Reverse the Walk: Long-Tail-Aware Sampling for Generalizable Tool Learning

Anonymous ACL submission

Abstract

Recent tool-augmented and -generated methods exhibit a heavy reliance on dominant-usage tools. This results in a severe long-tail imbalance, where a small fraction of frequently used APIs overshadows a vast number of rarely used ones. This imbalance critically limits a model's ability to generalize beyond memorized, high-frequency tool calls and severely hampers compositional reasoning across tools. To address this fundamental challenge, we introduce reverse walk, a novel data synthesis framework designed to democratize tool learning. We first construct an API dependency graph that captures the semantic relationships between tools based on their task descriptions. Departing from conventional forward generation, we perform a reverse random walk starting specifically from tail nodes (low-frequency tools) to generate multi-hop tool trajectories that naturally incorporate rare but meaningful tool combinations. This strategy compels models to learn the underlying semantic and logical dependencies rather than merely overfitting to frequency-based co-occurrence patterns. Our experimental results on challenging agentic benchmarks like $\tau 2$ and BFCL demonstrate that training on our generated data significantly improves both rare tool generalization and compositional reasoning effectiveness without degrading performance on frequent tools. Our findings highlight the critical importance of long-tail-aware data design for building robust and generalizable tool-using language models.

1 Introduction

The explosive development of large reasoning models (LRMs) has led to burgeoning agentic capabilities, spurring sophisticated methodologies for generating, training, and evaluating various agentic abilities, including decision making, planning, memory/context handling, and tool selection (Wang et al., 2023; Wu et al., 2024; Mialon et al., 2023a; Liu et al., 2023; Wang et al., 2024; Xi et al., 2025a). Consequently, the field is undergoing a paradigm shift towards frameworks that not only enhance a model's reasoning power but also systematically evaluate its performance within complex, interactive environments (Jimenez et al., 2023; Yao et al., 2024; Xi et al., 2025b; Barres et al., 2025; Lu et al., 2025).

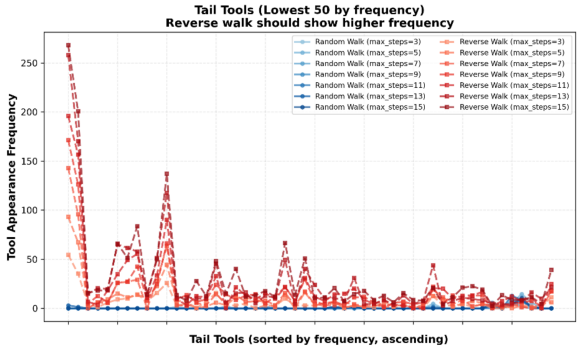

Figure 1: Random walk generation from head nodes rarely reaches tail tools, even when the max steps are increased from 3 to 15. By initiating random walks from tail nodes, meaningful trajectories for low-frequency tools are formed, which support multi-turn conversations and enhance the model's generalization capability.

However, despite the rapid advancements in agentic frameworks, a critical bottleneck remains in the way models learn and utilize tools. A closer examination of current tool-augmented and -generated methods reveals a significant imbalance in tool utilization, often adhering to a severe long-tail distribution. Existing approaches tend to rely heavily on dominant-usage tools, where a small fraction of frequently used APIs (e.g., weather, search, calculator) overshadows a vast number of rarely used but semantically important ones (Qin et al., 2023; Patil et al., 2024). This imbalance limits a model's ability to generalize beyond memorized, high-frequency tool calls and severely hampers reasoning when faced with novel or specialized tasks. As models become more capable, the

Related Works	Scalability	Real APIs	Real Conversations	Multi-step Reasoning	Argument Generation	Tool choice	Planning	Decision Making	Long-tail Handling
ToolLLM (Qin et al., 2023)	✓	✓	△	✓	✓	✓	✓	△	✗
StableToolBench (Guo et al., 2024)	✓	✗	△	✓	✓	✓	✓	✗	✗
Kimi-K2 (Team et al., 2025a)	✓	✓	✓	✓	✓	✓	✓	✗	✗
τ2-Bench (Barres et al., 2025)	△	✗	△	✓	✓	✓	✓	✓	✗
LongCat-Flash (Team et al., 2025b)	✓	✓	△	✓	✓	✓	✗	✗	✗
AgentGym-RL (Xi et al., 2025b)	✓	✗	△	✓	✓	✓	△	△	△
SFR-DeepResearch (Nguyen et al., 2025)	✓	✗	△	✓	✓	✓	△	△	✗
TGAI (Fang et al., 2025)	✓	✗	✓	✓	✓	✓	△	△	✗
ACEBench (Chen et al., 2025a)	△	✗	△	✓	✓	✓	△	△	△
AgentFlow (Li et al., 2025)	✓	△	△	✓	✓	✓	✓	✓	△

Table 1: Comparison of agentic frameworks and their handling of long-tail tool usage. This table highlights that most existing tool call data generation methods, while strong in scalability and multi-step reasoning, predominantly fail to address the critical challenge of long-tail tool usage. Our work provides a clear data construction pipeline to overcome this limitation and ensure generalization on rare tool uses.

ability to handle this "tail" of the distribution becomes a defining factor for true open-world generalization.

The root cause of this persistent limitation in tool generalization capability is directly attributable to the prevailing data generation paradigms currently in use (Team et al., 2025a; Prabhakar et al., 2025). As illustrated in Figure 1, standard data synthesis strategies—which typically rely on simulating random walks initiated from head nodes (frequently used tools)—prove fundamentally ineffective at thoroughly exploring the entire tool action space. Even when the maximum number of steps is increased (e.g., from 3 steps to 15 steps), these conventional forward-generation methods only rarely manage to reach the low-frequency tail nodes. The result of this process is a synthesized dataset that is overwhelmingly dominated by repetitive, high-frequency usage patterns. Consequently, models trained on such data exhibit a "rich-get-richer" bias, failing to form meaningful trajectories[1] for low-frequency tools. This key observation necessitates a paradigm shift in how we construct tool-use trajectories, ensuring that the learning model is systematically exposed to a diverse array of rare and specialized tool combinations.

To address these limitations and bridge the gap between head and tail tool usage, we propose a novel data synthesis framework designed for generalizable tool learning. We first construct an API dependency graph that captures the semantic relationships between tools based on their formal task descriptions. Departing from conventional forward generation methods, we introduce a **reverse random walk** strategy starting specifically from tail nodes. As visualized in the trajectories in Figure 1, this approach effectively generates multi-hop tool chains that naturally incorporate rare but meaningful tool combinations. By effectively forcing the model to navigate the graph in reverse—from the rare tail tool to the more common head tools—we compel it to learn the underlying semantic relationships and logical dependencies rather than merely developing an overfitting tendency toward simple frequency-based co-occurrence patterns.

Our experimental results on the τ2 (Barres et al., 2025) and BFCL (Patil et al.) agentic benchmarks demonstrate that training on our generated data significantly improves both rare tool generalization (low-frequency tools) and reasoning effectiveness without degrading performance on frequent tools (high-frequency tools), highlighting the importance of long-tail-aware data design for building robust agentic models. In addition, by training on low-frequency tools, we could achieve better tool efficiency by following the usage of tools in a multi-step and multi-turn inference stage. We will provide the data and code after the review.

