Training Verifier to Assessing Complex Real-World Tool-Use Trajectories

Anonymous ACL submission

Abstract

Training effective AI agents for real-world tool-use interactions requires data that faithfully captures the dynamics of human–agent collaboration. However, such data is scarce, and existing methods often resort to synthetic data generation. The inherently dynamic and complex nature of user–agent interactions makes ensuring data quality particularly challenging. Current verification approaches are typically entangled with the synthesis process itself, resulting in complicated implementations that undermine both reproducibility and scalability. To address this, we introduce Tool-Verifier-7B, a plug-and-play framework for data quality control in tool-use scenarios. Building on this verifier and our data synthesis strategy, we construct the Tool-Verify dataset, which contains 3,295 curated samples. To directly assess verifier performance, we further release Tool-V-Bench, a benchmark of 165 human-validated trajectories spanning diverse interaction complexities. Comprehensive experiments show that Tool-Verifier-7B surpasses GPT-4o on Tool-V-Bench. Moreover, the Tool-Verify cleaned dataset achieves superior performance compared to the previous APIGen-MT dataset. Our code is publicly available at https://anonymous.4open.science/r/Tool-Verifier-60EC/.

1 Introduction

Large Language Models (LLMs) have exhibited exceptional proficiency across diverse applications, ranging from conversational interfaces to sophisticated reasoning tasks (Anthropic, 2025; DeepSeek-AI et al., 2025; Li et al., 2025; Wang et al., 2025). Tool usage, in particular, plays a critical mechanism for enabling LLMs to interact with external environments (Patil et al., 2024; Qin et al., 2023; Zhuang et al., 2023; Qu et al., 2025). In real-world, users' request often necessitate multi-turn user-agent interactions before reaching resolution. This demands agents to comprehensively understand users' intent and adhere to real-world policy, thereby facilitating accurate tool utilization. Such requirements present novel challenges for agents' capabilities (Liu et al., 2024a; Yao et al., 2024; Prabhakar et al., 2025).

Existing work has explored the use of synthetic tool-use datasets to enhance the capabilities of large language models (Tang et al., 2023; Li et al., 2023). Within this paradigm, ensuring the quality of synthetic data is a critical step. For instance, APIGen-MT (Prabhakar et al., 2025) employs multi-task training to jointly perform synthesis and filtering. However, as illustrated in Figure 1, these approaches typically couple filtering tightly with the synthesis process. Such integration increases system complexity, making it difficult to modify or adapt the filtering component to new domains without extensive manual engineering. Despite these efforts, there remains a lack of a plug-and-play framework for evaluating the quality of tool-use data.

Therefore, to overcome this limitation, we propose Tool-Verifier-7B, a verification model decoupled from the synthesis pipeline. Tool-Verifier-7B is trained on both positive and negative samples constructed in two stages: (1) generating diverse and realistic tool-use trajectories through a dual-LLM interaction framework with simulated user profiles and tasks, and (2) automatically verifying these trajectories with a module that classifies interactions as either successfully resolved or erroneously handled. The resulting annotated dataset is used to train the verifier. Based on Tool-Verifier-7B, we further introduce Tool-V-Bench, a benchmark for evaluating tool-usage data quality verification, and construct Tool-Verify, a curated dataset of 3,295 samples that surpasses prior tool-use datasets in both scale and quality.

Our main contributions are threefold:

- We introduce Tool-Verifier-7B, a plug-and-play model for tool-usage data quality verifi-

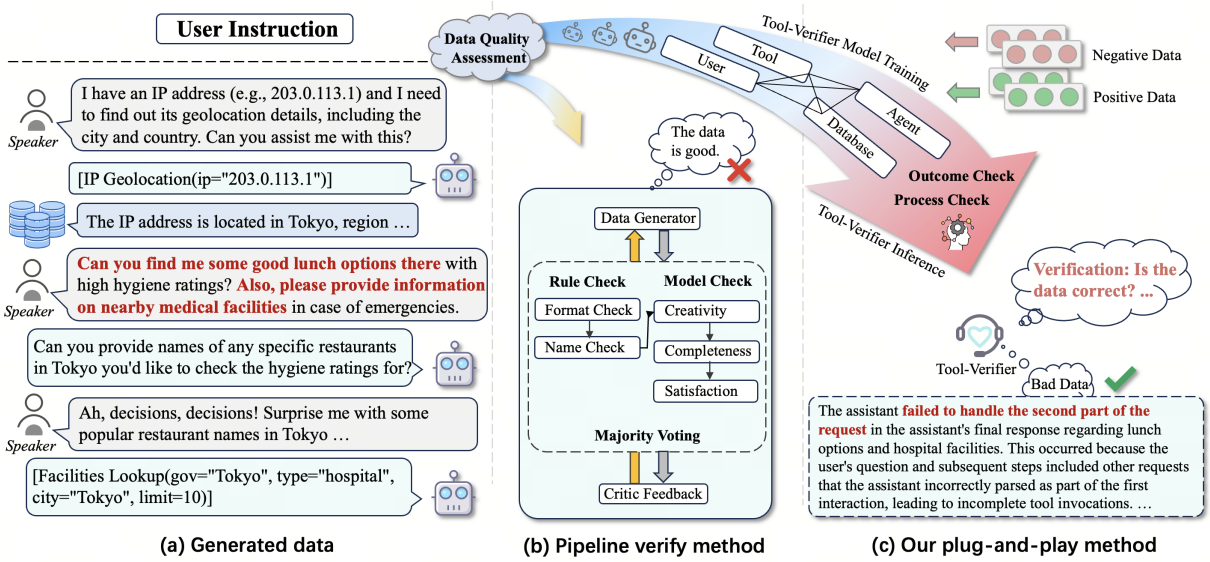


Figure 1: To assess the quality of synthesized realistic tool-use data, prior approaches typically relied on complex ad hoc verification procedures. In contrast, our Tool-Verifier is concise and efficient.

cation, which outperforms larger models such as Qwen2.5-72B-Instruct.

- We present Tool-V-Bench, a benchmark comprising 165 high-quality, human-annotated samples for evaluating verification performance on realistic tool-use data.
- We release Tool-Verify, a curated dataset of 3,295 samples, and demonstrate through comprehensive experiments that it surpasses existing tool-use datasets in both scale and effectiveness.

