

Atomic-to-Chain Planner: Token-Decoupled Policy Optimization with Chain-Awareness for Multi-Step Tool Use

Anonymous Author(s)

Abstract

Tool-augmented LLMs are increasingly deployed to operate web, mobile, and WoT services, yet prevailing training paradigms optimize for single-step accuracy and underperform on long-horizon, multi-step workflows. We introduce the Atomic-to-Chain Planner (AC-Planner), a learning framework that couples atomic correctness with chain-level coherence. At its core, Decoupled Unified Policy Optimization (DUPO) unifies supervised fine-tuning and reinforcement learning under a token-type-decoupled objective over `<plan>`, `<tool_call>`, and `<response>`, using dynamic clipping to retain high-signal positive-advantage updates. DUPO optimizes a chain-aware reward with three components—structural validity and two *novel* terms we introduce: semantic consistency and a sequential decay that credits early decisions—thereby aligning local tool execution with the global plan. To support training in complex scenarios, we propose PlanArena, a validated dataset of about 20k positive and negative multi-step trajectories (up to 7 actions) across 7,941 scenarios and 20,646 tools. Actions are checked by actual tool execution and further audited by humans. On two public benchmarks, AC-Planner sets a new state-of-the-art for multi-turn tool use, surpassing the best prior by +10.13 points on BFCL (Qwen-2.5 14B) and +1.53 points on API-Bank (14B). The code and dataset will be publicly available upon acceptance.

CCS Concepts

• Computing methodologies → Planning and scheduling; • Information systems → Web services; Web applications.

Keywords

Web agents, Tool-augmented LLMs, Multi-step tool use, Planning

ACM Reference Format:

Anonymous Author(s). 2026. Atomic-to-Chain Planner: Token-Decoupled Policy Optimization with Chain-Awareness for Multi-Step Tool Use. In . ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/nnnnnnnn>

1 Introduction

The proliferation of web, mobile, and Web of Things (WoT) services has created a rich ecosystem of tools (e.g., travel APIs, payment gateways, smart-home controllers) that empower users to accomplish increasingly complex real-world tasks. The recently introduced

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, Washington, DC, USA

© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
<https://doi.org/10.1145/nnnnnnnn>

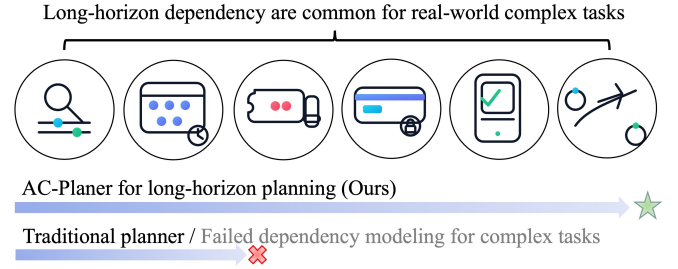


Figure 1: AC-Planner successfully completes complex multi-step tasks by preserving earlystage correctness and enforcing chain-level coherence, yielding robust long-horizon execution; in contrast, prior methods [12] often suffer cascading failures triggered by early missteps.

Model Context Protocol (MCP) [8] specifies a common protocol for large language model (LLM)–tool integration—covering tool discovery, invocation, and streaming I/O—thereby improving interoperability across web-oriented execution environments. By providing structured tool schemas and context-passing semantics, MCP enables LLM-based systems to plan and orchestrate sequences of tool invocations, supporting sophisticated multi-step workflows rather than just single-step queries.

Despite this potential, prevailing LLM training paradigms focus narrowly on single-step accuracy—choosing the right API and filling parameters [13–15]. While such local precision is necessary, it is insufficient for complex, multi-step tasks in web-centric environments, where global coherence and correct action sequencing are essential. As illustrated in Figure 2, successful orchestration depends on the coherence of the entire action chain. However, enabling LLMs to reason over long-horizon dependencies faces two core challenges: (1) annotated datasets for multi-step web workflows remain scarce, and (2) existing training frameworks often struggle to capture sequential dependencies and interactions among heterogeneous web tools.

To address these gaps, we present the Atomic-to-Chain Planner (*AC-Planner*), a framework that couples *atomic* correctness with *chain* coherence for LLMs orchestrating web/mobile/WoT tools. At the atomic level, *decoupled unified policy optimization* (DUPO) is the optimization engine that consumes the chain-aware reward: it unifies supervised fine-tuning and reinforcement learning under a token-type-decoupled objective (`<plan>`, `<tool_call>`, `<response>`) and uses dynamic clipping to retain high-signal positive-advantage updates. This provides a fine-grained learning control. At the chain level, the *chain-aware semantic reward* supplies the signal optimized by DUPO—a composite of (1) structural rewards (syntax/schema compliance) and two *novel components*: (2) a semantic reward that enforces consistency, and (3) a sequentially decaying reward that credits early decisions. Together, DUPO (optimizer)

and the chain-aware reward (signal) align local tool execution with context-aware planing, yielding executable atomic actions and coherent multi-step workflows.

To support modeling in complex scenarios, we construct PlanArena, a large-scale corpus of positive and negative multi-step action trajectories. Each trajectory is formed by composing atomic tasks proposed by multiple base LLMs (both closed- and open-source). Unlike prior datasets that emphasize short chains (e.g., 2–3 actions) [2], PlanArena includes trajectories of up to seven actions, better reflecting long-horizon real-world complexity. We explicitly elicit both goal-achieving (optimal) and semantically plausible but failure-prone (sub-optimal) variants at decision points, yielding contrastive supervision signals. All actions are first validated via actual tool execution (schema and runtime checks), then audited by human annotators to correct semantic and logical issues. The final dataset comprises approximately 20,000 trajectories across 7,941 scenarios, covering 20,646 distinct tools, providing a comprehensive resource for training LLMs in tool-augmented environments.

Leveraging *PlanArena* and *AC-Planner*, we achieve the state-of-the-art multi-turn tool-use performance, improving over the best prior by **10.13** points on BFCL (Qwen-2.5 14B) and **1.53** points on API-Bank (14B). In summary, the key contributions of this study are as follows:

- **AC-Planner**: a token- type-decoupled SFT-RL framework (DUPO) with dynamic clipping, aligning atomic correctness with chain coherence.
- **PlanArena**: a validated corpus of ~20k positive/negative multi- step trajectories (up to 7 actions) across 7,941 scenarios and 20,646 tools.
- **Outcomes**: state-of-the-art multi- turn tool use: +**10.13** on BFCL (Qwen-2.5 14B) and +**1.53** on API-Bank (14B).

