SIMULATING ENVIRONMENTS WITH LARGE LANGUAGE MODELS FOR GENERIC AGENT TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents excel in compact environments requiring deep reasoning but remain brittle when operating in broader, more complex contexts that demand robustness across diverse tools and schemas. Building bespoke environments for training is costly and brittle, limiting progress. We propose SIMAGENT, a framework that simulates diverse environments with reasoning models to generate complete trajectories from compact seed sets without executing real systems. Our approach combines diversity expansion with schema-verified generation, producing synthetic data that is both broad and training-ready. Fine-tuning open models on these trajectories yields consistent improvements across multiple benchmarks, in some cases surpassing GPT-4 and approaching GPT-4o, demonstrating that environment simulation provides a generic path for advancing agentic LLMs.

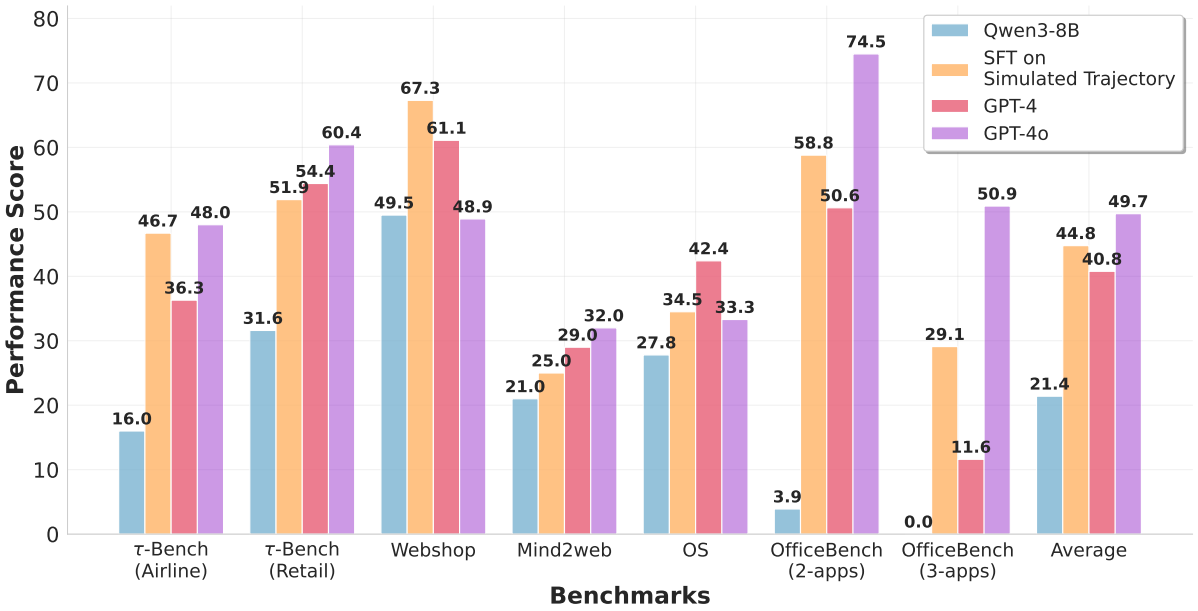

Figure 1: Performance of Qwen3-8B fine-tuned on synthetic trajectories generated by SIMAGENT without real environment implementations. Fine-tuning on simulated trajectories (orange) yields substantial gains over the base model and, on several benchmarks (e.g., τ-Bench), surpasses GPT-4 and approaches GPT-4o. Averaged across all benchmarks, the SIMAGENT-trained models achieve higher scores than GPT-4, highlighting the effectiveness of simulation-based synthetic data for improving agent performance.

1 INTRODUCTION

Large language model (LLM) agents are becoming increasingly capable and autonomous. Advances in reasoning mechanisms have enabled them to perform multi-step planning, tool use, and problem solving, to the point of surpassing human experts in narrow but cognitively demanding domains such as mathematics competitions (e.g. the International Mathematical Olympiad (Luong & Lockhart, 2025)) or programming contests (e.g. the ICPC (Lin & Cheng, 2025)). We refer to these as *complex-task/simple-environment* settings: the core challenge lies in solving a difficult reasoning problem, while the environment itself is well-specified, compact, and easily set up.

In contrast, LLM agents continue to struggle with tasks that are individually simple but situated in broad, messy, or dynamic contexts. Examples include operating vending machines, office workflows, or household tasks (Backlund

& Petersson, 2025). We call these *simple-task/complex-environment* settings. The reasoning required for each step may be minimal, yet success depends on robustness across a wide distribution of environments, tools, and edge cases.

However, scaling agents to handle diverse environments remains challenging. Progress requires not just reasoning skill within a single domain, but robustness across a wide variety of tools, interfaces, and contexts. Existing benchmarks and environments tend to be narrow in scope, often tied to specific APIs or domains, which limits their ability to capture the breadth of real-world variability. As a result, agents trained in these settings struggle to transfer effectively when faced with new tasks or altered schemas.

We argue that training agents across a wide variety of environments will be essential for progress on simple-task/complex-environment challenges. However, doing so is far from straightforward. Engineering bespoke environments is costly, brittle, and often reliant on mock data and fragile APIs, which poorly approximate real-world variability (Dulac-Arnold et al., 2019). In this paper, we propose a new agent trajectory synthetic approach called SIMAGENT: simulating diverse environments and trajectory with another LLM. By leveraging LLMs as environment simulators, SIMAGENT can cover a broader range of realistic contexts while maintaining controllability. This enables agents to explore a significant breadth of applications and provides a scalable path toward mastering simple-task/complex-environment challenges.

Two design pillars make this practical. Diversity expansion transforms seed trajectories into broader goal, tool, and query variants, while strict schema enforcement preserves structural fidelity in actions, tool calls, and observations. In practice, format fidelity is critical: even minor schema drift such as mismatched argument keys, omitted JSON fields, or inconsistent delimiters can materially degrade training yield. Schema enforcement converts informal textual rollouts into structured, machine-verifiable supervision, and automatic repair mitigates common deviations, increasing the proportion of trajectories that are immediately usable. Together, these components yield synthetic data that is both diverse and training-ready.

In that sense, our approach differs from traditional distillation. Rather than merely relabeling existing inputs, it expands the task distribution, enforces structural fidelity absent from raw teacher traces, and produces smaller models that are more robust to schema perturbations while still bounded by the semantic correctness of the synthesizer.

As an empirical preview, fine-tuning open models such as `Qwen3-8B` on SIMAGENT trajectories delivers substantial gains across multiple benchmarks (Wang et al., 2024; Yao et al., 2024; Barres et al., 2025; Liu et al., 2023), in some cases surpassing GPT-4 and approaching GPT-4o (Figure 1). Improvements include higher aggregate scores and enhanced resilience to tool schema variation and injected observation noise. This robustness is difficult to obtain from narrow augmentation strategies alone.

In summary, this paper makes the following contributions:

- **Benchmark-agnostic trajectory synthesis.** We introduce a framework that generates complete agent trajectories from compact seed sets without executing real environments, eliminating per-benchmark engineering overhead.
- **Schema-verified generation.** We design a strict format checker with automatic repair to enforce structural fidelity and prevent errors from propagating into training data.
- **Empirical gains and robustness.** Across diverse benchmarks, SIMAGENT-generated data yields consistent improvements for open models, enhancing resilience to schema perturbations and observation noise.

