AutoMaster: Autonomous Post-Training by Making the Same Model a Better Teacher

Guoxin Chen*, Shuang Sun*, Fanzhe Meng*, Xinyu Tang, Ruihua Song✉ and Wayne XinZhao✉

[1]Gaoling School of Artificial Intelligence, Renmin University of China

Task success alone does not specify which behaviors post-training should reinforce. Even successful teacher trajectories can contain unsupported assumptions, unproductive repetition, or superficial verification. We propose Skill-FT (Skill-Guided Fine-Tuning), which uses natural-language skills to suppress undesirable habits and elicit underrepresented effective behaviors during teacher rollouts, then fine-tunes a target model on the resulting trajectories. We introduce AutoMaster, an autonomous post-training system in which an LLM agent develops these interventions through trajectory analysis, skill design, data selection, and feedback from rollouts and target-model evaluation. Across Terminal, Search, and software engineering tasks with 4B and 9B target models, AutoMaster improves mean performance over Vanilla SFT on successful unguided teacher trajectories by 1.42–8.99 percentage points, with the teacher, task pool, training set size, and SFT recipe held fixed. Together with the ablations and trajectory analyses, these results suggest that behavioral shaping can make a fixed model a more effective teacher. We further find that retaining skill instructions during SFT yields higher mean performance than removing them across all six settings, despite their absence at evaluation and identical supervised outputs. These findings suggest that preserving skill context as part of a training demonstration can support better behavioral internalization.

*Equal Contributions. ✉Corresponding authors.

Date: Nov. 20, 2026.

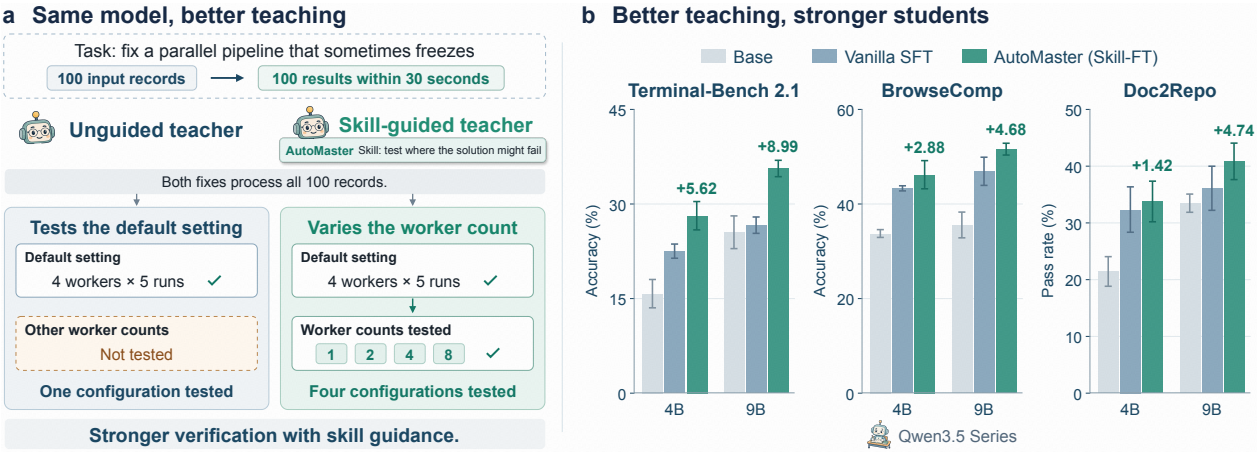


Figure 1. A correct result is only part of what a teacher demonstrates. AutoMaster designs natural-language skills to guide teacher rollouts and selects demonstrations for fine-tuning. **(a)** In this example, both pipelines process all 100 records with four workers. The skill-guided teacher also checks whether the fix holds with fewer or more workers. **(b)** With the same teacher, task pool, training set size, and SFT recipe, students trained with AutoMaster achieve higher performance than their Vanilla SFT counterparts on all three benchmarks at both model sizes (4B and 9B). Labels above the bars show gains of 1.42–8.99 percentage points over Vanilla SFT. Error bars show ±1 SD over three evaluation runs per checkpoint.