2 Related Works

2.1 Reasoning Models with Tool-Use

LRMs has spurred the creation of a wide range of benchmarks aimed at assessing autonomous agents operating in tool-mediated settings (Guo et al., 2025; Yang et al., 2025a; Team et al., 2025a,c; Zeng et al., 2025). Initial efforts largely emphasized single-step tool usage, such as one-shot function calling or selecting predefined APIs in narrowly scoped environments (Mialon et al., 2023b; Qin et al., 2023; Lee et al., 2025). As research attention shifted toward more realistic, multi-step decision-making, subsequent benchmarks—including Sta-

[1]We define the **trajectory** as a sequence of API calls, which are related (or correlated) with tool arguments for solving tasks

bleToolBench (Guo et al., 2024), BFCL (Patil et al., 2024), and the τ benchmarks (Yao et al., 2024; Barres et al., 2025)—were introduced to evaluate agents' ability to reason over and interact with large, heterogeneous tool ecosystems (Liu et al., 2025; Xu et al., 2025; Xi et al., 2025b).

2.2 Tool Invocation and Agent Strategies

Alongside progress in benchmarking, the design of tool-using agents has evolved from simple function invocation toward more autonomous and flexible orchestration strategies. Early approaches allowed LLMs to interpret tool specifications and issue calls, but they relied on fixed, predefined tool sets, which limited their effectiveness in open-ended environments (Schick et al., 2023; Hao et al., 2023). Subsequent work has explored ways to dynamically expand an agent's tool repertoire, including the on-demand synthesis of reusable tools (Cai et al., 2023) and the reuse of existing software artifacts through frameworks such as ToolMaker (Wölflein et al., 2025). In parallel, researchers have proposed complementary training paradigms to enhance agentic capabilities, such as critique-guided planning (Chen et al., 2025b), selective supervision of reasoning processes (Yang et al., 2025b), and methods that decouple internal reasoning from output formatting (Chen et al., 2024).

2.3 Agentic Benchmarks & Data Generation

We further highlighted the deficiency when comparing existing works, as summarized in Table 1. While contemporary frameworks such as ToolLLM (Qin et al., 2023), StableToolBench (Guo et al., 2024), and Kimi-K2 (Team et al., 2025a) demonstrate robust capabilities in scalability, multi-step reasoning, and argument generation, they largely overlook the critical challenge of long-tail handling. Most existing benchmarks, including τ-bench (Yao et al., 2024), τ2-Bench (Barres et al., 2025), and ACEBench (Chen et al., 2025a), and methods like SFR-DeepResearch (Nguyen et al., 2025), focus on optimizing performance for head tools or specific domains, often neglecting the generalization capability required for handling the long tail. Furthermore, many approaches struggle to simulate real conversations effectively, often resorting to static tool chains that lack the dynamic, multi-turn nature of realistic human-agent interaction. As shown in the Table 1, there is a clear absence of a framework that simultaneously addresses the effective handling of long-tail tool usage. In our work, we provide a data construction pipeline to handle long-tail tool usage and achieve generalization on those tool uses with effective performance.

3 Preliminary Experiments

Before generating long-tail, multi-turn conversation data, we conducted a rigorous foundational analysis of the agentic capabilities and structural properties inherent in the BFCL (Patil et al.) and τ2 (Barres et al., 2025) benchmarks. This preliminary investigation was essential to ensure that our synthesized trajectories would not only address tool frequency but also align with the complex reasoning demands of high-performance agentic environments. We define the six agentic abilities and eight properties based on the analysis of the agentic datasets (Zhang et al., 2025; Prabhakar et al., 2025) and evaluation benchmarks (Patil et al.; Barres et al., 2025). Six agentic abilities are as follows: decision making, planning, tool choice, argument generation, result analysis, and error handling. We detailed the eight properties observation in Appendix A. Our findings indicated that for a training dataset to be truly effective, it must go beyond static tool chains and incorporate multi-turn aware logic that integrate these complex abilities and properties alogside rare tail-tool usage.

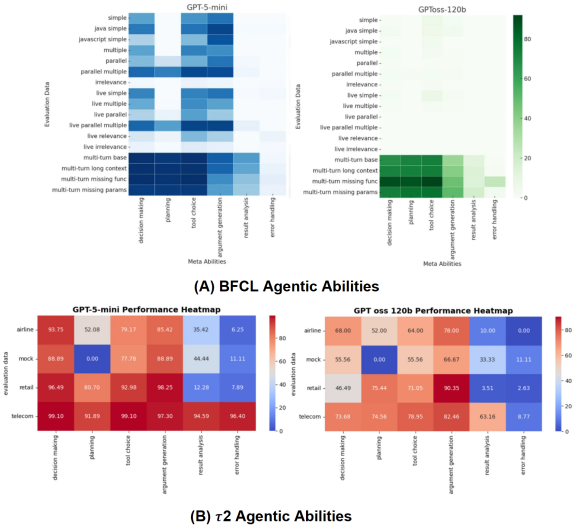

Figure 2: Agentic abilities of two benchmarks.

4 Reverse Walk: Long-Tail-Aware Sampling

The core challenge revealed in Figure 1 is that standard random-walk-based data generation overwhelmingly favors high-frequency tools (Team

et al., 2025a; Prabhakar et al., 2025). Even when walk lengths increase, forward exploration from head nodes rarely reaches tail nodes, leaving rare yet semantically important tools severely underrepresented. This imbalance is not addressed by existing tool-learning frameworks (Table 1), which typically rely on naturally occurring or head-biased synthetic tool chains.

To bridge this gap, we design a long-tail-aware dependency sampling framework that explicitly constructs and exploits the semantic dependency structure of APIs. Our objective is two-fold: (1) expose models to rare but valid multi-hop tool combinations, and (2) generate trajectories that reflect realistic task-solving processes rather than frequency-driven co-occurrence. We achieve this through three components: a semantic dependency graph, reverse random walks from tail nodes, and frequency-balanced sampling.

4.1 API Dependency Graph Construction

We first build a directed dependency graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that captures how tools can be composed to solve tasks. Each node $v_i \in \mathcal{V}$ represents a callable API or function, and edges $(v_i, v_j) \in \mathcal{E}$ represent semantically meaningful dependencies where the output or intent of v_i can serve as an input for v_j. In detail, we define edges (v_i, v_j) as directed from a prerequisite tool v_i to a dependent tool v_j, meaning v_i must precede v_j.

Unlike prior works that rely solely on call co-occurrence, we derive edges from semantic similarity between API descriptions. Specifically, we extract action verbs and purpose phrases from each API specification and connect tools whose semantic intents align (e.g., `search` → `fetch details` → `book`). This yields a semantically coherent graph that encodes compositional affordances. We also compute edge frequencies $f_{(v_i,v_j)}$ as the geometric mean of source and target node frequencies to capture **edge-level long-tail phenomena** rare but meaningful combinations of tools, scaled by semantic realism scores that penalize unrealistic combinations (e.g., `cancel` → `book`) and generic tools (e.g., `calculate`, `compute`). Further details are described in Appendix B.1.

4.2 Reverse Random Walk for Tail-Oriented Trajectories

Given this API graph, we perform reverse random walks over $\mathcal{G}$, starting from tail nodes (i.e., nodes with $f_i < \tau$) or tail edges (i.e., edges with $f_{(v_i,v_j)} < \tau_e$ when using edge-based sampling) and sampling predecessors along semantic dependencies. This contrasts with conventional forward random walks, which rarely traverse into tail regions.