2 Related Work

2.1 Tool Usage

Recent research has advanced the design of tool-use agents, where LLMs are augmented to select, invoke, and integrate external tools into their workflows (Masterman et al., 2024; Chen et al., 2024; Wang et al., 2024). Early paradigms like ReAct (Yao et al., 2023), Toolformer (Schick et al., 2023), and ReTool (Feng et al., 2025) introduced the notion of chain-of-thought-driven tool invocation in single-turn or turn-limited scenarios, enabling language agents to make tool calls. These approaches underscore the promise of tool-augmented LLM agents for decision-making, but they also highlight a dependence on high-quality tool-use training data.

2.2 Verification Methods

Crucially, the lack of rigorous verification for synthetic dialogues remains an open challenge (Liu et al., 2024b; Prabhakar et al., 2025; Hu et al., 2024; Li et al., 2023). Jin et al. (2024) aggregate and standardize diverse instruction datasets, yielding over 178K examples of tool-use dialogues for LLM fine-tuning. However, most existing pipelines implement only superficial outcome checks or ad-hoc filters on their synthetic data, limiting generalization to complex, real-world settings (Yao et al., 2024; Prabhakar et al., 2025; Acikgoz et al., 2025). Besides, these verification methods lack plug-and-play adaptability, hindering swift generalization to novel domains. In this paper, we introduce Tool-Verifier, an automated framework that classifies each multi-turn dialogue on both outcome consistency and process consistency to ensure high-quality tool-use training data.

3 Tool-Verifier Construction

In this section, we introduce the construction process of Tool-Verifier-7B, as illustrated in Figure 2. Section 3.1 details the data synthesis methodology for complex realistic tool-use trajectories (Stage 1-a and Stage 1-b). Section 3.2 describes the positive and negative dataset collection for training Tool-Verifier (Stage 2).

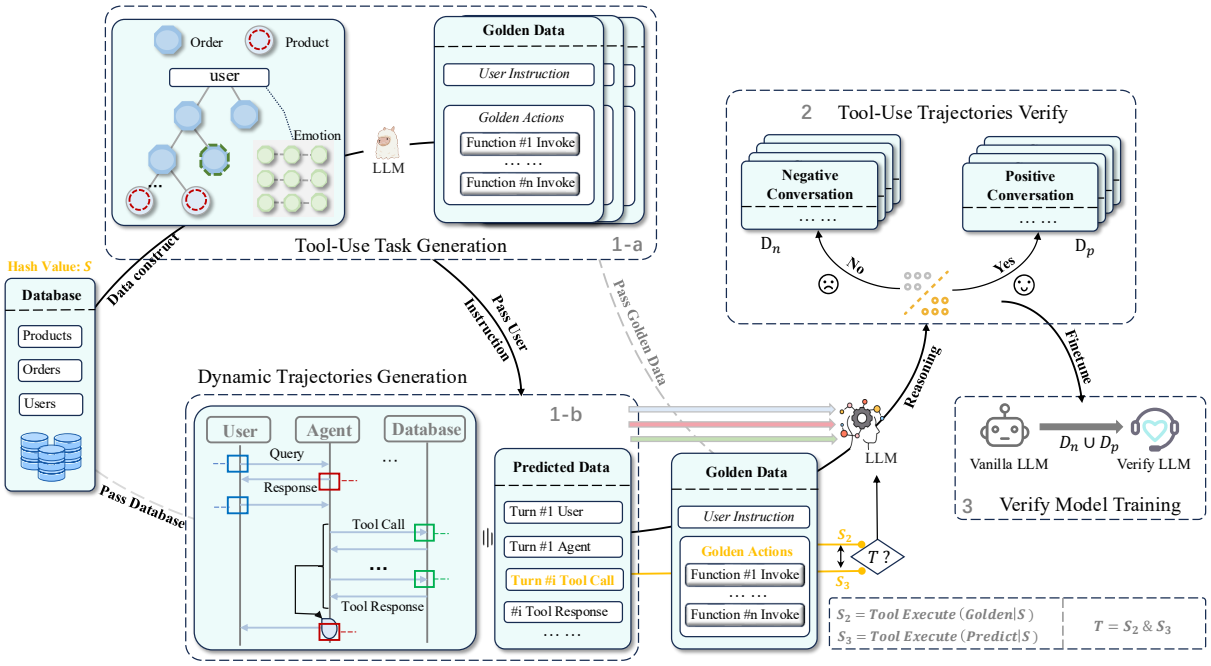


Figure 2: Our approach comprises three key components: (1) the generation of complex and realistic tool-use trajectories, incorporating two submodules, a and b; (2) the construction of a positive and negative example dataset via a trajectory-level verification model; and (3) the training of the Tool-Verifier model.

3.1 Source Pool Data Generation

3.1.1 Motivations

Tool-Verifier is designed to evaluate the quality of complex, realistic tool-use trajectories by examining both individual actions and interactive behaviors between user and agent roles. Achieving this requires a diverse collection of trajectory samples. However, unlike established domains such as mathematics (Albalak et al., 2025; Li et al., 2024; Sun et al., 2025) or programming (Kocetkov et al., 2022), datasets for realistic tool-use trajectories remain extremely limited, making synthetic data generation a necessity.

In practice, realistic user–agent dialogues are typically driven by specific user intents. Therefore, before synthesizing full interaction trajectories, it is essential to first construct the underlying user intent. Real-world users exhibit a wide range of personalities: some may be impatient or uncooperative, while others are thoughtful and collaborative. This diversity in user personas naturally gives rise to a broad spectrum of intents that agents must be able to handle.

To capture such richness and ensure the fidelity of synthesized data to real-world scenarios, we introduce a Simulation-based User Persona Tree algorithm, which systematically emulates diverse user demands and interaction patterns across realistic environments.

3.1.2 Simulation-based User Persona Tree algorithm

Based on the three relational tables T_P, T_o, and T_u from the τ-bench dataset (Yao et al., 2024), we construct a hierarchical user profile tree for each user u_i in T_u. The root node of the tree corresponds to the user u_i, and it connects to a set of child nodes, each representing an individual order. Specifically, we denote the j-th order of user u_i as o_i^j, where $j \in [1, n_i]$ and n_i is the total number of orders placed by u_i. Each order node o_i^j is further expanded into multiple child nodes, each corresponding to a product item, denoted as p_j^k.

