Skill2Task: Skill-based Agentic Trajectory Synthesis for Training Claw Agent

Anonymous ACL submission

Abstract

We present **Skill2Task**, a skill-grounded framework for synthesizing agentic trajectories to train open-ended agentic LLM. Addressing the scarcity of high-quality, long-horizon execution data, our pipeline systematically transforms a repository of over 50,000 real-world skills into executable task instances. Our method proceeds by first generating structured Standard Operating Procedures and synthesizing necessary initial environments for environment-dependent skills. A multi-agent collaboration framework then produces realistic task scenarios and corresponding evaluation rubrics grounded in process- and outcome-criteria. A key contribution is our dual-path trajectory sampling strategy, where an OpenClaw agent executes tasks with and without explicit skill documentation to capture both expert-level demonstrations and robustness-oriented exploration behaviors. We employ rejection sampling against generated rubrics to filter trajectories, yielding a high-quality dataset for training powerful agentic LLM.

1 Introduction

Recent advances in large language models (LLMs) have enabled increasingly capable autonomous agents that interact with external tools, APIs, and execution environments to accomplish complex user goals (Yao et al., 2023; sch, 2023). Beyond conventional function calling, modern agent systems are expected to perform long-horizon reasoning, invoke multiple tools sequentially, recover from intermediate failures, and maintain consistency across extended execution trajectories. Such capabilities are particularly important in open-ended agent frameworks such as `OpenClaw`, where models operate in realistic environments with dynamically composable skills and external tool ecosystems.

Despite rapid progress in agentic LLMs, current tool-use training data remains limited in several aspects. First, many existing datasets (Li et al., 2023b; Qin et al., 2023) focus on isolated API invocation or single-turn function calling, with limited emphasis on long-horizon multi-step execution. Second, several benchmarks primarily evaluate tool selection correctness rather than end-to-end task completion quality. Third, manually constructing realistic multi-tool trajectories is expensive and hard to scale for open-ended agent environments containing hundreds of heterogeneous skills.

Meanwhile, recent agent benchmarks such as PinchBench (web, 2026b) and Claw-Eval (Ye et al., 2026) have revealed a substantial gap between conventional language modeling capabilities and real-world agent execution performance. Models that perform strongly on standard NLP or reasoning benchmarks may still fail at practical agent workflows involving sequential tool usage, environment interaction, and execution planning (Mitra et al., 2024; Zeng et al., 2024). Existing studies (Ruan et al., 2024) similarly show that robustness in realistic tool-use environments remains a major challenge for open-source LLMs.

In this work, we propose a scalable synthetic trajectory construction framework for improving multi-step agent execution capabilities in `OpenClaw`. Instead of manually annotating trajectories, our framework automatically synthesizes executable training tasks from existing skill specifications hosted on ClawHub (web, 2026a). Specifically, we first leverage skill documentation (`skills.md`) to synthesize structured standard operating procedures (SOPs) describing valid multi-step tool-use workflows. We then utilize a multi-agent collaboration system to synthesize realistic execution environment and corresponding user scenarios, producing diverse task instances that require sequential tool interactions.

The synthesized instructions are executed by an `OpenClaw` agent in a dual-path sampling strategy to obtain complete execution trajectories, including

intermediate tool calls and environment interactions. To assess trajectory quality, we further construct task-specific evaluation rubrics that combine process adherence checks with outcome verification. These rubrics are subsequently used by an LLM judge to score execution trajectories. Finally, only high-quality trajectories are retained via rejection sampling to construct dataset for supervised fine-tuning.

Our framework differs from prior tool-use data construction approaches (Tang et al., 2023; Shen et al., 2024) in several important aspects. Rather than focusing on API-level supervision or static function calling, we target realistic agent execution trajectories in open environments. Our method jointly synthesizes task instructions, execution workflows, execution environments, and evaluation criteria in a unified pipeline, enabling scalable generation of high-quality long-horizon supervision signals.