At each step, we sample a predecessor node with probability proportional to the semantically normalized edge weight and inversely proportional to node frequency.

$$w_{(v_i,v_j)} = \left(1 - \frac{f_{(v_i,v_j)}}{f_{\max}} + \epsilon\right)^{\beta},$$

where β is the tail bias factor (typically 2.0–4.0) and $\epsilon = 0.01$ prevents division by zero. Higher β values more strongly favor tail edges, encouraging the walk to explore rare tool combinations. This ensures that (1) rare tools appear at the start of the chain, (2) the walk expands toward more general high-frequency tools, and (3) resulting trajectories reflect realistic multi-hop reasoning. The walk continues until reaching a head node (with $f_i \geq \tau$) or a root node (no predecessors), producing a dependency trajectory:

$$v_k \rightarrow v_{k-1} \rightarrow \cdots \rightarrow v_1,$$

which can be interpreted as a multi-step tool-calling plan.

To ensure validity, we apply several semantic filters: excluding overly generic utilities (e.g., `calculate`, `process`), enforcing temporal dependencies (e.g., no state-modifying operations after a `cancel` action), preventing repeated tools within the same trajectory, and ensuring all transitions reflect plausible operational orderings. (See more details in Appendix B.3)

As we mentioned above that we intend to use the reverse walk until reaching the head node. We try to build realistic conversations starting from the highly exposed tool to the generation model (in here we used GPT-oss-120b)[2]. We also reverse the order of dependency trajectories of the form:

$$\mathcal{T} = v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k,$$

naturally encoding how rare tools participate in multi-step solutions.

4.3 Frequency-Balanced Sampling

After observing the completed tool trajectories, we find that existing tools at the end of the trajectories

[2] https://huggingface.co/openai/gpt-oss-120b

are still commonly used. Since head tools dominate the frequency distribution, naïve sampling would largely reproduce head-heavy trajectories. To counter this, we adopt a frequency-based weighting strategy:

$$w_i \propto \frac{1}{f_i^{\alpha}}, \tag{1}$$

where $\alpha \in [0.5, 1.0]$ controls the strength of the long-tail bias. In our implementation, we use a normalized inverse weighting scheme $(1 - \text{normalized\_freq} + \epsilon)^{\gamma}$ that achieves the same goal of favoring tail nodes while maintaining numerical stability. The bias factor γ (typically 2.0 for nodes, 3.0 for edges) controls how aggressively we favor rare tools over frequent ones.

Critically, sampling always begins from tail nodes or tail edges, ensuring that each trajectory includes at least one rare tool by construction. This is more effective than post-hoc stratified filtering, which struggles under highly skewed long-tail distributions.

4.4 Complexity-Aware Validation

To maintain both realism and executability, each generated trajectory undergoes structural and argument-level validation. We restrict the maximum reasoning depth $d_{\max}$, verify that each tool call satisfies its schema, enforce cross-step consistency so that outputs of earlier tools satisfy prerequisites of later ones, and apply temporal logic constraints such as prohibiting state changes after deletion actions. Although simulator-based repair can be used, we adopt strict filtering to ensure that all retained trajectories exhibit clean, logically consistent structures.

5 Multi-turn Conversation Generation

Given the tool trajectories $\mathcal{T}$ generated through reverse random walks, we synthesize realistic multi-turn conversations that reflect how users and assistants interact in practice. Unlike static tool chains that merely enumerate API calls (Prabhakar et al., 2025; Team et al., 2025a), our approach generates dynamic, context-aware dialogues where user requests emerge naturally and assistant responses involve multi-step tool reasoning similar pipeline (Cho et al., 2026). We initiate conversation generation from the start node of each trajectory (i.e., the most frequently used tool), as these head tools are more familiar to the generation model and serve as natural entry points for realistic task formulation.

Dataset	Step	Turn	Task	Samples
Nemotron-based	3.65 (119)	3.28 (195)	2.52 (18)	15,287
Tau2-based	3.15 (58)	4.85 (218)	2.98 (21)	988

Table 2: Statistics of the Nemotron-based and Tau2-based tool conversation datasets. For each metric (Step, Turn, Task), we provide the average value with the maximum value provided in parentheses.

Our conversation generation pipeline operates through an iterative user-assistant interaction loop. First, we generate task descriptions that align with the tool trajectory $\mathcal{T}$, encoding the semantic dependencies as natural user goals ranging from simple single-tool queries to complex multi-step workflows. The user simulation module then generates realistic user turns by modeling human behavior: users ask for help naturally, provide context incrementally, and request follow-up actions based on previous assistant responses. The assistant response generation module, in turn, performs multi-step reasoning by calling tools sequentially, where each tool call may depend on outputs from previous calls. To ensure realism, we employ a tool simulator that generates plausible tool outputs (including occasional errors with probability p_{error}) based on tool specifications and call arguments, maintaining temporal consistency throughout the conversation.

The iterative process continues until the user indicates task completion, naturally producing conversations of varying lengths and complexities. This approach ensures that rare tools embedded in the trajectory are naturally integrated into realistic multi-turn dialogues, exposing models to both the compositional structure of tool dependencies and the dynamic nature of human-agent interaction. By starting from head nodes and expanding toward tail tools through natural conversation flow, we bridge the gap between frequency-biased tool exposure and long-tail generalization requirements.

6 Experiments

6.1 Experimental Details

Training & Inference Settings In Table 2, we provide the statistics of generated datasets. We generate the tool trajectories based on the seed tools provided by Nemotron post-training dataset (Nathawani et al., 2025) and $\tau 2$ database dataset (Barres et al., 2025). To provide hyperparameters of training setting, we use 1e-6 learning

Model	BFCL						Tau2			
	Memory	Multi-turn	Live	Non-Live	Hall. (Rel)	Hall. (Irrel)	Airline	Retail	Telecom	Telecom-Workflow
Claude Sonnet 4.5 (fc)	39.14	60.88	81.05	88.56	68.75	86.32	70.0	86.2	98.0	-
GPT-5.1-mini	44.09	27.38	58.55	70.19	68.75	91.76	-	-	-	-
GPT-5.1	57.63	36.12	58.99	73.04	68.75	91.35	77.9	-	95.6	-
Gemini-2.5-Pro	36.77	29.25	63.8	85.58	43.75	91.46	-	-	-	-
Gemini-3.0-Pro	-	-	-	-	-	-	73.0	85.3	85.4	-
xLAM-2-3b-fc-r†	13.33	56.00	58.69	82.94	94.44	57.94	32.0	44.4	-	-
xLAM-2-8b-fc-r†	15.7	69.25	66.73	84.35	83.33	64.11	35.2	58.2	-	-
xLAM-2-32b-fc-r†	16.77	66.38	73.79	89.50	83.33	76.25	45.0	64.3	-	-
xLAM-2-70b-fc-r†	17.63	75.12	72.63	88.48	66.67	78.74	45.2	67.1	-	-
Qwen3-4B-Thinking-2507†	19.35	48.1	82.9	86.3	100.0	78.9	58.0 (46.0)	53.5 (56.1)	27.2 (21.1)	- (21.9)
Qwen3-30B-A3B-Thinking-2507†	20.43	53.8	84.1	89.6	100.0	80.6	58.0 (56.0)	58.8 (54.4)	26.3 (22.8)	- (6.14)
Qwen3-235B-A22B-Thinking-2507†	29.25	50.12	83.35	90.17	93.75	82.26	50.0 (44.0)	74.6 (57.9)	32.5 (21.9)	- (22.8)
Qwen3-4B-Thinking-2507 + APIGEN	21.51	13.38	76.98	86.06	68.75	62.9	44.0	54.4	27.2	22.81
Qwen3-30B-A3B-Thinking-2507 + APIGEN	23.66	20.88	77.94	84.56	75.0	56.34	48.0	57.0	32.5	36.0
Qwen3-235B-A22B-Thinking-2507 + APIGEN	27.96	49.12	81.57	88.38	81.25	78.1	48.0	74.6	39.5	51.8
Qwen3-4B-Thinking-2507 + OURS	22.15	17.5	78.53	84.75	81.25	78.1	44.0	56.1	30.7	27.2
Qwen3-30B-A3B-Thinking-2507 + OURS	27.75	34.25	80.9	84.52	87.5	77.69	46.0	60.5	39.5	41.2
Qwen3-235B-A22B-Thinking-2507 + OURS	34.41	52.50	83.64	88.56	81.25	82.93	52.0	77.2	51.8	56.1

