum, with the result linearly scaled from $[0, 1]$ to$[-3, 3]$:

$$R_{\text{correct}}(\tau_k) = 6 \cdot (\lambda_1 r_{\text{syntax}}(\tau_k) + \lambda_2 r_{\text{name}}(\tau_k) + \lambda_3 r_{\text{param}}(\tau_k)) - 3 \tag{11}$$

where $\lambda_1, \lambda_2, \lambda_3 \in [0, 1]$ are weighting coefficients satisfying $\lambda_1 + \lambda_2 + \lambda_3 = 1$. This multi-dimensional verification strategy effectively captures the subtle distinctions between different tool invocation instructions. The overall outcome reward value $R_{\text{outcome}}(\tau_k)$ is finally derived as the sum of format reward and correctness reward:

$$R_{\text{outcome}}(\tau_k) = R_{\text{format}}(\tau_k) + R_{\text{correct}}(\tau_k) \in [-3, 4] \tag{12}$$

**Token Process Reward** Given that we restrict instruction-tuned models from generating reasoning traces, the freedom in their tool invocation outputs are reduced, thus requiring more precise reward estimation for effective training. For `<tool_call>...</tool_call>` responseswhile sequence-level outcome reward computation achieves considerable granularity, it systematically underestimates rewards for perfectly predicted tokens. To address this issue, we propose a process reward refinement mechanism that compares the model predictions with the ground truth token-by-token, assigning the reward upper bound directly to tokens with exact matches.

$$R_{\text{process}}(\tau_{k,t}) = \begin{cases} 4 & \text{if the } t\text{-th token is correct,} \\ -3 & \text{otherwise.} \end{cases} \tag{13}$$

$$R(\tau_k) = \{\max(R_{\text{process}}(\tau_{k,t}), R_{\text{outcome}}(\tau_k))\}_{t=1}^{|\tau_k|} \tag{14}$$

### 3.2.2 Token-Level Advantage Normalization

Building on the reward formulation above, we propose a novel Fine-Grained Advantage Estimation (FGAE) method that enables token-level normalization within each prompt group. Unlike conventional approaches that normalize at the sequence level, our method computes statistics across all tokens within a group, providing more precise and stable gradient signals. Specifically, for each group $i$, we calculate the token-level mean and standard deviation of the rewards across all $K$ sampled responses:

$$\tilde{\mu}_i = \frac{1}{\sum_{k=1}^{K} |\tau_{i,k}|} \sum_{k=1}^{K} \sum_{t=1}^{|\tau_{i,k}|} R(\tau_{i,k,t}) \tag{15}$$

$$\tilde{\sigma}_i = \sqrt{\frac{1}{\sum_{k=1}^{K}|\tau_{i,k}|}\sum_{k=1}^{K}\sum_{t=1}^{|\tau_{i,k}|}(R(\tau_{i,k,t}) - \tilde{\mu}_i)^2} \quad (16)$$

For each token within a group, we define the normalized advantage:

$$\tilde{A}_{i,k,t} = \frac{R(\tau_{i,k,t}) - \tilde{\mu}_i}{\tilde{\sigma}_i + \epsilon} \quad (17)$$

Equipped with our token-level normalized advantages, we reformulate the GRPO objective by incorporating DAPO's token-mean aggregation mechanism. The modified loss computes the clipped surrogate objective at each token position using our FGAE, then averages across all tokens in the batch:

$$\begin{aligned}\mathcal{L}_{\text{FG-DAPO}}(\theta) = -\frac{1}{\sum_{i=1}^{\hat{B}}\sum_{k=1}^{K}|\tau_{i,k}|}\sum_{i=1}^{\hat{B}}\sum_{k=1}^{K}\sum_{t=1}^{|\tau_{i,k}|} \\ \min(r_{i,k,t}(\theta)\tilde{A}_{i,k,t}, \text{clip}(r_{i,k,t}(\theta), \\ 1-\epsilon, 1+\epsilon)\tilde{A}_{i,k,t})\end{aligned} \quad (18)$$

This fully token-level loss function enables the RL training process to effectively leverage process supervision for enhancing LLM capabilities in tool use, while ensuring balanced learning across all generation steps.

### 3.3 Selective Preference Optimization

Although SFT enables LLMs to effectively imitate response patterns from expert demonstrations, it is limited by the absence of negative examples, resulting in poor out-of-distribution generalization. Moreover, in tool use scenarios, when no tool call is needed and the model should respond directly or ask for clarification, the distribution of expert responses can disrupt LLMs' learned patterns. To address these challenges, we propose the Selective Preference Optimization (**SPO**) approach.

