

Train Dirty, Recover Reliably: A Controlled Study of Data-Efficient Recovery Tuning for Tool Agents

Yumin Lee

Sookmyung Women’s University
Seoul, Republic of Korea
dldnaus1098@sookmyung.ac.kr

Chong-Kwon Kim

Korea Institute of Energy Technology
Seoul, Republic of Korea
ckim@kentech.ac.kr

Yoonhyuk Choi

Sookmyung Women’s University
Seoul, Republic of Korea
chldbsgur123@gmail.com

Abstract

Tool-using language model agents are commonly evaluated under successful interaction assumptions, where user requests are sufficiently specified and tool executions return valid responses. However, practical agents frequently encounter recoverable failures: users omit required information, refer to entities ambiguously, or tools return transient failures. This paper studies a focused question: whether a small amount of targeted supervision can improve a tool agent’s choice of immediate recovery actions under controlled failures. We introduce **Mini Recovery Tuning**, a deterministic benchmark and training protocol for evaluating this question. Starting from clean weather, calendar, and restaurant tool-use trajectories, the pipeline constructs two matched noise families, missing required slots and timeout failures, and two held-out surface variants, ambiguous references and empty tool responses. Noisy examples are labeled with one of two implemented recovery primitives, clarify or retry, and formatted for supervised LoRA/QLoRA adaptation. We separate Recovery-Action Success, which measures whether the model selects the correct immediate recovery action, from full task success after a recovered multi-turn interaction. Mini Recovery Tuning provides a controlled framework for studying immediate recovery-action selection under reproducible failure conditions. Oracle-based evaluation validates the data-generation and scoring pipeline, while model experiments serve as a diagnostic analysis of how targeted recovery supervision changes recovery behavior. Extending these findings to broader deployment robustness requires future evaluation with richer environments, multi-seed comparisons, and end-to-end recovery metrics.

Keywords

Tool-using agents, robustness, recovery behavior, data-efficient fine-tuning, parameter-efficient adaptation

1 Introduction

Large language model (LLM) agents increasingly solve tasks by interleaving natural-language reasoning with calls to external tools, APIs, databases, web environments, code execution, and retrieval systems [7, 11, 12, 21, 25]. This interface is powerful because tools can provide information and actions unavailable to the language model alone. It is also brittle: a single missing argument, ambiguous reference, malformed response, or transient tool failure can derail the trajectory even when the intended task is still recoverable.

Recent benchmarks have made tool-agent evaluation more realistic. API-Bank, BFCL, and ToolLLM evaluate tool selection and function calling at increasing scale [5, 10, 11]; WebShop, Mind2Web, WebArena, SWE-bench, and AgentBench study agents in interactive or environment-grounded tasks [1, 4, 7, 19, 25]; and

τ -bench, MCP-Bench, and WildToolBench emphasize multi-turn interaction, cross-tool coordination, and realistic user behavior [16, 20, 24]. Robustness-oriented work such as RoTBench shows that tool-learning performance can degrade under noisy conditions [22], while clarification and reliability-alignment studies argue that agents should sometimes ask, defer, or repair rather than call a tool immediately [14, 18]. Despite this progress, there remains a narrower but useful gap: controlled training protocols for measuring whether targeted, data-efficient supervision improves concrete recovery choices under specified user-side and tool-side failures.

Problem statement. Existing tool-agent evaluations primarily measure whether an agent can correctly invoke tools under successful execution. However, practical agents must also decide how to respond when an interaction becomes partially invalid but remains recoverable. We study the following question: given a noisy tool-use state, can data-efficient supervised recovery examples teach an agent to select an appropriate recovery action?

This paper addresses that gap with Mini Recovery Tuning, a small benchmark and experimental pipeline for recovery tuning. The benchmark is intentionally limited. It contains deterministic tools, synthetic template-based clean tasks, rule-based noise injection, and two implemented recovery primitives. The purpose is not to claim that a small local environment captures deployed agents, but to make recovery behavior auditable: each noisy example has a known cause, a known intended primitive, and a scorer that checks whether the model chooses the correct immediate repair action.