2 Related Work

2.1 Trajectory Planning in Tool Spaces

The construction of a trajectory - a sequence of actions to complete a task - becomes increasingly complex as the number of steps increases [12]. Recent studies improve LLM complex sequential planning by incorporating richer context and historical dependency information, enabling the model to remember previous tool outcomes and adjust subsequent actions accordingly [6]. Others introduce self-reflective mechanisms where the agent uses feedback from tool executions to correct its plan, or perform global pre-planning of the entire tool sequence before execution [21]. These techniques help avoid dead-ends and reduce redundant tool calls [1, 3, 18]. However, the expressiveness of any trajectory is inherently constrained by the available tool space. However, modeling inter-tool and inter-task dependencies is equally critical for effective multi-step planning. For model to understand the dependencies, some works attempt to expand tool space through synthesis, for example, APIgen [10] generates new API calls on the fly, and frameworks like StableToolBench [7] provide fallbacks to thousands of real-world APIs for more diverse tool usage - most agents still operate within a fixed, scenario-agnostic toolset. This means the agent cannot dynamically discover entirely new tools beyond its predefined repertoire. To date, few studies have explored on-the-fly, scenario-specific tool

discovery, which limits the generality and completeness of learned trajectories.

2.2 Reward Shaping for Multi-Step Tool Use

Designing effective rewards is critical for guiding reinforcement learning in multistep tool use scenarios [9, 14]. Naïve approaches that reward only the final task success or the correctness of the final answer [15] results in sparse feedback, overlooking whether each intermediate tool invocation was useful or not [13]. Recent work addresses this limitation through finer-grained reward shaping. For instance, StepTool [23] introduces step-grained rewards, assigning intermediate rewards for each tool call based on its success and contribution to overall progress. By evaluating every tool invocation (e.g. correct API used with correct parameters) and providing feedback at each step, StepTool’s RL agent learns not only whether to use a tool but also when and how to invoke it. Building on this idea, ToolRL [12] conducts a comprehensive study of reward design for tool use and proposes a principled per-step reward scheme. Using a Group Relative Policy Optimization (GRPO) [19] backbone, ToolRL [12] provides granular feedback for tool selection and usage, which yielded robust improvements over both base and SFT-tuned models. Such shaping encourages exploration of the tool-action space, helping the model learn not just to call tools correctly, but to sequence them effectively to solve complex tasks. Yet, these methods typically remain step-agnostic—assessing actions in isolation—without the contextual consideration in the full trajectory. As a result, sequence-aware reward signals, which capture the long-term impact of each decision, are still largely underexplored.

2.3 Reinforcement Learning for Sequential Rewards

Classical RL encourages efficiency with per-step penalties and discounted returns, but these linear schedules are ill-suited to LLM-based tool use where plans vary widely in length and exhibit hierarchical structure [4, 17]. Recent studies adopts GRPO [19] to enhance models ability to generate multi-step tool use due to GRPO’s improving sample efficiency for exploring multiple path at once. However, recent study shows incremental improvement on GRPO [19] and RL in general, DAPO [22] stabilizes training process updates via decoupled clipping (“clip-higher”), dynamic sampling, and token-level policy gradients—yielding strong results but with sensitivity to hyperparameters; CHORD [24] harmonizes SFT and on-policy RL through dynamic weighting and token-level filtering, improving stability yet potentially constraining exploration if the supervised signal dominates; lastly, Thinkless [5] reduces unnecessary chain-of-thought by learning when to think through a control token and a decoupled (mode vs. response) objective, allowing a more fine-grained control over the training. The current studies to date often only address part of the problem long-horizon planning faces, Our method combines these insights—dynamic SFT+RL unification (high SFT weight early), dynamic clipping, and truly decoupled losses across multiple reasoning and tool calling tags—to preserve stability while enabling broad exploration over long tool-use trajectories.

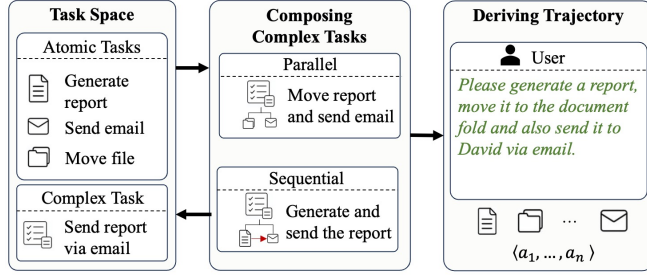


Figure 2: PlanArena is created with trajectories through three key steps: atomic tasks identification, complex tasks compositions and trajectory creation.

3 PlanArena Dataset

To facilitate training on complex multi-step tasks, we introduce the *PlanArena* dataset, a corpus of globally planned and human-validated tool-use trajectories that exhibits both *depth* (long horizons) and *breadth* (diverse tools and scenarios). Overall, the dataset contains approximately 20,000 trajectories spanning 7,941 scenarios and covering 20,646 distinct atomic actions (unique tool-operation types). We outline the construction of PlanArena as follows.

3.1 Trajectory Generation for Complex Tasks

Task Space and Query Inference. As illustrated in Figure 2, we model each scenario g (e.g., airline booking) by a *task space* seeded with schema-validated *atomic* (one-call) tasks. Let $\Gamma_1(g)$ denote the set of atomic tasks and each task deterministically maps to an *atomic action* via:

$$\alpha : \Gamma_1(g) \rightarrow \mathcal{A}, \alpha(\mathcal{T}_1) = a = (\text{plan}, \text{tool_call}, \text{response}), \quad (1)$$

be the mapping to atomic actions, where \mathcal{A} is the action set and $(\text{plan}, \text{tool_call}, \text{response})$ is a schema-validated tool call.

Starting from $\Gamma_1(g)$, we compose atoms into higher-level tasks using serial “,” and/or parallel “||” composition, expanding the task space level by level until a fixed budget is reached (no new tasks emerge). For each composed candidate, an off-the-shelf LLM *infers a natural-language query* intended to describe the task’s objective; this query is later used and validated by human against the composed task.

Trajectory Derivation. For each canonical (possibly multi-step) task, we deterministically unroll it into its constituent atomic subtasks and map them to actions, producing an ordered action trajectory. A *trajectory* is a finite action sequence $\mathbf{a} = \langle a_1, \dots, a_N \rangle \in \mathcal{A}^*$. We write \mathcal{A}^* for the free monoid over \mathcal{A} , i.e., the set of all finite action sequences.