These results suggest a practical recipe for scaling agent training: replace environment-specific code with reasoning-model simulators and enforce disciplined data formatting to safeguard supervision quality. While SIMAGENT substantially improves scalability, its fidelity remains bounded by the semantic accuracy of the synthesizer; we mitigate compounding errors through schema repair and filtering. This reframes environment engineering as an amortized prompt-and-schema design challenge, enabling broader and more reproducible progress in agentic language models.

2 END-TO-END SYNTHETIC AGENTIC TRAJECTORIES GENERATION

A key challenge in advancing LLM agents is producing synthetic trajectories that generalize across heterogeneous benchmarks. Prior methods often depend on environment-specific tools or APIs, making the data generation process difficult to scale or transfer. We introduce a framework for end-to-end synthetic agentic trajectory generation, where complete trajectories are simulated directly from seed data without requiring real environments. This design enables scalable, benchmark-agnostic data creation, forming the foundation for improving agentic performance across tasks.

2.1 AGENTIC TRAJECTORY SIMULATOR

We consider the problem of generating synthetic trajectories for agentic tasks in a manner that is independent of benchmark-specific environments. Traditional approaches construct such trajectories by implementing environments with explicit tool interfaces and APIs. While this yields realistic interactions, it ties the trajectory generation process to the specifics of each benchmark, limiting reusability. Instead, we seek to generate trajectories directly from *seed data*, minimal task specifications such as natural language instructions or evaluation examples, without requiring environment implementations. Formally, given a set of seeds $\mathcal{D}_{\text{seed}}$, our goal is to synthesize a set of trajectories $\mathcal{D}_{\text{traj}}$ that (i) include synthetic tasks, model responses, actions and environment feedback in a multi-turn manner, (ii) are diverse in reasoning and action sequences, (iii) are high-quality in task completion, and (iv) are broadly applicable across benchmarks.

2.1.1 FRAMEWORK OVERVIEW

Our framework addresses the challenge of generating benchmark-agnostic synthetic agentic trajectories by simulating complete trajectories end-to-end from seed data. The key idea is to leverage reasoning models to generate both synthetic tasks and their complete trajectories, including reasoning steps and tool interactions, without relying on explicit environment implementations. This design allows trajectories to be created at scale with minimal manual intervention, while avoiding the benchmark-specific coupling present in prior methods.

Seed data consists of tasks drawn from the training corpus, which are first solved by an LLM to establish reliable outputs. To ensure high-quality seeds, we employ highly capable proprietary models (e.g., GPT models), as these models can provide more accurate and complete solutions. Starting from this seed data, our framework proceeds in three stages: (i) an LLM synthesizes tasks and simulates corresponding trajectories, yielding both task descriptions and stepwise reasoning/action/observation sequences, (ii) the resulting trajectories are filtered and curated, and (iii) the curated set is compiled into a dataset for downstream use. This process decouples data generation from environment design and enables consistent trajectory creation across heterogeneous benchmarks. An illustration of the framework is provided in Figure 2.

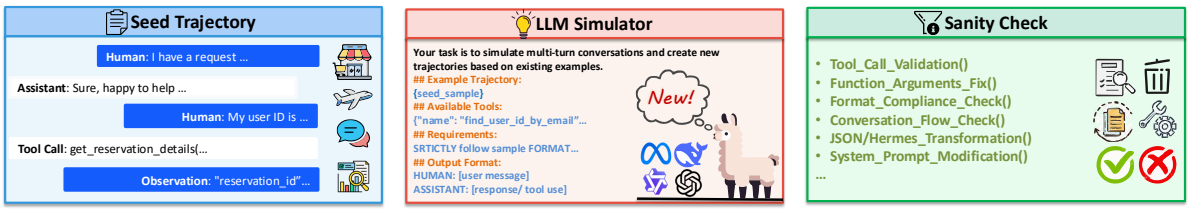

Figure 2: Trajectory simulation pipeline showing the flow from seed trajectory through LLM-based simulation to final sanity check.

2.1.2 TRAJECTORY GENERATION PROCESS

Given a set of seed tasks, our framework generates synthetic agentic trajectories through a structured process that leverages large language models as simulators. This process consists of three main components: prompt design, sampling and simulation, and filtering.

Prompt Design and Tool Constraints. Although our approach does not require implementing real environments, it is essential to ensure that generated trajectories remain valid with respect to the benchmark under consideration. To prevent the simulator from hallucinating non-existent tools or APIs, we incorporate the formal specification of the benchmark's tool and API interfaces directly into the prompt. This design anchors the generation process to the correct action space while avoiding the engineering burden of environment implementation. In practice, the prompt specifies the available tools, their input–output signatures, and usage rules, enabling the simulator to produce coherent sequences of reasoning steps and tool invocations.

The prompt also includes detailed instructions governing the format of the synthetic conversations. These instructions specify the number of turns, the expected conversation flow, the handling of context, and the output format to ensure consistency across generated trajectories. The complete prompt used is provided in Appendix C.

Sampling and Simulation. Starting from seed data, the LLM is prompted not only to generate a solution path but also to synthesize new agentic tasks inspired by the seed examples. For each such task, the model simulates a complete trajectory, producing alternating states and actions until task resolution. Sampling strategies such as temperature adjustment and multiple generation passes are employed to promote diversity, yielding a broad collection of trajectories that vary in task formulation, trajectory length, and action structure. This process enables the creation of rich and diverse synthetic trajectories from a small set of seeds.

Filtering and Quality Control. Not all generated trajectories are equally useful for downstream training. To curate high-quality data, we apply a filtering stage that discards trajectories with invalid tool usage, inconsistent reasoning, or trivial repetitions. Filtering criteria can be implemented either through automatic rule-based checks (e.g., verifying tool signatures) and model-based evaluation. The resulting dataset thus balances correctness, diversity, and task coverage, providing a reliable source of synthetic trajectories for improving LLM agents.

3 EXPERIMENTS

3.1 SETUP

Benchmarks.

To assess the generality of our approach, we evaluate on three complementary agentic suites that span distinct domains and interaction types: (1) τ-Bench (Airline, Retail) (Barres et al., 2025), a realistic tool-use benchmark emphasizing multi-turn API invocation, error recovery, and state tracking. We adopt the τ^2-Bench implementation (Barres et al., 2025) for evaluation, as it provides improved system optimization and task correction. Following prior work, we use GPT-4.1 as the user simulator (temperature set to 0) to generate user behaviors. (2) OfficeBench (Wang et al., 2024) (2-apps, 3-apps), which evaluates cross-application workflows, compositional tool use, and coordination across office utilities. (3) AgentBench (Liu et al., 2023) (Operating System, WebShop, Mind2Web), which spans software manipulation, e-commerce goal completion, and open-web browsing with long-horizon interactions. Together, these suites comprehensively stress multi-step reasoning and tool-use competency across distinct domains.

Training Details. To broaden coverage, we synthesize data from three seed datasets: (i) APIGen-MT (Prabhakar et al., 2025), which contains 5k training trajectories ($\sim$1.5k for τ-bench Airline and $\sim$3.5k for τ-bench Retail (Yao et al., 2024)); from this seed we generate 90k trajectories. (ii) AgentTuning (Zeng et al., 2023a), which provides 195 samples for Operating System (Liu et al., 2023), 351 for WebShop (Yao et al., 2022), and 122 for Mind2Web (Deng et al., 2023); we expand these into 15k trajectories spanning the three domains. (iii) OfficeBench (Wang et al., 2024), where we take 76 1-app tasks with o4-mini responses and generate 30k additional samples targeting multi-app settings. Unless otherwise noted, trajectory simulation is performed using GPT-5 and o4-mini models with temperature 1.0.