The resulting framing differs from broader deployment-robustness claims in three ways. First, the implemented method covers only clarify for missing required information and retry for transient tool failures; actions such as fallback, abstention, tool reselection, and output verification are future extensions. Second, the main noisy metric is Recovery-Action Success rather than end-to-end task success, because the current evaluator scores a one-step recovery action instead of simulating the full recovered dialogue. Third, the proposed primitive-ordered recovery tuning is treated as a hypothesis and must be compared against random and reverse-order controls under matched optimization budgets.

Our contributions are as follows.

- We define a deterministic tool-agent environment and noise-injection pipeline that converts clean weather, calendar, and restaurant trajectories into matched and held-out recovery examples with auditable clarify and retry labels.
- We specify a code-aligned recovery-tuning protocol using supervised LoRA/QLoRA adaptation, enabling controlled comparisons among prompt-only, clean-only, random recovery, and primitive-ordered recovery conditions under matched training budgets.

- We formalize evaluation around Recovery-Action Success, report oracle pipeline-validation results, and explicitly separate the current one-step recovery metric from the stronger claim of end-to-end recovered task success.

2 Related Work

2.1 Tool-Using and Agentic Language Models

Early tool-use and agentic prompting work shows that language models can combine reasoning traces with external actions [21] and can be trained to insert API calls in text [12]. ToolLLM and API-Bank introduce large-scale tool-use datasets and executable function-calling evaluations, enabling systematic study of tool selection and invocation behavior [5, 11]. Interactive environments such as WebShop, Mind2Web, WebArena, SWE-bench, and AgentBench evaluate agents in settings where action selection is coupled to environment state rather than isolated text prediction [1, 4, 7, 19, 25]. These benchmarks motivate our focus on tool-agent reliability, but our artifact is much smaller: it is designed for controlled diagnosis of recovery actions rather than broad coverage.

2.2 Tool-Agent Benchmarks Under Noise and Interaction

Recent benchmarks have moved beyond static function-call matching. BFCL evaluates function calling across progressively more agentic conditions [10]; τ -bench evaluates tool-agent-user interaction in realistic domains [20]; MCP-Bench emphasizes live MCP servers and cross-tool coordination [16]; and WildToolBench targets messy, realistic user behavior [24]. RoTBench is closest in spirit to the robustness motivation, as it evaluates tool learning under multiple noise levels and analyzes errors in tool selection, parameter identification, and content filling [22]. Our setting is narrower and training-oriented: rather than measuring robustness across many environments, it asks whether targeted recovery supervision changes a model’s immediate action under controlled failures.

2.3 Data-Efficient Adaptation and Recovery Behavior

Instruction tuning and self-generated supervision show that curated training data can substantially shift model behavior [9, 15, 17]. Parameter-efficient methods such as LoRA and QLoRA make it practical to test such shifts with limited compute [2, 3]. For tool agents specifically, ToolACE and MAGNET synthesize function-calling and multi-turn tool-use data at scale [6, 23], while environment tuning studies curriculum-based interaction with feedback [8]. Complementary work on feedback and reliability, including Reflexion, Learning to Ask, and Relign, studies agents that reflect, ask clarifying questions, defer, or dynamically adjust tool use [13, 14, 18]. Mini Recovery Tuning isolates a small subset of these behaviors: clarification for underspecified requests and retry for transient tool failures.

3 Task Formulation and Scope

3.1 Clean Tool-Use Tasks

A clean task is represented as

$$x = (u, \mathcal{T}, a^*, r^*, y^*), \quad (1)$$

where u is the user instruction and $\mathcal{T}$ is the available tool set. a^* denotes the ground-truth tool call generated by the deterministic environment, including the correct tool name and required arguments. r^* denotes the corresponding deterministic tool output, and y^* is the final response generated after tool execution. A clean-action prediction succeeds when the model emits a valid `tool_call` with the correct tool name and exact required arguments.