Experimental results show that supervised fine-tuning on the synthesized trajectories consistently improves agent performance on challenging benchmarks including PinchBench and Claw-Eval. Interestingly, we observe that performance gains remain significant regardless of whether the underlying agent is explicitly provided with skill documentation during execution, suggesting that the synthesized trajectories encode transferable procedural knowledge beyond explicit tool descriptions.

Overall, our work demonstrates that skill-grounded synthetic trajectory generation provides an effective and scalable paradigm for improving open-ended agent execution capabilities. We hope our findings can facilitate future research on realistic agent training, long-horizon tool use, and automatic construction of high-quality agent supervision data.

2 Related Work

2.1 Tool Use and Agentic LLMs

Large language models (LLMs) have driven research on autonomous agents that interact with tools and environments. Toolformer (sch, 2023) learned API usage via self-supervision, while Gorilla (Patil et al., 2024) addressed large-scale API retrieval. Systems such as HuggingGPT (Shen et al., 2023), Chameleon (Lu et al., 2023), and Visual ChatGPT (Wu et al., 2023a) orchestrated multiple expert models through language-based planning. Agentic reasoning and long-horizon decision making were advanced by ReAct (Yao et al., 2023) with interleaved reasoning and acting, and extended by Reflexion (Shinn et al., 2023), Voyager (Wang et al., 2023a), DEPS (Wang et al., 2023b), and AutoGen (Wu et al., 2023b), which explore self-reflection, iterative planning, and multi-agent collaboration. Frameworks such as MetaGPT (Hong et al., 2024), AutoGPT, and BabyAGI further demonstrated autonomous long-horizon workflows.

For evaluation, ToolBench and ToolLLM (Qin et al., 2023) provide large-scale API-invocation tasks, and APIBench (Patil et al., 2024), ToolAlpaca (Tang et al., 2023), and xLAM (Zhang et al., 2025a) extend tool-use instruction tuning. More recent multi-turn benchmarks include MTU-Bench (Wang et al., 2024), BFCL (Patil et al., 2025), and ACEBench (Chen et al., 2025).

However, most existing work focuses on isolated function calling, static tool descriptions, and short horizons; manually curated trajectories are costly to scale. In contrast, we target realistic multi-step execution in open agent environments and automatically synthesize executable tasks, SOPs, rubrics, and supervision from skill specifications, enabling scalable generation of long-horizon trajectories.

2.2 Synthetic Data Generation

Synthetic data generation scales instruction tuning and reasoning supervision. Self-Instruct (Wang et al., 2023c) pioneered automatic instruction generation using LLMs, and subsequent work (Alpaca (Dubois et al., 2023), Vicuna (Peng et al., 2023), UltraChat (Ding et al., 2023), Evol-Instruct (Xu et al., 2025)) showed that synthetic corpora improve conversational models.

For reasoning, chain-of-thought prompting (Wei et al., 2022; Kojima et al., 2022) revealed that intermediate traces enhance complex reasoning. Rationale distillation (Ho et al., 2023), process supervision (Lightman et al., 2023), self-refinement (Madaan et al., 2023), and self-improvement (Huang et al., 2023) further demonstrated the value of reasoning trajectories as supervision signals.

In agent settings, Voyager (Wang et al., 2023a) accumulates executable skills via interaction; CAMEL (Li et al., 2023a) and AutoGen (Wu et al., 2023b) generate multi-agent conversations; and SWE-agent (Yang et al., 2024) highlights the importance of execution trajectories for robustness.

Automatic feedback and evaluation: Constitu-

tional AI (Bai et al., 2022), RLAIF (Lee et al., 2024), and LLM-as-a-Judge studies (Zheng et al., 2023; Dubois et al., 2025) show that LLMs can provide scalable supervision, and recent benchmarks increasingly adopt LLM judges for open-ended assessment.

Unlike prior approaches, our framework synthesizes complete, skill-grounded multi-step execution workflows with task-specific evaluation rubrics derived from skill specifications and SOPs. Trajectories are explicitly filtered using auto-generated judging criteria, improving supervision quality and consistency.