Table 1: Performance on τ-Bench (Airline, Retail) across proprietary models, open baselines, and models fine-tuned on SIMAGENT trajectories. SIMAGENT-trained models (**Sim-Tau**) consistently outperform size-matched baselines, narrow the gap to much larger proprietary systems, and in some cases even surpass them.

Model	Airline	Retail	Average
GPT Models			
GPT-5	**58.0**	**77.2**	**67.6**
o4-mini	57.0	69.3	63.2
GPT-4.1	53.0	65.2	59.1
GPT-4o	48.0	60.4	54.2
GPT-4	36.3	54.4	45.4
$\geq$ 32B Models			
Sim-Tau (Qwen2.5-32B)	**56.0**	**61.7**	**58.9**
xLAM-2-70B	49.3	**63.2**	56.3
xLAM-2-32B	46.0	57.3	51.7
Qwen2.5-32B-Instruct	25.0	48.7	36.9
7B/8B Models			
Sim-Tau (Qwen3-8B)	**46.7**	51.9	**49.3**
xLAM-2-8B	37.3	**52.1**	44.7
Sim-Tau (Qwen2.5-Coder-7B)	41.3	43.4	42.4
Sim-Tau (Qwen2.5-7B)	39.3	40.9	40.1
Sim-Tau (Llama-3.1-8B)	36.0	36.3	36.2
Qwen3-8B-APIGen-MT-5k	22.7	48.7	35.7
Qwen2.5-7B-Instruct-APIGen-MT-5k	22.0	38.7	30.4
Qwen3-8B	16.0	31.6	23.8
Qwen2.5-7B-Instruct	18.0	11.4	14.7
Qwen2.5-Coder-7B-Instruct	24.0	5.3	14.7
Llama-3.1-8B-Instruct	22.0	5.3	13.7
3B Models			
Sim-Tau (Llama-3.2-3B)	**36.0**	**32.0**	**34.0**
Sim-Tau (Qwen2.5-3B)	32.3	31.1	31.7
xLAM-2-3B	19.3	18.7	19.0
Llama-3.2-3B-Instruct	30.0	6.1	18.1
Qwen2.5-3B-Instruct	12.7	6.2	9.5
1.5B Models			
Sim-Tau (Qwen2.5-1.5B)	24.0	**17.6**	**20.8**
xLAM-2-1.5B	**30.7**	6.7	18.7
Qwen2.5-1.5B-Instruct	22.0	5.3	13.7

We fine-tune the synthesized trajectories on the Qwen2.5/3 and Llama 3.1/3.2 model families across multiple sizes. Each trajectory is segmented at assistant turns, and training is performed to predict only assistant tokens while masking prompts and other messages. Fine-tuning is conducted with LLaMA-Factory (Zheng et al., 2024) under a full-parameter setting, with hyperparameter details provided in Appendix B.1.

Baselines. For reference, we report results from proprietary models including GPT-4, GPT-4o, GPT-4.1, o4-mini, and GPT-5. For open-source baselines, we fine-tune Qwen2.5-7B-Instruct and Qwen3-8B on the seed datasets of each suite; APIGen-MT (5k), AgentTuning (668), and OfficeBench 1-app tasks with o4-mini trajectories (76) using the same training recipe as our method. We further include the xLAM-2 model family (Prabhakar et al., 2025) and AgentLM family (Zeng et al., 2023b), spanning model sizes from 1B to 70B.

3.2 RESULTS

τ-bench. Table 1 shows that our Sim-Tau models demonstrate substantial performance gains across the Airline and Retail domains of the τ-Bench, consistently rivaling or exceeding the capabilities of much larger models such as GPT-4, which attains an average score of 45.4. Notably, Sim-Tau (Qwen2.5-32B) achieves an average score of 58.9 (56.0 on Airline, 61.7 on Retail), outperforming GPT-4 by 13.5 points while trailing GPT-4.1 by only 0.2 points and o4-mini by 4.3 points. It also surpasses the baseline xLAM-2-32B (average 51.7) by 7.2 points and even exceeds xLAM-2-70B (average 56.3) by 2.6 points.

For 7B/8B-scale models, Sim-Tau (Qwen3-8B) attains an average score of 49.3 (46.7 on Airline, 51.9 on Retail), outperforming GPT-4 by 3.9 points and xLAM-2-8B (average 44.7) by 4.6 points, while approaching the performance of the larger xLAM-2-32B (average 51.7). Furthermore, our Sim-Tau variants based on Qwen3-8B and Qwen2.5-7B substantially outperform models fine-tuned on API generation seeds (APIGen-MT-5k), with Sim-Tau (Qwen3-8B) exceeding Qwen3-8B-APIGen-MT-5k (average 35.7) by 13.6 points and Sim-Tau (Qwen2.5-7B) surpassing Qwen2.5-7B-Instruct-APIGen-MT-5k (average 30.4) by 9.7 points. In the 3B and smaller parameter regimes, Sim-Tau models continue to exhibit strong gains over baselines. Sim-Tau (Llama-3.2-3B) achieves an average of 34.0 (36.0 on Airline, 32.0 on Retail), outperforming xLAM-2-3B (average 19.0) by 15.0 points and Llama-3.2-3B-Instruct (average 18.1) by 15.9 points. Similarly, Sim-Tau (Qwen2.5-3B) attains 31.7 on average, more than tripling the score of Qwen2.5-3B-Instruct (average 9.5). At the 1.5B scale, Sim-Tau (Qwen2.5-1.5B) reaches 20.8, surpassing xLAM-2-1.5B (average 18.7) by 2.1 points and Qwen2.5-1.5B-Instruct (average 13.7) by 7.1 points.

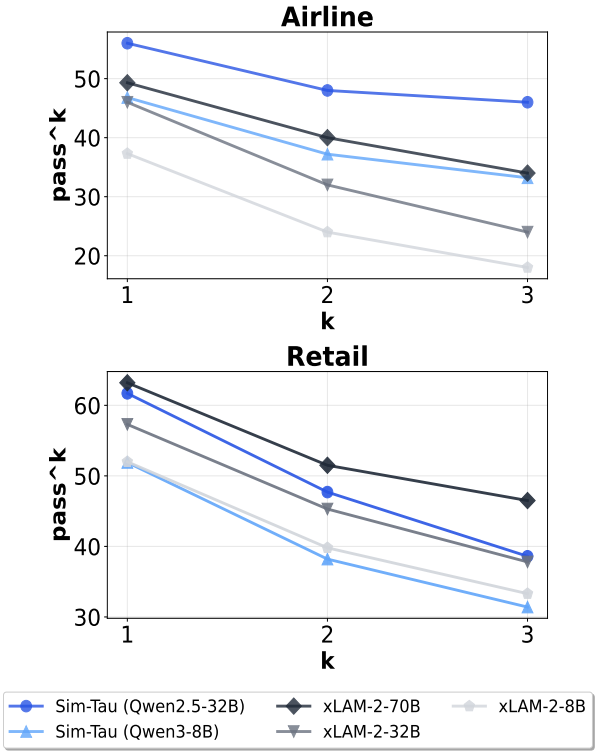

Figure 3: `Pass^k` performance comparison on the τ-Bench across Airline and Retail domains for Sim-Tau and xLAM-2 models, with k values of 1, 2, and 3. `Pass^k` requires that each task should be successful for all the k retries, highlighting the robustness.