3.2 Noisy Tasks and Recovery Primitives

A noisy task is produced by a noise function g :

$$\tilde{x} = g(x). \quad (2)$$

Each noisy task has a solvability label $s \in \{0, 1\}$ and a primitive label $p \in \{\text{clarify}, \text{retry}\}$. In the implemented training split, all noisy examples are solvable by construction. Missing required information maps to `clarify`; a transient tool failure after an otherwise valid call maps to `retry`. The model is asked to output exactly one compact JSON object with one of three actions: `tool_call`, `clarify`, or `retry`.

3.3 Claim Boundary

The formulation is intentionally restrictive. It does not model multiple simultaneous failures, unsafe tasks, tool schema drift, stale memory, conflicting constraints, or cases where the correct behavior is to abstain. It also does not yet evaluate full recovery after the model asks a question or retries a call. These restrictions are useful for controlled measurement but should prevent deployment-level interpretations of the current results.

4 Mini Recovery Tuning Benchmark

4.1 Deterministic Tools and Domains

The local environment contains three domains and five deterministic tools:

- **Weather:** `get_weather(city, date)`.
- **Calendar:** `create_event(title, date, time, location)` and `find_events(date)`.
- **Restaurant:** `search_restaurants(...)` and `book_restaurant(...)`.

Tool outputs are deterministic functions of the arguments. This choice removes external API variance and makes exact-action scoring auditable, at the cost of realism.

4.2 Clean Task Construction

The default artifact generates 120 clean tasks by round-robin sampling across the five tool types and then shuffling with a fixed seed. The split is 80 training tasks, 20 development tasks, and 20 test tasks. An optional larger setting generates 240 clean tasks with a 160/40/40 split. The generator reuses template families across splits, so the benchmark should not be described as template-disjoint.

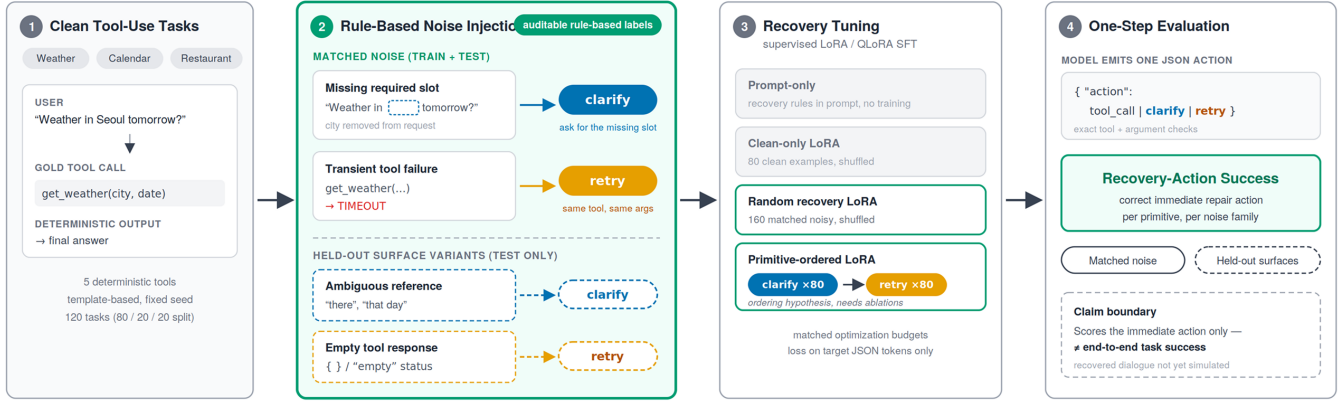


Figure 1: Overview of Mini Recovery Tuning. Clean deterministic tool-use trajectories are converted into noisy recovery examples. The current artifact evaluates whether a model selects the correct immediate recovery action, not whether it completes a full recovered task after user or tool simulation.

Table 1: Dataset statistics for the implemented default setting. Each clean task yields one matched missing-information example, one matched timeout example, one held-out ambiguous-reference example, and one held-out empty-response example.