2.3 Agent Benchmarks and Evaluation

Evaluation protocols have shifted from static language understanding (GLUE (Wang et al., 2018), SuperGLUE (Wang et al., 2019), MMLU (Hendrycks et al., 2021), BIG-Bench (Srivastava et al., 2023)) to realistic interactive settings. Agentic benchmarks now assess web/desktop tasks (WebGPT (Nakano et al., 2022), WebArena (Zhou et al., 2024), Mind2Web (Deng et al., 2023), OSWorld (Xie et al., 2024), GAIA (Mialon et al., 2023)) and tool use (ToolBench (Qin et al., 2023), MTU-Bench (Wang et al., 2024), BFCL (Patil et al., 2025), ACEBench (Chen et al., 2025)). Long-horizon evaluations (AgentBench (Liu et al., 2025), SWE-Bench (Jimenez et al., 2024), τ-Bench (Yao et al., 2024)) show that LLMs fail at persistent planning, error recovery, and execution consistency, motivating scalable trajectory supervision. Open-ended benchmarks executed in realistic environment such as PinchBench (web, 2026b) and Claw-Eval (Ye et al., 2026) further reveal that strong language understanding does not ensure robust multi-step execution with tool composition, planning, and file operations.

Our work targets multi-step execution robustness via scalable synthetic trajectory generation and filtering, rather than isolated API correctness. Gains on PinchBench and Claw-Eval demonstrate that skill-grounded synthetic supervision effectively improves realistic agent performance.

3 Method

In this section, we present Skill2Task, a systematic pipeline that transforms a large-scale skill repository into high-quality agentic training trajectories. The core insight is that real-world agent skills serve as natural anchors for synthesizing diverse, executable, and evaluable task instances, each skill representing a discrete, tool-mediated capability. As illustrated in Figure 1, the pipeline proceeds through multiple stages: (1) Data Cleaning and Tagging, (2) Initial Environment Synthesis, (3) SOP Generation, (4) Scenario Generation, (5) Evaluation Rubric Generation, and (6) Trajectory Sampling.

3.1 Skill Cleaning and Tagging

The raw skill repository comprises over 50,000 skills aggregated from diverse sources, including open-source MCP servers, public tool registries, and community plugins. To ensure quality and structural organization, we process these skills into a verified subset $\mathcal{S}_{clean}$ through the following steps.

Malicious skill removal. We employ a dual-strategy approach to filter harmful entries. A safety classifier flags skills involving system destruction, unauthorized access, or privacy violations, while keyword-based pattern matching catches overtly harmful skills that evade the classifier. Skills flagged by either strategy are removed.

Deduplication. To address functional overlap across skill sources, we compute semantic similarity between skill descriptions using embedding-based representations extracted by Qwen3-Embedding-4B (Zhang et al., 2025b). Skill embeddings with cosine similarity exceeding a high threshold 0.92 are merged into clusters, retaining only the most comprehensive description per cluster to eliminate redundancy.

Functionality verification. We exclude skills with incomplete parameter definitions, missing required fields, broken endpoints, or ambiguous descriptions. A structural validator checks each remaining skill against a unified specification format. This process reduces the repository to approximately 38,000 verified skills.

Tagging. To enable structured task synthesis, we categorize each skill into one of four functional domains:

- *Software Engineering.* Skills related to coding, debugging, testing, version control, system administration, log analysis, *etc*.
- *Information & Knowledge Processing.* Skills involving search, retrieval, summarization, translation, analytical reasoning, data visualization, *etc*.

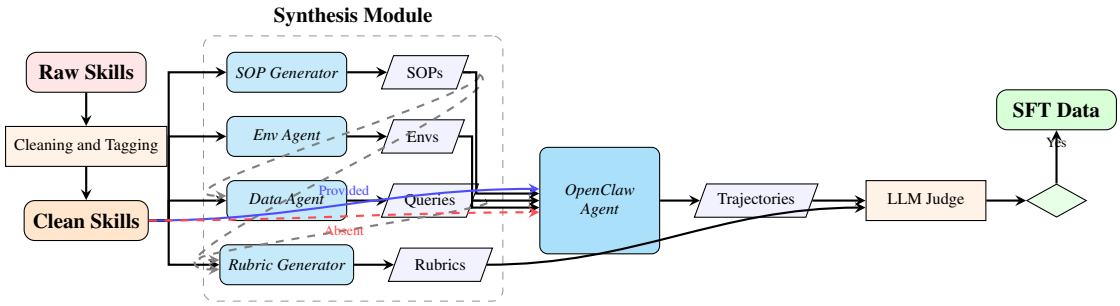

Figure 1: The Skill2Task pipeline.