Table 3: Agentic benchmark results. We provide the proprietary models (row 1-5), open-source function call models (row 6-9), baselines (row 10-12), and trained models with APIGEN and OURS which could be divided into forward and reverse tool trajectories while generating multi-turn conversation (row 13-18). The † replies the reported score from technical report, model card, and leaderboard. We observe a significant performance discrepancy in the baseline models (row 10–12) between the officially reported scores (†) and our re-evaluated scores (in parentheses).

rate with max sequence length of 64k[3]. We train the models for 5 epochs and choose the best performance epoch. We also perform full-finetuning using DeepSpeed ZeRO (Rasley et al., 2020) stage 3, Flash Attention 2 (Dao, 2023) in bfloat16 precision with AdamW optimizer (Loshchilov and Hutter, 2017). In our previous observations, higher learning rates make model to explode and lower sequence length provide timeout request error while evaluating agentic benchmarks. We use vLLM (Kwon et al., 2023) to serve the trained models with provided high quality hyperparameters.

We use three qwen-family reasoning models: Qwen3-4B-Thinking-2507, Qwen3-30B-A3B-Thinking-2507, and Qwen3-235B-A22B-Thinking-2507 (Yang et al., 2025a). In our experiments, we use two agentic benchmarks $\tau 2$ (Barres et al., 2025) and BFCL (Patil et al.). The $\tau 2$ benchmark comprises 5 realistic domains: Airline (50), Retail (114), Telecom (114), Telecom-Workflow (114), and Mock (9). We report a pass^1 result due to the high cost of API usage. We did not use the Mock domain because of its instability in its results. Also, the Gorilla BFCL V4 benchmark evaluates LLM function calling across 5,088 samples in categories weighted by importance. Key evaluation areas include Multi-turn (800) for conversational context and Agentic (665) for memory and web search tasks. We did not assess the web search part due to the failure case of using search API. It also assesses model accuracy in both Real-world Live (1,351) and standard Non-live (1,150) function calls, along with Hallucination (1,122) measurement to check for proper abstention.

[3]We explored the learning rate and maximum context length and detailed explorations in Appendix E

6.2 Experimental Results

Table 3 summarizes the agentic benchmark results for proprietary models, open-source function call models, and our trained models on the BFCL (Patil et al.) and $\tau 2$ (Barres et al., 2025) datasets. The models trained with APIGEN and OURS employ two distinct tool-use trajectory strategies: forward and reverse multi-turn conversation, respectively.

Training the baseline Qwen3 models with multi-turn conversation trajectories (APIGEN or OURS) yields dramatic performance gains on the $\tau 2$ benchmarks compared to the re-evaluated baseline scores. Notably, models trained with OURS, which utilizes a reverse-traversing tool trajectories, consistently show superior effectiveness in capturing complex conversational dynamics. For instance, the Qwen3-235B model, after training with OURS, achieves an F1 score of 77.2 in $\tau 2$ Retail, representing a substantial 19.3 point increase over its re-evaluated baseline score of 57.9. The most critical improvement is observed in the Qwen3-30B model on $\tau 2$ Telecom, where the baseline score of 22.8 is boosted to 39.5 by OURS, demonstrating a 16.7 point surge. This data highlights that while trajectory training is vital, adopting the reverse trajectory approach is significantly more effective at unlocking robust multi-turn tool usage capabilities, especially across challenging domains like Telecom and

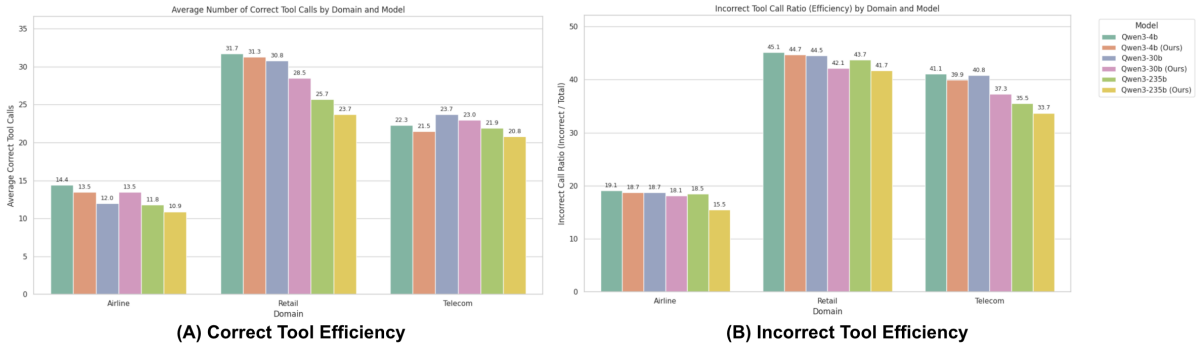

Figure 3: Average tool call efficiency by $\tau 2$ domains and models. These figures compare the average number of (A) Correct and (B) Incorrect tool usages across three $\tau 2$ domains. Models trained with our method (OURS) demonstrate improved efficiency, generally exhibiting a lower average number of incorrect calls while maintaining competitive correct tool usage, particularly in the complex Retail and Telecom domain.

Retail.

7 Analysis

7.1 Efficiency of tool usage

This subsection likely examines how efficiently the models utilize the available tools, particularly comparing the baseline models and those trained with the proposed reverse random walk (OURS) approach. The efficiency can be measured by comparing the average number of correct and incorrect tool usages across different domains, such as those in the $\tau 2$ benchmark (Airline, Retail, Telecom).

The provided Figure 3, labeled as (A) Correct Tool Efficiency and (B) Incorrect Tool Efficiency, visually supports this analysis. A robust model would be expected to demonstrate a high average number of correct tool calls and a low average number of incorrect tool calls. For the $\tau 2$ Retail domain, the Qwen3-235B model trained with OURS demonstrates superior performance, achieving a low average number of correct tool calls (23.7) and incorrect tool calls (41.7).