Split	Clean	Matched noisy	Held-out noisy
Train	80	160	160
Dev	20	40	40
Test	20	40	40

4.3 Matched and Held-Out Noise

Matched missing-information noise. For each clean task, the injector removes one required slot from the user instruction. For example, a complete weather request with both city and date may be converted into a request that retains the date but omits the city. The target is a `clarify` JSON object identifying the missing slot and asking a slot-specific question.

Matched timeout noise. For retry examples, the user request remains unchanged and the first assistant message contains the gold tool call. The following tool response is corrupted into a timeout error. The target is a `retry` action with the same tool name and arguments as the first call.

Held-out surface variants. The held-out noisy split uses the same primitives but different surface forms. Clarification transfer is tested with ambiguous references such as “there,” “that day,” or “that restaurant.” Retry transfer is tested with empty tool responses, represented either as an empty object or an explicit empty status. These examples test limited surface transfer inside the same toy environment; they are not a substitute for broad out-of-distribution evaluation.

4.4 Labeling Assumptions

Labels are produced by rules rather than by LLM judges. This makes the pipeline easy to audit, but it also creates a simple association between each noise family and its primitive. A model could therefore succeed by learning the mapping from noise type to action rather than a general recovery policy. Future versions should include multiple plausible recovery actions per noise family, unsolvable cases, and examples requiring more than one user or tool turn.

5 Recovery Tuning Protocol

5.1 Supervised Objective

All fine-tuned conditions use supervised next-token prediction over chat-formatted examples. For a noisy input $\tilde{x}$ and target recovery-action sequence z , the objective is

$$\mathcal{L}_{\text{SFT}}(\theta) = - \sum_{t=1}^{|z|} \log p_{\theta}(z_t \mid \tilde{x}, z_{<t}). \quad (3)$$

Only assistant target tokens are included in the loss; prompt tokens are masked. The implementation is designed for LoRA or QLoRA adaptation rather than full-parameter fine-tuning.

5.2 Training Conditions

Table 2 summarizes the implemented comparison set. The default protocol disables clean replay during recovery tuning. Clean replay can be useful diagnostically, but it should be reported only when total optimization steps or training tokens are matched, because adding clean examples changes the exposure budget.

5.3 Recovery Data Ordering Comparison

The current two-stage order trains on `clarify` before `retry`. Because the stages differ by recovery primitive rather than by a validated difficulty score, we describe the condition as primitive-ordered recovery tuning rather than a validated curriculum. A defensible curriculum claim requires at least a reverse-order condition, a random two-stage condition with the same optimizer or scheduler

Table 2: Implemented training conditions. The primitive-ordered condition evaluates one possible data ordering and does not establish an intrinsic curriculum effect.

Method	Training data/order	Purpose
Prompt-only	None	Tests whether the base model follows the explicit recovery rules in the prompt.
Clean-only LoRA	80 clean examples, shuffled	Tests whether clean tool-use supervision alone improves noisy recovery behavior.
Random recovery LoRA	160 matched noisy examples, shuffled	Controls for seeing the same recovery examples without primitive ordering.
Primitive-ordered recovery LoRA	80 clarify examples, then 80 retry examples	Tests whether the implemented two-stage order differs from random mixing.

reset, and a mixed random condition with matched total updates. Without these controls, gains could come from recency, reset effects, or unequal data exposure rather than curriculum structure.

6 Experiments and Results

6.1 Metrics

The experiments are designed to answer two questions: (1) whether targeted recovery supervision improves immediate recovery-action selection compared to clean-only or prompt-only baselines, and (2) whether the learned behavior transfers to held-out surface variations of the same recovery primitives.

Clean Action Success. On clean tasks, success requires a `tool_call` action with the correct tool name and exact required arguments.

Recovery-Action Success. On noisy tasks, Recovery-Action Success requires the correct immediate recovery action. For missing-information examples, the model must ask a relevant clarification question for the missing slot. For retry examples, the model must emit a retry action with the same tool and arguments, or equivalently repeat the same tool call after the error. This metric does not measure whether the final task is solved after recovery.