Category	Count	Ratio
Software Engineering	10,085	28%
Information & Knowledge Processing	8,228	23%
Productivity & Daily Automation	11,448	32%
Domain-Specific Professional	5,751	16%
Others	466	1%
Total	**35,978**	**100%**

Table 1: Distribution of cleaned skills across the four-category taxonomy.

- *Productivity & Daily Automation.* Skills for event organization, communication, scheduling, file management, email drafting, workflow automation, *etc*.
- *Domain-Specific Professional.* Skills requiring specialized domain knowledge in areas such as scientific computation, financial analysis, legal document review, creative design, *etc*.

We prompt an LLM to assign categories based on skill descriptions and parameter schemas.

3.2 Initial Environment Synthesis

Agent skills differ in their environmental prerequisites. Some can be invoked in a default state, while others require a specific initial environment. For example, a skill that modifies a database is non-operable without a pre-existing database. We use classifier to partition $\mathcal{S}_{clean}$ into $\mathcal{S}_{clean} = \mathcal{S}_{ind} \cup \mathcal{S}_{dep}$, where $\mathcal{S}_{ind}$ denotes environment-independent skills (e.g., text generation, web search) and $\mathcal{S}_{dep}$ denotes environment-dependent skills (e.g., skills operating on a code repository or database).

For each skill $s \in \mathcal{S}_{dep}$, we deploy an **Environment Agent** $\mathcal{A}_{env}$ that synthesizes the required initial state:

$$E_s = \mathcal{A}_{env}(s) \tag{1}$$

where environment E_s is a package that specifies the infrastructure (e.g., a git repository, a predefined database) and data population (e.g., mock records, configuration files). For skills in $\mathcal{S}_{ind}$, we assign E_s to be the default environment.

3.3 SOP Generation

For each skill $s \in \mathcal{S}_{clean}$, we generate a corresponding Standard Operation Procedure (SOP) that decomposes the skill into an ordered sequence of executable steps. The SOP serves dual purposes: (1) to serve as a structured reference that ensures the solvability of task execution, and (2) to serve as the foundation of the process-level evaluation process (Section 3.5).

Given a skill s with its description and parameter schema, the SOP generator produces an ordered sequence of K steps:

$$\text{SOP}(s) = \langle e_1, e_2, \ldots, e_K \rangle \tag{2}$$

where each step e_k specifies: (1) the action to perform, (2) the expected input, and (3) the expected intermediate output. Every step must be **executable** and **verifiable**, which means each e_k corresponds to a concrete tool call or an observable state transition.

The SOP generation is followed by a validation process that checks: (a) whether each step references a valid skill or tool, (b) whether the ordering forms a valid dependency chain (no circular dependencies, inputs depend only on preceding outputs), and (c) whether the final step produces the intended outcome. SOP that fails validation are re-generated iteratively.

3.4 Scenario Generation

While the SOP provides a generic procedure, training requires diverse, realistic task instances grounded in concrete scenarios. We employ a **Data Agent** $\mathcal{A}_{data}$ to synthesize contextualized scenarios and generate challenging task queries.

For a given skill s with SOP(s), the Data Agent generates a scenario:

$$\sigma_s = \mathcal{A}_{data}(s, \text{SOP}(s)) \tag{3}$$

where σ_s describes the contextual background and specifies the user persona and intent.

Combining σ_s with skill s, the Data Agent further produces a task query q_s:

$$q_s = \mathcal{A}_{data}(\sigma_s, s, \text{SOP}(s)) \tag{4}$$

The task query q_s is a natural language instruction incorporating both the user's goal and contextual constraints. It also specifies the final state after the task is completed, *e.g.* what type of files should be saved at which directory.