Emails: {gx.chen.chn, sunshuanguns, mengfanzhe16, batmanfly}@gmail.com

1 Introduction

LLM agents are beginning to undertake model development itself, from creating training data to executing post-training pipelines (Kulikov et al., 2026; Chen et al., 2026c; Rank et al., 2026). This opens a path toward autonomous post-training, in which an agent diagnoses model weaknesses, proposes interventions, and uses training feedback to guide further improvement. A central question is what such a system should seek to change.

A common approach to agent post-training is supervised fine-tuning (SFT) on successful trajectories (Chen et al., 2023; Ta et al., 2026). Yet *task success does not specify which behaviors post-training should reinforce*. Unsupported assumptions, unproductive repetition, or superficial verification can lead to failure, but they can also persist in trajectories that reach the correct outcome. Fine-tuning on such successful trajectories can reinforce these behaviors alongside the actions that support success. Meanwhile, desirable behaviors may remain underrepresented even among successful trajectories. We therefore argue that autonomous post-training should actively shape the behaviors demonstrated by the teacher, alongside selecting trajectories for training.

To this end, we propose **Skill-FT** (Skill-Guided Fine-Tuning), which uses natural-language skills to shape the teacher's behavioral patterns during rollout. Skills specify how to reason or act in relevant situations and are appended to the system prompt. By suppressing undesirable habits and eliciting effective behaviors that are underrepresented in unguided rollouts, this guidance aims to improve task execution and the demonstrations available for learning. The resulting trajectories are used to fine-tune a target model.

Developing effective Skill-FT interventions requires connecting changes in teacher behavior to their effects on target-model learning. Doing so involves repeated cycles of trajectory analysis, skill design and validation, data selection, model training, and evaluation, with accumulated evidence guiding what to investigate next. This combination of analysis, experimentation, and feedback makes the workflow well suited to automation by LLM agents. We therefore introduce **AutoMaster**, an autonomous post-training framework in which an LLM agent designs, executes, and refines Skill-FT interventions. The agent decides which behavioral patterns to target, how to design and apply skills, and which trajectories to use for training. It uses rollout evidence and evaluation of the fine-tuned model to revise these choices, linking the search for behavioral interventions to their effects on downstream model performance.

We evaluate AutoMaster on Terminal, Search, and software engineering tasks with Qwen3.5-4B and 9B (Qwen Team, 2026) as target models. Across all six benchmark–model settings, AutoMaster improves mean performance over Vanilla SFT by 1.42–8.99 percentage points, with the teacher, task pool, training set size, and SFT recipe held fixed. These gains indicate that successful-trajectory distillation can understate the post-training headroom available from a fixed teacher. Together with the component ablations and the task-matched comparison, these results suggest that Skill-FT can make a fixed model a more effective teacher by shaping the behaviors it demonstrates. Trajectory analyses further illustrate how skill guidance addresses behavioral weaknesses, even when both unguided and skill-guided rollouts succeed.

We further find that *reducing a training–evaluation prompt mismatch can make the same trajectories less effective for learning*. With the same trajectories and SFT recipe, retaining skill instructions during training yields higher mean performance than removing them across all six settings, even though all target models are evaluated without generated skills. The training examples suggest a possible explanation: retaining skill instructions keeps behavioral rules and their conditions of application explicit in the input. Removing them preserves the same reasoning and actions as supervision, but may require the target model to infer more of these rules from the task and

trajectory. For reasoning that explicitly references a skill, retaining the instructions also preserves the correspondence between the input guidance and the supervised output. These findings suggest that preserving skill context can help the target model better internalize the behaviors shaped during teacher rollouts.

Our contributions are summarized as follows:

- **Behavioral shaping beyond task success.** We propose Skill-FT, which uses natural-language skills during teacher rollouts to shape the behaviors represented in SFT demonstrations, including on tasks the teacher already solves. Our results suggest that such interventions can make a fixed model a more effective teacher.
- **Autonomous post-training framework.** We introduce AutoMaster, a framework in which an LLM agent designs, executes, and refines behavioral interventions. It jointly optimizes skill guidance and data selection using rollout evidence to guide subsequent exploration.
- **Skill context for behavioral internalization.** Controlled comparisons across three domains and two model sizes show that retaining skill instructions during SFT improves mean performance without test-time skills. This suggests that preserving the context behind supervised reasoning and actions can support better behavioral internalization.

