APIGen-MT: Agentic PIpeline for Multi-Turn Data Generation via Simulated Agent-Human Interplay

Akshara Prabhakar* **Zuxin Liu*** **Ming Zhu†** **Jianguo Zhang†** **Tulika Awalgaonkar†**
Shiyu Wang **Zhiwei Liu** **Haolin Chen** **Thai Hoang** **Juan Carlos Niebles**
Shelby Heinecke‡ **Weiran Yao‡** **Huan Wang‡** **Silvio Savarese‡** **Caiming Xiong‡**

Salesforce AI Research, USA

Abstract

Training effective AI agents for multi-turn interactions requires high-quality data that captures realistic human-agent dynamics, yet such data is scarce and expensive to collect manually. We introduce APIGen-MT, a two-phase framework that generates verifiable and diverse multi-turn agent data. In the first phase, our agentic pipeline produces detailed task blueprints with ground-truth actions, leveraging a committee of LLM reviewers and iterative feedback loops. These blueprints are then transformed into complete interaction trajectories through simulated human-agent interplay. We train a family of models—the xLAM-2-fc-r series with sizes ranging from 1B to 70B parameters. Our models outperform frontier models such as GPT-4o and Claude 3.5 on τ-bench and BFCL benchmarks, with the smaller models surpassing their larger counterparts, particularly in multi-turn settings, while maintaining superior consistency across multiple trials. Comprehensive experiments demonstrate that our verified blueprint-to-details approach yields high-quality training data, enabling the development of more reliable, efficient, and capable agents. We open-source both the synthetic data collected and the trained xLAM-2-fc-r models to advance research in AI agents.

Dataset https://huggingface.co/Salesforce/APIGen-MT-5k
Model https://huggingface.co/Salesforce/xLAM-2
Website https://apigen-mt.github.io

1 Introduction

The growth of Large Language Model (LLM) agents has been accelerating at an unprecedented rate, driven by advancements in AI capabilities and rising demand across industries [21, 1, 10, 5, 30, 55, 23, 7, 25]. Their role has evolved beyond simple conversational chatbots to AI agents capable of executing real-world tasks, such as managing financial transactions, scheduling appointments, and handling customer service requests. These applications demand not only linguistic fluency but also precise execution, reliability, and adherence to domain-specific policies. Realistic enterprise use cases require an assistant (also referred to as *agent* in this document) capable of fluently conversing with humans of different personalities, incrementally understanding their intent, extracting the background details needed, accurately invoke APIs, and operate over a complex business logic structure.

Despite their potential, building robust and reliable AI agents presents significant challenges [50]. Recent benchmarks reveal that even advanced LLMs struggle with multi-turn interactions, particularly

*Co-first Authors
†Core Contributors
‡Corresponding Authors

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Track on Datasets and Benchmarks.

those involving complex function calls, track long-term dependencies, or request missing information [51, 28, 47, 46, 19]. Although framework design and prompt engineering have shown promise, the underlying model capabilities remain the primary bottleneck, largely due to two fundamental obstacles: (1) the scarcity of high-quality agent interaction data in public pretraining corpora, and (2) the prohibitive cost and time required to manually collect and label such data, especially for domain-specific applications requiring specialized knowledge.

Several approaches have aimed to address these challenges. APIGen [26] generated single-turn function-calling data, and [42] investigated knowledge distillation for agent training. However, these focus predominantly on single-turn interactions, overlooking the complexity of real-world agent usage, which often involves multi-turn exchanges. While efforts such as [52, 51, 11] incorporate multi-turn elements, they lack human-agent interplay—essential for generating realistic data. The synthesis and verification of high-quality multi-turn trajectories that exhibit both linguistic diversity and grounded actions remains an open challenge, hindering progress in agent development.

To address these limitations, we introduce `APIGen-MT`, an agentic data synthesis pipeline for generating high-quality multi-turn agent data. It operates in two main steps: first, a data agent generates a detailed and verified task "blueprint", and second, this blueprint guides the generation of realistic multi-turn interactions through *simulated agent-human interplay* (Subsection 4.2). The blueprint generation includes sampling relevant APIs, policies, domain data, and user personas to create grounded general tasks configurations, and using *reverse task recombination* (SubSubsection 4.1.3) to enhance complexity. These blueprints are validated through format/execution checks and an LLM committee review using a reflection-based mechanism [39]. Subsequently, the validated blueprint seeds a simulated interaction between a human LM and an agent (e.g., `gpt-4o`), producing a complete interaction trajectory with dialogue, actions, and environment feedback for training.

The main contributions of our work are summarized as follows:

- We propose `APIGen-MT` (Figure 1), an agentic data synthesis pipeline that generates high-quality multi-turn agentic data leveraging environment execution feedback and a review committee.
- We develop a two-phase framework that first creates detailed task blueprints with verifiable groundtruth actions, then transforms these blueprints into realistic multi-turn conversational agent trajectories with tool-usage through simulated human-agent interplay.
- We train a series of models across multiple architectures and scales (Llama 3.1/3.2 and Qwen 2.5 at 1B to 70B parameters), demonstrating superior performance on two popular agentic benchmarks: τ-bench and BFCL, surpassing many frontier models including `gpt-4o`.
- We conduct comprehensive ablation studies investigating the impact of data quality and quantity on agent performance, providing valuable insights for future research in agent model training.
- We open-source our high-quality synthetic data containing 5K trajectories and trained models, i.e., the `xLAM-2-fc-r` series, to advance research in AI agent space.

2 Related Work

Tool-Use Agents. Tool-use capabilities enhance LLMs by enabling interaction with external tools, extending their reasoning and functionality [45, 33, 24]. Function-calling frameworks allow LLMs to parse queries, select tools, and interpret results, but often require predefined tools, limiting adaptability [26, 44]. Efforts were made to address this by creating reusable tools from scratch on the fly [9], built upon by ToolMaker [45] which leverages tools from existing code repositories. Others compose workflows or learn from demonstrations [33, 36]. Recently, several works have adopted specialized approaches for agent training—critique-informed planning [13], fine-tuning on selective steps [49], teasing apart reasoning from format following (Agent-FLAN) [12], and autonomously invoking tools without explicit post-training (ToRL) [22].

Interactive Conversational Benchmarks. Evaluating LLM agents in multi-turn settings requires specialized benchmarks. MultiChallenge [40] and ToolDial [38] assess agents on context maintenance and tool-augmented dialogue. InterCode [48] and CRMArena [19] evaluate iterative problem-solving and customer management. ToolSandbox [28] provides a stateful, interactive benchmark for tool use. User simulations have become essential in these benchmarks, offering systematic, realistic interactions [50, 28, 31]. Our work complements these efforts by generating synthetic multi-turn conversations to train and evaluate agents in such realistic settings.

Synthetic Data Generation. The scarcity of high-quality data drives synthetic data generation. Multi-agent frameworks like MAG-V [37], AgentInstruct [29], MATRIX [43], and IntellAgent [20] simulate agent interactions to create realistic datasets. Other approaches include instruction composition [18, 11], intermediate graphs [6], multi-turn planning [56] for generating complex dialogues. BUTTON [11], related to our multi-turn data generation efforts, synthesizes compositional instruction tuning data by combining 2–3 atomic tasks and collecting trajectories via a multi-agent setup. However, it needs manual API construction and lacks quality control during task composition, limiting data verification. MAGNET [51] uses a graph-based approach to generate function signature paths, which are iteratively transformed into sequences of queries and function calls.

3 APIGen-MT Method for Synthesizing High-Quality Multi-Turn Data

3.1 Multi-Turn Interaction Problem Formulation

Multi-turn interactions between an AI assistant and a human user present unique challenges that go beyond single-turn exchanges. We formalize this interaction as a Partially Observable Markov Decision Process (POMDP) defined by the tuple $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{U}$ represents the instruction space containing possible user intents; $\mathcal{S}$ denotes the state space of the environment and conversation history; $\mathcal{A} = \{tool\_call, response\}$ is the action space available to the assistant; $\mathcal{O} = \mathcal{O}_E \cup \mathcal{O}_H$ is the observation space comprising observations from the environment ($\mathcal{O}_E$) and response from the human ($\mathcal{O}_H$); $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S} \times \mathcal{O}$ is the transition function; and $\mathcal{R}$ is the reward function evaluating interaction success. The assistant must engage in dialogue to incrementally infer the human's intent $q \in \mathcal{U}$ and solve it through appropriate interactions with the environment while adhering to any domain rules. At turn t, the assistant predicts an action $a^t \in \mathcal{A}$ based on the interaction history and understanding of q thus far. When a^t is a $tool\_call$ compliant with the rules, it triggers a state transition $(s_E^t, tool\_call) \to (s_E^{t+1}, o_E)$, where $o_E \in \mathcal{O}_E$ is the tool output (typically in structured format like JSON). When a^t is a $response$ to the human, it causes a state transition $(s_H^t, response) \to (s_H^{t+1}, o_H)$, where $o_H \in \mathcal{O}_H$ is the human's follow-up message. Importantly, the environment state s_E^{t+1} remains latent to both the assistant and the human. The interaction terminates when the user sends a concluding message or a predefined turn limit is reached. The reward $\mathcal{R}(\Delta\mathcal{S}_E, a)$ is calculated based on the cumulative state change in the environment $\Delta\mathcal{S}_E$ and the sequence of responses $a = \{a_i \mid a_i \in response \text{ to human}\}$ provided by the assistant throughout the episode. The assistant's objective is to maximize this reward.