3.5 Evaluation Rubric Generation

To assess whether an agent has successfully completed a synthesized task, we generate an evaluational rubric $\mathcal{R}_s$ for each task query q_s. Derived from the SOP and task scenario, the rubric consists of two complementary components:

Process Evaluation Criteria $\mathcal{R}_s^{proc}$: These criteria evaluate whether the agent follows a reasonable execution process. For each step e_k in the SOP, we define a checklist item specifying: (1) whether the agent attempts the step, (2) whether the correct tool or approach is selected, and (3) whether intermediate outputs match expectations.

Outcome Evaluation Criteria $\mathcal{R}_s^{out}$: These criteria evaluate the final result, focusing on (1) whether the primary objective is achieved, and (2) whether the output files meet the standard defined in q_s.

The two components are combined into a unified rubric normalized to a total 10-point scale:

$$\text{Normalize}(\mathcal{R}_s) = \text{Normalize}(\mathcal{R}_s^{proc} + \mathcal{R}_s^{out}) = 10 \tag{5}$$

The rubric $\mathcal{R}_s$ is generated by considering task query q_s, SOP(s), scenario σ_s and environment E_s all together, to ensuring that every evaluation criterion directly corresponds to a task requirement and avoiding misalignment between training objectives and evaluation metrics.

3.6 Dual-Path Trajectory Sampling

The final stage samples execution trajectories using the **OpenClaw Agent** $\mathcal{A}_{claw}$, which operates in real environments initialized according to E_s (Section 3.2). For each task query q_s, the agent produces an execution trajectory τ consisting of a sequence of thought-action-observation triplets

$$\tau_s = \langle a_1, o_1, a_2, o_2, \ldots, a_T, o_T \rangle \tag{6}$$

where a_t and o_t denote the agent action and environment observation at step t, respectively. To augment the diversity of the training data and encourage robust policy learning, we introduce a dual-path sampling strategy. Specifically, we execute the task query q_s under two distinct information configurations regarding the availability of skill documentation.

Skill-Provided Execution. In the first path, the agent is explicitly provided with the specific skill definition s (e.g., the documentation from `skills.md` and the interface schema) alongside the task query q_s. This setting simulates a standard retrieval-augmented scenario where the agent has full access to the "manual" of the tools it needs to use. The trajectory is generated conditionally:

$$\tau_s^{prov} \sim \mathcal{A}_{claw}(q_s, s \mid E_s) \tag{7}$$

Trajectories sampled from this path tend to align closely with the generated SOPs, demonstrating efficient and correct tool usage. They serve as high-quality demonstrations for supervised fine-tuning.

Skill-Absent Execution. In the second path, we withhold the specific skill documentation s from the agent's context. Crucially, although the semantic skill definition is hidden, the necessary operational assets mentioned in SOP(s) remain in the execution environment E_s, such as accessible MCP server addresses, active service endpoints, and essential interface argument information. The agent must generate the trajectory τ_s^{abs} without explicit instructions on how to utilize the specific tools:

$$\tau_s^{abs} \sim \mathcal{A}_{claw}(q_s, \emptyset \mid E_s) \tag{8}$$

This setting challenges the agent to leverage its internal parametric knowledge or perform exploration-based problem solving to determine the correct tool invocations. Compared to the τ_s^{prov} guided by skill, τ_s^{abs} exhibits different behavioral patterns, such as trial-and-error or reasoning-chains based on tool name and arguments.

By sampling trajectories under both configurations, we construct a diverse dataset that captures both expert-level execution and robustness-oriented

exploration. This dual-path approach ensures the trained LLM learns not only to follow explicit instructions but also to adapt to scenarios where skill documentation is incomplete or unavailable.