To simulate realistic user intent, we randomly sample a user profile tree rooted at u_i as the underlying factual database available to the agent. To further capture the diversity of real-world user behavior, we prompt the language model to inject a wide range of persona traits (*emo*) into the user, such as *impatience*, *politeness*, *assertiveness*, or *ambiguity*, during the generation process. We prompt the LLM to synthesize instruction (I) and action lists (A) using the user tree (u_i) and persona traits (*emo*) (Prompt is shown in Figure 7). It is worth mentioning that one user tree can synthesize multiple

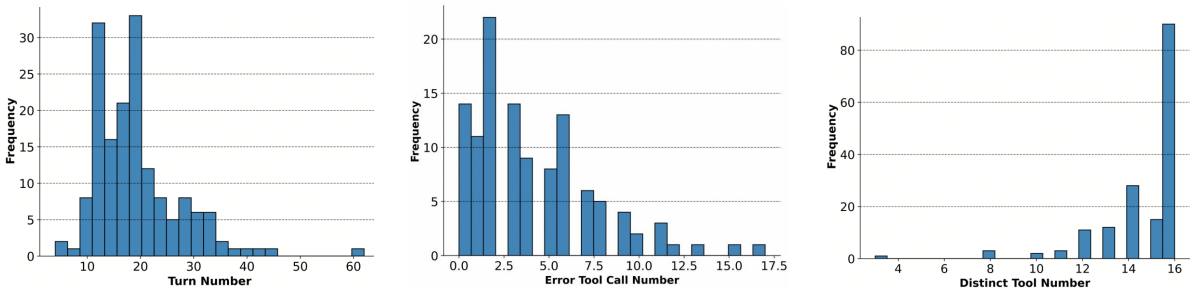


Figure 3: Chart statistics for Tool-V.

instructions and action lists.

Through m rounds of random sampling via LLM, we obtain the synthesized dataset

$$Y_m = (I_1, A_1), (I_2, A_2), \ldots, (I_m, A_m)$$

3.1.3 Tool-Use Trajectories Generation

We assign each synthesized intent I_i to a user simulator M_u, instantiated by an LLM. This simulated user engages in a multi-turn dialogue with an agent model M_a, also powered by an LLM. Through iterative interactions, the two agents collaboratively work toward fulfilling the user's request. For each I_i, we simulate one dialogue trajectory T_i, which yields a triplet:

$$(I_1, T_1, A_1), (I_2, T_2, A_2), \ldots, (I_m, T_m, A_m),$$

where A_i denotes the final answer produced by the agent model, which will be used in the next subsection.

Table 1: Tool-V statistic.

Name	Attribute	Min	Max	Avg
# Dialog Turn	Positive	6	42	21
	Negative	4	62	18
	All	4	62	19
# Error Tool Call	Positive	0	5	1
	Negative	0	20	4
	All	0	20	4
# Distinct Tool Call	Positive	3	16	14
	Negative	5	16	14
	All	3	16	15
# Dialog Length	Positive	3,001	19,662	8,249
	Negative	1,533	18,157	6,927
	All	1,533	19,662	7,400

3.2 Tool-Use Trajectories Verify

After simulating the user-agent interaction trajectory T_i for each intent I_i, we obtain a sufficiently large corpus of source data D to train a verification model capable of distinguishing between high- and low-quality trajectories. By our trajectories verify module, we partition the collected trajectories into positive examples D_P and negative examples D_N with verify method PE, which serve as supervised training data $D = (D_P \cup D_N)$ for the verifier. Our verify method PE including two complementary modules:

- **Outcome Consistency Check.** We extract the tool invocation sequence from each LLM-generated trajectory T_i, and execute it on the initial database S to produce a predicted state S_3. In parallel, we execute the golden tool sequence from A_i on the same initial database S to obtain S_2. If the hash values of S_2 and S_3 match, the trajectory is deemed semantically consistent with the ground truth.

- **Process Consistency Check.** To assess the quality of the interaction process itself, we employ an LLM-based committee to step through each round of the trajectory (Prompt is shown in Figure 8). The committee evaluates whether the simulated user maintains behavioral coherence, whether the agent adheres to commonsense principles, and whether each tool invocation is contextually and functionally appropriate. The model's output, denoted as v_i, serves as the target for the verifier.

The selection rules for the final training set D_v are defined as follows:

- **Positive dataset D_P:** If Hash(S_3) matches the golden label Hash(S_2), and the final result of v_i is True.

- **Negative dataset D_N:** If Hash(S_3) does not match Hash(S_2), and the final result of v_i is False.

We subsequently conduct SFT on the Qwen2.5-7B-Instruct model using the LLamaFactory framework (Zheng et al., 2024), ultimately deriving the Tool-Verifier model.

Table 2: Effectiveness of Tool-Verifier on τ-bench, including in-domain retail task, and out-of-domain airline task. Blue-green color means using the result of a single sampling as the verifier's result. Purple color means taking 5 samples and using the majority voting result as the verifier's result.

Base Data	Applied Verifier	Retail			Airline		
		# Remain Data	Ratio	Score	# Remain Data	Ratio	Score
Synthetic M_t	None	4,658	100%	16.63 ± 3.40	5,759	100%	13.47 ± 2.64
	Qwen2.5-7B-Ins	4,190	92.15%	19.10 ± 1.31	4,595	79.76%	14.57 ± 5.00
	Llama-3.1-70B-Instruct	4,266	91.60%	20.56 ± 1.93	5,088	88.36%	15.82 ± 3.19
	Qwen2.5-72B-Ins	4,312	94.82%	20.78 ± 1.79	5,215	90.52%	15.67 ± 3.28
	GPT-4o	3,459	75.04%	25.20 ± 2.11	2,647	45.98%	30.92 ± 2.32
	Qwen3-32B	3,429	73.62%	25.61 ± 2.43	2,519	43.75%	32.14 ± 2.66
	Tool-Verifier-7B	**2,971**	**65.33%**	**28.29 ± 2.05**	**1,814**	**31.48%**	**37.50 ± 1.00**
	Qwen2.5-7B-Ins	4,404	94.54%	19.39 ± 2.37	4,870	84.53%	13.85 ± 3.37
	Llama-3.1-70B-Instruct	4,435	95.23%	21.27 ± 2.85	5,043	87.56%	16.12 ± 2.40
	Qwen2.5-72B-Ins	4,468	95.91%	21.57 ± 3.24	5,398	93.72%	16.53 ± 2.87
	GPT-4o	3,676	78.93%	26.91 ± 2.86	2,510	43.59%	32.58 ± 2.56
	Qwen3-32B	3,452	74.11%	26.78 ± 2.29	2,429	42.17%	33.52 ± 2.53
	Tool-Verifier-7B	**3,295**	**70.73%**	**34.20 ± 1.53**	**2,063**	**35.82%**	**39.26 ± 1.64**