3.2 APIGen-MT Framework Overview

Generating high-quality multi-turn data that captures the complexities of agent-human interactions presents significant challenges. Directly synthesizing multi-turn conversations in one shot is difficult for two key reasons: (1) a single error or hallucination in any intermediate step can lead to complete failure, and (2) the content of each turn depends on previous function calls and their outputs, creating complex dependencies that are difficult to maintain consistently. To address these challenges, we introduce APIGen-MT, a two-phase framework for generating verifiable and diverse multi-turn data (Figure 1). Our approach extends the APIGen framework [26] by adding an agentic feedback loop and simulated human-agent interplay to generate realistic multi-turn conversations. The core insight of our approach is to separate the task generation process into two distinct phases: first creating a detailed "blueprint" of the task (Phase 1), and then using this blueprint to guide the generation of realistic multi-turn interactions that fill in the conversational details (Phase 2). This separation ensures both correctness of the underlying task structure and the naturalness of the resulting conversations.

3.2.1 Phase 1: Task Configuration and Groundtruth Generation

The initial phase of APIGen-MT focuses on systematically generating well-defined task configurations, each comprising a user intent (q), a corresponding sequence of verifiable groundtruth actions (a_{gt}), and the expected final outputs (o_{gt}). This phase establishes a solid, verifiable foundation for each interaction scenario before the complexities of conversational dynamics are introduced. As depicted in Figure 1, this is achieved through an agentic workflow incorporating multi-stage validation and refinement loops. More specifically, it has the following steps:

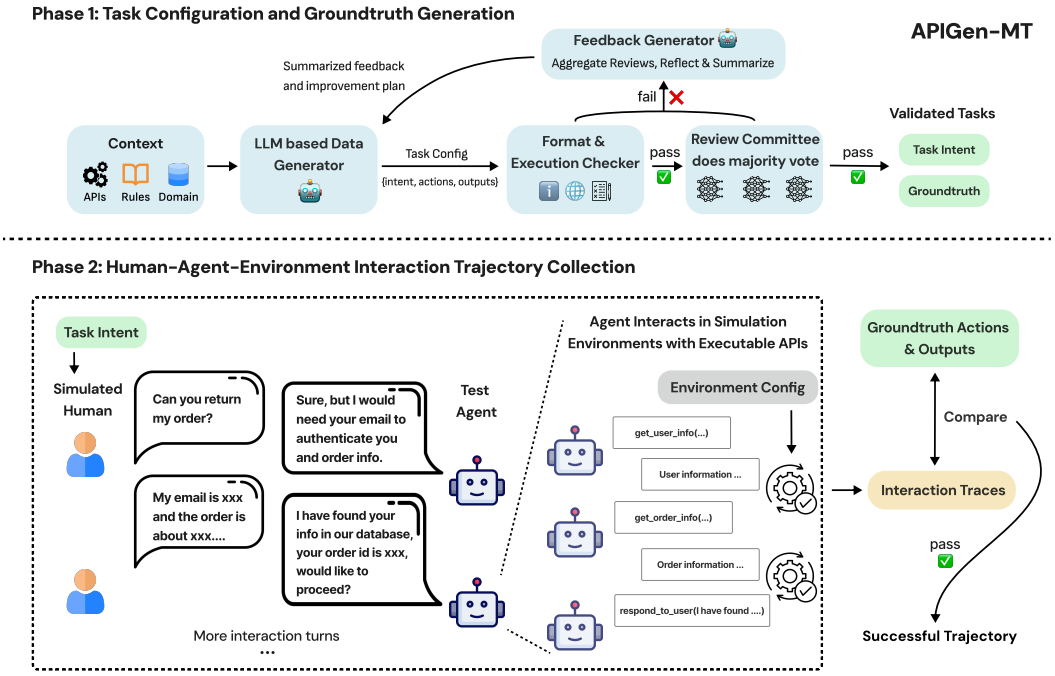

Figure 1: Overview of the `APIGen-MT`. Phase 1 generates task configurations and groundtruth actions through an agentic process with feedback loops. Phase 2 collects human-agent-environment interaction trajectories by simulating realistic conversations between a human user and a test agent in an executable environment.

1. **Context Preparation**: Relevant information such as available APIs, domain-specific rules or policies, and reference data is assembled. This context grounds the subsequent generation step in the specific constraints and capabilities of the target environment.
2. **LLM-based Data Generator**: An LLM utilizes the prepared context to propose initial task configurations. Each configuration consists of:
   - A detailed user intent q describing the high-level intent/instructions.
   - A sequence of groundtruth actions a_{gt} required to fulfill the intent.
   - Expected final outputs o_{gt} to be provided to the user.
3. **Format & Execution Checker**: Proposed configurations undergo automated technical validation. This component performs multiple checks:
   - Verifies the structural correctness of generated actions (e.g., valid API call formats) and outputs.
   - Confirms the executability of each action in a_{gt} within a simulated target environment E (checking API names, arguments, types).
4. **Review Committee**: Configurations passing rule-based checks proceed to semantic evaluation by a committee of multiple LLM reviewers. This committee assesses quality aspects like the coherence between q and a_{gt}, completeness, and overall task sensibility. We use majority voting to achieve a more stable assessment.
5. **Feedback Generation and Refinement**: If a task fails at either the validation (Step 3) or review (Step 4) stage, a Feedback Generator aggregates failure reasons and reviews, reflects upon them, and produces a summarized improvement plan. This plan guides the Data Generator (Step 2) in refining the task proposal in a subsequent iteration. Successfully validated tasks exit this loop.

This agentic design with feedback loops is crucial for generating high-quality tasks efficiently. By incorporating reflection and improvement based on validation results, the system can learn from failures and progressively generate better tasks. This structured exploration creates a vast combinatorial space of unique, goal-oriented scenarios that are both complex and coherent, deliberately trading infinite, noisy possibility for meaningful interactions.

3.2.2 Phase 2: Human-Agent-Environment Interaction Trajectory Collection

Building on validated task configurations q, a_{gt}, o_{gt} from Phase 1, Phase 2 simulates realistic multi-turn interactions between an LLM-based human user and a test agent in an executable environment.

Guided by the task intent q and often a specific persona, the simulated human naturally reveals information or sub-goals incrementally, while the agent interprets the evolving context, interacts with the environment via API calls when needed, and responds coherently. Importantly, the simulated user is unaware of the underlying environment and available APIs mimicking a real-world user. The simulation produces complete interaction trajectories that capture dialogue turns, agent actions, and environment responses. Each trajectory is validated by comparing its outcome against the groundtruth actions (a_{gt}) and expected outputs (o_{gt}) from Phase 1. Only trajectories that pass both state- and output-based checks are retained, ensuring that interactions are both dynamically plausible and grounded in a correct solution. This two-phase design offers several benefits. First, it provides verifiability by grounding interaction data in pre-validated task configurations. Second, it enhances realism by focusing the simulation on natural turn-by-turn dynamics without the simultaneous burden of task solution generation. Lastly, the modular approach isolates issues in task design from those in conversational modeling, facilitating debugging and scalability across diverse interaction patterns. In essence, by integrating agentic generation of verifiable task "blueprint" with realistic simulation of conversational dynamics, `APIGen-MT` produces high-quality, multi-turn interaction data that balances structural correctness with the naturalness required for training agent models.

4 A Case Study of `APIGen-MT` on τ-bench

This section details the instantiation of the `APIGen-MT` framework (Subsection 3.2) with τ-bench [50]. τ-bench, with its realistic domains, executable APIs, and specific policies, provides an ideal testbed for this methodology. Figure 7 in Appendix D illustrates this specific implementation.

4.1 Phase 1 Implementation: Task Configuration Generation and Validation

4.1.1 API Dependency Graph and Context Samplers

Generating realistic tasks for τ-bench requires navigating its specific APIs, policies, and data structures. We implemented the following techniques for task generation and validation.

API Graph Modeling. We model the available APIs in each τ-bench domain as a directed graph, where nodes represent APIs and edges represent dependencies between them. An edge exists from API A to API B if B's input arguments can depend on A's output and the co-occurrence of this tool-call pair is permitted under domain policies. This graph-based approach enables us to generate realistic task sequences by performing random walks through the API dependency graph.