7.2 Effectiveness on tail tool usage

This subsection focuses on the primary challenge addressed by the paper: the generalization capability of models on rarely used, or "long-tail" tools. Traditional forward-generation methods struggle to create meaningful trajectories for these low-frequency tools. Our reverse random walk strategy is specifically designed to generate multi-hop tool chains that naturally incorporate rare but meaningful tool combinations, forcing the model to learn semantic relationships over frequency-based patterns. The effectiveness of this approach is demonstrated by assessing the accuracy of tool usage specifically

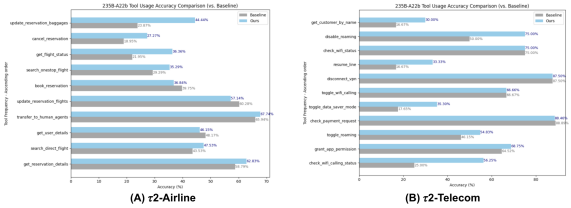

Figure 4: Tail tool accuracy comparison on $\tau 2$ Airline and Telecom Domains. This visualization focuses on the generalization capability for the bottom ten infrequent tools (long-tail tools). Training on the dataset generated by our reverse walk strategy (OURS) significantly enhances the proper usage and accuracy of these low-frequency tools compared to baselines.

for these long-tail tools. Figure 4, which provides $\tau 2$-Airline and $\tau 2$-Telecom tail tool accuracies under the bottom ten infrequent tool usages, is the core visual evidence for this analysis, showing that training on the dataset generated by the reverse walk (OURS) enhances the proper usage of these low-frequency tools.

7.3 Generalization of Tool distribution

We evaluate how well our model generalizes to tool distributions—tools that were either infrequent in the training set or appear during the evaluation phase. We compared the agentic performance of the model trained with our reverse walk (OURS) strategy against the toolsets used in the $\tau 2$-Airline and -Telecom domains of the evaluation benchmarks.

In the Telecom domain showed exceptional precision in state-control tools such as *check wifi status* (75.00%), proving that our reverse walk strategy effectively trains the model on diverse state-transition logic. Furthermore, stable results across tools like *get customer by phone* (56.76%) indicate that the

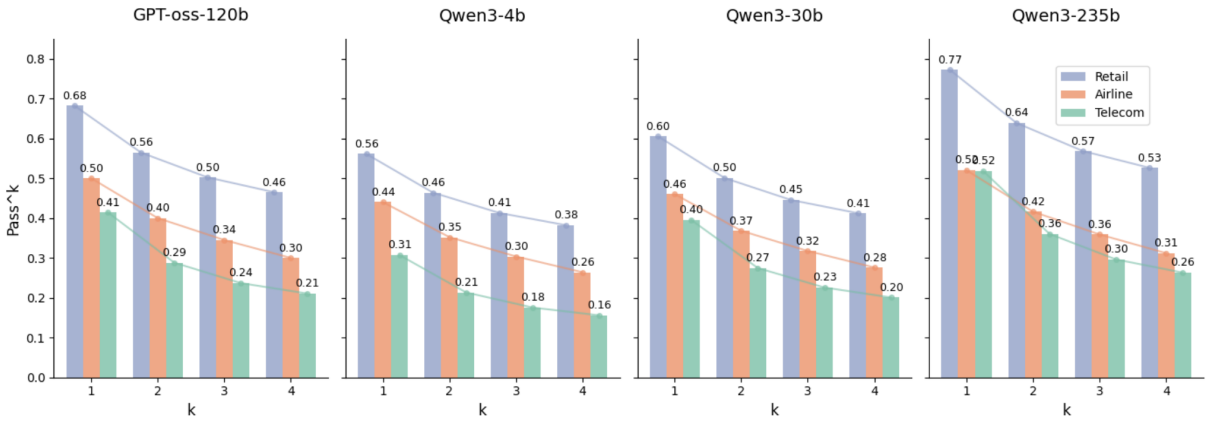

Figure 5: Average tool call consistency by $\tau 2$ domains and models.

model successfully generalizes by leveraging semantic tool descriptions rather than relying on frequency patterns. We want to note that by forcing the model to learn the underlying dependencies of rare tools, it enables the agent to navigate the tool action space effectively, even when encountering infrequent APIs in the evaluation benchmark. We further provide the detailed tool usage of $\tau 2$-Airline in Appendix D.

Tool Name	Correct	Incorrect	Total	Accuracy (%)
reboot device	59	96	155	38.06%
run speed test	81	69	150	54.00%
get customer by phone	63	48	111	56.76%
check network mode preference	56	46	102	54.90%
get data usage	37	43	80	46.25%
check sim status	25	55	80	31.25%
set apn settings	12	66	78	15.38%
send payment request	28	21	49	57.14%
make payment	23	20	43	53.49%
reseat sim card	16	27	43	37.21%
check installed apps	9	32	41	21.95%
check wifi calling status	18	14	32	56.25%
grant app permission	22	10	32	68.75%
toggle roaming	17	14	31	54.83%
toggle data saver mode	6	11	17	35.29%
toggle wifi calling	12	6	18	66.66%
resume line	4	8	12	33.33%
check wifi status	6	2	8	75.00%
get customer by name	2	4	6	30.00%
disable roaming	3	1	4	75.00%

Table 4: Tool Usage Comparison - Telecom Domain (Simulation)

7.4 Tool Consistency using pass^k

To evaluate the reliability of model-generated tool calls, we measure the average consistency across the Retail, Airline, and Telecom domains using the Pass^k metric. As illustrated in Figure 5, model performance scales positively with parameter size, where Qwen3-235b achieves the highest overall consistency (e.g., 0.77 in Retail at $k = 1$) compared to its smaller variants and GPT-oss-120b. Across all evaluated models, a consistent performance hierarchy emerges among domains, with Retail showing the highest stability and Telecom presenting the most significant challenge. Furthermore, the observed decline in Pass^k values as k increases suggests that while larger models provide a higher baseline for accuracy, maintaining consistent tool-calling logic across multiple iterations remains a complex task influenced heavily by domain-specific requirements.

8 Conclusion

This paper introduces a long-tail-aware sampling framework to enhance the generalization of tool-learning models by addressing the common bias toward high-frequency tool usage. We propose a reverse random walk strategy, starting from rare tail nodes/edges and moving toward head nodes along a semantically derived API Dependency Graph, which naturally compels the generation of trajectories featuring rare tool usage and complex multi-hop combinations. This is combined with a frequency-balanced sampling to mitigate the over-representation of frequent steps and the complexity-aware validation to ensure the generated data's realism and executability. Empirical results on agentic benchmarks like $\tau 2$ and BFCL confirm that training on the resulting dataset significantly improves robust multi-tool usage and generalization across diverse conversational scenarios, demonstrating the efficacy of our method for building highly capable tool-augmented reasoning models.

Limitations

While the framework successfully improves performance on tail tools, there is a risk that the model is simply memorizing the synthetic patterns generated

by the reverse walk rather than learning generalized reasoning. Since the open-sourced GPT-OSS-120b model is used to generate the dialogues, the trained models might be inheriting the specific linguistic and logical biases of the teacher model.

We also want to state that we did not assess the web search part of the BFCL benchmark due to API failure cases. Furthermore, many of the results are based on simulated tool outputs. The "long-tail" tools in the real world often involve unpredictable environmental states or complex authentication flows that simulations might oversimplify. The gap between "simulated accuracy" (e.g., Table 5) and "real-world deployment" remains a critical limitation for agentic frameworks.