2 Related Work

Autonomous Post-Training. Recent systems use LLM agents to iteratively revise training code in response to evaluation feedback (Karpathy, 2026; Chen et al., 2026a; Wei, 2026). For post-training, FT-Agent (Li et al., 2026) refines data and training strategies through structured planning and feedback analysis, while RSI-Master (Du et al., 2026) organizes research exploration through persistent experiment records and reviewer guidance. Within this paradigm, AutoMaster centers autonomous post-training on skill-guided behavioral shaping during teacher rollouts.

Skill-Based Agent Learning. Prompt and skill optimization methods use execution feedback to refine natural-language guidance without updating model parameters (Agrawal et al., 2025; Shi et al., 2026). Related work couples the evolution of skill libraries or procedural scaffolds with model training (Xia et al., 2026; Ding et al., 2026). Skill-to-LoRA (Zhang and Qi, 2026) trains skill-specific adapters on skill-guided demonstrations, while SkillGym (Ge et al., 2026) transforms skill documents into executable, verifiable training environments. Our work examines both how skills shape teacher demonstrations and whether retaining the skill instructions during SFT improves learning when those instructions are absent at evaluation.

Learning from Agent Trajectories. Prior work improves reasoning and agent capabilities by learning from generated trajectories (Chen et al., 2024b,a; Zeng et al., 2024), including trajectories that demonstrate error recovery (Fu et al., 2025; Yuan et al., 2025). Behavior Priming (Jin et al., 2026) selects trajectories exhibiting beneficial behaviors, while PROOF-Gen (Ta et al., 2026) uses per-task prompt optimization to recover verified demonstrations from teacher failures. Skill-FT complements trajectory selection and failure recovery by actively shaping behavioral patterns during teacher rollouts, including on tasks it already solves.

3 Autonomous Post-Training with Skill-FT

As illustrated in Figure 2, AutoMaster uses an LLM agent to develop and refine Skill-FT interventions through two nested loops. With a fixed task pool and teacher model, the inner loop designs