Specialized Context Samplers. To ensure task diversity, realism, and grounding, we utilize several domain-specific samplers that provide context to the LLM-based task generator.

- **API Sampler**: We distinguish between state-exploring ('read') APIs and state-changing ('write') APIs which can modify the environment states. The generator focuses on sampling the necessary 'write' APIs to form the core of a_{gt}, allowing flexibility in how 'read' APIs might be used during the subsequent interaction phase.
- **Policy Sampler**: We sample from the domain-specific policies and rules. These policies are incorporated into the task generation to ensure compliance of real-world use cases. Task complexity is influenced by the number of 'write' calls and the associated policy constraints.
- **Domain Data Sampler**: To ground tasks in realistic domain data without exceeding context limits, we sample domain-specific data with additional metadata (e.g., cost, time, attributes). This metadata enhances coverage and enables more creative and diverse task scenarios.
- **Persona Sampler**: We incorporate user persona descriptions from PersonaHub [17] to inform the user intent q and inject realistic human qualities and situational context, enhancing diversity for subsequent Phase 2 human-agent interaction simulation.
- **Example Sampler**: We provide few-shot examples of well-formed tasks relevant to the sampled APIs, guiding the generator on structure and format.

For each task generation iteration, we randomly vary the sampling frequency for each sampler to enhance diversity and prevent repetitive scenarios. The sampled information is compiled into a prompt instructing the LLM generator to produce a `<thought>` (its reasoning), the user `<intent>` (q), the corresponding groundtruth `<actions>` (a_{gt}), and the expected final `<outputs>` (o_{gt}).

4.1.2 Multi-Stage Validation for τ-bench

Stage 1: Action Validation.

- **Format Check**: Verifies the presence and basic structure of required task components (`<thought>`, `<intent>`, `<actions>`, `<outputs>`) and ensures all tool calls in `<actions>` are valid JSON and outputs in `<outputs>` are strings.
- **Execution Check**: Simulates each action in a_{gt} within the τ-bench environment, validating API names, argument names, and data types. The cumulative effect on the environment state ($\Delta\mathcal{S}_E$) is captured as a `diff_patch`, similar to `git diff`.
- **Policy Compliance Check**: Leverages the executable nature of τ-bench by translating domain policies into Python unit tests. These tests run against the simulated execution trace of a_{gt} to detect violations, especially those arising from interactions between multiple actions. Failures yield detailed feedback on the specific policy violation.

Stage 2: Alignment Validation. Tasks successfully passing Stage 1's action validation are then assessed for semantic alignment. Specifically, we evaluate whether the groundtruth actions (a_{gt}), as reflected by their environmental effects summarized in the `diff_patch`, accurately and comprehensively fulfill the user's intent expressed in the instruction (q). To mitigate the potential biases and inconsistencies of a single evaluator, we employ a committee of diverse LLM judges [55, 8]. These judges review each task based on a systematic rubric with metrics such as Correctness, Completeness, Satisfaction, and Creativity (refer Figure 9 in Appendix E for details). Each judge provides scores and qualitative feedback. We utilize a majority voting strategy across the committee's judgments to determine the final assessment for each metric and the overall task quality. This approach yields more stable and reliable evaluation results compared to single-judge assessments.

Stage 3: Final Semantic Review & Refinement. Based on the aggregated scores from the committee (determined via majority voting), tasks achieving an average score above a predefined threshold are accepted and added to the pool of validated task configurations. Failing tasks trigger the feedback loop mechanism. Consolidated feedback, summarizing the points raised by the committee majority, is sent back to the LLM task generator. This initiates a reflection process [39], guiding the generator to revise the task in the subsequent iteration to address the identified shortcomings.

4.1.3 Reverse Task Recombination for Complex Task Construction

While the refinement process improves generation quality and efficiency, directly generating complex, long-horizon tasks remains challenging. Validation failures can occur due to subtle policy conflicts or difficulties in ensuring perfect alignment across many steps. To overcome this and systematically construct more complicated scenarios, we implement *Reverse Task Recombination*, a technique that leverages the principle of compositionality [11, 18], similar to modular design in software engineering. The core idea is to build complex tasks from simpler, independently validated "building blocks":

1. **Select Validated Tasks**: Identify multiple simpler tasks ($T_1, T_2, ...$) that have successfully passed all validation stages (Stages 1-3) and are associated with the same user persona.
2. **Concatenate Components**: Combine their respective groundtruth actions ($a_{combined} = a_{gt,1} \circ a_{gt,2} \circ ...$) and expected outputs ($o_{combined} = o_{gt,1} \oplus o_{gt,2} \oplus ...$, where $\circ$ denotes action sequence concatenation and $\oplus$ denotes output aggregation).
3. **Re-Check Policy Compliance**: Rerun Policy Check on $a_{combined}$ to ensure that the cumulative action sequence remains logically sound and adheres to the domain rules as combinations could cause conflicting actions to appear together, for e.g., returning and canceling the same order.
4. **Synthesize Combined Intent**: Instruct the generator to create a new coherent user instruction ($q_{combined}$) that logically integrates the goals and steps represented by $a_{combined}$ and $o_{combined}$. This new instruction should frame the combined actions as a single, more complex user request.
5. **Re-Validate Semantics**: Submit the newly formed complex task $T_{combined} = \{q_{combined}, a_{combined}, o_{combined}\}$ for validation starting from Stage 2 (Alignment Validation). Stage 1 (Action Validation) can be safely skipped for $a_{combined}$ because each constituent action sequence ($a_{gt,1}, a_{gt,2}, ...$) has already been individually checked for format and execution within its original context, and policy compliance in the current context. Stage 3 (Final Semantic Review) proceeds based on the outcome of Stage 2 for the combined task.

This method allows for the generation of complex, multi-step tasks reliably, as it builds upon verified components while focusing the validation effort on the semantic coherence of the combined whole.

4.2 Phase 2: Simulated Human-Agent Interplay for Trajectory Collection

Building on the verified tasks from Phase 1—which include a detailed user intent q, groundtruth actions a_{gt}, and expected outputs o_{gt}—we simulate multi-turn interaction trajectories between an agent (A) and a human user (H) modeled by an LLM. Guided by the instruction q and an associated persona, the simulated human incrementally reveals task details to mimic realistic interactions. The agent, instantiated as `gpt-4o` with its function-calling mode, interprets the evolving intent and executes the necessary actions to complete the task. A critical challenge is maintaining its stability and fidelity. We implement a Best-of-N (BoN) with self-critique mechanism (more details in Appendix §B.1) to combat this and validate its effectiveness on the τ-bench test set. We observe improved success rate ($6\% \uparrow$) and reduced variance ($3\% \downarrow$) (refer Table 8 in Subsection B.1). We use rejection sampling to retain only successful trajectories ($r = 1$), verified by matching the final environment state to a_{gt} and agent responses to o_{gt}. Each task is attempted up to three times, and all unique successful runs are aggregated into an offline dataset for downstream use.

4.3 Data Collection & Statistics

Data Collection Procedure. We source APIs (5 'read' and 13 'write') implemented as Python functions and domain rules from the Retail and Airline environments of τ-bench. We utilize `gpt-4o` and DeepSeek V3 models in the task generation, validation and agent-human interplay stages to collect training data. The prompts used in every stage are provided in Appendix E. We set the maximum number of reflection-based feedback turns to 3 for retail and 5 for airline respectively.

Table 1: Statistics for the dataset generated using `APIGen-MT`. Success rates (S.R.) reported for the task configuration (w. and w/o agentic feedback in Phase 1) and trajectory simulation (Phase 2) stages.

Metric	Value
Task Config. S.R. (Phase 1)	70%
Task Config. S.R. w/o Agentic Feedback	28%
Trajectory Simulation S.R. (Phase 2)	67%
Min. Turns per Trajectory	1
Max. Turns per Trajectory	29
Avg. Tool Calls per Trajectory	7
Avg. User Turns per Trajectory	6

Statistics. A summary of the data collection is shown in Table 1. We can efficiently collect long trajectories requiring a strong model like `gpt-4o` to take an average 12 turns to complete the task using `APIGen-MT` (Figure 4 in Appendix). Our agentic pipeline involving review committee and iterative refinement via reflection provides a **2.5x** boost to the task collection success rate to attain 70%. This demonstrates that `APIGen-MT` can reliably generate high-quality, multi-turn data in complex domains with strict policy constraints, enabling the creation of diverse, realistic, and verifiable datasets for training and evaluation of conversational agents. Data curation cost is provided in Subsection C.2.

Human Expert Assessment of Synthesized Data.