Acknowledgments

References

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2023. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, and 1 others. 2025a. Acebench: Who wins the match point in tool usage? *arXiv preprint arXiv:2501.12851*.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024. Agent-flan: Designing data and methods of effective agent tuning for large language models. *arXiv preprint arXiv:2403.12881*.

Zhixun Chen, Ming Li, Yuxuan Huang, Yali Du, Meng Fang, and Tianyi Zhou. 2025b. Atlas: Agent tuning via learning critical steps. *arXiv preprint arXiv:2503.02197*.

Jungho Cho, Minbyul Jeong, and Sungrae Park. 2026. User-oriented multi-turn dialogue generation with tool use at scale. *arXiv preprint arXiv:*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Runnan Fang, Shihao Cai, Baixuan Li, Jialong Wu, Guangyu Li, Wenbiao Yin, Xinyu Wang, Xiaobin Wang, Liangcai Su, Zhen Zhang, and 1 others. 2025. Towards general agentic intelligence via environment scaling. *arXiv preprint arXiv:2509.13311*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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. *arXiv preprint arXiv:2403.07714*.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems*, 36.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

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*.

Gyubok Lee, Elea Bach, Eric Yang, Tom Pollard, Alistair Johnson, Edward Choi, Jong Ha Lee, and 1 others. 2025. Fhir-agentbench: Benchmarking llm agents for realistic interoperable ehr question answering. *arXiv preprint arXiv:2509.19319*.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. 2025. In-the-flow agentic system optimization for effective planning and tool use. *arXiv preprint arXiv:2510.05592*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Zhiwei Liu, Jielin Qiu, Shiyu Wang, Jianguo Zhang, Zuxin Liu, Roshan Ram, Haolin Chen, Weiran Yao, Shelby Heinecke, Silvio Savarese, and 1 others. 2025. Mcpeval: Automatic mcp-based deep evaluation for ai agent models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 373–402.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Siyuan Lu, Zechuan Wang, Hongxuan Zhang, Qintong Wu, Leilei Gan, Chenyi Zhuang, Jinjie Gu, and Tao Lin. 2025. Don't just fine-tune the agent, tune the environment. *arXiv preprint arXiv:2510.10197*.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu,

Asli Celikyilmaz, and 1 others. 2023a. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023b. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Dhruv Nathawani, Igor Gitman, Somshubra Majumdar, Evelina Bakhturina, Ameya Sunil Mahabaleshwarkar, , Jian Zhang, and Jane Polak Scowcroft. 2025. Nemotron-Post-Training-Dataset-v1.

Xuan-Phi Nguyen, Shrey Pandit, Revanth Gangi Reddy, Austin Xu, Silvio Savarese, Caiming Xiong, and Shafiq Joty. 2025. Sfr-deepresearch: Towards effective reinforcement learning for autonomously reasoning single agents. *arXiv preprint arXiv:2509.06283*.

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

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

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, and 1 others. 2025. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *arXiv preprint arXiv:2504.03601*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*.

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. *Advances in Neural Information Processing Systems*, 36.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025a. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Meituan LongCat Team, Bei Li, Bingye Lei, Bo Wang, Bolin Rong, Chao Wang, Chao Zhang, Chen Gao, Chen Zhang, Cheng Sun, and 1 others. 2025b. Longcat-flash technical report. *arXiv preprint arXiv:2509.01322*.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, and 1 others. 2025c. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*.

Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelovic, and Jakob Nikolas Kather. 2025. Llm agents making agent tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26092–26130.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025a. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, and 1 others. 2025b. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*.

Ran Xu, Yuchen Zhuang, Yishan Zhong, Yue Yu, Xiangru Tang, Hang Wu, May Dongmei Wang, Peifeng Ruan, Donghan Yang, Tao Wang, and 1 others. 2025. Medagentgym: Training llm agents for code-based medical reasoning at scale. In *The Second Workshop on GenAI for Health: Potential, Trust, and Policy Compliance*.

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

Ruihan Yang, Fanghua Ye, Jian Li, Siyu Yuan, Yikai Zhang, Zhaopeng Tu, Xiaolong Li, and Deqing Yang. 2025b. The lighthouse of language: Enhancing llm agents via critique-guided improvement. *arXiv preprint arXiv:2503.16024*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, and 1 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*.

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. 2025. Nemotron-research-tool-n1: Tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*.

A Background Observations

In Table 5, we define eight distinct agentic properties—such as Conditional Tool Invocation, File System State Tracking, and State Consistency Checking—to serve as qualitative benchmarks for our data. To assess these criteria, we use an LLM-as-a-judge (Zheng et al., 2023), scoring each property on a scale of 1 to 100. In Figure 2 and 6, our findings indicated that for a training dataset to be truly effective, it must go beyond static tool chains and incorporate multi-turn aware conditional logic and environmental state tracking. Consequently, our generation pipeline was specifically designed to produce multi-turn dialogues that integrate these complex properties alongside rare "tail-tool" API usage.

B Long-tail Data Generation

B.1 Motivation: Limitations of Node-Based Approach

Our initial implementation focused on **node-based long-tail phenomena**, where we identified tail nodes (tools with frequency $f_i < \tau$) and performed reverse random walks starting from these nodes. While this approach successfully generated chains containing rare tools, we observed several limitations:

- **Repetitive tool combinations**: The random walk often converged to common tool pairs (e.g., `getReservationDetails` → `updateReservationFlights`), even when starting from tail nodes. This occurred because edge weights were proportional to node frequencies, causing the walk to quickly transition to high-frequency tools.

- **Weak compositional reasoning**: The generated chains emphasized individual rare tools but failed to capture *rare but meaningful combinations* of tools. For instance, a chain might include a rare tool `sendCertificate`, but it would typically be connected to common tools like `getUserDetails`, rather than forming novel compositional patterns.

- **Insufficient edge diversity**: Even with inverse weighting (favoring tail nodes), the actual *dependencies* between tools remained dominated by frequent edges, limiting the diversity of multi-hop reasoning patterns.

B.2 Edge-Based Long-Tail: A Compositional Perspective

To address these limitations, we transitioned to an **edge-based long-tail** approach that emphasizes rare tool combinations rather than merely rare individual tools. This shift is motivated by the observation that compositional reasoning requires models to learn not just which tools exist, but how tools can be meaningfully combined.

Edge Frequency Computation. We compute edge frequencies $f_{(v_i,v_j)}$ as:

$$f_{(v_i,v_j)} = \sqrt{f_i \cdot f_j} \cdot s_{\text{realism}}(v_i, v_j),$$

where $s_{\text{realism}}(v_i, v_j) \in [0, 1]$ is a semantic realism score that penalizes unrealistic combinations. The geometric mean captures the joint probability of using both tools together, while the realism score ensures that semantically incoherent edges (e.g., `cancelReservation` → `bookReservation`) receive very low frequencies regardless of individual node frequencies.