Formal View. Let m denote the *hierarchical depth* and b the effective branching factor. The level- m task space is constructed recursively from lower levels:

$$\Gamma_m = \bigotimes_{i=1}^b \Gamma_{m-1}, \quad \Gamma_1 = \text{atomic tasks}. \quad (2)$$

Unrolling any $\mathcal{T}_m \in \Gamma_m$ to the atomic level induces an ordered list of atomic subtasks

$$\mathcal{T}_m \mapsto \langle \mathcal{T}_{1,1}, \dots, \mathcal{T}_{1,n}, \dots, \mathcal{T}_{1,N} \rangle, \quad \mathcal{T}_{1,n} \in \Gamma_1, \quad (3)$$

where N is the trajectory length, and a possible ground-truth action sequence is

$$\mathbf{a} = \langle a_1, \dots, a_n, \dots, a_N \rangle, \quad a_n = \alpha(\mathcal{T}_{1,n}) \in \mathcal{A}. \quad (4)$$

We measure task difficulty by the *minimal trajectory length*,

$$n(\mathcal{T}, g) := \min_{\mathbf{a}} |\mathbf{a}| \text{ s.t. } \mathbf{a} \text{ correctly executes } \mathcal{T} \text{ in scenario } g. \quad (5)$$

Execution Filter. Every candidate trajectory with atomic tasks is checked for schema validity and executed end-to-end. Trajectories that fail to execute (invalid calls, runtime errors, missing outputs) are *discarded*.

3.2 Context-Aware Planning.

To expose cross-tool dependencies upfront, we prepend each trajectory with a global, context-aware plan P . Let the target task \mathcal{T}_m be decomposed into atomic subtasks $\langle \mathcal{T}_{1,1}, \dots, \mathcal{T}_{1,n}, \dots, \mathcal{T}_{1,N} \rangle$. We serialize this list into a natural-language outline and place it in the `<plan>` segment *before* any `<tool_call>`. Unlike local CoT supervision that conditions only on past steps, P reveals the full sequence and inter-step dependencies from the start, improving long-horizon coherence during decoding and learning.

3.3 Validation of the Trajectories.

Human Validation. For each *executable* trajectory, annotators verify (i) that the off-the-shelf LLM-inferred query correctly states the task goal and (ii) that the trajectory actually achieves that goal. If both hold, the trajectory is labeled *positive for that task*; otherwise *negative for the same task*.

Contrastive Variants. To broaden coverage, we adopt a contrastive dual-generation scheme: for each scenario, external LLMs (e.g., Gemini, GPT, Qwen) produce a complete trajectory and, at each decision point, both correct atomic tasks and semantically relevant but incorrect alternatives, each assigned a relevance score. This teaches fine-grained validity and inter-tool dependencies. All data are then audited by human annotators, and low-quality or illogical trajectories are removed. This pipeline prevents the dataset from being dictated solely by external LLM outputs while improving both quality and variety.

4 Methodology

As illustrated in Figure 3, our *Atomic-to-Chain Planner* (AC-Planner) integrates *policy optimization* and *reward design* under a unified atomic-to-chain learning paradigm. It comprises two complementary components: (i) *Decoupled Unified Policy Optimization* (DUPO), which learns robust *atomic* behaviors through token-type decoupling and a dynamically bounded PPO objective; and (ii) *Chain-Aware Semantic Rewards*, which provide the structured, semantic, and temporal feedback that DUPO optimizes against. Together, they enable the model to align low-level tool execution with high-level chain reasoning—learning stable atomic actions while preserving long-horizon coherence.

4.1 Atomic Action and Token Representation

Given an action trajectory that is an ordered sequence:

$$\mathbf{a} = \langle a_1, \dots, a_n, \dots, a_N \rangle \in \mathcal{A}, \quad (6)$$

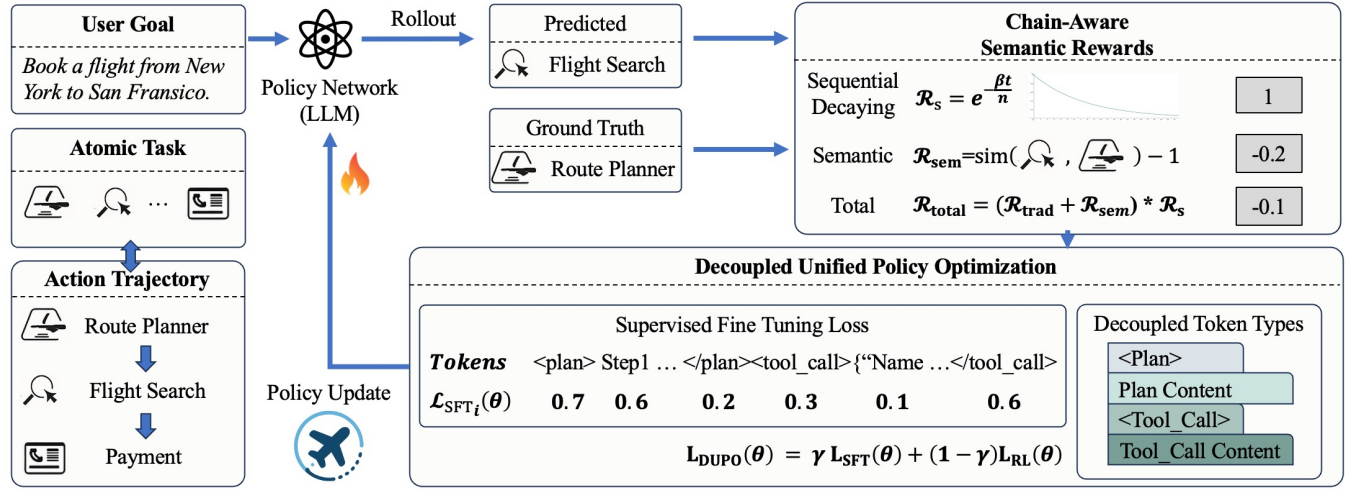


Figure 3: AC-Planner overall framework with Decoupled Unified Policy Optimization (DUPO) and Chain-Aware Semantic Rewards. During rollout, the policy network proposes the next action; the reward compares the predicted step with the ground truth via structural validity and our two *novel* terms—semantic consistency and a sequential decay that credits early decisions—to shape long-horizon coherence. At the bottom, DUPO unifies SFT and RL with token-type decoupling and dynamic clipping, assigning separate weights to tags and contents across <plan>, <tool_call>, and <response> so local tool execution aligns with the global plan.

each a_n contains a sequence of tokens $\{a_{n,i}\}$ from a set V structured with three tagged segments:

$$a_n \equiv \langle \text{plan} \rangle p_n \langle / \text{plan} \rangle \\ \langle \text{tool_call} \rangle u_n \langle / \text{tool_call} \rangle \\ \langle \text{response} \rangle r_n \langle / \text{response} \rangle, \quad (7)$$

where <plan> is for a natural-language task outline, <tool_call> contains a JSON API call that is validated by the tool schema, and <response> provides the final user-facing answer, produced with the action a_N . Correspondingly, we have six types of tokens:

- TAG-PLAN: structural tags <plan> and </plan>
- TAG-TOOL: structural tags <tool_call> and </tool_call>
- TAG-RESP: structural tags <response> and </response>
- PLAN: tokens inside <plan>...</plan>
- TOOL: tokens inside <tool_call>...</tool_call>
- RESP: tokens inside <response>...</response>

In a reinforcement learning setting, a policy model G_θ (an LLM in this study) recursively outputs an estimated trajectory $\hat{\mathbf{a}} = \langle \hat{a}_1, \dots, \hat{a}_n \rangle$ for \mathbf{a} .