To evaluate the quality and realism of data generated, we conducted an expert review of 200 sampled trajectories from the τ-bench dataset. As shown in Table 2, the data reflects high quality across both task query and trajectory dimensions. **99.4% of queries were deemed clearly understandable**, and the agent **successfully completed the task in 99%** of cases, with **zero fatal errors** and a low erroneous turn rate (0.75%). The sufficiency and achievability rates are naturally lower and improve over the turns as user intent is gradually revealed. Full annotation guidelines and further review insights are in Appendix C.

Table 2: Annotation results evaluating the clarity and feasibility of user queries and correctness of agent behavior.

Type	Metric (at turn level)	Score %
User Query	Clarity: Easy to read and understand	**99.4**
	Sufficient Info: Has all needed context	70.2
	Achievability: Solvable with domain APIs	77.4
Agent Action	Fatal Error: Unrecoverable mistake	**0.0**
	Erroneous Step: Minor mistake in action	0.75
	Self-Correction: Agent fixes own error	0.0
	Task Completed: Solved by final turn	**99.0**

5 Experiments

5.1 Experimental Setup

We perform filtered Behavioral Cloning (BC) using the collected trajectories with Llama 3.1/3.2 Instruct models [16] and Qwen 2.5 Instruct models [34]. We evaluate on two challenging benchmarks designed specifically for assessing agent capabilities, that focus on multi-turn interactions and tool use capabilities, which are central to our data generation methodology – **(1) BFCL v3** [46], a leading benchmark for tool-use evaluation, specifically designed to assess LLMs' function calling capabilities

and **(2) τ-bench** [50], a comprehensive benchmark for evaluating AI agents in realistic scenarios. More experimental details are in Appendix A.

5.2 Experiment Results

Table 3: Performance of different models on BFCL leaderboard (as of date 04/03/2025). The rank is based on the overall accuracy, which is a weighted average of different evaluation categories. "FC" stands for function-calling mode in contrast to using a customized "prompt" to extract the function calls. See the benchmark [46] for details.

Rank	Overall Acc	Model	Single-Turn			Multi-Turn	Hallucination	
			Non-live (AST)	Non-live (Exec)	Live (AST)	Overall Acc	Relevance	Irrelevance
1	**78.19**	**xLAM-2-70b-fc-r (FC)**	**88.48**	**85.98**	**72.63**	**75.12**	**66.67**	**78.74**
2	**75.83**	**xLAM-2-32b-fc-r (FC)**	**89.50**	**86.48**	**73.79**	**66.38**	**83.33**	**76.25**
3	74.31	watt-tool-70b (FC)	84.06	89.39	77.74	58.75	94.44	76.32
4	**72.83**	**xLAM-2-8b-fc-r (FC)**	**84.35**	**85.59**	**66.73**	**69.25**	**83.33**	**64.11**
5	72.08	GPT-4o-2024-11-20 (Prompt)	88.1	89.38	79.83	47.62	83.33	83.76
6	69.94	GPT-4.5-Preview-02-27 (FC)	86.12	83.98	79.34	45.25	66.67	83.64
7	69.58	GPT-4o-2024-11-20 (FC)	87.42	89.2	79.65	41	83.33	83.15
8	68.39	ToolACE-2-8B (FC)	87.58	87.11	80.05	36.88	72.22	90.11
9	67.98	watt-tool-8B (FC)	86.56	89.34	76.5	39.12	83.33	83.15
10	67.88	GPT-4-2024-04-09 (FC)	84.73	85.21	80.5	38.12	72.22	83.81
11	67.87	o1-2024-12-17 (Prompt)	85.67	87.45	80.63	36	72.22	87.78
12	67.72	BitAgent-8B	86.92	89.52	76.14	38.5	83.33	82.38
13	65.12	o3-mini-25-01-31 (Prompt)	86.15	89.46	79.08	28.75	72.22	82.96
14	**65.11**	**xLAM-2-3b-fc-r (FC)**	**82.94**	**81.88**	**58.69**	**56.00**	**94.44**	**57.94**
15	64.1	CoALM-405B	90.58	89.07	74.5	28.75	100	71.79
16	64.1	GPT-4o-mini-24-07-18 (FC)	85.21	83.57	74.41	34.12	83.33	74.75
...	...	...	...					
34	58.93	Gemini-2-Flash-Thinking	87.4	87.07	75.97	14.5	77.78	72.75
35	58.9	Qwen2.5-14B-Instruct (FC)	85.42	84.86	76.68	15.88	55.56	77.69
36	**58.90**	**xLAM-2-1b-fc-r (FC)**	**76.23**	**74.86**	**59.88**	**43.12**	**88.89**	**56.87**
37	58.55	DeepSeek-V3 (FC)	89.17	92.32	68.41	18.62	88.89	59.36
38	58.45	mistral-large-2407 (FC)	86.81	84.38	69.88	23.75	72.22	52.85
39	58.42	ToolACE-8B (FC)	87.54	89.21	78.59	7.75	83.33	87.88

BFCL Results. Our models demonstrate exceptional performance on the BFCL v3 benchmark. As shown in Table 3, `xLAM-2-70b-fc-r` and `xLAM-2-32b-fc-r` top the leaderboard with overall accuracies of 78.19% and 75.83%, outperforming all proprietary and open-source models. The most striking advantage appears in multi-turn scenarios, where our models excel across all parameter scales: `xLAM-2-70b-fc-r` achieves 75.12% accuracy, while smaller models also excel—`xLAM-2-8b-fc-r` at 69.25%, `xLAM-2-3b-fc-r` at 56.00%, and `xLAM-2-1b-fc-r` at 43.12%—all significantly ahead of o1 (36%) and `gpt-4o` (41%) in function-calling mode. Additionally, our models demonstrate strong hallucination detection, with `xLAM-2-3b-fc-r` achieving 94.44% on relevance detection, matching the best score in this category.

τ-bench Results. Table 4 presents results under the default *naive* user setting on τ-bench. Our `xLAM-2-70b-fc-r` model achieves a 56.2% success rate, outperforming Llama 3.1 70B Instruct (38.2%), DeepSeek v3 (40.6%), and even proprietary models like GPT-4o (52.9%), while approaching more recent models like Claude 3.5 Sonnet (60.1%). Notably, our smaller variants like `xLAM-2-32b-fc-r` (54.6%) and `xLAM-2-8b-fc-r` (46.7%) surpass larger baselines, demonstrating that our synthetic data approach enables efficient knowledge transfer and strong performance with fewer parameters.

These results demonstrate that our `APIGen-MT` approach—using simulated agent-human interplay to generate multi-turn data—is highly effective. Models trained on this data consistently outperform open-source baselines and rival proprietary models, with especially strong multi-turn performance. Notably, it enables smaller models to match or exceed the performance of much larger ones, underscoring the efficiency of our method.

Table 4: Success Rate ($pass@1$) of various models on the Retail and Airline settings of τ-bench (averaged across at least 5 trials). The `xLAM-2-fc-r` models are trained on the data generated using `APIGen-MT`. Overall indicates the average score across both domains. [1] indicates results from [14]; [2] indicates results from [2]; [3] indicate results from [3]; [4] indicates from [4]. **Note.** We evaluate only with the benchmark's *think* tool and no prompt optimizations.

Model	**τ-Retail**	**τ-Airline**	**Overall**
Open-Source Models			
Qwen 2.5 32B Instruct	24.4	25.0	24.7
Llama 3.1 70B Instruct	50.4	26.0	38.2
DeepSeek v3[1]	58.3	22.8	40.6
xLAM-2-70b-fc-r	67.1	45.2	56.2
xLAM-2-32b-fc-r	64.3	45.0	54.6
xLAM-2-8b-fc-r	58.2	35.2	46.7
xLAM-2-3b-fc-r	44.4	32.0	38.2
xLAM-2-1b-fc-r	22.5	21.0	21.8
Proprietary Models			
Gemini 1.5 pro[1]	54.9	25.2	40.1
`gpt-4o-2024-11-20`	62.8	43.0	52.9
`o1`[3]	73.5	54.2	63.9
Claude 3.5 Haiku[2]	51.0	22.8	36.9
Calude 3.5 Sonnet[2]	62.6	36.0	49.3
Claude 3.5 Sonnet (new)[3]	71.5	48.8	60.1
Claude 3.7 Sonnet[4]	78.3	41.2	59.8
Claude 3.7 Sonnet + prompt optim[4]	81.2	58.4	69.8

5.3 Consistency Experiments

We plot the **passˆk** curves [50] in Figure 2 on τ-bench using the default *naive* user LM setting. passˆk measures the probability that all k i.i.d. task trials succeed, averaged across tasks. As k increases, our models show a smaller drop in success rate (SR). Notably, on the more complex airline domain, `xLAM-2-70b-fc-r` achieves a higher passˆ5 than Claude, despite a slightly lower passˆ1—indicating greater reliability and consistency across trials. This is crucial for real-world deployment, where consistent performance is essential.