Figure 3 illustrates the `Pass^k` (Barres et al., 2025) performance of Sim-Tau and xLAM-2 models on the τ-Bench across Airline and Retail domains, evaluating the effect of varying k values (1, 2, and 3) on task success rates. `Pass^k` (Barres et al., 2025) requires that each task should be successful for all the k retries to measure the robustness. Our Sim-Tau (Qwen2.5-32B) consistently achieves higher pass rates, with scores of 56.0, 48.0, and 46.0 for Airline, and 61.7, 47.7, and 38.6 for Retail, outperforming xLAM-2-70B Airline (49.3, 40.0, 34.0) and slightly less for retail (63.2, 51.5, 46.5) as k increases. Similarly, Sim-Tau (Qwen3-8B) demonstrates resilience with scores of 46.8, 37.2, and 33.2 for Airline, and 51.9, 38.2, and 31.4 for Retail, surpassing xLAM-2-8B (37.3, 24.0, 18.0 for Airline; 52.0, 39.8, 33.3 for Retail), underscoring the robustness of agentic ability for Our Sim-Tau.

These consistent gains highlight the quality of our synthesized dataset and the enhanced capabilities of our Sim-Tau models in schema understanding, precise API/tool calling, and robust state transitions within complex, multi-step, and multi-turn workflows.

OfficeBench. Table 2 shows that our Sim-OB models demonstrate substantial advancements in office automation tasks on the OfficeBench benchmark, which evaluates language agents' ability to handle complex workflows involving multiple applications such as Word, Excel, calendars, and emails. Notably, Sim-OB (Qwen3-8B) achieves an average score of 44.0 (58.8 on 2-apps, 29.1 on 3-apps), outperforming GPT-4 (average 31.1) by 12.9 points. Other variants, such as Sim-OB (Qwen2.5-7B) with 42.6 average and Sim-OB (Qwen2.5-Coder-7B) with 39.7, further highlight the efficacy of our approach in bridging the gap between small models and proprietary giants, even on multi-app scenarios that demand intricate coordination. We would like to mention that for the Qwen3-8B model, the thinking behavior leads to excessively long CoT sequences and repetitive patterns. This behavior results in performance scores approaching zero.

Compared with the model fine-tuned on the seed data with real environments, Sim-OB (Qwen3-8B) outperforms baselines Qwen3-8B-1apps-seed (average 29.6) by 14.4 points, demonstrating superior generalization beyond single-app seeds. Similarly, Sim-OB (Qwen2.5-7B) exceeds Qwen2.5-7B-Instruct-1apps-seed (average 25.9) by 16.7 points. In the 3B and 1.5B regimes, Sim-OB (Qwen2.5-3B) attains 26.1 on average, nearly tripling Qwen2.5-

Table 3: Performance on Webshop, Mind2Web and Operating System (Liu et al., 2023) across proprietary models, open baselines, and models fine-tuned on SIMAGENT trajectories. *GPT-4 results are reported from (Liu et al., 2023) and GPT-4o results are reported from leaderboard of SuperBench (Superbench, 2025).

Model	Mind2Web	OS	Webshop	Average
GPT Models				
GPT-4*	29	**42.4**	**61.1**	**44.2**
GPT-4o*	**32**	33.3	48.9	38.1
≥7B Models				
Sim-AB (Qwen3-8B)	25	**34.5**	67.3	**42.6**
Sim-AB (Qwen2.5-Coder-7B)	**29**	33	66	42.6
Sim-AB (Qwen2.5-7B)	23	32.6	69.2	41.6
Qwen3-8B-seed	22	31.3	68.6	40.6
Qwen2.5-7B-Instruct-seed	12	29.1	68.2	36.4
AgentLM-70B	13.5	21.5	64.9	33.3
Qwen3-8B	21	27.8	49.5	32.8
AgentLM-13B	8.4	18.1	**70.8**	32.4
Qwen2.5-7B-Instruct	7	27.8	58.8	31.2
Llama-3.1-8B-Instruct	18	19.4	54.9	30.8
Qwen2.5-Coder-7B-Instruct	17	22	53	30.5
AgentLM-7B	6.4	17.4	63.6	29.1
3B Models				
Sim-AB (Llama-3.2-3B)	**26**	28	**69**	**40.9**
Sim-AB (Qwen2.5-3B)	23	**31**	65	39.9
Qwen2.5-3B-Instruct	16	24	49	29.5
Llama-3.2-3B-Instruct	15	17	49	27.0
1.5B Models				
Sim-AB (Qwen2.5-1.5B)	17	**15**	**54**	**28.5**
Qwen2.5-1.5B-Instruct	**20**	13	40	24.0

3B-Instruct (8.8), while Sim-OB (Qwen2.5-1.5B) reaches 19.4, outperforming Qwen2.5-1.5B-Instruct (3.0) by 16.4 points.

AgentBench. Our models exhibit competitive performance across Mind2Web, Operating System (OS), and Webshop, as presented in Table 3. Our Sim-AB (Qwen3-8B) achieves an average score of 42.6, with a standout 34.5 on OS and 67.3 on Webshop, closely matching GPT-4's 44.2 average and surpassing GPT-4o's 38.1. The Sim-AB (Qwen2.5-Coder-7B) model ties with Qwen3-8B at 42.6 average, excelling with 29 on Mind2Web and 66 on Webshop, though it lags slightly on OS (33). The Qwen2.5-7B model leads with a Webshop score of 69.2, outperforming GPT-4 (61.1) and GPT-4o (48.9), with an overall average of 41.6. Both of our fine-tuned Qwen3-8B and Qwen2.5-7B-instruct models are better than their variants fine-tuned on the seed dataset generated from real environment. In the 3B category, Sim-AB (Llama-3.2-3B) achieves 40.9, driven by 26 on Mind2Web and 69 on Webshop, surpassing Qwen2.5-3B (39.9) and baseline models like Qwen2.5-3B-Instruct (29.5). For 1.5B models, Sim-AB (Qwen2.5-1.5B) reaches 28.5, outperforming Qwen2.5-1.5B-Instruct (24.0) across all tasks, with a notable 54 on Webshop, demonstrating the efficacy of our synthesized data from simulated trajectory compared to baselines.

Table 2: Performance on OfficeBench (Wang et al., 2024) across proprietary models, open baselines, and models fine-tuned on SIMAGENT trajectories.

Model	2-apps	3-apps	Average
GPT Models			
GPT-5	84.3	**76.4**	**80.4**
o4-mini	**86.3**	70.9	78.6
GPT-4.1	78.4	63.6	71
GPT-4o	74.5	50.9	62.7
GPT-4	50.6	11.6	31.1
7B/8B Models			
Sim-OB (Qwen3-8B)	58.8	**29.1**	**44.0**
Sim-OB (Qwen2.5-7B)	57.8	27.3	42.6
Sim-OB (Qwen2.5-Coder-7B)	64.7	14.6	39.7
Sim-OB (Llama-3.1-8B)	**64.9**	12.7	33.8
Qwen3-8B-1apps-seed	39.2	20	29.6
Qwen2.5-7B-Instruct-1apps-seed	35.3	16.4	25.9
Qwen2.5-Coder-7B-Instruct	31.4	10.7	21.1
Qwen2.5-7B-Instruct	27.5	10.9	19.2
Llama-3.1-8B-Instruct	7.8	3.6	5.7
Qwen3-8B	3.9	0	2.0
3B Models			
Sim-OB (Qwen2.5-3B)	**43.1**	**9.1**	**26.1**
Sim-OB (Llama-3.2-3B)	**43.1**	7.3	25.2
Qwen2.5-3B-Instruct	15.7	1.8	8.8
Llama-3.2-3B-Instruct	7.6	0	3.8
1.5B Models			
Sim-OB (Qwen2.5-1.5B)	**33.3**	**5.5**	**19.4**
Qwen2.5-1.5B-Instruct	5.9	0	3.0