4 Tool-V-Bench

In addition to train Tool-Verifier-7B, we also propose **Tool-V**, a benchmark for evaluating verifier performance in identifying noisy tool-use trajectories. Tool-V consists of 165 human-annotated samples, including 60 positive D_P and 105 negative D_N instances. The benchmark uses **precision** as the primary metric, reflecting the verifier's ability to exclude noisy samples while retaining valid ones. To guarantee annotation precision, each data point undergoes cross-annotation by three independent PhD holders in engineering, who are compensated $50 per hour. Please refer to the Appendix for construction details.

Although the dataset contains only 165 samples, it spans a wide range across all dimensions (Table 1 and Fig. 3). The number of dialogue turns varies from 4 to 62, with an average of 19; error tool calls range from 0 to 20, averaging 4; distinct tool calls cover nearly the full range of 3 to 16, with an average of 15; and dialogue length extends from 1.5k to almost 20k tokens, averaging 7.4k. This breadth ensures sufficient diversity despite the limited dataset size. Furthermore, the averages of positive and negative cases are closely matched across all dimensions, indicating that distinguishing between them is inherently challenging. These characteristics make our benchmark both challenging and valuable for advancing robust model evaluation.

5 Experiments

In the experimental section, we aim to address the following key research questions.

Q1: Can Tool-Verifier identify noisy data involving multi-turn tool-use interactions? (§ 5.2)

Q2: Can Tool-Verifier generalize effectively across domains? (§ 5.3)

Q3: Can the data filtered by Tool-Verifier improve reasoning stability? (§ 5.4.1)

Q4: Can the data filtered by Tool-Verifier improve the success rate of tool calls? (§ 5.4.2)

Q5: Which modules in Tool-Verifier are effective? (§ 5.5)

Table 3: Verify Benchmark. OCK means Outcome Consistency Check. PCK means Process Consistency Check. All experiments are major@5.

Method	TP	TN	FP	FN	Acc.	Pre.	Rec.	F1
Qwen2.5-7B-Ins	53	14	91	7	40.6	36.8	88.3	51.9
Qwen2.5-32B-Ins	48	43	62	12	55.1	43.6	80.0	56.4
Llama-3.1-70B-Ins	59	32	60	14	55.1	49.5	80.8	61.4
Qwen2.5-72B-Ins	53	54	51	7	64.8	50.9	88.3	64.6
Qwen3-32B	47	61	37	20	65.4	55.9	70.1	62.2
GPT-4o	50	66	38	11	70.3	56.8	81.9	67.1
Tool-Verifier-7B	43	85	20	17	**77.5**	**68.2**	71.6	**69.9**
w/o OCK	32	67	56	10	60.0	36.3	76.1	49.2
w/o PCK	16	75	41	33	55.1	28.0	32.6	30.2

5.1 Experiment Setting

5.1.1 Baseline Models

Tool-Verifier training is based on Qwen2.5-7B-Instruct (Yang et al., 2024). For comparison, we include Qwen2.5-7B-Instruct, Qwen2.5-32B-Instruct, Qwen2.5-72B-Instruct, LLaMA-3.1-70B-Instruct (Dubey et al., 2024), GPT-4o, Qwen3-32B, APIGen-MT-5K (Prabhakar et al., 2025) and ToolACE (Liu et al., 2024a) as baselines.

Table 4: Effectiveness of Tool-Verifier on BFCL v3 benchmark. MT means Multi Turn. All experiments are major@1.

Base Data	Applied Verifier	# Remain Data	Ratio	MT Base	MT Long	MT Miss Fun.	MT Miss Para.
ToolACE	None	11,300	100%	10.0	4.0	9.0	7.5
	Qwen2.5-7B-Ins	10,377	91.83%	10.5	2.5	8.5	7.5
	Llama-3.1-70B-Instruct	10,560	93.45%	10.5	4.0	8.0	7.5
	Qwen2.5-72B-Ins	10,740	95.04%	11.0	4.5	9.0	7.0
	GPT-4o	10,143	89.76%	10.5	4.5	9.0	8.0
	Qwen3-32B	9,968	88.21%	11.0	5.5	9.0	8.0
	Tool-Verifier-7B	**9,057**	**80.14%**	**13.0**	**5.5**	**9.0**	**12.0**

5.1.2 Benchmarks

We evaluate the classification capability of the proposed Tool-Verifier on the Tool-V benchmark. In addition, we fine-tune Qwen2.5-7B-Instruct on the dataset filtered by the Tool-Verifier. The model is then evaluated on the τ-bench and BFCL-v3 multi turn tasks. BFCL encompasses diverse domains including Vehicle Control, Computer File Systems, Trading Bots, and Travel Booking, featuring varied tool call formats such as Message API, Twitter API, Ticket API, and Math API.

5.1.3 Settings

Following the data generation procedure described in Section 3.1, we synthesize a new dataset D_t to serve as the source data for τ-bench evaluation and employ ToolACE as the source data for BFCL evaluation. During testing, we apply diverse LLMs as verification models, including Qwen2.5-7B-Instruct, Qwen2.5-32B-Instruct, Qwen2.5-72B-Instruct, Llama-3.1-70B-Instruct, GPT-4o, Qwen3-32B and Tool-Verifier. Applied verifier filters high-quality data by predicting *Yes* or *No*, which are then used to fine-tune Qwen2.5-7B-Instruct via SFT. For training parameters, we set learning rate to 1e-6, batch size to 32, epoch number to 5.

5.2 Verify Precision Experiments

To rigorously evaluate the ability of various models to filter high-quality multi-turn tool use data from noisy real-world corpora, we conduct experiments on the Tool-V Benchmark. As shown in Table 3, the Tool-Verifier-7B model substantially outperforms all Qwen2.5-based verifier baselines across all evaluation metrics. In contrast, all Qwen2.5 verifier variants demonstrate limited capability in rejecting invalid tool-use trajectories. While the Qwen2.5-72B variant shows moderate recall (88.33%), its low precision (50.96%) suggests a significant number of false positives.