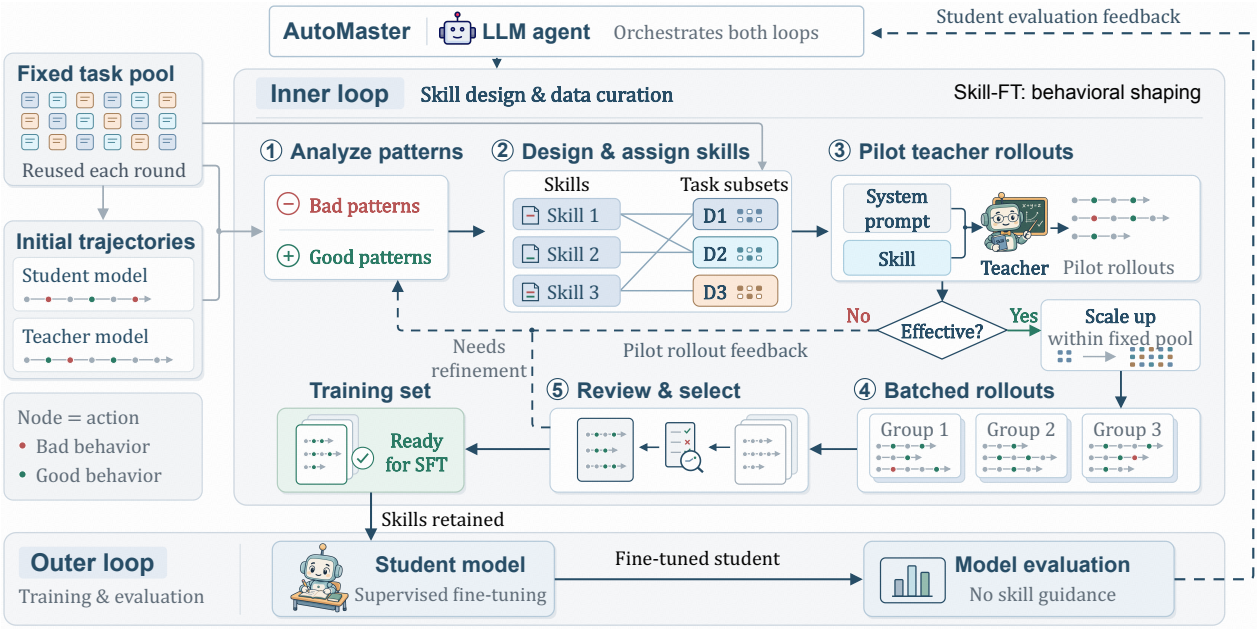


Figure 2. Overview of AutoMaster. An LLM agent coordinates two nested loops for skill-guided autonomous post-training. In the **inner loop**, the agent analyzes tasks and trajectories from a fixed task pool to design skills for relevant task subsets. These skills are injected into the teacher's system prompt to suppress undesirable habits and elicit effective but underrepresented behaviors during rollout. Pilot rollouts inform skill refinement before scaling up, while feedback from batched rollouts and data selection can trigger further revision. The **outer loop** fine-tunes and evaluates the target model, providing feedback that guides subsequent behavioral interventions.

and tests skills and selects training trajectories. The outer loop fine-tunes the target model from a fixed initial checkpoint and uses evaluation feedback to guide subsequent interventions.

AutoMaster uses an LLM agent to develop and refine Skill-FT interventions through two nested loops (Figure 2). Within each setting, the teacher, training task pool, and initial student checkpoint are fixed. In the inner loop, the agent decides how to guide the teacher and which demonstrations to use for training. In the outer loop, it fine-tunes the student on the selected demonstrations and uses the student's subsequent evaluation scores and trajectories to guide the next round.

3.1 Skill-FT: Skill-Guided Fine-Tuning

To shape the behavioral patterns represented in training trajectories, we guide the teacher with natural-language skills. These skills are designed to suppress undesirable habits and elicit effective behaviors that are underrepresented in unguided rollouts. A skill specifies how to reason or act in relevant situations and may include behavioral principles, concrete procedures, or domain-specific knowledge. Skills can be shared across tasks or combined to address different aspects of the same task. For a task x, let s denote the combined text of the selected skills. We append s to the base system prompt p and generate a trajectory through interaction with the task environment:

$$\tau \sim \text{Rollout}(\pi_T, x; p \oplus s), \tag{1}$$

where π_T denotes the teacher model and $\oplus$ denotes text concatenation. This guidance conditions the teacher's reasoning and actions during execution, enabling behavioral interventions even on tasks it already solves.

To internalize the demonstrated behaviors, we fine-tune the target model with standard SFT on selected teacher trajectories. We retain each trajectory's original system prompt, including any skill instructions, to preserve the conditions under which the demonstration was generated. The target model is evaluated under each benchmark's official protocol without adding the generated skills.

3.2 Inner Loop: Developing Behavioral Interventions

The inner loop develops and refines Skill-FT interventions through trajectory analysis and rollout experiments. The agent analyzes tasks and available trajectories to identify behavioral patterns worth suppressing or reinforcing. In subsequent rounds, it also examines evaluation trajectories from the fine-tuned target model to assess behavioral internalization and identify remaining weaknesses. These observations guide skill design and the selection of task subsets for intervention.

As shown in Figure 2, pilot rollouts test whether candidate skills elicit the intended behaviors without introducing new problems during task execution. The agent uses the resulting trajectories to refine both the skill instructions and their scope of application, repeating trials as needed.

The agent then scales promising interventions to batched rollouts and selects trajectories it judges valuable for training, considering both task outcomes and demonstrated behaviors. The agent may also select unguided teacher trajectories that already exhibit useful behavioral patterns. Feedback from generation and selection can prompt further revisions to skills or task coverage, followed by additional rollouts as needed. The selected trajectories form the training set for the outer loop.