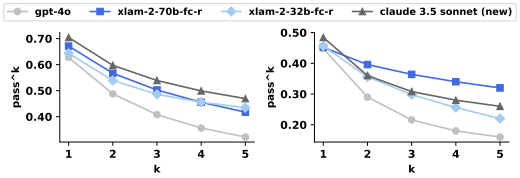

Figure 2: Passˆk curves measuring the probability that all 5 trials succeed for a given task, averaged across tasks for retail (left) and airline (right) domains.

5.4 RQ. What is the effect of task granularity / trajectory length?

As mentioned in SubSubsection 4.1.3, we use Reverse Task Recombination to synthesize denser task instructions requiring more interaction turns by combining the intents of individual successful tasks. This allows us to study the effect of having complex tasks producing more granular trajectories. We train a 32B model without data from Reverse Task Recombination, thereby comprising majorly of short trajectories, but maintain the number of trajectories same as the original setup to control for confounding factors related to scale. We categorize tasks into 'short', 'medium' and 'long' based on the number of turns Claude 3.5 requires to solve them across the union of 8 trials, using the 33rd and 66th percentiles as thresholds. Figure 3 clearly illustrates how performance drops across all categories suggesting the importance of having diverse length trajectories. The effect is more pronounced in the Airline domain ($45\% \rightarrow 33\%$) than in Retail ($64\% \rightarrow 61\%$).

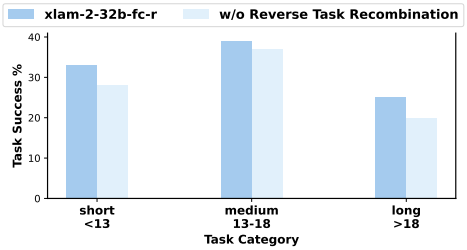

Figure 3: Effect of removing Reverse Task Recombination data on τ-bench.

5.5 RQ. What do component-wise ablations reveal about system behavior?

We further perform an ablation analysis to disentangle the contributions of key components across the task generation and trajectory construction pipeline. Removing individual modules substantially degrades both the validity and diversity of generated trajectories (Table 1). Specifically, excluding the feedback and refinement stages reduces the overall task success rate from 70% to 28%, indicating their central role in stabilizing generation quality. Omitting validation or consistency checks introduces ill-formed or infeasible task formulations that later cascade into execution failures; while disabling the recombination module produces shorter, less interactive trajectories, highlighting its importance for maintaining interaction depth. Collectively, these findings illustrate that the system's effectiveness emerges from the interplay of complementary components, with robustness hinging on preserving structural and semantic feedback loops throughout task construction.

5.6 RQ. Can multi-turn data generated for one domain benefit another?

Domains where the assistant interacts with a simulated human turn-wise, invoking APIs to fulfill user intent while following policy constraints, are considered in-domain; this includes the Retail and Airline settings from τ-bench. In contrast, BFCL is considered out-of-distribution relative to τ-bench, with a broader scope spanning 8 domains. As shown in Table 5, while filtered BC benefits generalization, synthesizing interaction trajectories for specific domains remains essential for achieving better performance.

Additionally, we evaluate our xLAM models on two agentic benchmarks that exhibit substantial distributional shifts relative to the original training data—introducing novel APIs, action spaces, intents, and reasoning patterns. **NexusRaven [41]** is a diverse function-calling benchmark comprising

318 test examples spanning 65 distinct APIs. **CRMAgentBench**[4] assesses an agent's proficiency in realistic Customer Relationship Management (CRM) scenarios, testing topic identification, precise function execution, and context-aware response generation. Derived from hundreds of expert-assessed, real-world CRM interactions, it provides a rigorous measure of a model's practical utility and reliability for enterprise deployment.

Table 5: (Left) In-Domain; (Right) Cross-Domain Generalization. The setting refers to the environment from which the synthetic data to train the base model is sourced. (base) indicates the base model and (all) indicates the full training set used to train the `xLAM-2-fc-r` series. ↑ is the relative performance gain w.r.t to the base setting.

Model (setting)	τ-Retail	τ-Airline
Qwen 32B (base)	24.4	25.0
Qwen 32B (retail only)	–	36.0 (44% ↑)
Qwen 32B (airline only)	49.1 (101% ↑)	–
xLAM-2-32b-fc-r (all)	**64.3** (163% ↑)	**45.0** (80% ↑)

Model (setting)	τ-bench	BFCL
Qwen 32B (base)	24.7	57.8
Qwen 32B (retail + airline)	–	65.0 (13% ↑)
xLAM-2-32b-fc-r (all)	**54.6** (121% ↑)	**75.8** (31% ↑)

Table 6: Out-of-Domain Generalization. Performance on NexusRaven function-calling benchmark.

Model	P_{api}	R_{api}	$F1_{api}$
Llama-3.3-70b-inst	0.917	0.934	0.925
Mistral-latest-12b-inst	0.906	0.940	0.923
GPT-4o-2024-11-20	0.943	0.840	0.889
GPT-4.1-2025-04-14	0.841	0.846	0.843
DeepSeek-r1-671b	0.837	0.840	0.838
xLAM-2-70b-fc-r	**0.934**	**0.940**	**0.937**

Table 7: Out-of-Domain Generalization. Performance on CRMAgentBench benchmark.

Model	Topic Acc	Function Call Acc	Free Text Acc	Average Acc
DeepSeek-r1-671b	0.82	0.83	0.94	0.86
o1-preview	0.98	0.75	0.81	0.85
Llama-3.3-70b-inst	0.99	0.72	0.80	0.84
GPT-4-turbo	0.99	0.60	0.92	0.83
xLAM-2-70b-fc-r	**0.95**	**0.83**	**0.84**	**0.874**

Results in Table 6 and Table 7 show that xLAM-2, trained with `APIGen-MT` data, generalizes effectively to these out-of-distribution domains. This highlights the strength of our blueprint-driven framework in producing domain-adaptive training data that supports the development of models that are both robust and practically applicable.

6 Discussion

Conclusion. We introduced `APIGen-MT`, a two-phase framework for generating high-quality multi-turn agent data via simulated human-agent interactions. By decoupling the creation of detailed task blueprints from the simulation of conversational trajectories, our approach ensures both structural correctness and natural dialogue dynamics. Experiments on τ-bench and BFCL v3 demonstrate that models trained on our synthetic data outperform existing baselines, with even smaller models showing competitive performance in multi-turn scenarios. Moreover, our stabilization techniques yield more consistent and reliable agent behavior. By open-sourcing our synthetic data and trained models, we aim to foster further advances in AI agent development.

Limitations and future directions. Despite its advantages, `APIGen-MT` has limitations that open avenues for future work. First, while our Best-of-N sampling and self-critique mechanisms reduce human simulation variance, some stochasticity still remains; more deterministic simulation methods or refined filtering metrics could improve stability. Second, our current approach discards failed trajectories in the second phase, yet these cases may offer valuable insights; future work could leverage such failures as additional contrastive signal during model training. Third, the multi-stage validation process, though effective, incurs computational overhead; developing more efficient validation or adaptive sampling strategies could improve scalability. Finally, extending to new domains and integrating self-improvement through reinforcement learning present promising future directions.

[4]https://www.salesforceairesearch.com/crm-benchmark

References

[1] S. Agashe, J. Han, S. Gan, J. Yang, A. Li, and X. E. Wang. Agent s: An open agentic framework that uses computers like a human. *arXiv preprint arXiv:2410.08164*, 2024.

[2] Anthropic. Claude 3.5 sonnet, 2024. URL https://www.anthropic.com/news/3-5-models-and-computer-use.

[3] Anthropic. Claude 3.7 sonnet, 2025. URL https://www.anthropic.com/news/claude-3-7-sonnet.

[4] Anthropic. Claude think tool, 2025. URL https://www.anthropic.com/engineering/claude-think-tool.

[5] A. Antoniades, A. Örwall, K. Zhang, Y. Xie, A. Goyal, and W. Wang. Swe-search: Enhancing software agents with monte carlo tree search and iterative refinement. *arXiv preprint arXiv:2410.20285*, 2024.

[6] S. Arcadinho, D. Aparício, and M. Almeida. Automated test generation to evaluate tool-augmented llms as conversational ai agents. *arXiv preprint arXiv:2409.15934*, 2024.

[7] D. Bahdanau, N. Gontier, G. Huang, E. Kamalloo, R. Pardinas, A. Piché, T. Scholak, O. Shliazhko, J. P. Tremblay, K. Ghanem, et al. Tapeagents: a holistic framework for agent development and optimization. *arXiv preprint arXiv:2412.08445*, 2024.

[8] Z. Bi, K. Han, C. Liu, Y. Tang, and Y. Wang. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*, 2024.