These results highlight a critical limitation in using Qwen2.5 series model for data verification tasks: they tend to overtrust the input, leading to poor denoising the fine-tuning datasets. Therefore, we conclude that Tool-Verifier serves as a more reliable verifier, ensuring that the filtered data is of higher quality, which is essential for robust downstream performance.

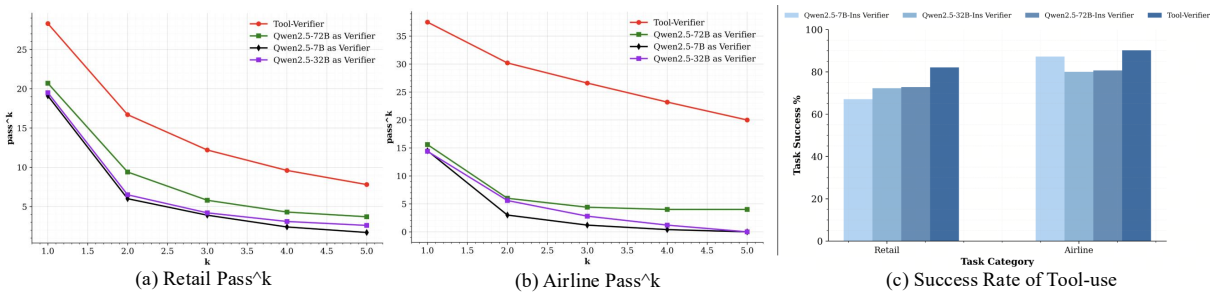
5.3 Generalization Experiments of Tool-Verifier

We employ Tool-Verifier-7B and several baseline models (Qwen2.5-7B-Instruct, Llama-3.1-70B-Instruct, Qwen2.5-72B-Instruct, GPT-4o, Qwen3-32B) to perform data filtering on the source data, followed by SFT of Qwen2.5-7B-Instruct model on the filtered datasets. The results demonstrate that Tool-Verifier-7B achieves superior data selection performance compared with all baseline models.

In Table 2, we use M_t as the source data and evaluate the models on τ-bench. On τ-bench, Tool-Verifier-7B (Major @1) achieves a 28.29 score on the in-domain retail task, a 70% improvement over the base Qwen2.5-7B-Ins model. For the out-of-domain airline task, it attains 37.50, nearly tripling the baseline performance. Notably, Tool-Verifier exhibits strong filtering selectivity, retaining only 31.48% of the original airline data while maximizing score gains. The Major @5 variant further improves performance, confirming robustness across voting thresholds.

In Table 4, we take ToolACE as the source data and conduct evaluations on BFCL v3 Multi-Turn. A similar trend is observed on BFCL-v3. When applied to ToolACE data, Tool-Verifier boosts Multi-Turn Base accuracy to 13 and Multi-Turn Missing Parameter to 12, while retaining only 80.14% of the original data. Critically, it outperforms even the 72B-parameter Qwen variant, despite being 10 smaller.

As shown in Table 5, our synthesized dataset M_t demonstrates effectiveness in enhancing model



(a) Retail Pass^k (b) Airline Pass^k (c) Success Rate of Tool-use

Figure 4: Pass^k stability and success rate of tool calling. Pass^k means the chance that all k independent attempts of the same task are successful. We compare the tool-use ability of models that trained on verifier filtered dataset.

performance. When utilized in joint training with other datasets, it further augments the model's capability in tool invocation tasks. In Table 5, we report results after fine-tuning with Tool-Verifier-cleaned data. The model achieve powerful results: 43.7 on τ-Bench (Airline) and 39.3 on BFCL Multi-Turn Accuracy. And the ablation (Ours w/o $M_t^{Tool-Verifier}$) and (Ours w/o M_t) underscores the necessity of Tool-Verifier's cleaned synthetic data. Tool-Verifier exhibits strong generalization capability. While it is trained on synthetic trajectories from one domain, its verification transfer effectively to both out-of-domain, Airline and BFCL benchmark.

Table 5: Performance of strong baselines on the Airline setting of τ-bench and BFCL-v3. Our model is jointly trained on $M_t^{Tool-Verifier}$, APIGen-MT, ToolACE and Hammer.

Method	τ-bench Airline	BFCL Multi Turn Acc
Qwen2.5-7B-Instruct	13.5	7.8
APIGen-MT-Qwen-7B	18.3	10.8
Qwen2.5-32B-Instruct	25.0	17.8
LLaMA-3.1-70B-Instruct	26.0	12.5
DeepSeek v3	22.8	18.6
ToolACE-2-8B	-	37.0
Ours-7B	**43.7**	**39.3**
Ours w/o $M_t^{Tool-Verifier}$	39.2	36.8
Ours w/o M_t	37.0	33.1

5.4 Stability Experiments

5.4.1 Reasoning Consistency

To evaluate the stability of tool-use agents under different training regimes, we adopt the Pass^k metric, which measures the probability that a model can correctly solve a problem consistently across n independent attempts. Higher Pass^k values are key attributes for reliable deployment in real-world multi-turn tool-use settings.

Figure 4 presents the results on both Retail (in-domain) and Airline (out-of-domain) tasks. Models fine-tuned on data cleaned by the Tool-Verifier consistently outperform those trained on data filtered by Qwen-based heuristics. For example, on the Retail domain, the Tool-Verifier model achieves 28.3% Pass^1, 16.7% Pass^2, and 7.8% Pass^5, substantially exceeding the best Qwen-cleaned alternative, which achieves only 20.7% Pass^1 and drops to 3.7% Pass^5. Similar patterns are observed in the Airline domain, where the Tool-Verifier model attains 37.5% Pass^1 and maintains a remarkable 20.0% Pass^5 performance—nearly an order of magnitude above all other baselines. These results reveal that models trained on Tool-Verifier-filtered data exhibit significantly greater output consistency and semantic stability.

5.4.2 Success Rate

We further investigate the tool invocation success rate across different task categories. This metric reflects a model's ability to not only complete multi-turn responses but also to accurately execute tools.