4.2 Decoupled Unified Policy Optimization (DUPO)

For the i -th token $a_{n,i}$ of a_n , we sample K rollout candidates $\{\hat{a}_{n,i,k}\}_{k=1}^K$ from the policy distribution (over logits) and assign per-candidate rewards $r_{n,i,k}$. DUPO leverages these rewards in a PPO-style surrogate with two components: (i) decoupled token-type weighting and (ii) a dynamic upper bound for the policy ratio to preserve large positive-advantage updates.

Decoupled token-type weighting. Tags enforce structure, while contents drive task completion; they benefit from different learning

pressures. We therefore decouple the objective by token type. Let the function $\tau(\hat{a}_{n,i,k})$ assign the weights in line with the six token types. Define nonnegative type weights $\{\rho_c\}$ for $c \in C$ (optionally normalized so $\sum_c \rho_c = 1$). Given a generic per-token (reward-based) objective \mathcal{L} , the type-weighted loss is:

$$\mathcal{L}_{RL} = \text{avg}_{n,i,k} [\rho(\hat{a}_{n,i,k}) \cdot \mathcal{L}(\hat{a}_{n,i,k})]. \quad (8)$$

This formulation cleanly separates structural tags from content tokens and allows assigning distinct learning pressure to planning text (PLAN), tool JSON (TOOL), and user-facing response (RESP), while ensuring tag correctness via (TAG-PLAN, TAG-TOOL, TAG-RESP).

Unified Loss with Dynamic Upper Clipping. Compared to DAPO [22] and GRPO [19], we propose to relax the fixed PPO clipping range with a *dynamic upper bound* on the policy ratio r , so that updates with large positive advantage are not prematurely truncated. While this increases exploration and learning signal from rollouts, it also introduces optimization instability. We mitigate this by a unified loss (DUPO) that linearly anneals from supervised cross-entropy (SFT) to the RL surrogate, providing a strong curvature prior early on and stabilizing training as the dynamic upper clipping becomes effective.

In detail, we have:

$$\mathcal{L}(\hat{a}_{n,i,k}) = \min \left[r(\hat{a}_{n,i,k}) \hat{A}_{n,i,k}, \Phi(r(\hat{a}_{n,i,k}), 1 - \epsilon_L, 1 + \epsilon_H) \hat{A}_{n,i,k} \right], \quad (9)$$

where the policy ratio is

$$r(\hat{a}_{n,i,k}) = \frac{\pi(\hat{a}_{n,i,k} \mid a_{n,<i}, \theta_{\text{new}})}{\pi(\hat{a}_{n,i,k} \mid a_{n,<i}, \theta_{\text{old}})}, \quad (10)$$

with $\pi_{\theta_{\text{new}}}$ and $\pi_{\theta_{\text{old}}}$ denoting the current and reference policies, respectively. The mean-normalized advantage is

$$\hat{A}_{n,i,k} = \frac{\mathcal{R}_{n,i,k} - \bar{\mathcal{R}}_{n,i}}{\bar{\mathcal{R}}_{n,i}}, \quad \bar{\mathcal{R}}_{n,i} = \frac{1}{K} \sum_{k=1}^K \mathcal{R}_{n,i,k}, \quad (11)$$

and $\Phi(x; \ell, u) = \min\{\max(x, \ell), u\}$ is the clipping operator with lower / upper bounds $1 - \epsilon_L$ and $1 + \epsilon_H$.

The upper clipping limit ϵ_H is dynamically adjusted as an asymptotic function of the past *average reward* at the same context. This yields a bounded, monotone, self-annealing upper clip that reduces premature truncation of large positive-advantage updates. We use

$$\epsilon_H = \frac{\text{avg}(\mathcal{R}_{n,<i,k}) \cdot (T - t + 1)}{\text{avg}(\mathcal{R}_{n,<i,k}) + (T - t + 1)}, \quad (12)$$

where t indicates the current learning step and T is the total learning steps. Early in training, $(T - t)$ is large, it allows responsiveness to strong signals; late in training, it makes the policy update more conservative for convergence. The construction guarantees $\epsilon_{\text{high}} \leq \min(\text{avg}(\mathcal{R}_{n,<i,k}), \text{total_step} - \text{current_step} + 1)$ so it never explodes to unreasonable values—protecting against instability.

DUPO Unified Loss. To further stabilise the training, we introduce a unified loss:

$$\mathcal{L}_{\text{DUPO}} = \gamma \mathcal{L}_{\text{SFT}} + (1 - \gamma) \mathcal{L}_{\text{RL}}, \quad (13)$$

where γ is a decaying function of the step t , $\gamma = e^{-\zeta t}$, and $\zeta \in (0, 1]$ is a hyperparameter controlling the rate of decay. The SFT loss is defined as:

$$\mathcal{L}_{\text{SFT}} = \text{avg}_i [-\log \pi(a_{n,i} | a_{n,<i}, \theta_{\text{old}})]. \quad (14)$$

The γ schedule ensures smooth continuation, so the loss remains well-behaved. As t increases, γ decays exponentially and the RL loss gradually takes over. So, at the start of training, when dynamic clipping is high, the unified loss is dominated by supervised cross-entropy, which forces the policy to stay close to the reference ground truth. This prevents policy exploitation or divergence at the beginning and provides a stable starting point.

4.3 Chain-Aware Semantic Rewards

Long-horizon tool use demands *chain-level* coherence: early choices constrain later calls, and local correctness is not sufficient. We therefore design a *chain-aware* reward that scores each candidate action for a_n along three complementary axes—(i) structural validity (common in prior work) and two *novel* components, (ii) semantic alignment and (iii) sequence-aware credit assignment—so the policy learns to plan and act consistently across the entire trajectory.