3.3 Outer Loop: Model Training and Feedback

The outer loop tests whether the selected demonstrations translate into improved target-model performance. At round r, the training set $\mathcal{D}_r$ produced by the inner loop is used to fine-tune the target model from the same initial checkpoint θ_0:

$$\theta_r = \mathrm{SFT}(\theta_0, \mathcal{D}_r). \tag{2}$$

Evaluation scores measure the downstream performance of θ_r, while evaluation trajectories allow the agent to examine whether the intended behaviors appear in the fine-tuned model and identify remaining weaknesses. The agent uses this feedback to revise behavioral targets, skill instructions, task coverage, and data selection in subsequent inner-loop exploration. Prior skills, trajectories, and experimental findings remain available across rounds, allowing the search to build on accumulated evidence while each candidate model is trained from θ_0.

3.4 Infrastructure

The infrastructure lets the AutoMaster agent test how a skill shapes teacher behavior and examine whether the student exhibits that behavior after fine-tuning (Figure 3). From its Docker container, the agent uses the Model Context Protocol (MCP) (Anthropic, 2024) to access tasks and initial trajectories, submit teacher rollouts, and launch SFT on selected demonstrations. The rollout harness calls the teacher API, while the evaluation harness queries the student model served by SGLang (Zheng et al., 2024). Both harnesses use a shared sandbox cluster that runs task tools and verifiers in isolated Docker containers, allowing multiple tasks to execute concurrently.

When an SFT job completes, the execution controller checks that the checkpoint is available and automatically schedules evaluation. The training and evaluation GPU clusters share a checkpoint store: SGLang workers load the weights directly from the location where the SFT service wrote them, avoiding an intermediate copy to the evaluation cluster. The controller also records job

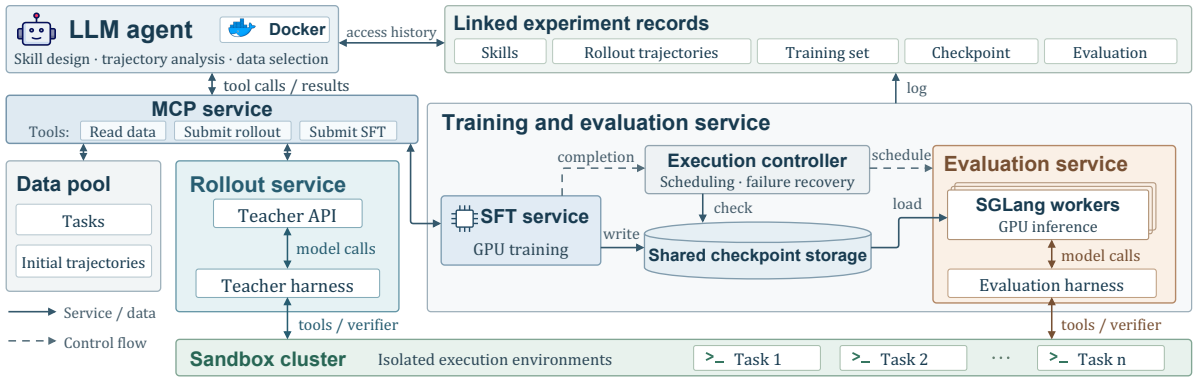


Figure 3. Infrastructure of AutoMaster. The agent accesses data and submits rollout and SFT jobs through a single MCP interface. Teacher rollouts and student evaluations share a sandbox cluster for isolated task execution. After SFT, the execution controller schedules evaluation on a separate GPU cluster, where SGLang workers load the checkpoint from shared storage. Linked experiment records let the agent trace each evaluation back to the training set and the skills that guided the teacher.

and stage progress so that it can restart a failed inference worker and resume evaluation without rerunning SFT.

Across rounds, AutoMaster links each student's evaluation scores and trajectories to its checkpoint, its training set, and the skills used during teacher rollouts. When a weakness recurs in student evaluations, the agent can inspect earlier rollouts to see whether the teacher demonstrated the desired behavior and check which of those demonstrations entered the training set. This evidence informs whether the agent revises a skill, changes its data selection, or both in the next round.