3.7 Dataset Construction by Rejection Sampling

After trajectory generation, an LLM judge assess each trajectory τ against the rubric $\mathcal{R}_s$, yielding a normalized score $r_s \in [0, 10]$ based on both process- and outcome- correctness. We perform N sampling rounds per task to encourage behavioral diversity. Trajectories are then filtered via rejection sampling: only those meet a high quality threshold ($r_s \geq 8$) are retained to construct the SFT dataset:

$$\mathcal{D}_{SFT} = \{\tau_s \mid r_s \geq 8\}. \tag{9}$$

This filtering mechanism eliminates hallucinated tool uses and failed execution paths, ensuring the LLM is trained by high-fidelity trajectories.

For tasks with average score $\bar{r}_s < 8$ from N-round sampling, we keep these tasks for the later reinforcement learning stage after SFT.

4 Experiments

4.1 Experimental Setup

We evaluate the efficacy of **Skill2Task** by training a general-purpose LLM within the OpenClaw framework. Our base model is Qwen3.6-35B-A3B. We utilize GLM-5.1 as the teacher model to generate synthetic trajectories via the dual-path sampling strategy described in Section 3. We compare three configurations: (1) the original base model; (2) the model fine-tuned exclusively on "with-skill" trajectories (τ^{prov}); and (3) the model fine-tuned on a mixture of "with-skill" and "w/o-skill" trajectories ($\tau^{prov} \cup \tau^{abs}$). All SFT training experiments are conducted on NVIDIA H100 GPUs with Slime framework (Zhu et al., 2025) .

4.2 Benchmarks

We adopt two rigorous benchmarks to assess agentic capabilities:

Claw-Eval. A trajectory-aware benchmark for autonomous agents, focusing on end-to-end execution reliability. We report `pass`^{3} (success in all three runs), `pass@3` (success in any of three runs), and the average task score.

PinchBench. A real-world benchmark for OpenClaw agents comprising 147 tasks across productivity, coding, and research domains. Performance is measured by the average task score over three runs.

4.3 Result on Claw-Eval

Table 2 presents the results on Claw-Eval. Supervised fine-tuning on synthesized trajectories yields substantial improvements across all metrics. Specifically, the SFT model achieves a `pass`^{3} of 53%, outperforming the baseline by 7 absolute points (15.2% relative gain). Similarly, `pass@3` improves from 61% to 68%, and the average score rises by 5.52 points. These results demonstrate that procedural knowledge distilled from Skill2Task trajectories effectively transfers to held-out tasks, enhancing both execution reliability and planning consistency.

4.4 Result on PinchBench

Table 3 reports the PinchBench results. Training on "with-skill" trajectories alone improves the score from 88.70% to 90.67%, validating the quality of expert demonstrations. Crucially, incorporating "w/o-skill" trajectories yields a further improvement to 92.79%. This ablation confirms the value of the dual-path sampling strategy: while "with-skill" trajectories teach correct tool usage and adherence to documentation, "w/o-skill" trajectories enhance the model's ability to handle scenarios where explicit instructions are absent, thereby improving robustness and generalization in open-ended environments.

4.5 Analysis

Effectiveness of skill-grounded synthetic supervision. The consistent improvements across both benchmarks validate our hypothesis: synthetic tasks derived from skill descriptions yield effective procedural supervision. For instance, the SFT model achieves a 7-point gain in `pass`^{3} on Claw-Eval and a 4.09-point gain on PinchBench. These results suggest that synthesized trajectories encode dense execution signals—such as environment interaction and tool orchestration—that are more effective for agentic training than isolated instruction-response pairs.

Why trajectory filtering matters. The rejection sampling mechanism is pivotal for data quality. By retaining only trajectories that meet high evalua-

Model	pass^3	pass@3	Avg. Score
Qwen3.6-35B-A3B (Baseline)	46	61	62.7
Qwen3.6-35B-A3B (SFT)	**53**	**68**	**68.2**

Table 2: Main evaluation results on Claw-Eval. We report pass^3, pass@3, and average score (%).