**Selective Chosen Supervision** For data requiring direct model responses, we apply masking to the answer content and the end </answer> tag, guiding the model to focus the learning on two aspects: (1) determining whether a direct answer is needed, and (2) generating the first few special tokens appropriately. The substantive answer content can be freely generated by the model based on its own capabilities. For data requiring tool calls, we apply no additional processing. We define an indicator function for token-level loss computation as

$$\mathbf{1}_{\text{preserve}}(y^*_{i,t}) = \begin{cases} 0, & \text{if } (y^*_i \in \mathcal{A}_{\text{answer}}) \\ & \wedge (t > |\textbf{tokenize}(\texttt{<answer>})|), \\ 1, & \text{otherwise.} \end{cases} \quad (19)$$

where $|\textbf{tokenize}(\texttt{<answer>})|$ denotes the number of tokens obtained from tokenizing the <answer> tag.

**Selective Rejected Preference** To mitigate model overfitting to expert data, we draw inspiration from KTO (Ethayarajh et al., 2024) and dynamically treat tokens with negative advantage values in each online rollout as rejected preference data, thereby enhancing the model's preference alignment capability in tool-use scenarios.

Prior work on KTO has established that "at sufficient scale, KTO does not need SFT" (Ethayarajh et al., 2024). Leveraging this insight, we integrate selective chosen supervision and rejected preference into a unified loss function defined as:

$$\begin{aligned}&\mathcal{L}_{\text{SPO}}(\theta) \\ &= -\frac{1}{\sum_{i=1}^{B}|y^*_i|}\sum_{i=1}^{B}\sum_{t=1}^{|y^*_i|}\mathbf{1}_{\text{preserve}}(y^*_{i,t}) \\ &\quad [1 - \text{sigm}(\beta\, log\pi^*_{i,t}(\theta))] \\ &\quad - \frac{1}{\sum_{i=1}^{\hat{B}}\sum_{k=1}^{K}|\tau_{i,k}|}\sum_{i=1}^{\hat{B}}\sum_{k=1}^{K}\sum_{t=1}^{|\tau_{i,k}|} \\ &\quad \mathbf{1}(\tilde{A}_{i,k,t} < 0)\,[1 - \text{sigm}(-\beta\, log\pi_{i,k,t}(\theta))]\end{aligned} \quad (20)$$

where $\pi^*_{i,t}(\theta) = \pi_\theta(y^*_{i,t}|x_i, y^*_{i,<t})$, $\pi_{i,k,t}(\theta) = \pi_\theta(\tau_{i,k,t}|x_i, \tau_{i,k,<t})$, $\text{sigm}(z) = \frac{1}{1+e^{-z}}$, and $\beta$ acts as a regularization hyperparameter.

### 3.4 Single-Stage Unified Post-Training

Previous approaches (Zhang et al., 2025c) have demonstrated the challenge of selecting the optimal extent of SFT and how offline expert data can disrupt the established patterns of LLMs in SFT-then-RL trainning. Motivated by this observation, we propose to reframe SPO objective as a dynamically weighted auxiliary loss within the on-policy RL process.

**Stabilize Preference Optimization with $\phi(\cdot)$** We adopt a token-wise weighting function to down-weight the learning signal for tokens at both ends of the probability spectrum, thereby preventing entropy collapse from highly probable tokens and avoiding disruption from extremely improbable tokens. Specifically, we define the weighting function $\phi(\cdot)$ based on the policy's probability $p_t$ for a given token as follows:

$$\phi(p_t) = p_t(1 - p_t) \quad (21)$$

which forms a parabolic curve that peaks at $p_t = 0.5$ and decays to zero as $p_t$ approaches 0 or 1. The SPO objective function can be represented as:

$$
\begin{aligned}
&\tilde{\mathcal{L}}_{\text{SPO}}(\theta) \\
&= -\frac{1}{\sum_{i=1}^{B}|y_i^*|}\sum_{i=1}^{B}\sum_{t=1}^{|y_i^*|}\phi(\pi_{i,t}^*(\theta)) \\
&\quad \mathbf{1}_{\text{preserve}}(y_{i,t}^*)\,[1-\text{sigm}(\beta\, log\pi_{i,t}^*(\theta))] \\
&\quad -\frac{1}{\sum_{i=1}^{\hat{B}}\sum_{k=1}^{K}|\tau_{i,k}|}\sum_{i=1}^{\hat{B}}\sum_{k=1}^{K}\sum_{t=1}^{|\tau_{i,k}|}\phi(\pi_{i,k,t}(\theta)) \\
&\quad \mathbf{1}(\tilde{A}_{i,k,t}<0)\,[1-\text{sigm}(-\beta\, log\pi_{i,k,t}(\theta))]
\end{aligned} \tag{22}
$$

This approach focuses on learning tokens with high policy uncertainty, striking a balance between informative novelty and policy stability.