4 Experiments

4.1 Experimental Setup

Benchmarks and Task Pools. We evaluate Terminal, Search, and software engineering (SWE) tasks using Terminal-Bench 2.1 (Merrill et al., 2026), BrowseComp (BC; using the Lite-185 subset released by Traverse) (Wei et al., 2025; Ma et al., 2026), and Doc2Repo (Chen et al., 2026b), respectively. For each domain, we randomly sample 1,000 training tasks from public datasets (Meng et al., 2026; Deng et al., 2026; Zhao et al., 2026) to form a fixed task pool shared across all methods.

Models and Baselines. We use Qwen3.5-4B and Qwen3.5-9B (Qwen Team, 2026) as target models. Across all experiments, the LLM agent runs in Codex with GPT-6 Astra (OpenAI, 2026) at `xhigh` reasoning effort, while DeepSeek-V4-Pro-Preview (DeepSeek-AI, 2026) serves as the teacher at `high` reasoning effort. We compare against **Base**, the target model without additional fine-tuning, and **Vanilla SFT**, which performs standard SFT on 512 trajectories randomly sampled from successful unguided teacher rollouts on the shared task pool.

Training and Evaluation. For AutoMaster, the agent starts with trajectories generated by the initial target model and the unguided teacher on the training task pool, with a budget of five outer-loop rounds. Across all settings, each training run starts from the corresponding initial checkpoint and performs full-parameter SFT on 512 trajectories for three epochs, with a learning rate of 10^{-5} and a global batch size of 32. We evaluate final-epoch checkpoints without generated skills using domain-specific harnesses (Meng et al., 2026; Sun et al., 2026; Chen et al., 2026b), with identical

Table 1. Main results with Qwen3.5-4B and Qwen3.5-9B. We report task accuracy for Terminal-Bench 2.1 and BrowseComp, and mean test pass rate for Doc2Repo, all in percent (higher is better). Results are mean ± SD over three evaluation runs per checkpoint. Bold marks the highest mean in each column. Δ gives the gain over Vanilla SFT in percentage points.

	Terminal-Bench 2.1		BrowseComp		Doc2Repo	
Method	**4B**	**9B**	**4B**	**9B**	**4B**	**9B**
Base	15.73 ± 2.25	25.47 ± 2.59	33.69 ± 0.82	35.50 ± 2.72	21.39 ± 2.61	33.42 ± 1.60
Vanilla SFT	22.47 ± 1.12	26.59 ± 1.30	43.24 ± 0.54	46.85 ± 2.98	32.30 ± 4.00	36.05 ± 3.89
AutoMaster	**28.09** ± 2.25	**35.58** ± 1.30	**46.13** ± 2.98	**51.53** ± 1.25	**33.72** ± 3.58	**40.79** ± 3.21
Δ over Vanilla SFT	+5.62	+8.99	+2.88	+4.68	+1.42	+4.74

settings across methods within each domain. We measure task accuracy on Terminal-Bench 2.1 and BC and mean test pass rate on Doc2Repo, reporting mean ± SD over three evaluation runs per checkpoint. Further experimental details are provided in Appendix A.

4.2 Main Results

Table 1 shows that AutoMaster achieves the highest mean score across all six benchmark–model comparisons. Vanilla SFT improves over Base in every setting, demonstrating the effectiveness of learning from successful teacher trajectories. Building on this baseline, AutoMaster yields further gains of 1.42–8.99 percentage points with the same training set size and SFT recipe. Paired confidence intervals for these gains are reported in Appendix A.6. The gains are larger for the 9B model than for the 4B model across all three domains.

The Terminal-Bench results illustrate how Vanilla SFT gains can understate the available post-training headroom. For the 9B model, Vanilla SFT increases accuracy from 25.47% to 26.59%, whereas AutoMaster reaches 35.58% using the same teacher and training task pool. This comparison highlights the potential of jointly refining how teacher trajectories are generated and selected to improve downstream learning.