Trajectory by Real Env V.S. Simulated Env. Figure 4 shows the performance of Qwen2.5-7B-Instruct fine-tuned on trajectories generated from real environment (seed dataset) and simulated environments (our synthesized dataset). When trained on datasets of identical size (e.g., for τ-bench we compare the model fine-tuned on 5k seed data and 5k synthesized data), we find that synthesized trajectories achieve performance comparable to real-environment-based trajectories on OfficeBench and AgentBench, even better than the seed dataset on τ-bench. As dataset size expands, e.g., from 5k to 90k in simulated settings, synthesized trajectories significantly outperform seed-based ones (e.g., on τ-Bench rising from 28.0 to 39.3 for Airline), demonstrating the scalability advantage of synthesized data with simulated environments in enhancing reasoning capabilities. This flexibility

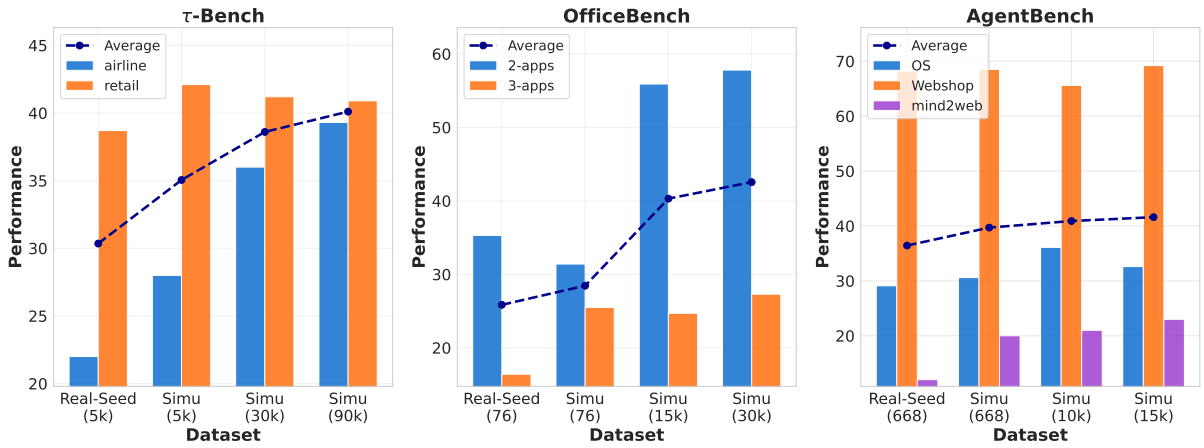

Figure 4: Ablation study on dataset generated by real environment and simulated environment. When the dataset size is identical, we show that simulated trajectories achieve performance comparable to real-environment-based trajectories on OfficeBench and AgentBench, and even better on τ-bench. As dataset size scales, simulated trajectories significantly outperform real-environment-based trajectories. This flexibility highlights the potential of simulated environments to support larger, more diverse datasets, addressing the limitations of real-world seed data constrained by collection efforts.

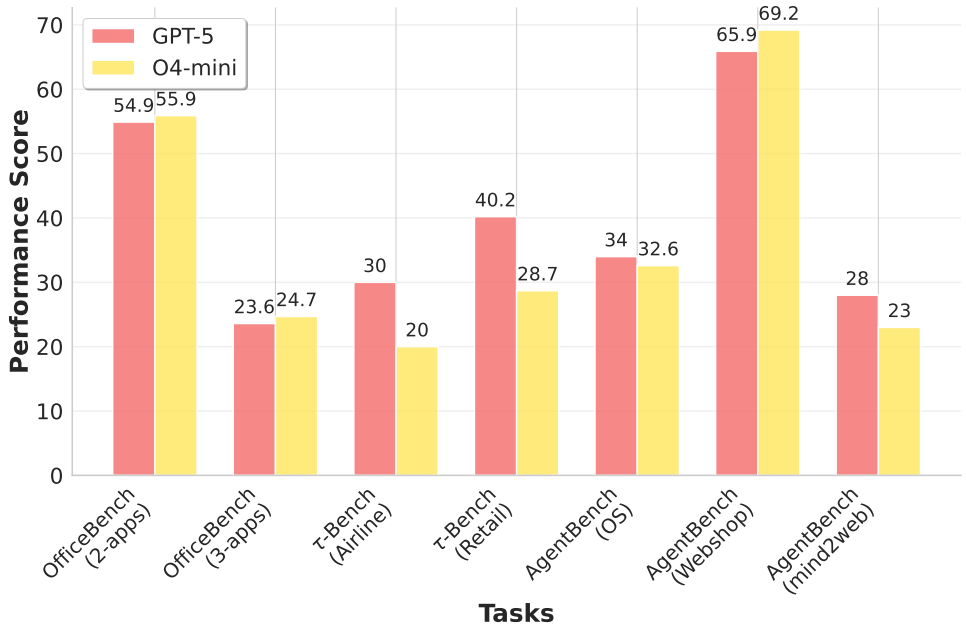

Figure 5: Ablation on different trajectory simulators. Performance comparison of 15k synthetic simulated trajectories generated by GPT-5 and o4-mini across OfficeBench, τ-Bench, and AgentBench, showing comparable results for both of them as synthesizers across most tasks, with GPT-5 significantly outperforming o4-mini on τ-Bench tasks.

highlights the potential of simulated environments to support larger, more diverse datasets, addressing the limitations of real-world seed data constrained by collection efforts.

Ablation on Different Trajectory Simulators. We compare the performance of 15k synthetic trajectories generated by GPT-5 and o4-mini across OfficeBench, τ-bench, and AgentBench, as shown in Figure 5. We fine-tune their simulated trajectories on Qwen2.5-7B-Instruct model. As the simulated trajectory synthesizer, the performance results of o4-mini and GPT-5 are similar across most tasks, with o4-mini slightly outperforming GPT-5 on OfficeBench 2-apps (55.9 vs. 54.9), 3-apps (24.7 vs. 23.6), and Webshop (69.2 vs. 65.9), while GPT-5 excels on AgentBench mind2web (28 vs. 23). Notably, GPT-5 achieves significantly higher scores on τ-bench, particularly in Retail (40.2 vs. 28.7) and airline (30 vs. 20) domains.

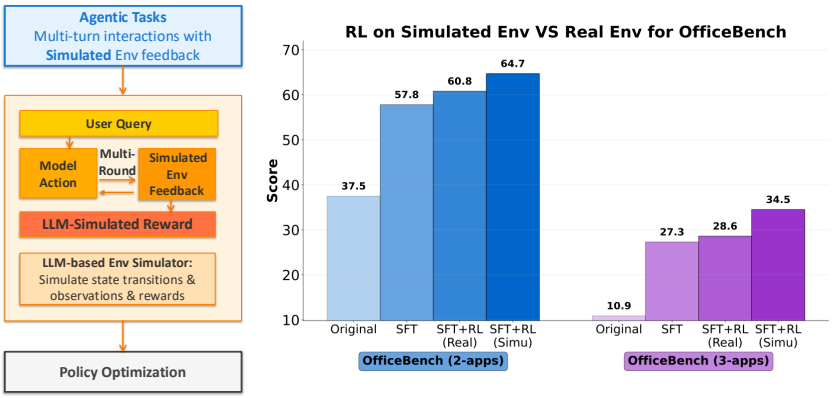

Figure 6: RL on simulated environment. (left) The diagram illustrates the agentic task framework with multi-turn interactions with simulated environment, where LLM simulator provides the simulated environment feedback and rewards. (right) The figure shows performance scores across RL on simulated and real environments. We show that the simulated environment (64.7, 34.5) outperforms the real environment (60.8 28.6) OfficeBench (2-apps, 3-apps), yielding total improvements of 6.9 and 7.2 points over the model after SFT.