**Balance Preference Optimization and Reinforcement Learning via $\gamma$** We adopt a unified loss function that minimizes a weighted sum of the SPO and RL losses:

$$\mathcal{L}_{\text{ToolUPT}}(\theta) = \gamma \cdot \tilde{\mathcal{L}}_{\text{SPO}}(\theta) + (1-\gamma)\cdot \mathcal{L}_{\text{FG-DAPO}}(\theta) \tag{23}$$

where $\tilde{\mathcal{L}}_{\text{SPO}}$ (Equation 22) and $\mathcal{L}_{\text{FG-DAPO}}$ (Equation 18) are the SPO and fine-grained DAPO losses, and $\gamma \in [0, 1]$ is a balancing coefficient. We dynamically adjust the $\gamma$ using warm-up combined with periodic scheduling. Starting from a small value, $\gamma$ gradually increases linearly to a larger value, then decays back down. This cyclical schedule enables the model distribution to adapt smoothly to tool-use tasks while maintaining the exploration-exploitation balance.

# 4 Experimental Setup

## 4.1 Training Dataset

We construct the training set by sampling examples from six open-source datasets: Hermes Glaive Function Calling, ToolACE, xLAM, APIGen-MT, FunReason-MT, and Hammer. We convert multi-turn and multi-step instances into single-turn, single-step format by incorporating dialogue history and previous tool calls as context, simplifying RL training without compromising tool-use capabilities (refer to Appendix A for details).

## 4.2 Benchmark

To examine generalization, we evaluate our model on the **BFCL-v3** benchmark (Patil et al., 2025), which spans a diverse set of challenges, including single-step reasoning, parallel multi-tool selection, irrelevant tool rejection, missing parameter clarification, multi-turn and multi-step tool use. This benchmark provides a robust and scalable framework for assessing the too-use capabilities critical to agentic AI systems.

## 4.3 Baselines

We compare the proposed ToolUPT with a comprehensive set of baselines. **Original Models**: The zero-shot base models without any post-training. **SFT**: The best-performing SFT checkpoint selected based on validation set across different training steps. **GRPO**: Models trained with GRPO with binary (0/1) reward. **DAPO**: Models trained with DAPO using our proposed sequence-level outcome rewards. **DAPO***: Models trained via DAPO with our proposed sequence-level outcome reward, excluding the dynamic sampling and overlong reward shaping components that provide negligible gains. **SFT+DAPO***: The two-stage SFT-then-DAPO* method, with the same modifications as DAPO*. **LUFFY**, **SRFT**, and **CHORD**: Three single-stage approaches that combine expert demonstrations with on-policy rollouts during training.

## 4.4 Training Settings

We evaluate both existing post-training algorithms and our proposed approach across multiple model scales: Llama-3.1-8B-Instruct (Team, 2024), Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct, and Qwen2.5-32B-Instruct (Yang et al., 2024). All methods employ full-parameter fine-tuning. RL baselines and ToolUPT are trained on-policy with group size 8 for 600 steps. SFT baseline is trained for 3,000 steps. Additional hyperparameters are detailed in Appendix B.

# 5 Experimental Results and Analyses

## 5.1 Main Results

We report the average performance across four base models in Table 1, with the corresponding detailed results for each model provided in Tables 2–5. Table 1 shows ToolUPT's superior performance with an average score of 56.6%, surpassing all baseline methods (highlighted in dark blue). Specifically, ToolUPT demonstrates improvements of +9.25 percentage points (pp) over SFT, +8.43 pp over RL, +3.58 pp over the SFT-then-RL pipeline, and +4.83 pp over previous methods combining expert demonstrations with on-policy self-exploration. Furthermore, our FGAE provides consistent improvements when integrated into existing methods (highlighted in light blue): +4.11 pp for DAPO* and +1.40 pp for SFT-then-DAPO*, validating its effectiveness in providing precise and stable gradient signals. Tables 2–5 demonstrate that ToolUPT's gains concentrate in the multi-turn category. This is precisely

the challenging regime where base models are inherently weak and baseline approaches consistently struggle to improve.