Recovery Accuracy. Recovery accuracy measures only whether the predicted primitive matches the gold primitive, ignoring finer argument or question-quality checks.

Robustness Drop. For the current artifact, robustness drop is computed as clean action success minus noisy Recovery-Action Success. This quantity should not be compared directly with end-to-end task-success drops in larger benchmarks.

Extra Cost. The evaluator records one extra user turn for a correct clarify action and one extra tool call for a correct retry action. This is a proxy and should not be interpreted as a measurement of latency, token cost, or complete interaction length.

6.2 Oracle Pipeline Validation

The repository includes an oracle mode that emits the gold action for each example. Table 3 reports oracle results on the default test split. These numbers verify that data generation and scoring are internally aligned. They are smoke-test results, not model-learning results.

Table 3: Oracle pipeline validation on the default test split. These values are generated by the oracle policy and should not be reported as trained-model performance.

Metric	Matched	Held-out
Clean action success	100.0%	
Noisy Recovery-Action Success	100.0%	100.0%
Recovery accuracy	100.0%	100.0%
Robustness drop	0.0%	0.0%
Avg. extra user turns	0.50	0.50
Avg. extra tool calls	0.50	0.50

6.3 Recovery Tuning Results

Tables 4, 5, and 6 present controlled evaluations of immediate recovery-action selection. We compare whether targeted noisy supervision changes recovery behavior across matched and held-out failure patterns. The current results are diagnostic: they measure recovery-action selection rather than end-to-end task completion, and stronger robustness claims require additional seed variation and ordering controls.

7 Discussion and Claim Boundary

Table 7 summarizes the boundary between implementation and interpretation. The table is included to make the paper auditable and to avoid presenting the current pipeline as a general robustness result.

7.1 Immediate Extensions

The most important extension is end-to-end recovery evaluation. A stronger evaluator should allow the model to decide whether to call a tool, ask a question, retry, or stop; simulate the user response or second tool response; and then score final task completion. The noise taxonomy should also be expanded to include typos, paraphrases, multiple missing slots, conflicting constraints, multilingual requests, malformed JSON, partial results, rate limits, schema mismatch, stale results, and wrong-but-plausible outputs. Finally, the primitive set should be ablated with `clarify-only`, `retry-only`, `clarify+retry`, and variants containing `re-check`, `fallback`, `abstain`, or `dynamic tool reselection`.

8 Limitations

This study has several limitations. The environment is small, deterministic, and template-based. All training-time noisy examples are solvable, so the model is not trained to abstain under impossible, unsafe, or underspecified conditions that cannot be repaired with one clarification. The evaluator measures one-step recovery-action correctness rather than full task completion after recovery. The system prompt explicitly states the recovery rules, making prompt-only inference an important and potentially strong baseline. The curriculum effect is not established without reverse-order and matched two-stage controls. Finally, the current paper includes single-seed trained-model diagnostics in addition to oracle pipeline validation; these diagnostics do not establish improved recovery behavior without seed variation, reverse-order controls, and end-to-end recovery evaluation.

Table 4: Recovery tuning results on clean and matched noisy test sets. Higher is better for success rates. Lower is better for robustness drop and extra cost.

Method	Clean SR $\uparrow$	Missing-info SR $\uparrow$	Timeout SR $\uparrow$	Robustness Drop $\downarrow$	Extra Turns $\downarrow$
Prompt-only	0.0%	0.0%	100.0%	-50.0%	0.00
Clean-only LoRA	82.5%	0.0%	100.0%	32.5%	0.00
Random recovery LoRA	82.5%	95.0%	100.0%	-15.0%	0.47
Primitive-ordered Recovery LoRA	85.0%	0.0%	100.0%	35.0%	0.00

Table 5: Held-out surface transfer results. The held-out test set uses new surface forms that require the same recovery primitives as the training noise.