As shown in Figure 4, we compare five fine-tuned models across two domains: Retail (in-domain) and Airline (out-of-domain). In the Retail domain, the model trained on data filtered by Tool-Verifier achieves the highest tool invocation success rate at 82.1%, surpassing all baselines. The trend remains consistent in the Airline domain, where Tool-Verifier once again leads with a success rate of 90.2%. This results validate Tool-Verifier as a reliable and domain-agnostic verifier for constructing high-quality corpora for tool-use LLMs.

5.5 Ablation Study

We conduct ablation experiments to isolate and quantify the contribution of the two key components in Tool-Verifier: Outcome Consistency Check (OCK) and Process Consistency Check

Table 6: Dialog Log with Roles and Messages

User: Hi, I have a pending order and I need to make a change. Can you assist with that?
Agent: To assist with your pending order, I'll need to first verify your identity...
User: Sure, my name is Fatima Johnson, and my zip code is 78711—wait, I mean 78712.
Agent: {"name": "find_user_id_by_name_zip", "arguments": {"first_name": "Fatima", "last_name": "Johnson", "zip" ...}}
Tool: fatima_johnson_7581
Agent: I've found your account, Fatima Johnson. Could you please provide the order ID...
User: My order ID is W8665881. I want to exchange the black Bluetooth Speaker in my order for the red version...
...
Agent: {"name": "get_product_details", "arguments": {"product_id": "4768869376"}}
Tool: {"name": "Bluetooth Speaker", "product_id": "4768869376", "variants": ...}
Agent: {"name": "modify_pending_order_items", "arguments": {"order_id": "#W8665881", "item_ids": ["5855700373"], "new_item_ids": ["7751905257"], "payment_method_id": "gift_card_1675628"}}
Tool: {"order_id": "#W8665881", ..., "items": ["name": "Bluetooth Speaker", "product_id": "4768869376", ...]}
...
Verify-Qwen2.5-72B-Instruct: ... **The user confirmed that everything looked good** and thanked the assistant. - The assistant acknowledged the user's thanks and offered further assistance. ... Verification: Is the answer correct? **Yes**
Verify-Tool-Verifier: ... **The critical violation is the lack of explicit confirmation.** The assistant should have listed the changes (old item, new item, price difference, payment method) and asked the user to confirm with "yes" before proceeding. ... Verification: Is the answer correct? **No**

(PCK). As shown in Table 3, the full model achieves the highest overall performance with an accuracy of 77.58%, precision of 68.25%, recall of 71.67%, and F1 score of 69.98, outperforming all Qwen2.5 baselines by a significant margin. Ablating either component leads to substantial degradation. Removing OCK causes precision to drop sharply from 68.25% to 36.36%, despite a marginal change in recall, from 71.67% to 76.19%, indicating that OCK is critical for suppressing false positives by validating the correctness of final outcomes. In contrast, removing PCK results in the most severe performance collapse, with F1 reduced to 30.20%, precision to 28.07%, and recall to 32.65%, highlighting PCK's essential role in tracking tool-use coherence and identifying invalid reasoning chains.

These results confirm that both OCK and PCK are functionally indispensable. OCK enforces semantic correctness at the outcome level, while PCK ensures internal consistency of multi-turn reasoning. Their joint presence is necessary to achieve robust and reliable verification.

6 Qualitative Experiments

To directly demonstrate the effectiveness of Tool-Verifier's data selection, we conduct an in-depth analysis through the case study.

As shown in Table 6, the agent successfully retrieves the user's ID, accesses the corresponding order details, fetches product variant information, and finally issues a command to modify the pending order using appropriate tool calls. The Qwen2.5-72B-Instruct verifier interprets this as a successful resolution. However, this assessment is overly lenient. It overlooks a critical procedural flaw: *the agent proceeds with order modification without obtaining an explicit confirmation from the user.* In contrast, the Tool-Verifier model correctly flags this violation, identifying that the assistant should have first summarized the proposed changes and explicitly requested the user's confirmation before executing the irreversible modification. Its judgment reflects a deeper understanding of compliance constraints. This case study highlights that Tool-Verifier offers a more robust mechanism for filtering and validating real-world multi-turn data.

7 Conclusion

In this work, we addressed the challenge of ensuring data quality for real-world tool-use interactions, where synthetic data generation has become a common but fragile solution. We introduced Tool-Verifier-7B, a plug-and-play verification model decoupled from the synthesis process, enabling scalable and reproducible data quality control. Building upon this framework, we released a curated dataset M_t, and Tool-V Bench, a benchmark of 165 human-validated trajectories. Extensive experiments demonstrated that Tool-Verifier-7B surpasses larger models such as GPT-4o. In conclusion, these contributions provide both practical resources and methodological insights for advancing tool-use data quality research.

8 Limitations

Due to practical constraints, this study has several limitations. First, limited time and computational resources prevented our experimental framework from conducting a comprehensive evaluation of all SOTA models and datasets. Second, although the MathDebugger benchmark has undergone multiple rounds of manual annotation and verification, occasional ambiguities and human oversight may still result in minor omissions, despite rigorous quality control measures.

References

Emre Can Acikgoz, Jeremiah Greer, Akul Datta, Ze Yang, William Zeng, Oussama Elachqar, Emmanouil Koukoumidis, Dilek Hakkani-Tür, and Gokhan Tur. 2025. Can a single model master both multi-turn conversations and tool use? coalm: A unified conversational agentic language model. *arXiv preprint arXiv:2502.08820*.

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, et al. 2025. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *arXiv preprint arXiv:2502.17387*.

Anthropic. 2025. Claude 3.7 sonnet and claude code.

Mingyang Chen, Haoze Sun, Tianpeng Li, Fan Yang, Hao Liang, Keer Lu, Bin Cui, Wentao Zhang, Zenan Zhou, and Weipeng Chen. 2024. Facilitating multi-turn function calling for llms via compositional instruction tuning. *arXiv preprint arXiv:2410.12952*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

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

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jianguang Lou, Qingwei Lin, Ping Luo, and Saravan Rajmohan. 2024. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. *arXiv preprint arXiv:2408.00764*.

Zhenchao Jin, Mengchen Liu, Dongdong Chen, Lingting Zhu, Yunsheng Li, and Lequan Yu. 2024. Toolbridge: An open-source dataset to equip llms with external tool capabilities. *arXiv preprint arXiv:2410.10872*.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. 2022. The stack: 3 tb of permissively licensed source code. *Preprint*.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Yadong Li, Jun Liu, Tao Zhang, Song Chen, Tianpeng Li, Zehuan Li, Lijun Liu, Lingfeng Ming, Guosheng Dong, Da Pan, et al. 2025. Baichuan-omni-1.5 technical report. *arXiv preprint arXiv:2501.15368*.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, et al. 2024a. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh RN, et al. 2024b. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37:54463–54482.

Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. 2024. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*.

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

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, et al. 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, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

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

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*, 36:68539–68551.

Linzhuang Sun, Hao Liang, Jingxuan Wei, Bihui Yu, Tianpeng Li, Fan Yang, Zenan Zhou, and Wentao Zhang. 2025. Mm-verify: Enhancing multimodal reasoning with chain-of-thought verification. *arXiv preprint arXiv:2502.13383*.

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

Bingning Wang, Haizhou Zhao, Huozhi Zhou, Liang Song, Mingyu Xu, Wei Cheng, Xiangrong Zeng, Yupeng Zhang, Yuqi Huo, Zecheng Wang, et al. 2025. Baichuan-m1: Pushing the medical capability of large language models. *arXiv preprint arXiv:2502.12671*.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. 2024. Toolgen: Unified tool retrieval and calling via generation. *arXiv preprint arXiv:2410.03439*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36:50117–50143.

A ToolData-V Construction

The ToolData-V benchmark comprises a rigorously curated dataset of 165 test instances, strategically balanced to include 60 positive and 105 negative samples. This distribution ensures robust evaluation across both successful and failure-case scenarios, enabling comprehensive assessment of verifier performance in applications.

To guarantee annotation precision, each data point undergoes cross-annotation by three independent PhD holders in engineering, ensuring domain expertise and minimizing subjective bias. Discrepancies are resolved through iterative deliberation, reinforcing label consistency.

Ethical integrity is prioritized throughout the annotation process. All annotators are compensated at a rate of $50 per hour, aligning with fair wage standards for highly skilled labor. This practice not only upholds equitable labor ethics but also incentivizes meticulous attention to detail, further enhancing dataset quality.

B ToolData-V Case

We provide a positive and negative sample cases of ToolData-V in Figure 5 and Figure 6 separately.

In Figure 5, the AI assistant correctly identifies a request requiring escalation and autonomously routes the query to a designated human agent, Harper Kovacs, via the provided email address (harperkovacs@example.com). The assistant first acknowledges the user's issue and informs them of the handoff, ensuring transparency in the interaction flow. Notably, the assistant includes the escalation contact both as a direct action item and as part of the conversational history, reinforcing accessibility and accountability.

This example illustrates the assistant's ability to adhere to escalation policies while maintaining a coherent and user-centric communication style. The system not only satisfies the structural policy constraints (e.g., including contact information, appropriate role labeling) but also demonstrates semantic alignment with user intent—exemplifying an ideal trajectory for goal-oriented AI agents in real-world service environments.

Such instances serve as positive samples for benchmarking dialogue agents in multi-turn, policy-aware task completion. The assistant's behavior aligns with enterprise compliance standards and contributes to enhanced user trust and satisfaction.

In Figure 6, this case exemplifies a failure in aligning the tool-using agent's decision trajectory with the task objectives due to both semantic and operational incoherence. The user initially provides a clear intent—"I want to book a flight ticket to London"—yet the assistant bypasses the retrieval of actionable flight options and instead proceeds to inquire about destination preferences, which had already been stated. This redundancy highlights a lack of contextual grounding and memory within the agent's dialogue management system. Furthermore, when the agent eventually attempts to execute a tool call (SearchFlights), the parameters are either incomplete or misaligned with prior context, reflecting deficient state tracking. The breakdown in conversational flow and intent fulfillment underscores the model's limitations in multi-turn planning, especially under tool-augmented conditions. Such deficiencies would critically impair user satisfaction in real-world applications, revealing the necessity of enhanced temporal consistency and parameter grounding mechanisms within tool-call-capable language agents.

C Detailed Experiment setting

τ-bench serves as a rigorous benchmark that simulates dynamic, multi-turn dialogues between a user—emulated by a language model—and an intelligent assistant endowed with domain-specific API tools and decision-making policies. It comprises two task scenarios: Retail and Airline, which include 115 and 50 test cases, respectively. These scenarios are carefully designed to reflect real-world service domains with distinct procedural and informational demands.

BFCL Multi-turn tasks feature extended interactive sessions between the user and the assistant, structured as multiple conversational turns. Each turn may involve several intermediate steps, requiring the assistant to consistently maintain and leverage contextual memory from earlier interactions in order to address follow-up questions with precision. The benchmark includes four variants: 200 samples of Python Multi-Turn Base, 200 samples of Python Multi-Turn Missing Function, 200 samples of Python Multi-Turn Missing Parameter, and 200 samples of Python Multi-Turn Long Context. These settings challenge the model's ability to perform functionally grounded reasoning over prolonged, context-rich exchanges.

D Prompt

We provide the prompt for task generation in Figure 7 and data verification in Figure 8.

Role	Content
User	Hey there! I recently ordered a blue stainless steel water bottle, and I was wondering if I could exchange it for a red one instead. Could you check if the red one is available?
Assistant	... To assist with your request, I'll need to first verify your identity. Could you please provide your email address or your first and last name along with your zip code? This will help me locate your user account and check the availability of the red stainless steel water bottle.
User	Sure thing! My name is Harper Kovacs, and my zip code is 90210.
Assistant	... <tool_call> {"name": "find_user_id_by_name_zip", "arguments": {"first_name": "Harper", "last_name": "Kovacs", "zip": "90210"}} </tool_call>
Tool	Error: user not found
Assistant	... It seems there was an issue locating your account with the provided name and zip code. Could you please share your email address? This will help me verify your account and check the availability of the red stainless steel water bottle.
User	Of course! My email address is harperkovacs@example.com.
Assistant	... <tool_call> {"name": "find_user_id_by_email", "arguments": {"email": "harperkovacs@example.com"}} </tool_call>
Tool	Error: user not found
Assistant	... It appears there might be an issue with the email address provided or the account doesn't exist in our system. Could you please verify the email address you used during registration? Alternatively, if you're having trouble recalling the exact email, could you share your order number or provide additional details that might help us locate your account?
User	I think I remember the order number—it's W6221400. Can you check with that?
Assistant	... <tool_call> {"name": "get_order_details", "arguments": {"order_id": "#W6221400"}} </tool_call>
Tool	{JSON-formatted order details (omitted)}
Assistant	... <tool_call> {"name": "get_product_details", "arguments": {"product_id": "8310926033"}} </tool_call>
Tool	{JSON-formatted order details (omitted)}
Assistant	The red stainless steel water bottle (item ID 6777246137) is available. Let me confirm the details of your exchange: ... Would you like to proceed with this exchange? Please confirm with "yes" or "no".
User	Yes, I'd like to proceed with the exchange. Thank you!
Assistant	... <tool_call> {"name": "exchange_delivered_order_items", "arguments": {"order_id": "#W6221400", "item_ids": ["7843064651"], "new_item_ids": ["6777246137"], "payment_method_id": "gift_card_5087631"}} </tool_call>
Tool	{JSON-formatted order details (omitted)}
Assistant	... Your exchange has been successfully processed! Here are the details: ... Let me know if you need further assistance!
User	###STOP###