Semantic Realism Scoring. The realism score s_{realism} combines three factors:

$$s_{\text{realism}}(v_i, v_j) = 0.3 \cdot s_{\text{domain}} + 0.5 \cdot s_{\text{action}} + 0.2 \cdot s_{\text{pattern}}, \quad (2)$$

where:

Label	Name	Definition
(A)	**Complex Policy Presence**	Whether the conversation contains or implies a complex, multi-condition policy that governs how the agent should behave or make decisions.
(B)	**User Information Verification**	Whether the conversation involves the agent explicitly checking, confirming, or requesting user-related information (e.g., name, status, role, identity, preferences) before proceeding.
(C)	**User-Dependent Policy Variation**	Whether the conversation shows that different policies or behaviors are applied depending on user type, grade, or classification.
(D)	**Excessive Step Decomposition**	Whether the solving process is overly decomposed into too many sub-steps, indicating unnecessary procedural granularity.
(E)	**Incorrect Tool Execution Sequence**	Whether the conversation demonstrates incorrect or logically inconsistent tool usage, such as invoking tools in the wrong order or with wrong arguments.
(F)	**Conditional Tool Invocation**	Whether the conversation involves conditional decisions about whether or not to use a tool, based on state, input, or environment.
(G)	**File System State Tracking**	Whether the conversation involves monitoring, updating, or reasoning about file system states (e.g., file creation, deletion, modification).
(H)	**State Consistency Checking**	Whether the conversation explicitly verifies or ensures consistency between stored states, memory, or environment variables during execution.

Table 5: Definition of the eight agentic properties.

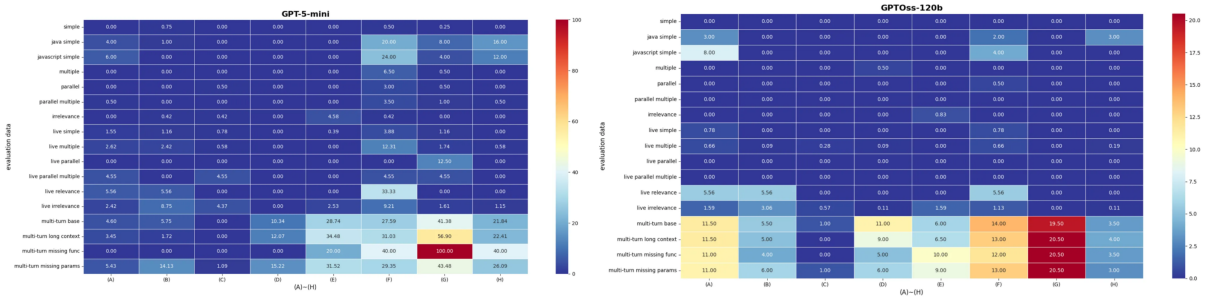

Figure 6: Agentic properties of two benchmarks using GPT-5-mini and GPT-OSS-120b as a judger.

- s_{domain}: Domain consistency (1.0 for same domain, 0.7 for cross-domain)
- s_{action}: Temporal logic based on action types (e.g., `search` $\rightarrow$ `book` = 1.0, `cancel` $\rightarrow$ `update` = 0.01)
- s_{pattern}: Tool name pattern matching (same entity operations score higher)

This scoring mechanism ensures that even if two tools individually have moderate frequencies, their combination receives a low edge frequency if it is semantically unrealistic, effectively filtering out nonsensical dependencies during graph construction.

B.3 Implementation Details

Generic Tool Filtering. We identify and exclude generic tools (e.g., `calculate`, `compute`, `process`) that lack specific semantic meaning. These tools are filtered at two stages:

1. **Graph construction**: Generic tools are excluded as source nodes in dependency edges, preventing them from creating spurious connections.
2. **Random walk**: During chain generation, generic tools are filtered from predecessor candidates, ensuring only semantically meaningful tools appear in the final chains.

Temporal Logic Constraints. To prevent unrealistic sequences, we enforce temporal logic rules:

- After `cancel` or `delete` actions, only `read` and `search` operations are allowed. This prevents chains like `cancelReservation` $\rightarrow$ `updateReservationFlights`, which would be semantically invalid.
- Action type classification enables pattern-based validation: `search` $\rightarrow$ `create` sequences receive high scores, while `delete` $\rightarrow$ `create` sequences are heavily penalized.

B.4 Empirical Observations

Our transition to edge-based long-tail yielded several improvements:

- **Increased compositional diversity**: Chains now contain more diverse tool combinations, such as `searchOnestopFlight` $\rightarrow$ `sendCertificate` $\rightarrow$ `updatePassengerDetails`, which represent novel multi-hop reasoning patterns.
- **Reduced unrealistic sequences**: Semantic validation eliminated problematic patterns like `cancel` $\rightarrow$ `update` or `calculate` $\rightarrow$ `bookReservation`, improving the quality of generated tasks.
- **Better long-tail coverage**: Edge-based sampling ensures that even when individual tools have moderate frequencies, their rare combinations are properly emphasized, leading to more challenging compositional reasoning tasks.

B.5 Parameter Selection

For edge-based long-tail, we use the following parameter settings:

- $\tau_e = 0.0001$: Edge frequency threshold for identifying tail edges. This lower threshold (compared to node-based $\tau = 0.01$) reflects that edge frequencies are typically lower than node frequencies due to the geometric mean computation.
- $\beta_e = 3.0$: Edge tail bias factor, higher than node-based $\beta = 2.0$ to more aggressively favor rare edges.
- Semantic validation threshold: $s_{\text{realism}} \geq 0.35$ to filter out unrealistic edges while allowing some flexibility for cross-domain combinations.

These parameters were selected through iterative refinement to balance between generating sufficiently diverse chains and maintaining semantic coherence.

C Multi-Turn Conversation Data Generation Details

The dataset is constructed using an iterative, multi-stage pipeline designed to simulate complex, multi-turn tool-use interactions. Each conversation consists of four primary roles: system instructions, user queries, assistant responses (including reasoning and tool calls), and tool execution results.

C.1 Initial Task and Response Generation

The process begins with the Task Generation Pipeline, which utilizes tool specifications to create realistic user instructions. Each task is assigned a complexity level (easy, medium, or hard) and a detailed evaluation rubric that defines success criteria, expected tool-use patterns, and checkpoints. Once a task is defined, the Response Generation Pipeline produces the assistant's turns. This stage is iterative: the model generates reasoning and tool calls, the pipeline simulates tool execution (introducing a probabilistic tool error probability to model real-world failures), and the results are fed back into the conversation until the assistant provides a final answer.

C.2 Validity and Self-correction

To ensure high data quality, every generated interaction undergoes LLM-based validation. The validator compares the assistant's entire reasoning and tool-use process against the predefined rubric and tool specifications. If the response fails to meet the success criteria or deviates from the expected pattern, the conversation is reverted to the last user turn and sent back for regeneration. This retry mechanism creates a self-correcting loop that filters out hallucinations or suboptimal tool usage.

C.3 Continual Turn Extension

Following a successful validation, the pipeline determines whether to extend the conversation. Based on a continuation probability and a maximum turn limit, the LLM generates a follow-up user turn. This new turn is designed to follow the conversation context naturally—often depending on previous tool outputs—while maintaining the original complexity level. Each new turn includes its own unique rubric, and the cycle of response generation and validation repeats.

C.4 Key design philosophies

The pipeline incorporates several features to enhance model robustness: (1) Complexity Preservation: Ensuring all turns in a single session adhere to the initial difficulty setting. (2) Error Simulation: Training the model to recover from tool failures by injecting realistic errors into the simulation. (3) Dynamic Variation: Using probabilistic continuation to ensure a diverse mix of short-form and long-form multi-turn dialogues. The final output is a collection of comprehensive JSONL objects, where each entry contains the full conversation history and metadata, providing a rich training signal for complex context management and tool interaction patterns.