Method	Ambiguous Paraphrase SR $\uparrow$	Empty Response SR $\uparrow$	Recovery Acc. $\uparrow$	Extra Tool Calls $\downarrow$
Prompt-only	0.0%	100.0%	42.9%	0.29
Clean-only LoRA	0.0%	100.0%	28.6%	0.29
Random recovery LoRA	22.5%	100.0%	58.9%	0.29
Primitive-ordered Recovery LoRA	0.0%	100.0%	28.6%	0.29

Table 6: Primitive-level recovery analysis.

Method	Clarify Acc. $\uparrow$	Retry Acc. $\uparrow$	Unnecessary Clarify $\downarrow$
Prompt-only	0.0%	100.0%	20.0%
Clean-only LoRA	0.0%	100.0%	0.0%
Random recovery LoRA	95.0%	100.0%	10.0%
Primitive-ordered Recovery LoRA	0.0%	100.0%	0.0%

Table 7: Claim-to-implementation alignment after revision.

Topic	Current implementation	Revised interpretation
Noise	Missing slot, timeout, ambiguous reference, empty response	Controlled toy noise, not general deployment noise.
Primitives	clarify, retry	Other recovery actions are future work.
Curriculum	clarify stage followed by retry stage	Ordering hypothesis requiring ablations.
Evaluation	One-step action scoring	Recovery-action correctness, not full task success.
Clean replay	Optional, disabled by default	Report only under a matched budget.
Data split	Template families reused across splits	Do not claim template-disjoint generalization.
Results	Oracle smoke test plus single-seed model diagnostics	Do not claim model robustness gains without multi-seed, reverse-order, and end-to-end evaluations.

9 Conclusion

Mini Recovery Tuning studies a narrow but important failure mode for tool-using agents: choosing a repair action when required user information is missing or a tool call fails transiently. The artifact provides deterministic task generation, rule-based noise injection, supervised recovery labels, LoRA/QLoRA training recipes, oracle validation of the scoring pipeline, and preliminary single-seed model diagnostics. Its main value is methodological: it separates immediate recovery-action selection from full recovered task success and defines fair comparisons for future model experiments. Broader deployment robustness remains an open empirical question requiring multi-seed trained-model results, richer noise, stronger baselines, and end-to-end recovery evaluation.

Acknowledgments

This work was supported by Sookmyung Women’s University Research Grants (1-2503-2027).

References

- [1] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, Vol. 36. <https://openreview.net/forum?id=kiYqbO3wqw>
- [2] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems*, Vol. 36. https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html
- [3] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=nZeVKeFYf9>
- [4] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2024. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=VTF8yNQM66> Oral presentation.
- [5] Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, Yongbin Li, and Mengyuan Zhou. 2023. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, 3102–3116. <https://doi.org/10.18653/v1/2023.emnlp-main.187>
- [6] 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. 2025. ToolACE: Winning the Points of LLM Function Calling. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=8EB8k6DdCU> Poster presentation.
- [7] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2024. AgentBench: Evaluating LLMs as Agents. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=zAdUB0aCTQ>
- [8] Siyuan Lu, Zechuan Wang, Hongxuan Zhang, Qintong Wu, Leilei Gan, Chenyi Zhuang, Jinjie Gu, and Tao Lin. 2026. Don’t Just Fine-Tune the Agent, Tune the Environment. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=nzodtGccEM> Poster presentation.
- [9] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training Language Models to Follow Instructions with Human Feedback. In *Advances in Neural Information Processing Systems*, Vol. 35. 27730–27744. https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html
- [10] Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 267)*. PMLR, 48371–48392. <https://proceedings.mlr.press/v267/patil25a.html>