- (1) **Structure reward** $\mathcal{R}_{\text{struct}}$ evaluates correctness of tool name, parameter values, and output format;
- (2) **Semantic reward** \mathcal{R}_{sem} measures the semantic relevance of the candidate action to the ground-truth task;
- (3) **Sequentially decaying reward** $\mathcal{R}_{\text{S-decay}}$ assigns higher weights to early-stage decisions, penalizing early mistakes more severely.

Formally, the combined reward for the k -th rollout of the i -th token in a_n is computed as:

$$\mathcal{R}_{n,i,k} = \left(\mathcal{R}_{\text{struct}}^{n,i,k} + \mathcal{R}_{\text{sem}}^{n,i,k} \right) \times \mathcal{R}_{\text{S-decay}}^{n,i,k}. \quad (15)$$

Structure Reward. $\mathcal{R}_{\text{struct}}^{n,i,k}$ is defined as a combination of three aspects:

$$\mathcal{R}_{\text{trad}}^{n,i,k} = r_{\text{name}}^{n,i,k} + r_{\text{param}}^{n,i,k} + r_{\text{format}}^{n,i,k}. \quad (16)$$

Each term is computed as follows:

$$\begin{cases} r_{\text{name}}^{n,i,k} = \text{IoU}_{\text{name}}(\hat{a}_{n,1:i-1} \oplus \hat{a}_{n,i,k}, a_{n,1:i}), \\ r_{\text{param}}^{n,i,k} = \text{IoU}_{\text{param}}(\hat{a}_{n,1:i-1} \oplus \hat{a}_{n,i,k}, a_{n,1:i}), \\ r_{\text{format}}^{n,i,k} = \mathbb{I}(\hat{a}_{n,i,k} \text{ format is valid}), \end{cases} \quad (17)$$

where IoU indicates Intersection-over-Union, ranging from 0 to 1 and reflects the degree of overlap between two components, and \mathbb{I} is an indicator function; and $\hat{a}_{n,1:i-1} \oplus \hat{a}_{n,i,k}$ denotes the partial hypothesis formed by taking the greedy (top-1) prefix at positions $1:i-1$ and appending candidate k at i , which we compare against the gold prefix $a_{n,1:i}$ (i.e., ground truth).

Semantic Reward. To assess the semantic relevance of each sampled action $\hat{\mathcal{T}}_{1,m}^{(t)}$, we leverage a pretrained LLM (denoted as LLM-S) to compute a similarity score:

$$\text{sim}_{n,i,k} = \text{sim}(\hat{a}_{n,1:i-1} \oplus \hat{a}_{n,i,k}, a_{n,1:i}) \in [0, 1]. \quad (18)$$

The semantic reward is then defined as:

$$\mathcal{R}_{\text{sem}}^{n,i,k} = \lambda_{\text{sem}} \cdot (\text{sim}_{n,i,k} - 1), \quad (19)$$

where λ_{sem} is a tunable coefficient controlling the contribution of semantic alignment to the final reward.

Sequentially Decaying Reward. In long-horizon task trajectories, early-stage decisions often play a more critical role in determining overall success. To reflect this intuition, we introduce a sequentially decaying reward scheme, which assigns higher importance to earlier steps in a trajectory. Specifically, for the n -th action a_n , we define a sequential importance weight:

$$\mathcal{R}_{\text{S-decay}}^{n,i,k} = e^{-\frac{\beta n}{N}}, \quad (20)$$

where $\beta = \log(N + 1)$, controlling the smoothness of the decay while ensuring adaptive decay for tasks of varying lengths. This scheme encourages the model to avoid early-stage errors and to pay greater attention to initial planning steps. The smooth reward shaping also contributes to training stability, as previously observed in ToolRL [12] and ToolLLM [15].

5 Experiments & Discussions

5.1 Benchmarks & Implementation

Benchmarks. To evaluate the capabilities of our method (AC-Planner) and dataset (PlanArena for training) for complex task handling, we benchmark on two suites spanning diverse tasks and difficulty levels: BFCL [11], which assesses multi-step, multi-turn tool use, and API-Bank [10], which covers diverse API calls under increasing difficulty.

Implementation Details. We train on the *PlanArena* dataset using a framework built on VERL [20]. We fine-tune *Qwen-2.5-Instruct* (Qwen2.5) [16], which serves as the planner. DUPO serves as the reinforcement-learning optimizer, enhanced with the chain-aware semantic reward. Experiments are conducted on 8xNVIDIA A100 GPUs with a global batch size of 256. More details will be available in our codes that will be released upon acceptance.

Model	Overall Acc	Non-Live AST Acc	Live Acc	Multi-Turn Acc	Relevance Detection	Irrelevance Detection
Qwen2.5-3B (Raw)	33.04%	42.52%	53.96%	1.00%	64.71%	56.01%
Qwen2.5-3B (SFT)	47.82%	66.67%	67.75%	2.38%	77.78%	100.00%
Qwen2.5-3B (ToolRL)	52.98%	81.58%	73.78%	3.75%	88.24%	84.85%
Qwen2.5-3B (AC-Planner)	52.10%	81.29%	65.57%	10.12%	83.33%	69.34%
Qwen2.5-7B (Raw)	41.97%	66.02%	53.51%	4.25%	76.47%	62.66%
Qwen2.5-7B (SFT)	43.05%	51.35%	65.53%	4.62%	50%	86.86%
Qwen2.5-7B (ToolRL)	58.38%	86.17%	74.9%	18.12%	83.33%	76.68%
Qwen2.5-7B (AC-Planner)	59.91%	83.62%	72.55%	24.63%	83.33%	71.99%
Qwen2.5-14B (Raw)	56.32%	83.38%	74.01%	14.75%	83.33%	70.09%
Qwen2.5-14B (SFT)	56.89%	81.65%	75.92%	11.50%	72.22%	86.70%
Qwen2.5-14B (ToolRL)	62.40%	88.48%	75.97%	23.87%	72.22%	79.16%
Qwen2.5-14B (AC-Planner)	64.79%	86.29%	74.68%	34.00%	66.67%	78.00%

Table 1: Performance evaluation on BFCL (↑). Overall accuracy include both multi-turn and single-step accuracy, with the weighting dominated by the single-step accuracy; Multi-turn reports multi-step success.