| | Llama3.1-8B -Instruct | Qwen2.5-7B -Instruct | Qwen2.5-14B -Instruct | Qwen2.5-32B -Instruct | Avg. |
|---|---|---|---|---|---|
| Zero Shot | 17.03 | 28.87 | 30.01 | 43.59 | 29.88 |
| SFT | 26.84 | 44.59 | 47.77 | 52.35 | 42.89 |
| Pure RL | | | | | |
| Naive GRPO | 27.96 | 40.84 | 42.40 | 50.39 | 40.40 |
| DAPO | 25.58 | 41.62 | 45.64 | 47.77 | 40.15 |
| DAPO* | 30.54 | 41.89 | 46.67 | 55.73 | 43.71 |
| FG-DAPO* | 37.92 | 45.11 | 50.22 | 58.03 | 47.82 |
| Two-stage SFT-then-RL | | | | | |
| SFT+DAPO* | 39.79 | 46.06 | 49.08 | 59.29 | 48.56 |
| SFT+FG-DAPO* | 40.32 | 46.99 | 51.61 | 60.92 | 49.96 |
| Single-stage | | | | | |
| LUFFY | 22.94 | 43.13 | 52.20 | 58.15 | 44.11 |
| SRFT | 38.38 | 44.69 | 46.29 | 45.70 | 43.77 |
| CHORD | 33.48 | 44.25 | 51.04 | 60.46 | 47.31 |
| ToolUPT | **42.66** | **48.65** | **54.92** | **62.32** | **52.14** |

Table 1: Main experiment results on BFCL-v3 benchmarks based on four base models. **Bold** and underline indicate the best and second-best results, respectively.

| | Overall Acc | Non-live (AST) | Live (AST) | Multi Turn | Relevance | Irrelevance |
|---|---|---|---|---|---|---|
| Zero Shot | 17.03 | 27.21 | 36.79 | 0.00 | 56.25 | 38.16 |
| SFT | 26.84 | 55.98 | 36.49 | 0.88 | 62.50 | 65.94 |
| Pure RL | | | | | | |
| Naive GRPO | 27.96 | 40.35 | 31.01 | 0.00 | 25.00 | 96.39 |
| DAPO | 25.58 | 41.56 | 26.65 | 0.00 | 25.00 | 85.26 |
| DAPO* | 30.54 | 62.33 | 49.22 | 1.37 | 62.50 | 67.56 |
| FG-DAPO* | 37.92 | 75.90 | 60.40 | 0.62 | 68.75 | 89.33 |
| Two-stage SFT-then-RL | | | | | | |
| SFT+DAPO* | 39.79 | 86.31 | 63.51 | 0.25 | 75.00 | 88.17 |
| SFT+FG-DAPO* | 40.32 | 84.33 | 66.10 | 3.75 | 81.25 | 80.22 |
| Single-stage | | | | | | |
| LUFFY | 22.94 | 23.54 | 28.28 | 0.75 | 43.75 | 83.59 |
| SRFT | 38.38 | 80.90 | 62.62 | 0.00 | 68.75 | 86.77 |
| CHORD | 33.48 | 72.25 | 56.40 | 1.00 | 75.00 | 69.20 |
| ToolUPT | **42.66** | 85.90 | 73.95 | 5.88 | 81.25 | 78.50 |

Table 2: BFCL-v3 results with Llama3.1-8B-Instruct.