- [11] 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. 2024. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=dHng200Jjr>
- [12] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*, Vol. 36. https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html
- [13] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik R. Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 36. <https://openreview.net/forum?id=vAElhFcKW6>
- [14] Wenxuan Wang, Juluan Shi, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jen-tse Huang, Wenxiang Jiao, and Michael R. Lyu. 2025. Learning to Ask: When LLM Agents Meet Unclear Instruction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Suzhou, China, 21773–21784. <https://doi.org/10.18653/v1/2025.emnlp-main.1104>
- [15] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, 13484–13508. <https://doi.org/10.18653/v1/2023.acl-long.754>
- [16] Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Shah, Yujia Bao, and Eugene Siow. 2026. MCP-Bench: Benchmarking Tool-Using LLM Agents with Complex Real-World Tasks via MCP Servers. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=fe8mzHwMxN> Poster presentation.
- [17] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models Are Zero-Shot Learners. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=gEZrGCozdqR>
- [18] Hongshen Xu, Zichen Zhu, Lei Pan, Zihan Wang, Su Zhu, Da Ma, Ruisheng Cao, Lu Chen, and Kai Yu. 2025. Reducing Tool Hallucination via Reliability Alignment. In *Proceedings of the 42nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 267)*. PMLR, 69992–70006. <https://proceedings.mlr.press/v267/xu25ap.html>
- [19] Shunyu Yao, Howard Chen, John Yang, and Karthik R. Narasimhan. 2022. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. In *Advances in Neural Information Processing Systems*, Vol. 35. 20744–20757. https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html
- [20] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R. Narasimhan. 2025. τ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=roNSXZpUDN> Poster presentation.
- [21] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=WE_vluYUL-X Notable top 5%.
- [22] Junjie Ye, Yilong Wu, Songyang Gao, Caishuang Huang, Sixian Li, Guanyu Li, Xiaoran Fan, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. RoTBench: A Multi-Level Benchmark for Evaluating the Robustness of Large Language Models in Tool Learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Miami, Florida, USA, 313–333. <https://doi.org/10.18653/v1/2024.emnlp-main.19>
- [23] Fan Yin, Zifeng Wang, I-Hung Hsu, Jun Yan, Ke Jiang, Yanfei Chen, Jindong Gu, Long T. Le, Kai-Wei Chang, Chen-Yu Lee, Hamid Palangi, and Tomas Pfister. 2025. MAGNET: Multi-Turn Tool-Use Data Synthesis and Distillation via Graph Translation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vienna, Austria, 32600–32616. <https://doi.org/10.18653/v1/2025.acl-long.1566>
- [24] Peijie Yu, Wei Liu, Yifan Yang, Jinjian Li, Zelong Zhang, Xiao Feng, and Feng Zhang. 2026. Benchmarking LLM Tool-Use in the Wild. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=yz7fL5vfnp> Poster presentation.
- [25] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=oKn9c6ytLx>

A Implementation Details

A.1 Data Generation

The default generator creates clean tool-use examples from deterministic templates over weather, calendar, and restaurant domains. Tool type is selected by round-robin assignment, concrete slot values are sampled from finite lists, and the resulting examples are shuffled with a fixed seed before splitting into train, development, and test sets. Each clean example stores the user request, available tool schema, oracle tool call, deterministic tool response, and final answer.

A.2 Noise Injection

For matched clarification examples, the injector removes one required slot from the user-facing request and records the deleted slot as the clarification target. For matched retry examples, the first assistant message contains the oracle tool call and the following tool message is replaced with a timeout error. Held-out clarification examples replace explicit arguments with ambiguous references, while held-out retry examples replace the timeout with an empty response. The implementation assigns labels by rule: missing or

ambiguous required information maps to clarify, and timeout or empty tool response maps to retry.

A.3 LoRA/QLoRA Training Setup

Training examples are serialized as chat-style supervised fine-tuning records. Prompt and context tokens are masked in the loss, and only the assistant target JSON is optimized. The intended fine-tuning path uses LoRA or QLoRA adapters rather than full-model updates. The default recovery setting trains only on matched noisy examples; clean replay is treated as an optional diagnostic and should be budget-matched if reported.

A.4 Evaluation Pipeline

The evaluator parses the model output as a compact JSON action. Clean examples are scored by exact tool name and required-argument match. Noisy examples are scored by Recovery-Action Success: clarification examples require a relevant question for the missing slot, while retry examples require retrying or repeating the same tool call with the same arguments. Oracle mode bypasses model generation and emits the gold action, allowing the data-generation and scoring code to be tested independently of model quality.