Model	Overall Acc	Level 1	Level 2	Level 3
3B (Raw)	35.68%	43.36%	22.39%	19.08%
3B (SFT)	57.12%	61.15%	49.25%	48.85%
3B (ToolRL)	67.00%	73.43%	67.16%	47.33%
3B (AC-Planner)	65.33%	71.18%	62.69%	48.85%
7B (Raw)	58.29%	65.66%	41.79%	44.27%
7B (SFT)	59.63%	66.67%	53.73%	41.22%
7B (ToolRL)	64.66%	73.93%	61.19%	38.17%
7B (AC-Planner)	67.84%	72.93%	65.67%	53.44%
14B (Raw)	45.56%	51.88%	31.34%	33.59%
14B (SFT)	64.82%	72.43%	62.69%	42.75%
14B (ToolRL)	66.16%	71.93%	62.69%	50.38%
14B (AC-Planner)	68.01%	74.19%	62.69%	51.91%

Table 2: Performance evaluation on API-Bank (↑). Overall accuracy include both multi-turn and single-step accuracy, with the weighting dominated by the single-step accuracy; Level- x indicates multi-step success for tasks with x atomic steps.

5.2 Overall Performance

Results and Analysis. Our method (AC-Planner), instantiated with Qwen-2.5 7B and 14B backbones, achieves the best overall accuracy (a single-step based evaluation metric) on both BFCL and API-Bank (Tables 1 and 2). Given the complexity of the real-world scenarios, we adopt *multi-turn* performance as the primary metric, as it more faithfully reflects long-horizon reasoning and tool-use robustness. On the BCFL benchmark [11], AC-Planner outperforms the strongest existing method ToolRL by 10.13%, with even larger margins over standard SFT. On API-Bank [10], we observe consistent improvements and new state-of-the-art results on tasks requiring complex tool use, indicating the method’s generalizability. **Impact of Model Size.** Additionally, we observe a consistent improvement with planner model scale on both BFCL and API-Bank. Larger backbones better handle long-context inputs and capture

cross-step logical dependencies—both critical for multi-step planning. Our method is able to leverage the greater capacity of larger models to improve both multi-turn and single-step metrics.

5.3 Ablation Study

We analyze the contribution of each core component to understand its impact and establish stronger baselines for future work.

Impact of Context-Aware Planning (PlanArena Dataset). We assess the effect of context-aware plans by removing the <plan> segment from the training data. As shown in Table 3, adding global plans in training consistently improves performance across benchmarks, with an average gain of 4.4% over models trained without planning. We found that introducing planning greatly benefits smaller models, as they gain more context with long-range dependencies and structured task priors, which they would otherwise lack the ability to capture or generate without training. This indicates that *PlanArena* provides highquality supervision and is especially useful for training smaller models in line with recent smallmodel training practices.

Model	Overall Acc	Multi-Turn Acc
Qwen2.5-3B (Without)	48.70%	3.25%
Qwen2.5-3B (With)	52.10%	10.12%
Qwen2.5-7B (Without)	59.26%	22.62%
Qwen2.5-7B (With)	59.91%	24.63%

Table 3: Global plan information analysis on BFCL (↑).

Moreover, Figure 4 shows an example that providing a global plan improves reasoning over tool interdependencies. By learning to create an explicit outline in the <plan> segment, the model understands clear order constraints and task priors, which guide next-token generation toward coherent, goal-aligned sequences. As illustrated, our method first emits a global plan and then follows it with an ordered action chain; in contrast, ToolRL often miss the sequential structure. Without such guidance, LLMs may incorrectly

parallelize dependent calls (e.g., running *cat* and *cd* in parallel when *cd* must precede *cat*). The benefit is most pronounced on long-horizon tasks that require strict cross-step dependencies.

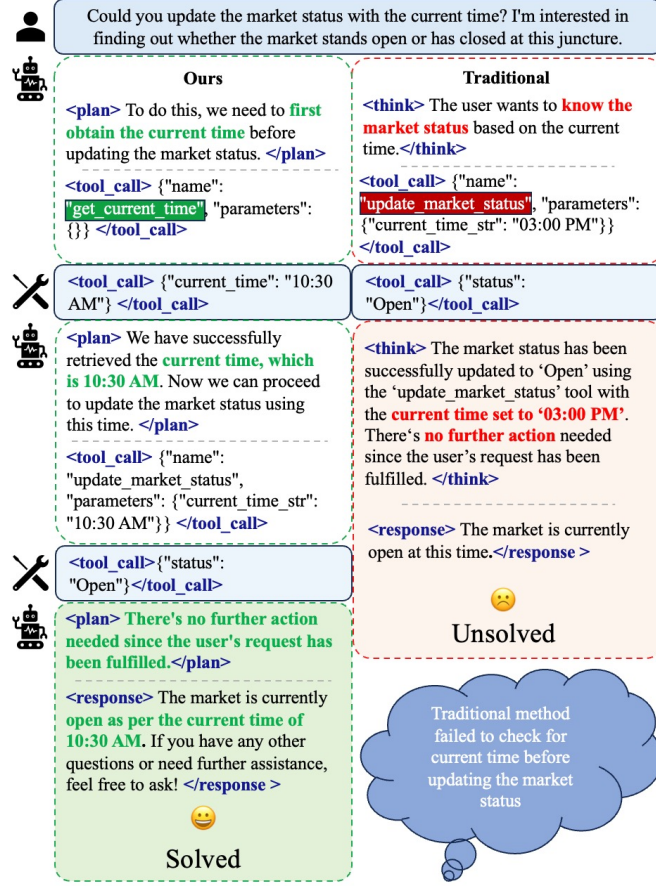


Figure 4: AC-Planner completes complex task by utilizing Global plan, whereas ToolRL failed to capture the sequential relationship between actions.

Impact of Decoupled Weights in DUPO. We implement a decoupled loss by separating structural tags from their segment content at the token level and assigning distinct weights ρ . As shown in Figure 5, performance is more sensitive to content weight than to tag weight: once ρ_{tag} exceeds a modest threshold, tag production becomes reliable and further up-weighting brings diminishing returns, whereas increasing ρ_{content} consistently improves both Overall and Multi-turn metrics.

Impact of Clipping in DUPO. To isolate the effect of clipping, we compare our dynamic upper clipping with a static PPO clip (DAPO [22]). As reported in Table 4, dynamic clipping yields higher accuracy on the multi-turn benchmark, indicating that it better leverages strong positive-advantage updates to enhance the model's exploratory capabilities without compromising stability.