[9] T. Cai, X. Wang, T. Ma, X. Chen, and D. Zhou. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*, 2023.

[10] CAMEL-AI.org. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. https://github.com/camel-ai/owl, 2025. Accessed: 2025-03-07.

[11] M. Chen, sunhaoze, T. Li, F. Yang, H. Liang, KeerLu, B. CUI, W. Zhang, Z. Zhou, and weipeng chen. Facilitating multi-turn function calling for LLMs via compositional instruction tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=owP2mymrTD.

[12] Z. Chen, K. Liu, Q. Wang, W. Zhang, J. Liu, D. Lin, K. Chen, and F. Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models. *arXiv preprint arXiv:2403.12881*, 2024.

[13] Z. Chen, M. Li, Y. Huang, Y. Du, M. Fang, and T. Zhou. Atlas: Agent tuning via learning critical steps. *arXiv preprint arXiv:2503.02197*, 2025.

[14] S. Cognition. Apt-1 blog, 2025. URL https://www.scaledcognition.com/blog/apt-1.

[15] T. Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

[16] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[17] T. Ge, X. Chan, X. Wang, D. Yu, H. Mi, and D. Yu. Scaling synthetic data creation with 1,000,000,000 personas, 2024. URL https://arxiv.org/abs/2406.20094.

[18] S. A. Hayati, T. Jung, T. Bodding-Long, S. Kar, A. Sethy, J.-K. Kim, and D. Kang. Chain-of-instructions: Compositional instruction tuning on large language models. *arXiv preprint arXiv:2402.11532*, 2024.

[19] K.-H. Huang, A. Prabhakar, S. Dhawan, Y. Mao, H. Wang, S. Savarese, C. Xiong, P. Laban, and C.-S. Wu. Crmarena: Understanding the capacity of llm agents to perform professional crm tasks in realistic environments, 2025. URL https://arxiv.org/abs/2411.02305.

[20] E. Levi and I. Kadar. Intellagent: A multi-agent framework for evaluating conversational ai systems. *arXiv preprint arXiv:2501.11067*, 2025.

[21] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[22] X. Li, H. Zou, and P. Liu. Torl: Scaling tool-integrated rl, 2025. URL `https://arxiv.org/abs/2503.23383`.

[23] Y. Li, Y. Li, X. Wang, Y. Jiang, Z. Zhang, X. Zheng, H. Wang, H.-T. Zheng, P. Xie, P. S. Yu, et al. Benchmarking multimodal retrieval augmented generation with dynamic vqa dataset and self-adaptive planning agent. *arXiv preprint arXiv:2411.02937*, 2024.

[24] W. Liu, X. Huang, X. Zeng, X. Hao, S. Yu, D. Li, S. Wang, W. Gan, Z. Liu, Y. Yu, et al. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*, 2024.

[25] Z. Liu, J. Zhang, K. Asadi, Y. Liu, D. Zhao, S. Sabach, and R. Fakoor. Tail: Task-specific adapters for imitation learning with large pretrained models. *arXiv preprint arXiv:2310.05905*, 2023.

[26] Z. Liu, T. Hoang, J. Zhang, M. Zhu, T. Lan, J. Tan, W. Yao, Z. Liu, Y. Feng, R. RN, et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37:54463–54482, 2024.

[27] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[28] J. Lu, T. Holleis, Y. Zhang, B. Aumayer, F. Nan, F. Bai, S. Ma, S. Ma, M. Li, G. Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. *arXiv preprint arXiv:2408.04682*, 2024.

[29] A. Mitra, S. Patel, T. Chakrabarty, and C. Baral. Agentinstruct: An agentic framework for generating high-quality synthetic instruction data. *arXiv preprint arXiv:2402.12360*, 2024.

[30] J. Pan, X. Wang, G. Neubig, N. Jaitly, H. Ji, A. Suhr, and Y. Zhang. Training software engineering agents and verifiers with swe-gym. *arXiv preprint arXiv:2412.21139*, 2024.

[31] J. Pan, R. Shar, J. Pfau, A. Talwalkar, H. He, and V. Chen. When benchmarks talk: Re-evaluating code llms with interactive feedback, 2025. URL `https://arxiv.org/abs/2502.18413`.

[32] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

[33] Y. Qin, S. Hu, Y. Lin, W. Chen, N. Ding, G. Cui, Z. Zeng, X. Zhou, Y. Huang, C. Xiao, et al. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):1–40, 2024.

[34] Qwen, :, A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

[35] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He. 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*, KDD '20, page 3505–3506, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3406703. URL `https://doi.org/10.1145/3394486.3406703`.

[36] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

[37] S. Sengupta, K. Curtis, A. Mallipeddi, A. Mathur, J. Ross, and L. Gou. Mag-v: A multi-agent framework for synthetic data generation and verification. *arXiv preprint arXiv:2412.04494*, 2024.

[38] J. Shim, G. Seo, C. Lim, and Y. Jo. Tooldial: Multi-turn dialogue generation method for tool-augmented language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=J1J5eGJsKZ.

[39] N. Shinn, F. Cassano, A. Gopinath, K. R. Narasimhan, and S. Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

[40] V. Sirdeshmukh, K. Deshpande, J. Mols, L. Jin, E.-Y. Cardona, D. Lee, J. Kritz, W. Primack, S. Yue, and C. Xing. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. *arXiv preprint arXiv:2501.17399*, 2025.

[41] V. K. Srinivasan, Z. Dong, B. Zhu, B. Yu, H. Mao, D. Mosk-Aoyama, K. Keutzer, J. Jiao, and J. Zhang. Nexusraven: a commercially-permissive language model for function calling. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023. URL https://openreview.net/forum?id=Md6RUrGz67.

[42] H. Su, R. Sun, J. Yoon, P. Yin, T. Yu, and S. Ö. Arık. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. *arXiv preprint arXiv:2501.10893*, 2025.

[43] S. Tang, X. Pang, Z. Liu, B. Tang, R. Ye, X. Dong, Y. Wang, and S. Chen. Synthesizing post-training data for llms through multi-agent simulation. *arXiv preprint arXiv:2410.14251*, 2024.

[44] J. Wang, J. Zhou, M. Wen, X. Mo, H. Zhang, Q. Lin, C. Jin, X. Wang, W. Zhang, and Q. Peng. Hammerbench: Fine-grained function-calling evaluation in real mobile device scenarios. *arXiv preprint arXiv:2412.16516*, 2024.

[45] G. Wölflein, D. Ferber, D. Truhn, O. Arandjelović, and J. N. Kather. Llm agents making agent tools. *arXiv preprint arXiv:2502.11705*, 2025.

[46] F. Yan, H. Mao, C. C.-J. Ji, T. Zhang, S. G. Patil, I. Stoica, and J. E. Gonzalez. Berkeley function calling leaderboard. 2024.

[47] J. Yang, A. Prabhakar, S. Yao, K. Pei, and K. R. Narasimhan. Language agents as hackers: Evaluating cybersecurity skills with capture the flag. In *Multi-Agent Security Workshop @ NeurIPS'23*, 2023. URL https://openreview.net/forum?id=KOZwk7BFc3.

[48] J. Yang, A. Prabhakar, K. Narasimhan, and S. Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

[49] R. Yang, F. Ye, J. Li, S. Yuan, Y. Zhang, Z. Tu, X. Li, and D. Yang. The lighthouse of language: Enhancing llm agents via critique-guided improvement. *arXiv preprint arXiv:2503.16024*, 2025.

[50] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan. Tau-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

[51] F. Yin, Z. Wang, I.-H. Hsu, J. Yan, K. Jiang, Y. Chen, J. Gu, L. T. Le, K.-W. Chang, C.-Y. Lee, H. Palangi, and T. Pfister. Magnet: Multi-turn tool-use data synthesis and distillation via graph translation, 2025. URL https://arxiv.org/abs/2503.07826.

[52] Y. Zeng, X. Ding, Y. Wang, W. Liu, W. Ning, Y. Hou, X. Huang, B. Qin, and T. Liu. Boosting tool use of large language models via iterative reinforced fine-tuning. *arXiv preprint arXiv:2501.09766*, 2025.

[53] J. Zhang, T. Lan, M. Zhu, Z. Liu, T. Hoang, S. Kokane, W. Yao, J. Tan, A. Prabhakar, H. Chen, et al. xlam: A family of large action models to empower ai agent systems. *arXiv preprint arXiv:2409.03215*, 2024.

[54] J. Zhang, T. Hoang, M. Zhu, Z. Liu, S. Wang, T. Awalgaonkar, A. Prabhakar, H. Chen, W. Yao, Z. Liu, et al. Actionstudio: A lightweight framework for data and training of action models. *arXiv preprint arXiv:2503.22673*, 2025.