| | Overall Acc | Non-live (AST) | Live (AST) | Multi Turn | Relevance | Irrelevance |
|---|---|---|---|---|---|---|
| Zero Shot | 28.87 | 35.79 | 61.81 | 2.75 | 87.50 | 67.39 |
| SFT | 44.59 | 85.10 | 70.74 | 7.75 | 68.75 | 88.67 |
| Pure RL | | | | | | |
| Naive GRPO | 40.84 | 84.88 | 71.80 | 0.00 | 56.25 | 88.26 |
| DAPO | 41.62 | 85.71 | 71.50 | 0.87 | 62.50 | 89.89 |
| DAPO* | 41.89 | 88.25 | 75.94 | 4.62 | 87.50 | 73.27 |
| FG-DAPO* | 45.11 | 85.54 | 73.87 | 8.00 | 62.50 | 87.27 |
| Two-stage SFT-then-RL | | | | | | |
| SFT+DAPO* | 46.06 | 85.71 | 70.91 | 9.38 | 62.50 | 91.59 |
| SFT+FG-DAPO* | 46.99 | 86.52 | 73.58 | 13.12 | 75.00 | 82.45 |
| Single-stage | | | | | | |
| LUFFY | 43.13 | 88.42 | 76.83 | 3.12 | 81.25 | 84.15 |
| SRFT | 44.69 | 87.69 | 69.58 | 6.38 | 50.00 | 91.74 |
| CHORD | 44.25 | 81.83 | 74.76 | 8.38 | 75.00 | 83.80 |
| ToolUPT | **48.65** | 86.19 | 73.87 | 16.25 | 87.50 | 83.11 |

Table 3: BFCL-v3 results with Qwen2.5-7B-Instruct.

## 5.2 Analyses

**Ablation Study I: Impact of Fine-Grained Advantage Estimation** To evaluate the contribution of FGAE, we replace it with standard sequence-level advantage estimation and outcome reward from DAPO* on Qwen2.5-14B-Instruct. As shown in Figure 4, removing this component decreases overall accuracy from 54.92% to 52.2%. This decline demonstrates that token-level process rewards and advantage estimation are essential for optimal performance.

**Ablation Study II: Impact of Selective Preference Optimization** We further ablate the SPO component on Qwen2.5-14B-Instruct, reducing ToolUPT to a unified loss function that minimizes the weighted sum of SFT and fine-grained DAPO* losses. Figure 4 shows that overall accuracy decreases from 54.92% to 53.52%, confirming the effectiveness of selective supervision and preference optimization.

| | Overall Acc | Non-live (AST) | Live (AST) | Multi Turn | Relevance | Irrelevance |
|---|---|---|---|---|---|---|
| Zero Shot | 30.01 | 28.71 | 43.15 | 8.25 | 56.25 | 83.43 |
| SFT | 47.77 | 84.40 | 72.83 | 7.62 | 68.75 | 88.51 |
| Pure RL | | | | | | |
| Naive GRPO | 42.40 | 86.73 | 75.43 | 2.12 | 62.50 | 85.86 |
| DAPO | 45.64 | 88.48 | 74.39 | 8.50 | 68.75 | 85.49 |
| DAPO* | 46.67 | 86.90 | 75.13 | 11.25 | 56.25 | 84.26 |
| FG-DAPO* | 50.22 | 88.10 | 76.46 | 16.88 | 68.75 | 86.15 |
| Two-stage SFT-then-RL | | | | | | |
| SFT+DAPO* | 49.08 | 86.27 | 72.32 | 19.25 | 81.25 | 78.13 |
| SFT+FG-DAPO* | 51.61 | 88.58 | 72.98 | 19.62 | 62.50 | 89.19 |
| Single-stage | | | | | | |
| LUFFY | 52.20 | 89.67 | 77.65 | 22.00 | 87.50 | 79.91 |
| SRFT | 46.29 | 86.04 | 69.80 | 9.12 | 37.50 | 94.52 |
| CHORD | 51.04 | 91.38 | 73.80 | 18.38 | 68.75 | 85.94 |
| ToolUPT | **54.92** | 87.79 | 74.76 | 27.50 | 75.00 | 84.47 |

Table 4: BFCL-v3 results with Qwen2.5-14B-Instruct.

| | Overall Acc | Non-live (AST) | Live (AST) | Multi Turn | Relevance | Irrelevance |
|---|---|---|---|---|---|---|
| Zero Shot | 43.59 | 82.69 | 77.05 | 8.00 | 93.75 | 77.81 |
| SFT | 52.35 | 88.23 | 76.17 | 20.13 | 81.25 | 89.34 |
| Pure RL | | | | | | |
| Naive GRPO | 50.39 | 88.83 | 76.76 | 15.12 | 68.75 | 91.40 |
| DAPO | 47.77 | 88.08 | 77.79 | 10.38 | 68.75 | 89.61 |
| DAPO* | 55.73 | 87.81 | 76.17 | 26.75 | 75.00 | 90.14 |
| FG-DAPO* | 58.03 | 88.27 | 73.72 | 31.37 | 75.00 | 92.05 |
| Two-stage SFT-then-RL | | | | | | |
| SFT+DAPO* | 59.29 | 89.50 | 76.24 | 35.25 | 81.25 | 84.22 |
| SFT+FG-DAPO* | 60.92 | 88.40 | 76.24 | 37.12 | 87.50 | 89.51 |
| Single-stage | | | | | | |
| LUFFY | 58.15 | 84.44 | 75.94 | 31.63 | 75.00 | 93.67 |
| SRFT | 45.70 | 87.77 | 72.98 | 6.38 | 43.75 | 94.33 |
| CHORD | 60.46 | 89.65 | 79.20 | 36.88 | 81.25 | 83.28 |
| ToolUPT | **62.32** | 88.13 | 76.68 | 40.88 | 87.50 | 86.47 |