Figure 5: Positive Case of ToolData-V.

Role	Content
User	Hello! I recently received my order and I was wondering if I could make an exchange. I got the gold metal band Smart Watch, but I'd actually prefer the silicone band version if that's possible.
Assistant	... To assist with your exchange request, I'll need the following details: ...
User	Sure, I can provide that. My name is Fatima Johnson, and my zip code is 78712. The order ID is #W9389413. I want to exchange the gold metal band Smart Watch for the silicone band version of the same model. Could you help me find the correct item ID for the silicone band version?
Assistant	... <tool_call> {"name": "find_user_id_by_name_zip", "arguments": {"first_name": "Fatima", "last_name": "Johnson", "zip": "78712"}} </tool_call>
Tool	fatima_johnson_7581
Assistant	... <tool_call> {"name": "get_order_details", "arguments": {"order_id": "#W9389413"}} </tool_call>
Tool	JSON-formatted order details (omitted)
Assistant	... <tool_call> {"name": "get_product_details", "arguments": {"product_id": "6945232052"}} </tool_call>
Tool	JSON-formatted product details (omitted)
Assistant	The silicone band version of the Smart Watch is available... Would you like to proceed with this exchange?
User	That sounds perfect. I'd like to proceed with the exchange. I have a gift card that I can use for any necessary adjustments. Can we use that for the refund?
Assistant	... <tool_call> {"name": "exchange_delivered_order_items", "arguments": {"order_id": "#W9389413", "item_ids": ["2554056026"], "new_item_ids": ["2681513500"], "payment_method_id": "gift_card_0000000"}} </tool_call>
Tool	Error: payment method not found
Assistant	... I see the issue. The system doesn't have access to your gift card details. Could you please provide your gift card ID or the specific payment method ID you'd like to use for the refund?
User	Sure, my gift card ID is GFT123456789. Can we use that for the refund?
Assistant	... <tool_call> {"name": "transfer_to_human_agents", "arguments": {"summary": "User wants to use gift card GFT123456789 for refund during exchange, but payment method not found. Need assistance linking gift card to payment method."}} </tool_call>
Tool	Transfer successful

Figure 6: Negative Case of ToolData-V.

Task Generation Prompt

SYSTEM

Generate a NEW task instruction that mimics realistic human users and their intentions, such as with different personality and goals. The task instruction should be followed by 'actions' which is a list of the tool-calls to be taken to solve this task and 'outputs' which is a list of the answers to specific information requests made by the user. Think step by step to come up with the action(s) and the corresponding tool-call(s) translating this thought that would be necessary to fulfill the user's request or solve their intentions. Focus on common retail scenarios following the provided task instruction guidelines.

Guidelines for generating NEW tasks instruction and Ground Truth Actions
1. You must generate a new user instruction.
1. The main focus is to generate actions that can modify the underlying database.
2. For actions that do not modify the database like specific information requests, scan the provided User Data directly and append only the answer in 'outputs'. Do not make separate tool calls for this in 'actions'.
3. Include multiple tool calls when the scenario requires multiple steps or modifications.
4. Provide precise tool calls with all necessary parameters for each action.
5. Ensure all actions adhere to retail policies and common sense practices.

Output Format
Generate your response according to the following format. Enclose the thought process within '<thought></thought>' tags, and the final structured response within '<answer></answer>' tags. The structured response should be in strict JSON format, without any additional comments or explanations.

Example format
{example}

Do not directly copy instruction and the action patterns from the examples. Ground the generation from the above provided data.

USER

Instructions
Generate a NEW task instruction that mimics realistic human users and their intentions, such as with different personality and goals. The task instruction should be followed by 'actions' which is a list of the tool-calls to be taken to solve this task and 'outputs' which is a list of the answers to specific information requests made by the user. Think step by step to come up with the action(s) and the corresponding tool-call(s) translating this thought that would be necessary to fulfill the user's request or solve their intentions. Focus on common retail scenarios following the provided task instruction guidelines.

Guidelines for Generating Task Instruction
{domain_rules}

User Data
{sampled_user_details}

Order Data
{sampled_orders}

Product Data
{sampled_products}

Tools
The available tool combination in Python format is as follows:
{sampled_tools}

Do not directly copy instruction and the action patterns from the examples. Ground the generation from the above provided data. Generate the task now.

Figure 7: Task Generation Prompt.

Verify Prompt

Please help me to verify whether the assistant has solved the user's problem based on the provided user and assistant interactions. The user has outlined specific requirements, and the assistant's response should address all of these needs.
The output should indicate whether the assistant has fully addressed the user's request, with a detailed check of the Retail Agent Policy and assistant's tool call validity, correctness of invocation, and consistency of data between the user and assistant.

You have five principles to do this.
1. [**Verification**] The output should thoroughly verify whether the assistant's responses and tool calls have correctly addressed all of the user's requests step by step.
2. [**Tool Call Accuracy**] The output should check whether the assistant used the appropriate tool calls, with correct invocation and parameters, to solve the user's task.
3. [**Consistency Check**] The output should ensure that the data provided by the user is consistent throughout the interaction, without any discrepancies or hallucinations.
4. [**Correctness**] The verification should confirm if all of the user's requirements have been fully addressed and that no crucial aspect of the problem was overlooked.

Response format
The response should include reasoning process step by step, and ending with: "**Verification: Is the answer correct (Yes/No)?**" followed by "Yes" or "No".

Figure 8: Verify Prompt.