Model / Training Data	Score	Runs
Qwen3.6-35B-A3B (Base)	88.70	3
+ SFT (With-Skill Trajectories)	90.67	3
+ SFT (With- & W/O-Skill Trajectories)	**92.79**	3

Table 3: Evaluation results on PinchBench. We report the average score (%) over 3 runs.

tion standards ($r_s \geq 8$), we eliminate hallucinated tool uses and failed execution paths. This quality-oriented filtering ensures that the model learns exclusively from successful, executable demonstrations. The resulting performance gains confirm that a moderate volume of high-quality trajectories provides stronger supervision than a larger set of unfiltered, noisy data.

With-skill versus w/o-skill collection. The ablation study highlights the complementary nature of our dual-path strategy. "With-skill" trajectories offer structured, documentation-aligned demonstrations, teaching the model canonical workflows. Conversely, "w/o-skill" trajectories foster robustness by forcing the model to operate without explicit manuals, simulating real-world scenarios where tool documentation may be absent. The superior performance of the mixed-data setting demonstrates that combining these approaches balances adherence to procedures with adaptive problem-solving capabilities.

Transfer beyond the generated tasks. The significant gains on held-out benchmarks (Claw-Eval and PinchBench) indicate that knowledge transfers effectively from synthesized Skill Hub tasks to real-world evaluations. This validates Skill2Task as a scalable alternative to expensive manual annotation. Rather than overfitting to specific benchmarks, the model acquires generalizable execution behaviors—such as planning and error recovery—from the broad procedural knowledge encoded in the skill repository.

Pass metrics and execution robustness. The parallel improvements in Claw-Eval's pass^3 and pass@3 metrics reveal enhanced reliability and coverage. The gain in pass@3 indicates the model solves a wider range of tasks, while the gain in pass^3 confirms greater consistency across retries. Additionally, the rise in average score suggests that even when tasks are not fully solved, the model produces more coherent partial solutions, reducing the frequency of severe execution failures.

Data efficiency. Despite the final training set containing only 4,624 tasks, the model achieves substantial performance improvements. This data efficiency underscores the value of structured, skill-grounded trajectories. The dense, end-to-end supervision signals provided by these trajectories prove to be a highly efficient training resource compared to large-scale but shallow instruction data.

5 Conclusion

We presented **Skill2Task**, a skill-grounded framework for synthesizing agentic trajectories to train open-ended agentic LLMs. Addressing the critical bottleneck of high-quality supervision data scarcity, our pipeline systematically transforms a large repository of real-world skills into executable tasks, complete with structured SOPs, synthesized environments, and process-aware evaluation rubrics. A central contribution is our dual-path trajectory sampling strategy, which captures both expert-level demonstrations via skill-provided execution and robustness-oriented exploration via skill-absent execution. Experimental results on Claw-Eval and PinchBench demonstrate that models trained on our filtered synthetic data achieve significant improvements in task completion success rates and execution reliability.

6 Limitations

While Skill2Task demonstrates promising results in synthesizing high-quality training data for agentic LLMs, several limitations remain. The current experimental study focuses exclusively on the efficacy of supervised fine-tuning (SFT). Although our pipeline naturally generates a subset of trajectories suitable for reinforcement learning, specifically those with partial success scores that were filtered out during rejection sampling, we have not yet validated the utility of this data in a subsequent RL phase. Future work should investigate how RL algorithms can exploit these exploration-oriented trajectories to refine policy robustness, potentially enabling the agent to learn error recovery strategies that are difficult to acquire through imitation learning alone.

References

2023. Toolformer: language models can teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

2026a. Claw hub. https://clawhub.ai/.

2026b. Pinchbench. https://github.com/pinchbench/skill.

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, and 32 others. 2022. Constitutional ai: Harmlessness from ai feedback. *Preprint*, arXiv:2212.08073.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Yuefeng Huang, Xiangcheng Liu, Wang Xinzhi, and Wu Liu. 2025. ACEBench: A comprehensive evaluation of LLM tool usage. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12970–12998, Suzhou, China. Association for Computational Linguistics.