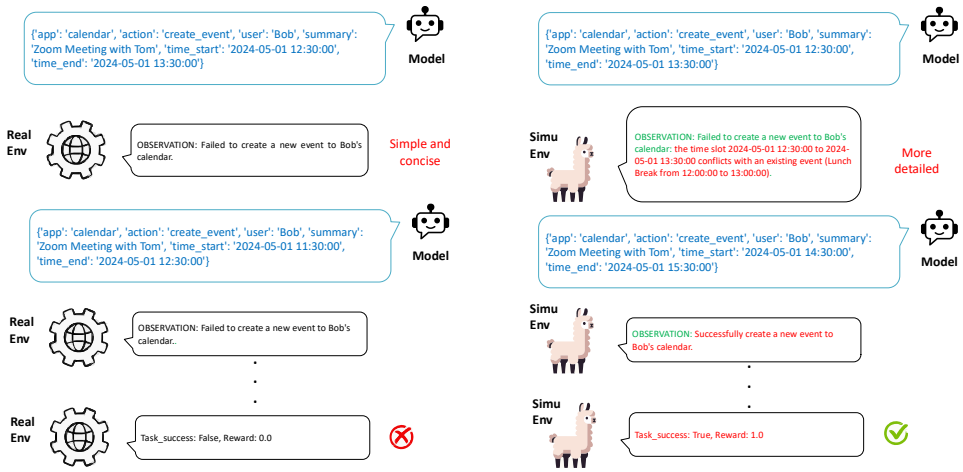

Figure 7: Case study comparing RL on real and simulated environments. In the real environment (left), the system provides fixed and concise failure messages. In contrast, the simulated environment (right) generates more flexible and detailed feedback, explaining conflicts (e.g., overlapping with a lunch break) and enabling the model to adjust and eventually get the reward.

4 Discussion: Reinforcement Learning on the Simulated Agentic Environment

While our main study focuses on supervised fine-tuning with simulated trajectories, reinforcement learning (RL) presents another natural avenue for leveraging reasoning-model simulators. RL allows agents to optimize policies directly through interaction, rather than relying solely on offline trajectory data. The simulated environment circumvents the need for extensive agent-environment setups and enabling the scalable adaptation of diverse RL agentic tasks within a unified framework. In this section, we present preliminary experiments on RL within the simulated environment, illustrating both the feasibility and the challenges of extending SimAgent beyond SFT. We leverage LLM-based environment simulators, incorporating (1) *Environment Feedback Simulation*: process agent actions to produce simulated state transitions, tool outputs, user interfaces, and error messages through structured dialogue in each round; and (2) *Reward Computation*: assess trajectory completion and assigns reward signals based on interactive trajectories and predefined success criteria and objectives.

We implement RL experiments on OfficeBench in a simulated environment using RAGEN (Wang et al., 2025b) built on VeRL (Sheng et al., 2024). The Sim-OB (Qwen2.5-7B-Instruct) model is trained with GRPO (Shao et al., 2024) on OfficeBench 1-apps tasks. Detailed RL training hyperparameters are provided in Appendix B. The model's performance is evaluated on OfficeBench 2-apps and 3-apps tasks, using the same evaluation setup in Section 3.

We utilize o4-mini to simulate the OfficeBench environment, which interacts with the model in each round, and computes the final reward. o4-mini is configured with a temperature of 1.0 and a maximum context length of 60,000 tokens. The complete OfficeBench tool usage specification, environment response format, testbed data for each task, and the interaction history are converted into text and incorporated into the o4-mini's prompt. Given the model action at the current round, we prompt o4-mini to provide simulated environment feedback. Upon task completion or reaching the maximum number of interaction rounds, o4-mini will assess whether the task was successfully completed based on the trajectory. We assign a reward of 1 for successful task completion and 0 for failure. Detailed prompt are shown in Appendix D.

Figure 6 (right) demonstrates the evaluation results of RL on real versus simulated environments on OfficeBench. We observe that the simulated environment (64.7, 34.5) outperforms the real environment (60.8 28.6) OfficeBench (2-apps, 3-apps), yielding total improvements of 6.9 and 7.2 points over the model after SFT. See more results of the training curves in Appendix E.

Case Study. We present in Figure 7 a case study highlighting how RL surprisingly benefits more from the simulated environment than from the real one. In the real setting (left), the system only produces fixed, concise failure messages. By contrast, the simulated environment (right) offers richer and more adaptive feedback, e.g., pointing out conflicts such as overlapping with a lunch break, which helps the model adjust its behavior and ultimately achieve the reward.

5 Related Work

Tool-using LLMs. Despite strong reasoning and generation abilities, LLMs fall short on tasks requiring real-time access, accurate computation, or environment interaction, motivating the development of *LLM-based agents* that leverage external tools. Representative examples include LLMs that use a web-browsing environment for question answering (Nakano et al., 2022), a Python interpreter for arithmetic and symbolic reasoning (Gao et al., 2023), and information retrieval for grounding in dialogue (Thoppilan et al., 2022). The range of tools explored in the literature is far broader, and we refer readers to the comprehensive survey in Qu et al. (2025) for full coverage.

Synthetic agentic datasets. While tool-using LLMs demonstrate the promise of agents, scaling their development requires large volumes of training data. To this end, recent efforts have introduced synthetic agentic datasets that emulate queries, tools, and environment feedback at scale. Gorilla (Patil et al., 2023) uses a self-instruct approach to generate instruction–API pairs. ToolAlpaca (Tang et al., 2023) leverages a multi-agent simulation environment to build a diverse tool-use corpus, while ToolLLM (Qin et al., 2023) employs ChatGPT to synthesize instruction–solution pairs involving real-world RESTful APIs. AgentTuning (Zeng et al., 2023b) adopts GPT-4 as an agent to generate trajectories across six task domains. API-Bank (Li et al., 2023) uses a multi-agent pipeline to generate domains, simulate APIs, construct queries, and validate responses. APIGen (Liu et al., 2024), built on ToolBench (Qin et al., 2023), applies multi-stage verification with diverse prompt templates to ensure accuracy and coverage. ToolBridge (Jin et al., 2024) curates a dataset of Python-based tool invocations via selection, conversion, and filtering of public corpora. BUTTON (Chen et al., 2025) generates multi-turn function-calling data with GPT-4o, combining bottom-up task evolution with top-down decomposition of complex tasks. Finally, ToolACE (Liu et al., 2025) introduces a pipeline for synthesizing diverse, complex, and accurate function-calling data through iterative tool evolution, complexity-guided dialog synthesis, and dual-layer verification.

Existing synthetic datasets have made significant progress in enabling tool-augmented LLMs, introducing diverse APIs, generation pipelines, and verification strategies. Our work takes a complementary direction: rather than targeting individual tool calls, SimAgent simulates complete end-to-end trajectories that integrate reasoning steps, tool use, and environment feedback. This benchmark-agnostic design allows synthetic data to be scaled across diverse domains and tasks, providing a flexible way to augment existing datasets with richer, trajectory-level supervision. In this sense, SimAgent does not replace prior efforts but instead offers a means of extending their coverage and utility by contributing multi-step, training-ready agentic data.