Table 5: BFCL-v3 results with Qwen2.5-32B-Instruct.

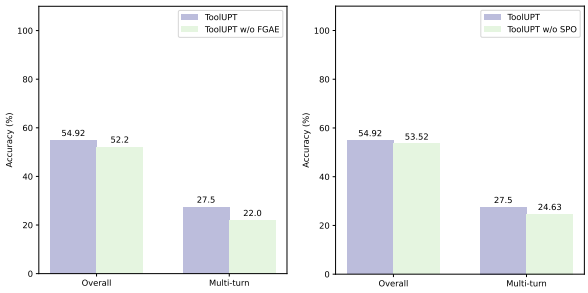


Figure 4: Ablation study on FGAE (left) and SPO (right).

## 6 Conclusion

In this paper, we propose ToolUPT, a novel single-stage unified post-training methodology that synergizes offline expert guidance with online self-exploration. To improve gradient update precision and model robustness, ToolUPT incorporates FGAE into DAPO and employs SPO that leverages expert supervision while enhancing rejected token identification. Extensive experiments conducted on BFCL-v3 validate the effectiveness of our proposed method and framework for agentic tool use.

## Limitations

While our study demonstrates the superiority of the proposed ToolUPT methodology, the following limitations remain. First, our methodology is primarily designed and evaluated on instruction-tuned models; its generalization to reasoning models has not been fully explored. Extending our approach to such models would likely require two key modifications: (1) incorporating process reward models (PRMs) to provide supervision signals at intermediate reasoning steps, and (2) adapting the selective chosen supervision component to effectively identify and leverage high-quality reasoning trajectories. Second, our experiments focus exclusively on dense model architectures. Whether our approach generalizes to Mixture-of-Experts (MoE) models, which exhibit fundamentally different training dynamics, requires further investigation.

## References


Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A general theoretical paradigm to understand learning from human preferences. *CoRR*, abs/2310.12036.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. 2025. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Model alignment as prospect theoretic optimization. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *CoRR*, abs/2504.11536.

Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. 2025. SRFT: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *CoRR*, abs/2506.19767.

Shijue Huang, Wanjun Zhong, Jianqiao Lu, Qi Zhu, Jiahui Gao, Weiwen Liu, Yutai Hou, Xingshan Zeng, Yasheng Wang, Lifeng Shang, Xin Jiang, Ruifeng Xu, and Qun Liu. 2024. Planning, creation, usage: Benchmarking llms for comprehensive tool utilization in real-world complex scenarios. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 4363–4400. Association for Computational Linguistics.

interstellarninja, NousResearch, teknium, and THEODOROS. 2024. Hermes function calling dataset v1. ModelScope, https://modelscope.cn/datasets/AI-ModelScope/hermes-function-calling-v1.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, Jun Wang, and Weinan Zhang. 2025. Robust function-calling for on-device language model via function masking. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Mingyang Liu, Gabriele Farina, and Asuman E. Ozdaglar. 2025a. UFT: Unifying supervised and reinforcement fine-tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, and 8 others. 2025b. Toolace: Winning the points of LLM function calling. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Yifei Lu, Fanghua Ye, Jian Li, Qiang Gao, Cheng Liu, Haibo Luo, Nan Du, Xiaolong Li, and Feiliang Ren. 2025. Codetool: Enhancing programmatic tool invocation of llms via process supervision. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 18287–18304. Association for Computational Linguistics.

Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Hao Wong, Junbo Niu, Chengyu Shen, Runming He, Bin Cui, and Wentao Zhang. 2025. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. *CoRR*, abs/2506.07527.