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 *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051, Singapore. Association for Computational Linguistics.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2025. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *Preprint*, arXiv:2404.04475.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacafarm: a simulation framework for methods that learn from human feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14852–14882, Toronto, Canada. Association for Computational Linguistics.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. *Preprint*, arXiv:2308.00352.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large

language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, Singapore. Association for Computational Linguistics.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? *Preprint*, arXiv:2310.06770.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2024. Rlaif vs. rlhf: scaling reinforcement learning from human feedback with ai feedback. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. Camel: communicative agents for "mind" exploration of large language model society. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023b. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, Junjie Gao, Junjun Shan, Kangning Liu, Shudan Zhang, Shuntian Yao, Siyi Cheng, Wentao Yao, Wenyi Zhao, Xinghan Liu, and 11 others. 2024. Autoglm: Autonomous foundation agents for guis. *Preprint*, arXiv:2411.00820.

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, and 3 others. 2025. Agentbench: Evaluating llms as agents. *Preprint*, arXiv:2308.03688.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: plug-and-play compositional reasoning with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: iterative refinement with self-feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. *Preprint*, arXiv:2311.12983.

Arindam Mitra, Luciano Del Corro, Guoqing Zheng, Shweti Mahajan, Dany Rouhana, Andres Codas, Yadong Lu, Wei ge Chen, Olga Vrousgos, Corby Rosset, Fillipe Silva, Hamed Khanpour, Yash Lara, and Ahmed Awadallah. 2024. Agentinstruct: Toward generative teaching with agentic flows. *Preprint*, arXiv:2407.03502.

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. 2022. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:2112.09332.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff 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 *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

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*, ICML'25. JMLR.org.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: large language model connected with massive apis. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *Preprint*, arXiv:2304.03277.

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. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. 2024. Identifying the risks of lm agents with an lm-emulated sandbox. *Preprint*, arXiv:2309.15817.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: solving ai tasks with chatgpt and its friends in hugging face. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. 2024. Taskbench: benchmarking large language models for task automation. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 432 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Preprint*, arXiv:2206.04615.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: a stickier benchmark for general-purpose language understanding systems. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA. Curran Associates Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

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

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023b. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Pei Wang, Yanan Wu, Zekun Wang, Jiaheng Liu, Xiaoshuai Song, Zhongyuan Peng, Ken Deng, Chenchen Zhang, Jiakai Wang, Junran Peng, Ge Zhang, Hangyu Guo, Zhaoxiang Zhang, Wenbo Su, and Bo Zheng. 2024. Mtu-bench: A multi-granularity tool-use benchmark for large language models. *Preprint*, arXiv:2410.11710.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023c. 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)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023a. Visual chatgpt: Talking, drawing and editing with visual foundation models. *Preprint*, arXiv:2303.04671.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023b. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. Osworld: benchmarking multimodal agents for open-ended tasks in real computer environments. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. 2024. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *Preprint*, arXiv:2405.14333.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2025. Wizardlm: Empowering large pre-trained language models to follow complex instructions. *Preprint*, arXiv:2304.12244.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023. On the tool manipulation capability of open-source large language models. *Preprint*, arXiv:2305.16504.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: agent-computer interfaces enable automated software engineering. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, and 1 others. 2026. Claw-eval: Toward trustworthy evaluation of autonomous agents. *arXiv preprint arXiv:2604.06132*.

Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, Chenliang Li, Yuanhong Xu, Hehong Chen, Junfeng Tian, Qi Qian, Ji Zhang, Fei Huang, and Jingren Zhou. 2024. mplug-owl: Modularization empowers large language models with multimodality. *Preprint*, arXiv:2304.14178.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, Bangkok, Thailand. Association for Computational Linguistics.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Zhiwei Liu, Yihao Feng, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese, and Caiming Xiong. 2025a. xLAM: A family of large action models to empower AI agent systems. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11583–11597, Albuquerque, New Mexico. Association for Computational Linguistics.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. Lima: less is more for alignment. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

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. *Preprint*, arXiv:2307.13854.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Xiangru Tang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. 2023b. Agents: An open-source framework for autonomous language agents. *Preprint*, arXiv:2309.07870.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. 2025. slime: An llm post-training framework for rl scaling. https://github.com/THUDM/slime. GitHub repository. Corresponding author: Xin Lv.