[55] K. Zhang, W. Yao, Z. Liu, Y. Feng, Z. Liu, R. Rithesh, T. Lan, L. Li, R. Lou, J. Xu, et al. Diversity empowers intelligence: Integrating expertise of software engineering agents. In *The Thirteenth International Conference on Learning Representations*, 2024.

[56] Y. Zhang, J. Lu, and N. Jaitly. Probing the multi-turn planning capabilities of LLMs via 20 question games. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1495–1516, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.82. URL https://aclanthology.org/2024.acl-long.82/.

[57] Y. Zheng, R. Zhang, J. Zhang, Y. Ye, Z. Luo, Z. Feng, and Y. Ma. 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, 2024. Association for Computational Linguistics. URL http://arxiv.org/abs/2403.13372.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See the Abstract and Introduction sections.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Limitations paragraph in Section 6

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5 and Subsection A.2 describes the experimental setup and training details. Code and data is also provided in Supplementary.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Dataset collected is provided on HuggingFace. Code to reproduce evaluation experiments is provided.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 and Subsection A.2 describes the experimental setup and training details. Code and data is provided.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Where relevant, we re-run experiments over 5 trials. Table 4

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All details are provided in Section 5

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm that our research conform with the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See the discussions in Appendix F for details.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: See the discussions in Appendix F for details.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All licenses are included in the HuggingFace assets.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Dataset and model details are included in the submission.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: A small sample of our data annotation and evaluation was performed by human experts from a third-party supplier. The detailed annotation guidelines provided to the experts are included in the Appendix C. Compensation was based on the number and complexity of annotated trajectories, and the supplier confirmed that its personnel receive a living wage.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA] .

    Justification: IRB approval was not required, as our work only involved synthetic data generation and professional annotators performing routine tasks with no sensitive personal information collected.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

A Experimental Details

A.1 Benchmarks Description

- **BFCL v3**: It introduces comprehensive evaluation across single-turn, multi-turn, and multi-step function calling scenarios. BFCL v3 evaluates models on their ability to understand user requests, select appropriate functions, generate valid parameters, and interpret function outputs across multiple interaction turns. The benchmark uses a weighted average of different evaluation categories to provide an overall accuracy score.
- **τ-bench**: It measures an agent's ability to interact with simulated human users (powered by language models) and programmatic APIs while following domain-specific policies. τ-bench emulates dynamic conversations across multiple domains, including retail and airline customer service, requiring agents to maintain context across turns, understand user intents, and follow complex domain-specific rules. The benchmark emphasizes the importance of multi-turn interactions and policy adherence in real-world applications.

A.2 Training Details

The collected trajectories are split at every assistant response and we train to predict only the assistant response tokens by masking the prompt and other messages. To enhance the dataset diversity, we also jointly train our `xLAM-2-fc-r` models with function-calling data from [26] and other domains of agentic data from [53, 54]. We utilize the LLama-Factory library [57] and perform full-finetuning using DeepSpeed ZeRO [35] stage 3, Flash Attention 2 [15] in bfloat16 precision with AdamW optimizer [27] and train for at most 3 epochs on a NVIDIA H200 node.

B Further Insights

B.1 Best-Of-N (BoN) Simulated Human Stabilization

Over multiple conversational turns, the human LLM may drift from the original instruction or be unduly influenced by the agent's responses [32], introducing variability that hinders reliable evaluation [50]. To address this, we adopt a Best-of-N (N=4) sampling strategy in combination with a self-critique mechanism for the human LLM's responses (see Figure 12 in Appendix E for details), allowing it to adhere to the task intent more accurately and not be mislead by the test agent responses. Its effectiveness was validated on the τ-bench test set, where improved consistency in agent performance evaluation across multiple trials was observed (Table 8). Although this enhancement is applied to the user LM, it suggests that enhancing the user simulation strategy with a simple self-critiquing mechanism can not only increase stability but also improve agent performance.

Table 8: Success Rate (SR) across 5 trials on the Retail domain of τ-bench using `gpt-4o` and `xLAM-2-70b-fc-r` as the test assistants. The average SR is higher with lower variance using *BoN*, indicative of a more stable evaluation.

Model (User LM setting)	t1	t2	t3	t4	t5	Avg	Var
`gpt-4o` (Naive)	61.7	57.4	65.2	65.2	64.4	62.8	11.1
`gpt-4o` (BoN)	65.2	69.6	67.0	66.1	67.0	**67.0**	**2.6**
`xLAM-2-70b-fc-r` (Naive)	69.6	65.2	62.6	68.7	69.6	67.1	9.7
`xLAM-2-70b-fc-r` (BoN)	66.9	71.3	68.7	66.9	70.4	**68.8**	**4.0**

B.2 Dataset Statistics.

B.3 How does performance scale with data size?

We investigate how model performance improves as the size of synthetic data increases. To control for external variables, we source data from and test on τ-bench by training a Qwen2.5-32b model. We include 3K single-turn function calling samples from [26] to prevent overfitting and demonstrate the tool-calling format to the model, and train for 2 epochs. Figure 5 shows that the model is benefited as the the data scales up. Remarkably, even a small dataset of **just 32 samples** enhances the base model's capability in function calling, especially for the retail environment.

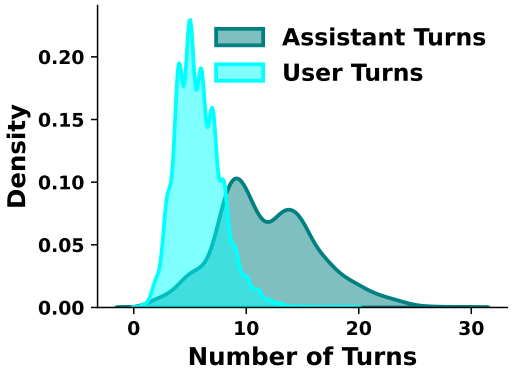

Figure 4: Density distribution of assistant and user turns in collected trajectories.

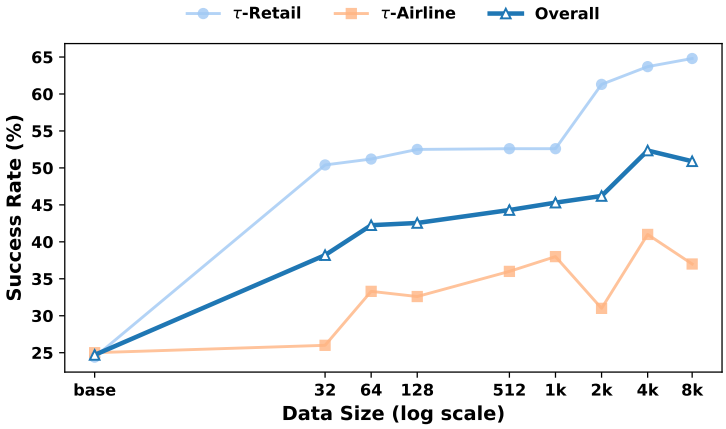

Figure 5: Data scaling laws on τ-bench Retail and Airline environments.

B.4 In-Depth Analysis of Model Behavior

To better understand the behavior of our trained models, we perform an in-depth investigation of the tasks solved by `xLAM-2-70b-fc-r` and a state-of-the-art model Claude 3.5 Sonnet (new) on τ-bench. From Figure 6 we observe that particularly on the 'long' task category, the success rate for `xLAM-2-70b-fc-r` is much higher than `gpt-4o` but lags behind Claude. Further, we assess the efficiency of the agent by measuring the number of interactions needed with the user for the agent to fully comprehend the intent and successfully complete the task. The plot reveals that `xLAM-2-70b-fc-r` is at par with `gpt-4o` but requires more interactions compared to Claude, which can be attributed to its method of retrieving user details in stages, necessitating more turns. These observations suggest potential areas for improvement in future iterations.

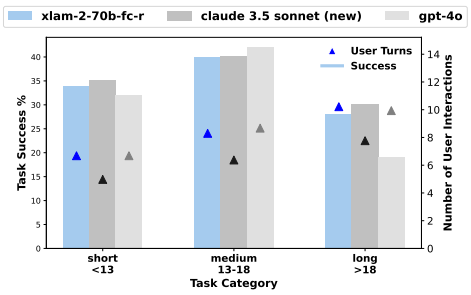

Figure 6: Performance and efficiency comparisons of `xLAM-2-70b-fc-r` with frontier models on τ-bench.

C Human Expert Annotation Insights

C.1 Guidelines for Evaluating the Synthesized Dataset.