D Overall Tool Usages

Tool Name	Correct	Incorrect	Total	Accuracy (%)
get reservation details	71	42	113	62.83%
search direct flight	49	54	103	47.5%
get user details	18	21	39	46.15%
transfer to human agents	21	10	31	67.74%
update reservation flights	12	9	21	57.14%
book reservation	7	12	19	36.84%
search onestop flight	6	11	17	35.29%
get flight status	4	7	11	36.36%
cancel reservation	3	8	11	27.27%
update reservation baggages	4	5	9	44.44%

Table 6: Tool Usage Comparison - Airline Domain (Simulation)

E Exploration of Hyperparameters

To ensure the robustness and optimal performance of our proposed long-tail-aware sampling framework, we conducted detailed ablation studies focusing on two critical training hyperparameters: the maximum context length and the learning rate.

Increasing Maximum Context Length. The ability to handle long, multi-turn conversations and complex tool trajectories is directly dependent on the model's maximum sequence length. Our initial observations indicated that an insufficient context length (less than 32k) often resulted in timeout request errors (approx. $16 \sim 22\%$) during the evaluation of agentic benchmarks.

We anchor our experiments on a sequence length of 64k and compare performance against shorter (e.g., 8k, 16k, and 32k) settings. This exploration is crucial for verifying that the model can effectively leverage the full compositional structure encoded by our reverse random walks. We primarily analyze the impact of context length on metrics sensitive to conversational depth, such as the $\tau 2$ Telecom-Workflow accuracy and the BFCL Multi-turn F1 score. A substantial increase in performance on these metrics with greater context length would confirm that the generated multi-turn, multi-step trajectories are being fully utilized by the model.

Learning Rate. The learning rate is a critical factor for achieving stable and effective training, especially when dealing with skewed data distributions like the long tail of tool usage. We previously observed that higher learning rates lead the model to explode. Thus, we set our baseline learning rate conservatively at 1e-6 for 5 epochs of full-finetuning using AdamW optimizer (Loshchilov and Hutter, 2017).

We conduct experiments by varying the learning rate around the baseline (e.g., 1e-7, 5e-7, 5e-6, 1e-5) to determine the optimal balance between convergence speed and stability. The primary goal is to find the maximum effective learning rate that preserves or enhances the model's ability to generalize to rare, long-tail tool usages. We assess this by monitoring the overall performance on the $\tau 2$-Airline and $\tau 2$-Telecom domains (as highlighted in Figure 3), alongside the overall BFCL scores and the stability of the training loss curve. This analysis will validate our conservative learning rate choice and demonstrate its efficacy in navigating the unique challenges posed by long-tail tool learning.

F Scientific Artifacts Usage

To ensure the highest level of linguistic quality and technical clarity, generative AI tools—including Gemini and ChatGPT—were utilized during the preparation of this manuscript. Specifically, these models were employed for grammatical refinement, improving prose fluency, and polishing technical descriptions within the text. The use of these tools was strictly limited to language enhancement and editorial support. All core scientific ideas, data generation logic, experimental results, and interpretations remain the original work of the authors. The final manuscript was thoroughly reviewed and edited by the authors to ensure accuracy, integrity, and adherence to academic standards.

Preliminary (Abilities)

You are an expert evaluator trained to analyze Python dictionary–formatted samples involving an agent performing reasoning and tool-using tasks. Your goal is to assess how strongly the given sample provides evidence that the data could help train or evaluate the model's agentic abilities. Each sample includes structured keys such as "function", "question", "context", "response", "arguments", or "output". You should carefully examine all fields and their relationships to determine whether the data meaningfully demonstrates or could supervise each ability.

Scoring

For each ability, assign a score between 1 and 100, where:
1 → The ability is essentially absent or only trivially implied.
25 → Weak or ambiguous learning signal.
50 → Moderate, partial supervision signal.
75 → Clear and strong evidence supporting this ability.
100 → Explicit, central, and unambiguous supervision signal.
If the aspect is not meaningfully present, assign a score close to 1 rather than 0.

Criteria and Definitions

1. **Decision Making**: Degree to which the sample provides supervision for deciding the next optimal action (e.g., whether to plan, invoke a tool, ask for clarification, or directly answer).
2. **Planning**: Extent to which the sample demonstrates decomposition of a high-level task into ordered sub-tasks or structured steps.
3. **Tool Choice**: Strength of evidence that the sample teaches selecting an appropriate tool from an available toolset.
4. **Argument Generation**: Degree to which the sample provides supervision for generating correct and contextually appropriate arguments (parameters or inputs) for a selected tool.
5. **Result Analysis**: Extent to which the sample demonstrates interpreting tool outputs and deciding whether the task is complete or further action is required.
6. **Error Handling**: Strength of evidence that the sample teaches recognition of errors, unexpected outputs, or failed executions and appropriate responses to them.

Required Output Format

```
{
  "decision_making": 1-100,
  "planning": 1-100,
  "tool_choice": 1-100,
  "argument_generation": 1-100,
  "result_analysis": 1-100,
  "error_handling": 1-100,
  "reasoning": "Brief 1-2 sentence explanation highlighting which abilities are strongly
  or weakly supported."
}
```

Preliminary (Properties)

You are an expert evaluator analyzing Python dictionary–formatted samples that describe an agent's reasoning, tool use, or policy-driven behavior. Your goal is to determine how strongly the sample demonstrates or implies specific meta-structural elements, such as complex policies, user-dependent variations, or state management logic.
Each sample may include fields such as "function", "question", "context", "response", or "arguments".

Scoring

For each criterion (A–H), assign a score between 1 and 100, where:
1 → The aspect is essentially absent or only trivially hinted.
25 → Weak or implicit signal; ambiguous relevance.
50 → Moderately present; partially demonstrated.
75 → Clearly present with strong evidence.
100 → Explicit, central, and unambiguous presence.
If the aspect is not meaningfully present, assign a score close to 1 rather than 0.

Criteria and Definitions

(A) **Complex Policy Presence** Degree to which the sample defines or implies a multi-condition, non-trivial policy governing agent behavior.
(B) **User Information Verification** Extent to which the sample checks, validates, or requests user-related information before proceeding.
(C) **User-Dependent Policy Variation** Strength of evidence that rules or actions vary depending on user role, identity, or attributes.
(D) **Excessive Step Decomposition** Degree to which the reasoning or task flow is broken into unnecessarily granular or redundant steps.
(E) **Incorrect Tool Execution Sequence** Severity of illogical, inefficient, or incorrect ordering of tool calls.
(F) **Conditional Tool Invocation** Strength of conditional logic determining whether, when, or how tools are invoked.
(G) **File System State Tracking** Extent of reasoning about file system state changes (creation, deletion, updates, persistence).
(H) **State Consistency Checking** Degree to which the sample explicitly verifies consistency across states, memory, or stored artifacts.

Required Output Format

```
{
  "A_complex_policy_presence": 1-100,
  "B_user_information_verification": 1-100,
  "C_user_dependent_policy_variation": 1-100,
  "D_excessive_step_decomposition": 1-100,
  "E_incorrect_tool_execution_sequence": 1-100,
  "F_conditional_tool_invocation": 1-100,
  "G_file_system_state_tracking": 1-100,
  "H_state_consistency_checking": 1-100,
  "reasoning": "Brief 1-2 sentence explanation highlighting the strongest and weakest dimensions."
}
```