4.3 Ablation Studies

To understand how AutoMaster improves on Vanilla SFT, we separately ablate skill guidance and data selection. In **w/o Skill Guidance**, the LLM agent selects training trajectories from unguided teacher rollouts. In **w/o Data Selection**, the agent designs and refines skills for a fixed, randomly sampled subset of training tasks, without agent-directed trajectory selection. Both variants retain autonomous experimentation with the same teacher, task pool, training set size, and SFT recipe.

Figure 4 shows that removing either component lowers mean performance across all six benchmark–model settings. On Terminal-Bench 2.1 with the 4B model, both ablations fall below Vanilla SFT, whereas the complete framework improves upon it. The relative performance of the ablations also varies across settings. For the 9B model on Terminal-Bench 2.1, retaining skill guidance alone yields higher mean accuracy than retaining data selection alone (34.08% versus 28.46%). For the 4B model on BrowseComp, the ordering reverses: retaining data selection yields the higher mean (45.77% versus 43.78%). These results suggest complementary roles for the two components, with their relative benefits varying across tasks and target models.

We further test whether the choice of training tasks alone can account for the gains. **Task-Matched**

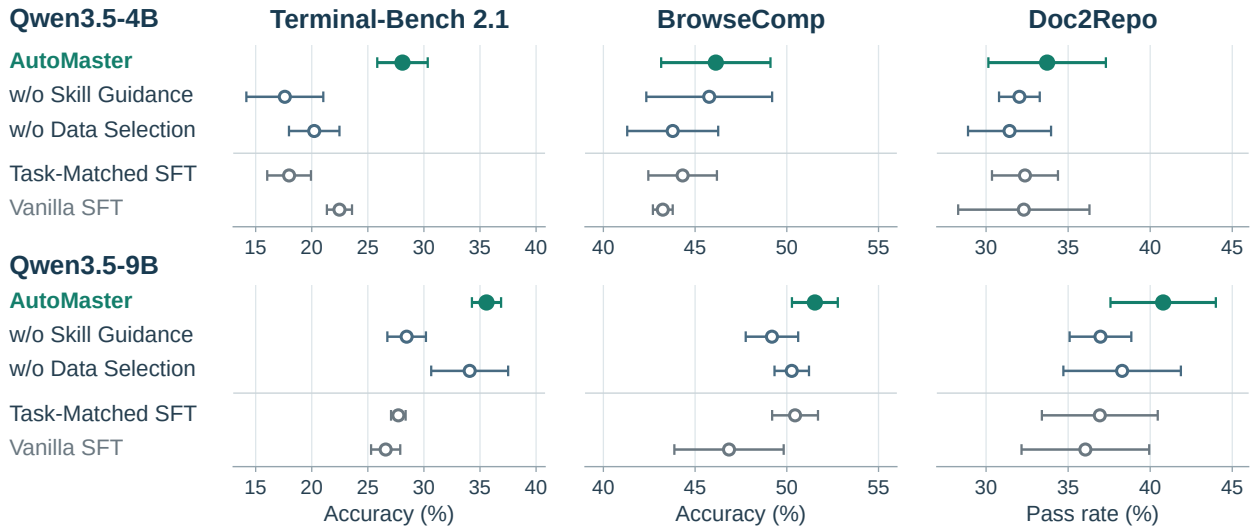


Figure 4. Contributions of skill guidance and data selection. *w/o Skill Guidance* retains agent-directed selection from unguided teacher rollouts. *w/o Data Selection* retains skill guidance on a fixed random subset of tasks but omits agent-directed trajectory selection. Removing either component lowers the mean score in each of the six settings. *Task-Matched SFT* uses unguided rollouts from the same teacher and matches the final training set of AutoMaster. Even with this match, AutoMaster scores 1.08–10.11 percentage points higher on average within each setting. Points show mean scores over three evaluation runs per checkpoint. Horizontal bars indicate ± 1 standard deviation.