The human experts were provided with synthesized trajectory data consisting of one or multiple queries, tools, and a number of rounds. A "round" is defined as one inference from the model (i.e., one message from the "assistant"). The annotation task included the following components:

- **Query Annotation:** Experts evaluated if the user's query was clear, contained sufficient information for the agent to proceed, and was achievable with the given tools. Query complexity was assessed by estimating the number of steps and potential solutions.
- **Plan Annotation:** Experts wrote a brief plan outlining the steps required to address the query.
- **Agent Response Annotation:** Experts checked if the agent responded with plain text or described its thought process and made tool calls. They identified self-correction steps, errors in tool choice, argument correctness, hallucinations, logical errors, and evaluated the reasonableness of the action plan. Errors were categorized, described, and fixes suggested (except for redundant steps). Finally, they determined if the task was completed or if a fatal error occurred.

C.2 Cost Comparison

`APIGen-MT` **Costs.** The average cost for generating one task blueprint was $0.33, while generating one full trajectory averaged $0.75.

Human Costs. Human annotation costs were based on the number of interaction rounds per trajectory: $10 for fewer than 5 rounds, $13 for 5–10 rounds, $18 for 10–15 rounds, and $35 for 15–20 rounds.

Notably, human experts were only asked to evaluate and annotate existing data, not to synthesize new data. Even with this limited scope, the associated costs were relatively high. In contrast, our data synthesis pipeline using `APIGen-MT` proves to be significantly more efficient both in terms of cost and scalability.

D `APIGen-MT` adaptation for τ-bench

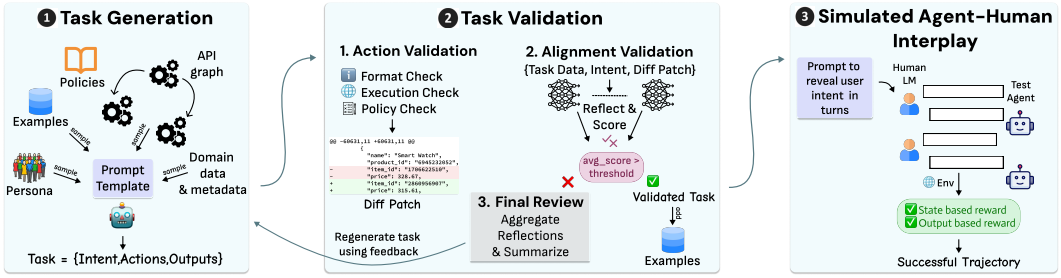

Figure 7: Realization of `APIGen-MT` framework for τ-bench. We first generate realistic task instances by random walk down the API graph and sampling. Next the tasks are validated following a multi-stage pipeline. Instances which fail are sent back to the Generator to be refined based on the validation feedback. Finally, trajectories are generated by a simulated human user that interacts with a test agent by supplying the query details in a turn-wise manner. Trajectories which pass state- and output- based evaluations are collected.

E Prompts

The prompts used across the various stages of `APIGen-MT` implemented for τ-bench are shown here – Task Configuration Generation (Figure 8), Alignment Validation (Figure 9), Final Semantic Review (Figure 10), Trajectory Collection (Figure 11), Stabilized Human Simulation (Figure 12).

Task Configuration Generation Prompt

Instructions
Generate a 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 (q)
{task_rules + domain_rules}

User Data
{sampled_user_details}

Order Data
{sampled_orders}

Guidelines for generating Groundtruth Actions (a_{gt})
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' (o_{gt}). 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.

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

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 Tasks
{example}

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 8: Task configuration generation prompt for retail domain of τ-bench.

F Broader Impacts and Safeguards

Broader impacts: Releasing APIGen-MT models and data can enable researchers and developers to build more capable conversational agents for beneficial applications such as customer service, education, or accessibility, thereby advancing AI and productivity. Providing verified multi-turn agent blueprints can improve the reliability and transparency of agent behaviors. However, open access also poses risks: malicious actors might exploit the data or models to generate realistic disinformation, spam, or automate harmful tasks. Additionally, any biases or unsafe behaviors present in the generated data could be propagated if not carefully filtered. We therefore recommend that users apply our data responsibly, monitor outputs for misuse or bias, and implement appropriate oversight when deploying agent systems.

Safeguards: We acknowledge that open-sourcing agent models poses misuse risks. Our release on HuggingFace will include clear documentation, usage guidelines, and a license that emphasize responsible use. We stress that the models are research prototypes that may produce unsafe or biased outputs and should not be deployed in critical systems without human oversight. We recommend that downstream users implement additional content filtering or monitoring, and we will advise adhering to any relevant terms of service. These minimal safeguards are intended to raise awareness of limitations and encourage responsible use, rather than imposing strict access restrictions.

Task Alignment Validation Prompt

You are an AI judge and your goal is to judge the quality and validity of the provided task object based on the guidelines, following the rubric.

Guidelines
- The task object contains an 'intent' (q) from a user, 'actions' (a_{gt}), and 'outputs' (o_{gt}).
- The 'actions' correspond to the tool_calls made by an AI assistant to satisfy the instruction.
- A description of the 'tools' available to the AI assistant is provided.
- The 'diff_patch' is the difference in the database state after the tool_calls are made. It should only reflect changes corresponding to the 'intent'. There should be no extraneous changes. If the 'diff_patch' is empty, it means that the tool_calls did not change the database state, which is possible if the instruction was to provide information only.
- Perform a brief reflection on the task based on the below Rubrics.
- Think step-by-step to generate a score of 0 or 1 for each of these criteria (1 means follows criterion and 0 means does not)

Rubric
- *Correctness*: Do the actions (a_{gt}) accurately implement the instruction (q)?
- *Completeness*: Is the instruction (q) sufficiently detailed, and is it fully addressed by the actions? (Includes rule-based checks).
- *Satisfaction*: Do the expected outputs (o_{gt}) fulfill any explicit or implicit information requests within the instruction (q)?
- *Creativity*: Does the task represent a non-trivial, plausible, and potentially interesting scenario within the domain?

Task Object
{task}

Tools in Python format
{tools}

Diff Patch
{diff_patch}

Output format
<scores>
{{
    "reflection": str, <a brief high-level review of the task>
    "correctness": int, <0/1>,
    "completeness": int, <0/1>,
    "satisfaction": int, <0/1>,
    "creativity": int, <0/1>,
    "total": int, <total score out of 4>
    "correction": str, <brief explanation and suggested correction (if needed)>
}}
</scores>

Figure 9: Task alignment validation prompt for τ-bench. This is sent to each LM in the review committee to get their scores, following which we employ majority voting.

Final Semantic Review Prompt

You are responsible for analyzing and summarizing feedback from multiple AI judges. Your primary goal is to provide clear, actionable feedback that will help the generator LLM improve its future outputs. You do not evaluate the task directly; instead, you review and grounding the existing feedback from the AI judges.

Review Process
- Begin by analyzing individual reflections and scores from each judge.
- Summarize common points of agreement or disagreement.
- Offer a concise summary of actionable feedback to be sent back to the data generator, which aims to improve the next round of data quality.

Diff Patch
{diff_patch}

Generated Task Data
{task}

AI Judges' Feedback
{reviews}

Output Format
Generate your response according to the following format. Enclose the thought process within '<thought></thought>' tags, and the final summary of actionable feedback within '<summary></summary>' tags.

Figure 10: Final semantic review prompt for τ-bench.

Trajectory Collection Prompt

You are a detail-oriented user interacting with an AI agent.

Intent
{intent}

Rules
- Generate one line at a time to simulate the user's message.
- Do not give away all the intent at once. Only provide the information that is necessary for the current step.
- Do not hallucinate information that is not provided in the intent.
- If the intent goal is satisfied, generate '###STOP###' to end the conversation.
- Do not repeat the exact intent in the conversation. Instead, use your own words to convey the same information.
- Try to make the conversation as natural as possible and stick to the personalities in the intent.

Figure 11: Trajectory collection prompt for τ-bench.

BoN User LM Setting Prompt

You are a fair judge and an expert in following details.

A human is interacting with a retail assistant to get help on solving their task. You are provided with the description of the human and the task the human wants to accomplish (wrapped with <description></description>), and a candidate response (wrapped with <response></response>) the human wants to give the assistant. Please help the human evaluate this candidate response, give an integer score (ranging from 0 to 10) to indicate the correctness of the response, higher score means better quality.

1. If the response includes specific item / order / personal details, and they correctly match the task description you should give full score of 10. If there is some change in details, give a corresponding lower score (more incorrect details gets lower score).
2. The response can include any normal conversation otherwise (e.g. asking details, saying ###STOP###) etc. which are all correct responses.
3. Additionally, if the candidate_response keeps the conversation flowing by describing the task clearly / gives information properly then give a high score and if not (e.g. "I don't remember" or unhelpful response) should get a corresponding lower score.

<description> {description} </description>

<response> {response} </response>

After scoring using the mentioned guideline, tell me your score, wrap it in <score></score> tags.

Figure 12: Best-of-N (BoN) User LM setting prompt used in the retail domain of τ-bench.