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*.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, Shelby Heinecke, Weiran Yao, Huan Wang, Silvio Savarese, and Caiming Xiong. 2025. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *CoRR*, abs/2504.03601.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. ToolRL: Reward is all tool learning needs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, Chi Han, Yi R. Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, and 24 others. 2025. Tool learning with foundation models. *ACM Comput. Surv.*, 57(4):101:1–101:40.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: a survey. *Frontiers Comput. Sci.*, 19(8):198343.

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. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/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, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, pages 1279–1297. ACM.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Trans. Mach. Learn. Res.*, 2025.

Hongze Tan and Jianfei Pan. 2025. GTPO and GRPO-S: token and sequence-level reward shaping with policy entropy. *CoRR*, abs/2508.04349.

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. *CoRR*, abs/2306.05301.

Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Zengzhuang Xu, Bingguang Hao, Zechuan Wang, Yuntao Wen, Maolin Wang, Yang Liu, Long Chen, Dong Wang, Yicheng Chen, Cunyin Peng, Chenyi Zhuang, Jinjie Gu, Leilei Gan, Xiangyu Zhao, and Shi Gu. 2025. Funreason-mt technical report: Overcoming the complexity barrier in multi-turn function calling. *CoRR*, abs/2510.24645.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. Learning to reason under off-policy guidance. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Fan Yin, Zifeng Wang, I-Hung Hsu, Jun Yan, Ke Jiang, Yanfei Chen, Jindong Gu, Long T. Le, Kai-Wei Chang, Chen-Yu Lee, Hamid Palangi, and Tomas Pfister. 2025. Magnet: Multi-turn tool-use data synthesis and distillation via graph translation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 32600–32616. Association for Computational Linguistics.

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

Yuanqing Yu, Zhefan Wang, Weizhi Ma, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. 2025b. Steptool: Enhancing multi-step tool usage in llms via step-grained reinforcement learning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management, CIKM 2025, Seoul, Republic of Korea, November 10-14, 2025*, pages 3952–3962. ACM.

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. 2025. 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.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025a. AdaptThink: Reasoning models can learn when to think. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3716–3730, Suzhou, China. Association for Computational Linguistics.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, Zhiwei Liu, Yihao Feng, Tulika Manoj Awalgaonkar, Rithesh R. N., Zeyuan Chen, Ran Xu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, and 2 others. 2025b. xlam: A family of large action models to empower AI agent systems. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 11583–11597. Association for Computational Linguistics.

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2025c. On-policy RL meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. *CoRR*, abs/2508.11408.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025d. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 10495–10516. Association for Computational Linguistics.

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. 2025. Group sequence policy optimization. *CoRR*, abs/2507.18071.

## A Dataset Details

We construct the training set by sampling a proportion of examples from each of the following open-source datasets from prior work:

- **Hermes Glaive Function Calling** (interstellarninja et al., 2024): A dataset containing 5k function-calling conversations, JSON-mode, and structured extraction samples for training LLMs to generate structured outputs from natural language instructions.

- **ToolACE** (Liu et al., 2025b): It leverages a self-evolution synthesis process to curate incorporates a comprehensive API pool of 26,507 diverse APIs, supports complex nested parameters, accommodates both parallel and dependent function calls, and addresses various types of tool-related data.

- **xLAM** (Zhang et al., 2025b): The dataset contains 60,000 data collected by an automated data generation pipeline designed to produce verifiable high-quality datasets for function-calling applications. It collects 3,673 executable APIs across 21 different categories.

| | Hermes Glaive Function Calling | ToolACE | xLAM | APIGen-MT | FunReason-MT | Hammer (Irrelevance) |
|---|---|---|---|---|---|---|
| Original Size | 5,000 | 11,300 | 60,000 | 5,000 | 16,000 | 7,500 |
| Sampled Size | 2,500 | 11,300 | 10,000 | 2,500 | 10,000 | 3,500 |
| Processed Size | 8,829 | 12,253 | 10,000 | 21,622 | 23,915 | 3,500 |

Table 6: Dataset statistics

- **APIGen-MT** (Prabhakar et al., 2025): A high-quality multi-turn agent interaction dataset that generates realistic trajectories where an AI agent gradually comprehends the human's intent and interacts with tools and the environment step-by-step. The dataset has released 5k trajectories, which we use in our experiments.
- **FunReason-MT** (Xu et al., 2025): The dataset comprises 16,000 high-quality multi-turn samples. It was generated using a three-phase data synthesis framework, which focuses on generating complex trajectories that require: environment-API graph interactions, advanced tool-query synthesis and guided iterative chain.
- **Hammer (Irrelevance)** (Lin et al., 2025): A specialized subset of Hammer designed to activate the ability of irrelevant function detection for LLMs.