6 Conclusion

In this work, we introduced *SIMAGENT*, a benchmark-agnostic framework for end-to-end synthetic trajectory generation that leverages LLMs as environment simulators. By combining diversity expansion and schema-enforced fidelity, SIMAGENT produces high-quality trajectories that generalize across heterogeneous benchmarks without requiring laborious environment engineering. Our experiments demonstrate consistent improvements in robustness, schema understanding, and tool-use competency, enabling smaller open models to rival or surpass much larger proprietary systems. We further demonstrate reinforcement learning on simulated environments, highlighting the promise of agent RL training without costly real-world implementations. Together, these results establish environment simulation as a practical pathway for advancing agentic LLMs, opening opportunities for scalable research and deployment across diverse real-world tasks.

Ethics Statement

This work investigates synthetic data and environments for agentic tasks. It **does not** involve human subjects, user studies, or the collection of personally identifiable information. All experiments are conducted on **publicly available** benchmarks released under their respective licenses, which, to the best of our knowledge, do not contain sensitive personal data.

Reproducible Statement

We synthesize the task trajectories and simulate the agent environments using the Azure OpenAI endpoint with fixed model snapshots to ensure reproducibility. Our implementation relies on the LLaMA-Factory library (Zheng et al., 2024) for supervised finetuning, VeRL (Sheng et al., 2024) and RAGEN (Wang et al., 2025b) for reinforcement learning, and vLLM (Kwon et al., 2023) for inference and evaluation. We will also release the full codebase in a public GitHub repository and make our training models/ datasets publicly available on the Hugging Face platform.

References

Axel Backlund and Lukas Petersson. Vending-bench: A benchmark for long-term coherence of autonomous agents, 2025. URL https://arxiv.org/abs/2502.15840.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Mingyang Chen, Haoze Sun, Tianpeng Li, Fan Yang, Hao Liang, Keer Lu, Bin Cui, Wentao Zhang, Zenan Zhou, and Weipeng Chen. Facilitating multi-turn function calling for llms via compositional instruction tuning. *arXiv:2410.12952*, 2025.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *arXiv:2211.10435*, 2023.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

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

Hanzhao (Maggie) Lin and Heng-Tze Cheng. Gemini achieves gold-level performance at the international collegiate programming contest world finals. https://deepmind.google/discover/blog/gemini-achieves-gold-level-performance-at-the-international-collegiate-programming-co 2025.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. Toolace: Winning the points of llm function calling. *arXiv:2409.00920*, 2025.

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

Zuxin Liu, Thai Hoang, Jianguo Zhang, and et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *arXiv:2406.18518*, 2024.

Thang Luong and Edward Lockhart. Advanced version of gemini with deep think officially achieves gold-medal standard at the international mathematical olympiad. https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at 2025.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback. *arXiv:2112.09332*, 2022.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv:2305.15334*, 2023.

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

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv:2307.16789*, 2023.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-rong Wen. Tool learning with large language models: a survey. *Frontiers of Computer Science*, 19(8), January 2025. ISSN 2095-2236. doi: 10.1007/s11704-024-40678-2. URL http://dx.doi.org/10.1007/s11704-024-40678-2.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Superbench. Superbench leaderboard – model detail. https://fm.ai.tsinghua.edu.cn/superbench/leaderboard/modelDetail, 2025. Tsinghua University Fundamental Model Research Center. Accessed: 2025-09-24.

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

Romal Thoppilan, Daniel De Freitas, Jamie Hall, and et al. Lamda: Language models for dialog applications. *arXiv:2201.08239*, 2022.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025a. URL https://arxiv.org/abs/2504.20073.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. Officebench: Benchmarking language agents across multiple applications for office automation. *arXiv preprint arXiv:2407.19056*, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

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

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*, 2023a.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. *arXiv:2310.12823*, 2023b.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang 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.

A LLM USE STATEMENT

We use LLMs as general-purpose assist tools to check typos and grammar errors in writing.

B EXPERIMENT SETUP

B.1 SUPERVISED FINE-TUNING

Our model SFT is conducted using LLaMA-Factory (Zheng et al., 2024), on a server with eight NVIDIA A100-SXM4-80GB GPUs. We follow Prabhakar et al. (2025) for the training parameters. Table 4 lists hyper-parameters for full parameter supervised fine-tuning.

Table 4: Hyper-parameters used for full parameter supervised fine-tuning.

Hyper-parameter	**Value**
Learning Rate	5×10^{-6}
Number of Epochs	2
Number of Devices	8
Per-device Batch Size	2
Optimizer	`Adamw`
Learning Rate Scheduler	`cosine`
Max Sequence Length	16384

B.2 GRPO

We use the following hyper-parameters detailed in Table 5 for RL training on the simulated environment. We perform experiments on eight A100 GPUs. The model is trained using RAGEN (Wang et al., 2025a)and VeRL (Sheng et al., 2024).

Table 5: Hyper-parameters for RL training.

Hyper-parameter	**Value**
Learning Rate	1×10^{-6}
Number of Steps	64
Number of Devices	8
Temperature	0.7
Top P	1.0
PPO Mini Batch Size	32
Max Response Length	16384
KL Coefficient	0.001
Rollout Engine	VLLM (V0.8.2)
Optimizer	`Adamw`
Learning Rate Scheduler	`cosine`
Warmup Ratio	0.1
Agent Max Turn	25
Agent Max Action Per Turn	1
Clip Ratio High	0.28

C FULL PROMPT FOR SYNTHETIC TRAJECTORY GENERATION

See figure 8 for synthesize APIGen-MT and Agenttuning, and see figure 9 for OfficeBench.

D FULL PROMPT FOR RL ON THE SIMULATED ENVIRONMENT

Please see Figure 10 for prompt to simulated RL environment feedback and Figure 11 for the prompt to compute the reward.