Impact of Unified Optimization in DUPO. To fully understand the impact of unified optimization, we conducted two sets of experiments. In Table 5 we first compared unified optimization against

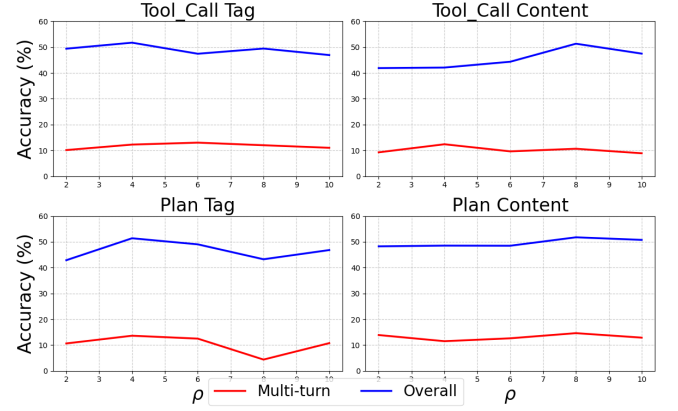


Figure 5: Sensitivity of overall (single-step dominated) and multi-turn accuracy to ζ on BFCL (†).

Model	Overall Acc	Multi-Turn Acc
Qwen2.5-3B (Without)	49.27%	6.62%
Qwen2.5-3B (With)	52.10%	10.12%
Qwen2.5-7B (Without)	57.31%	20.57%
Qwen2.5-7B (With)	59.91%	24.63%

Table 4: Dynamic clipping analysis on BFCL (†).

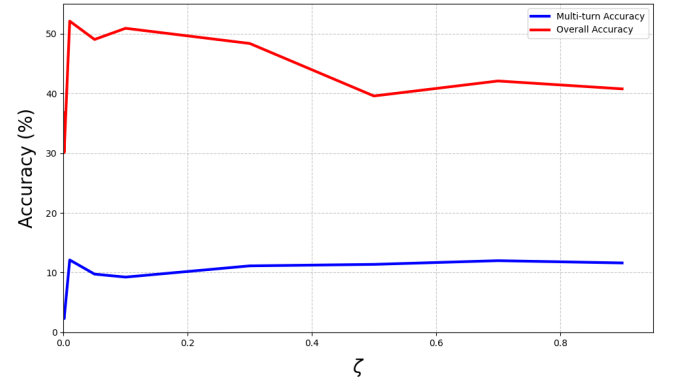


Figure 6: Sensitivity of overall and multi-turn accuracy to ζ .

both RL cold start and SFT-only baselines. Results demonstrate that unified optimization yields superior performance and serves as a more effective training paradigm. This supports the intuition that large-scale pre-training partially mitigates cold-start instability; however, by implementing a decay schedule, we further improve model performance while maintaining training stability. Finding the right balance between exploration freedom and training stability is therefore key.

We further conducted a sensitivity analysis on ζ as shown in Figure 6. Notably, the model requires supervision initially to stabilise training, but excessive supervision actually degrades performance.

This occurs because early-stage supervised learning overly constrains policy updates, leaving insufficient room for exploration in later steps.

Model	Overall Acc	Multi-Turn Acc
3B (SFT)	47.82%	2.38%
3B (Cold Start)	50.21%	5.25%
3B (Unified)	52.10%	10.12%
7B (SFT)	43.05%	4.62%
7B (Cold Start)	59.54%	21.62%
7B (Unified)	59.91%	24.63%

Table 5: Unified optimisation performance gain on BFCL (†).

Impact of Semantic Reward. Table 6 reports the impact of incorporating semantic rewards on BFCL, controlled by a sensitivity parameter λ_{sem} (see Eq. (19)). We observe that λ_{sem} is independent of model choice, as both the 3B and 7B models yield the same result, with $\lambda_{\text{sem}} = 0.5$ achieving the best performance. However, we cannot conclude that λ_{sem} depends on the task space, since our experiments were conducted within the same task space (i.e., the same dataset).

Nevertheless, integrating semantic rewards consistently improves performance by enabling the model to better evaluate the semantic correctness of individual actions thereby refining the learning signal and enhancing step-level decision-making.

Model	Overall Acc	Multi-Turn Acc
Qwen2.5-3B (Without)	48.74%	3.88%
Qwen2.5-3B ($\lambda_{\text{sem}} = 0.2$)	48.26%	3.88%
Qwen2.5-3B ($\lambda_{\text{sem}} = 0.8$)	52.10%	10.12%
Qwen2.5-3B ($\lambda_{\text{sem}} = 0.5$)	49.27%	6.62%
Qwen2.5-7B (Without)	58.80%	22.12%
Qwen2.5-7B ($\lambda_{\text{sem}} = 0.2$)	59.01%	23.25%
Qwen2.5-7B ($\lambda_{\text{sem}} = 0.8$)	59.54%	23.50%
Qwen2.5-7B ($\lambda_{\text{sem}} = 0.5$)	59.91%	24.63%

Table 6: Semantic reward analysis regarding λ_{sem} on BFCL (†), where without this reward is for $\lambda_{\text{sem}} = 0$.

Impact of Sequential Decaying Reward. We investigate the effectiveness of our sequential decaying reward scheme by comparing three different decaying formulations in addition to a non-decaying option (Table 7):

$$\mathcal{R}_{\text{Discrete-decay}}^{n,i,k} = [2, 1.5, 1, 0.5, 0.4, 0.3, 0.2], \quad (21)$$

for $c \in (-\infty, 0], (0, 1], \dots, (4, 5], (5, +\infty)$, respectively,

$$\mathcal{R}_{\text{Linear-decay}}^{n,i,k} = \frac{2}{c+1}, \quad (22)$$

$$\mathcal{R}_{\text{S-decay}}^{n,i,k} = e^{-\frac{\beta c}{\pi}} \quad (\text{Ours}). \quad (23)$$

All reward functions decay over time, but differ in smoothness and sensitivity to step position.

We observe that the exponential decay variant yields the strongest multi-turn performance, whereas the overall (single-step dominated) accuracy shows some variability. A plausible explanation is a misalignment between the step-dependent decay reward and the semantics / correctness-based rewards: the latter are conditioned on action validity, while the sequential decay depends only on position and is agnostic to outcome quality. This misalignment can introduce optimization tension—particularly in larger models. Nonetheless, the smoother decay curve supplies a more consistent training signal, which appears to aid convergence and improve long-horizon planning.

Model	Overall Acc	Multi-Turn Acc
Qwen2.5-3B (Without)	48.26%	5.25%
Qwen2.5-3B (Linear)	48.77%	4.25%
Qwen2.5-3B (Discrete)	48.27%	3.62%
Qwen2.5-3B (Ours)	52.10%	10.12%
Qwen2.5-7B (Without)	59.37%	24.12%
Qwen2.5-7B (Linear)	59.82%	23.88%
Qwen2.5-7B (Discrete)	60.17%	23.75%
Qwen2.5-7B (Ours)	59.91%	24.63%

Table 7: Sequential decaying reward analysis on on BFCL (†).