After random sampling, we post-process multi-turn and multi-step tool-use instances into single-turn, single-step format, concatenating dialogue history and previous tool calls as context in the user prompt (as shown in Figure 6). This processing strategy simplifies the multi-turn, multi-step rollout process in RL trainning without affecting the model's tool-use capability. Table 6 shows the dataset statistics across three stages: original size, sampled size, and processed size after format conversion.

## B Complete Training Configuration

We conduct all experiments using the veRL (Sheng et al., 2025) framework, with the vLLM (Kwon et al., 2023) backend for RL rollouts. All hyperparameters used in our experiments are summarized in Table 7.

| Hyperparameter | RL Methods | SFT |
|---|---|---|
| Training Mode | On-policy | - |
| Training Steps | 600 (2 epochs) | 3,000 |
| Learning Rate | $1 \times 10^{-6}$ | $1 \times 10^{-5}$ |
| LR Scheduler | constant | cosine |
| Batch Size | 256 | 256 |
| Max Prompt Length | 8,192 | 8,192 |
| Max Response Length | 1,024 | 1,024 |
| Group Size | 8 | - |
| Rollout Temperature | 1.0 | - |
| Entropy Coef. | $3 \times 10^{-4}$ | - |

Table 7: Hyperparameters in experiments for training.

## C Prompt Templates

The system prompt we employ during the post-training is shown in Figure 5. The user prompt is used to store the trajectory history, including intermediate dialogue history, tool calls, environment observations, and any additional user commands. The complete user instruction is presented in Figure 6.

**System Prompt for Training**

## **Role**
You are a helpful assistant skilled at using tools to solve user tasks.

## **Task Definition**
Based on the [User Query] and [Chat Short Memory] (user, assistant, etc.), and with reference to the [Intermediate Tools Calls and Returns], plan the tools necessary to solve the current [User Query]. If multiple tools are required, only need to output the tool names and parameter values for the first tool or the first group of tools that could to be called in parallel!

## **Available Tools Description**
__TOOL_DESCS__

## **Overall Response Format**:
MUST choose one of the following two:
### **Tool Usage**
<tool_call>
If a tool is needed, specify the tool's name and parameter values
</tool_call>
### **Answer Appropriately**
<answer>
If an answer is needed, summary or ask user for necessary information
</answer>

## **Tool Usage Output Requirements**
For each tool call, return a json object with tool name and parameters within <tool_call> </tool_call> XML tags
### **Format Requirements**
1. When a single tool is required, the tool call must include the tool's name and parameter values. If no parameters are needed, leave the 'params' field an empty dictionary. The format is:
<tool_call>
{"name": "Tool name", "params": {"Parameter name": "Parameter value", "...": "..."}}
</tool_call>
2. When a single tool or multiple tools can be called IN PARALLEL, enclose each call in [], with tools separated by ', '. The format is:
<tool_call>
[{"name": "Tool-1 name", "params": {"Parameter name": "Parameter value", "...": "..."}}, {"name": "Tool-2 name", "params": {"Parameter name": "Parameter value", "...": "..."}}, ...]
</tool_call>
### **Content Requirements**
1. In the tool description, "required" corresponds to the list of mandatory parameters. Do not extract optional parameters that are not in this list and are not mentioned in the input content!

## **Answer Output Requirements**
When one of the following conditions is met, response an answer directly within <answer> </answer> XML tags
1. When all tools related to the current task have been called, summarize the tool returns and deliver a comprehensive response
2. When no available tool can fulfill the user request, point out that the requested task cannot be achieved due to lack of tools
3. When the value of the tool's required input parameter is missing, ambiguous, or cannot be inferred from the context, ask for clarifications

## **Important Notes**
1. Response at most one of <tool_call> or <answer>

Figure 5: The system prompt used for post-training.

**User Prompt for Training**

**[Chat Short Memory]**
user: {Round 1 user input}
assistant: {Round 1 model answer}
user: {Round 2 user input}
assistant: {Round 2 model answer}
…

**[User Query]**
__QUERY__

**[Intermediate Tools Calls and Returns]**
Step-1 tool call: …
Step-1 observation: …
Step-2 tool call: …
Step-2 observation: …
…

**[Response]**

Figure 6: The user prompt used for post-training.