Prompt to synthesize APIGen-MT and Agenttunig

```
You are an AI assistant that generates multi-turn conversation data for agent training. Your
     task is to create new agent trajectories based on existing examples.

## Example Trajectory:
{sample_text}

## Available Tools:
{available_tools}

CRITICAL FORMAT PRESERVATION REQUIREMENTS - ABSOLUTE COMPLIANCE:
1. **STRICTLY PRESERVE ORIGINAL FORMAT**: You MUST maintain the EXACT format structure from
     the example trajectory (EXCEPTION: function_call turns may include <think> tags when
     reasoning is needed)
2. **NO SYSTEM PROMPT GENERATION**: Do NOT generate any SYSTEM messages - follow the system
     prompt from the original example and that will be preserved separately
3. **TOOL CONSTRAINT ADHERENCE**: You MUST STRICTLY use ONLY the tools listed in the "
     Available Tools" section above. DO NOT use any tools outside this specified allowed tool
      set. This is MANDATORY.
4. **FORMAT CONSISTENCY**: Maintain identical conversation structure, role naming conventions
     , and response patterns as shown in the example (EXCEPTION: function_call turns may
     include <think> tags when reasoning is added)
5. **TURN COUNT MATCHING**: Generate approximately the SAME NUMBER of conversation turns as
     the example trajectory - the generated conversation should have a comparable length and
     depth to the sample data

## FUNCTION_CALL TURN REQUIREMENTS:
1. **REASONING IN THINK TAGS**: When making function calls, add brief reasoning (1-3
     sentences) inside `<think> </think>` tags ONLY in FUNCTION_CALL turns after you output '
     FUNCTION_CALL:'
2. **SELECTIVE REASONING**: Not every function call needs reasoning. Only include it when it
     helps explain the complex decision-making process
3. **STRICT TURN CONSTRAINT**: Reasoning in `<think> </think>` tags should ONLY appear in
     FUNCTION_CALL turns, NEVER in HUMAN, GPT, OBSERRVATION or additional turns
4. **FORMAT REQUIREMENT**: If reasoning is included, the FUNCTION_CALL turn are allowed to
     add the thinking sentences instead of only JSON format. You should follow this format:
'''
FUNCTION_CALL:
<think>
Brief reasoning about why this function call is needed (1-3 sentences). Ended with: I will
     call the function <function_name>.
</think>
{{"name": "function_name", "arguments": {{...}}}}
'''

## ABSOLUTE PROHIBITIONS:
- DO NOT use ANY tools that are not explicitly listed in the "Available Tools" section above
- DO NOT change the conversation format structure (human/gpt roles, value formatting, etc.) -
      EXCEPTION: function_call turns may include <think> tags when reasoning is needed
- DO NOT violate any fixed formatting elements, tool specifications, or requirements in
     system instructions from the example - EXCEPTION: function_call turns may include <think
     > tags when reasoning is added
- DO NOT generate significantly fewer or more turns than the example trajectory
- DO NOT invent or create new tools - use ONLY the provided tools

## Requirements:
1. Generate a completely NEW scenario/task that is different from the example but requires
     similar problem-solving patterns
2. Create a multi-turn conversation between Human and Assistant that demonstrates systematic
     problem-solving
3. The conversation should show the agent's reasoning process and step-by-step approach
4. **Start directly with a HUMAN message - do not include the SYSTEM content**

## Output Format:
Generate the conversation:
HUMAN: [user message content]
ASSISTANT: [assistant reply content]
HUMAN: [user message content]
ASSISTANT: [assistant reply content]
HUMAN: [user message content]
ASSISTANT: [assistant reply content]
...(until the task is finished and the conversation is complete - matching the turn count of
     the example)
```

Figure 8: Prompt to synthesize APIGen-MT and Agenttunig

Prompt to synthesize OfficeBench

```
You are an AI assistant that generates multi-turn conversation data. Based on the provided
     example conversation, create a new conversation that expands the task to use multiple
     apps in a natural workflow.

Example conversation:\{seed\_sample\}

Your task:
1. Analyze the example task
2. Create a new task that naturally extends this workflow to use multiple different apps
3. The new task should be realistic and make sense - don't force apps together artificially
4. New created task should be diverse

Available Tools:
{availabel tools}

## CRITICAL REQUIREMENTS - AVOID COMMON ERRORS:

### Critical Error Pattern - Insufficient Path/Directory Validation:
**Root Cause: Failure to perform directory ls or mkdir validation before operations**
- **File Path/Directory Errors**: Working directory or data directory spelling, hierarchy
     does not match expectations, no prior checking or directory creation
- **Mandatory Operation**: Before any file operations, use ls to confirm current directory
     structure and file existence
- **Directory Creation**: If creating new directories, must first use mkdir -p to ensure
     directory exists

## Workflow Best Practices:
1. **File Discovery**: Use shell commands (ls, find) to discover actual file names before
     operations
2. **App Context**: Always switch\_app before using different app APIs
3. **Data Operations**: Count/filter accurately, verify each step, preserve headers
4. **File Creation**: Use proper app APIs (Excel: new_file, Word: new_document), NOT shell
     touch

Requirements:
1. Follow the same conversation format as the example (\#\#Task: format, <think> and <answer>
      structure)
2. Generate a completely new task - don't copy the example
3. Make sure the workflow feels natural and logical
4. DON'T specify exact file names in the task - let the workflow discover them with shell
     commands
5. Include app switching with "Successfully switched to app: [app\_name]. Available actions:"
      messages
6. End with finish\_task action

Create a conversation that shows the complete workflow from start to finish with proper error
      prevention.

Please generate the conversation content directly in this format:
HUMAN: [human message content]
GPT: [GPT reply content]
HUMAN: [human message content]
GPT: [GPT reply content]
...
```

Figure 9: Prompt to synthesize OfficeBench

Prompt to simulated OfficeBench Environment Feedback

```
You are an office environment simulator that needs to generate realistic environment feedback
      based on eval model's actions.

Current application: {self.current_app or "None selected"}

Available applications and action formats:
{action_formats}

{response_guidance}

Testbed Data of the task:
{testbed_text}

Please simulate the execution result of this action. You need to:
1. FIRST check if the action is valid according to the available action formats above
2. If the action is invalid (wrong app, wrong action name, missing parameters), return
     failure immediately.
3. If valid, simulate realistic execution results
4. Return appropriate observation results

IMPORTANT: Only actions listed in the "Available applications and action formats" section are
      valid. If the action is not in that list, it should fail.

Response format:
```json
{{
   "success": true/false,
   "observation": "Specific observation result explaining success or failure."
}}
Please ensure your response is realistic. Invalid actions should always return success=false.

Current Task:
{current_task}

Previous history of the eval model's actions:
{self._get_interaction_history()}

Action to simulate:
{eval_context}
"""
```

Figure 10: Prompt to simulate OfficeBench Environment Feedback

Prompt to compute OfficeBench RL Reward

```
You need to evaluate whether the eval model has successfully completed the given task based
     on the interaction history.

Complete interaction history:
{interaction_history}

Please analyze the above information and determine whether the model has successfully
     completed the task based on the trajectory history and whether the final outcome meets
     the requirements if following all the actions. Do NOT simply judge success based on
     whether the model claimed "task finished".

Please provide judgment in the following format:
```json
{{
   "reasoning": "Detailed explanation of your judgment about whether the task is successfully
        completed",
   "evidence": "Key evidence from the interaction history that supports your judgment",
   "task_success": true/false,
   "confidence": 0.0-1.0
}}
```

Figure 11: Prompt to compute OfficeBench RL Reward

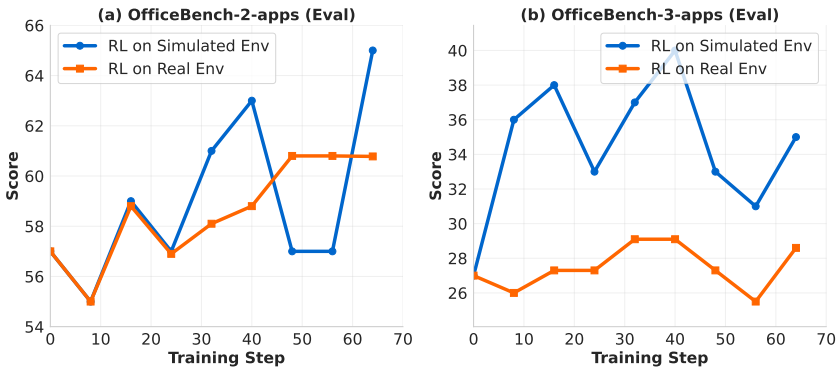

Figure 12: Evaluation performance of RL on simulated environment and real environment across training steps.

E More Results of RL on Simulated Environment

We present the experimental results for the RL performance on the simulated environment for OfficeBench across training steps, as depicted in Figure 12. The evaluation performance on the OfficeBench 2-apps and 3-apps tasks demonstrates improvement, with 2-apps scores rising steadily from around 57.8 to approximately 64.7, and 3-apps scores increasing from 27.3 to 34.5. We can find that for the 3-apps performance RL on simulated environment is always better than real environemnt.