5.4 Limitations

Due to hardware constraints, we limited our training to relatively small models (3-14B) and capped the task composition depth at $M \leq 11$, where $\Gamma_m = \bigotimes_{i=1}^M \Gamma_{m-1}$. Future work could explore more cost-effective training strategies to scale our framework to larger models and datasets of problem scale. Moreover, while our study focuses on general-purpose and web relevant scenarios, domain-specific applications (e.g., medical, legal, financial) may necessitate additional domain adaptation. This includes incorporating domain knowledge through specialized instruction tuning or Retrieval-Augmented Generation (RAG) to support accurate generation of atomic-level tasks and trajectories.

6 Conclusion

We presented *AC-Planner*, a framework that couples token-type-decoupled policy optimization (DUPO) with chain-aware semantic rewards to align atomic correctness with chain-level coherence in multi-step tool use. Supported by the *PlanArena* dataset of validated positive/negative trajectories, our approach achieves state-of-the-art results on BFCL and API-Bank, with ablations confirming the contribution of type decoupling, dynamic clipping, and chain-aware rewards. Grounded in web-centric tool ecosystems (web/mobile/WoT), our method and dataset constitute practical building blocks—an interoperable training corpus (PlanArena) and an optimization recipe (DUPO with chain-aware rewards)—for tool-augmented LLMs, enabling broader adoption and progress in multi-step, chain-coherent modeling across real-world web services while improving robustness to long-horizon dependencies and easing integration with existing APIs and execution stacks.

References

- [1] Junjie Chen, Haitao Li, Jingli Yang, Yiqun Liu, and Qingyao Ai. 2025. Enhancing LLM-Based Agents via Global Planning and Hierarchical Execution. *arXiv:2504.16563* [cs.LG] <https://arxiv.org/abs/2504.16563>
- [2] Mingyang Chen, Haoze Sun, Tianpeng Li, Fan Yang, Hao Liang, Keer Lu, Bin Cui, Wentao Zhang, Zenan Zhou, and Weipeng Chen. 2025. Facilitating Multi-turn Function Calling for LLMs via Compositional Instruction Tuning. *arXiv:2410.12952* [cs.CL] <https://arxiv.org/abs/2410.12952>
- [3] Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-Act: Improving Planning of Agents for Long-Horizon Tasks. *arXiv:2503.09572* [cs.CL] <https://arxiv.org/abs/2503.09572>
- [4] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Model alignment as prospect theoretic optimization. In *Proceedings of the 41st International Conference on Machine Learning (Vienna, Austria) (ICML'24)*. JMLR.org, Article 504, 18 pages.
- [5] Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Thinkless: LLM Learns When to Think. *Advances in neural information processing systems* (2025).
- [6] Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 79081–79094. https://proceedings.neurips.cc/paper_files/paper/2023/file/f9f54762cbb4fe4dbffdd4f792c31221-Paper-Conference.pdf
- [7] Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. StableToolBench: Towards Stable Large-Scale Benchmarking on Tool Learning of Large Language Models. In *Association for Computational Linguistics*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 11143–11156. doi:10.18653/v1/2024.findings-acl.664
- [8] Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. 2025. Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. *arXiv:2503.23278* [cs.CR] <https://arxiv.org/abs/2503.23278>
- [9] Weiwen Liu, Xu Huang, Xingshan Zeng, xinlong hao, Shuai Yu, Dexun Li, and et al. 2025. ToolACE: Winning the Points of LLM Function Calling. In *The Thirteenth International Conference on Learning Representations*. <https://openreview.net/forum?id=8EB8k6DdCU>
- [10] Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, and et al. Kokane. 2024. APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Datasets. In *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 54463–54482. https://proceedings.neurips.cc/paper_files/paper/2024/file/61cce86d180b1184949e58939c4f983d-Paper-Datasets_and_Benchmarks_Track.pdf
- [11] 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*.
- [12] Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiuxi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. ToolRL: Reward is All Tool Learning Needs. *arXiv:2504.13958* [cs.LG] <https://arxiv.org/abs/2504.13958>
- [13] Cheng Qian, Chi Han, Yi R Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. 2023. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. *arXiv preprint arXiv:2305.14318* (2023).
- [14] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, and et al. Zeng. 2024. Tool Learning with Foundation Models. *ACM Comput. Surv.* 57, 4, Article 101 (Dec. 2024), 40 pages. doi:10.1145/3704435
- [15] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, and et al. Lu. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789* (2023).
- [16] Qwen, ., An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, and Dayiheng Liu and et al. 2025. Qwen2.5 Technical Report. *arXiv:2412.15115* [cs.CL] <https://arxiv.org/abs/2412.15115>
- [17] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: your language model is secretly a reward model. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (New Orleans, LA, USA) (NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 2338, 14 pages.
- [18] Mrinal Rawat, Ambuje Gupta, Rushil Goomer, Alessandro Di Bari, Neha Gupta, and Roberto Pieraccini. 2025. Pre-Act: Multi-Step Planning and Reasoning Improves Acting in LLM Agents. *arXiv:2505.09970* [cs.AI] <https://arxiv.org/abs/2505.09970>
- [19] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, and et al. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv:2402.03300* [cs.CL] <https://arxiv.org/abs/2402.03300>
- [20] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, and et al. 2024. HybridFlow: A Flexible and Efficient RLHF Framework. *arXiv preprint arXiv: 2409.19256* (2024).
- [21] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian (Shawn) Ma, and Yitao Liang. 2023. Describe, Explain, Plan and Select: Interactive Planning with LLMs Enables Open-World Multi-Task Agents. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 34153–34189. https://proceedings.neurips.cc/paper_files/paper/2023/file/6b8dfb8c0c12e6fafc6c256cb08a5ca7-Paper-Conference.pdf
- [22] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. 2025. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv:2503.14476* [cs.LG] <https://arxiv.org/abs/2503.14476>
- [23] Yuanqing Yu, Zhefan Wang, Weizhi Ma, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. 2025. StepTool: Enhancing Multi-Step Tool Usage in LLMs through Step-Grained Reinforcement Learning. *arXiv:2410.07745* [cs.CL] <https://arxiv.org/abs/2410.07745>
- [24] Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2025. On-Policy RL Meets Off-Policy Experts: Harmonizing Supervised Fine-Tuning and Reinforcement Learning via Dynamic Weighting. *arXiv:2508.11408* [cs.LG] <https://arxiv.org/abs/2508.11408>

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009