SFT uses unguided trajectories from the same teacher and matches both the task IDs and the per-task sample counts of AutoMaster's final training set. Even with this match, AutoMaster achieves higher mean scores in all six settings, with gains of 1.08–10.11 percentage points. This gap persists with training-task composition held fixed, and removing either skill guidance or data selection lowers mean performance in every setting. Together, these results support *jointly optimizing how training demonstrations are generated and selected*.

5 Analysis

5.1 Behavioral Shaping Beyond Task Success

Task success can conceal weaknesses in a teacher's reasoning and actions. To examine the behaviors targeted by Skill-FT, Figure 5 contrasts unguided and skill-guided teacher trajectories on the same tasks across three domains.

The first two cases illustrate behavioral differences that task outcomes alone cannot distinguish. In Figure 5(a), both executions satisfy the numerical error threshold, yet the unguided teacher claims fourth-order accuracy despite measured convergence of approximately second order. The guided teacher uses independent numerical checks to revise and validate its method. Similarly, both search executions in Figure 5(b) reach the correct answer. The unguided trajectory repeatedly pursues unverified award guesses before finding supporting records, while the guided trajectory uses retrieved episode–director records to identify and verify a candidate. A filter based solely on successful completion would leave these behavioral differences unexamined.

The third case highlights the importance of diagnosing verification failures. Both executions initially construct a self-test with an input different from the specification while retaining the original expected order. The unguided teacher reverses the global sorting rule to satisfy this mistaken

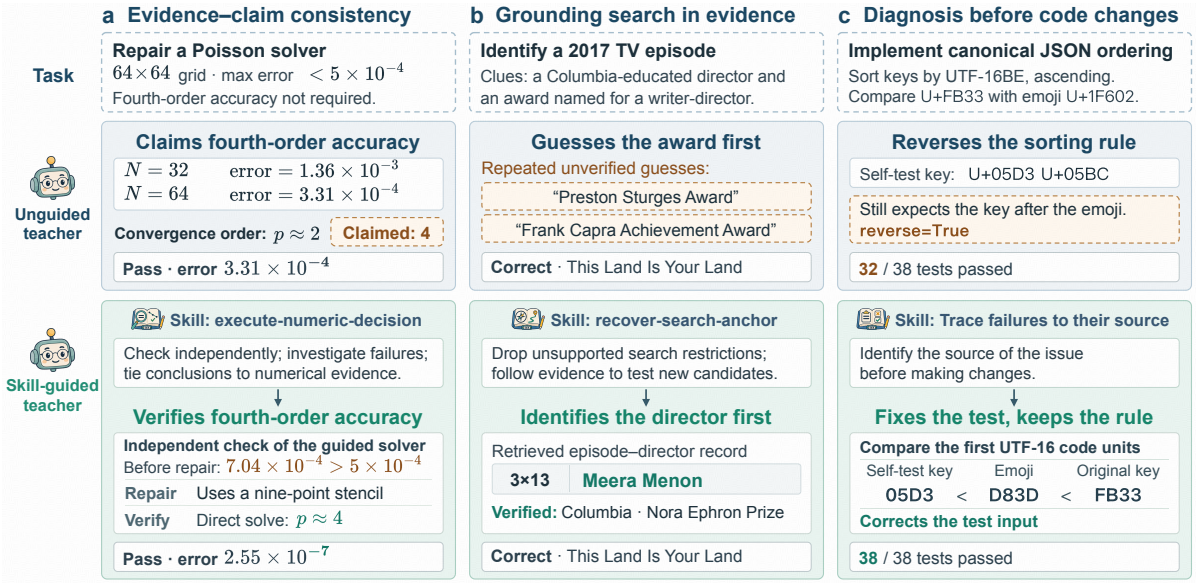


Figure 5. Skill-guided changes in teacher behavior. Each column compares unguided and skill-guided rollouts by the same teacher on the same task. **(a) Terminal:** Both solvers meet the error threshold. The unguided teacher claims fourth-order accuracy despite evidence of second-order convergence. The skill-guided teacher switches to a nine-point stencil and verifies fourth-order convergence using direct solves on successively finer grids. **(b) Search:** Both rollouts identify the same episode. The unguided teacher repeatedly guesses the award, while the skill-guided teacher identifies the director from episode credits and verifies the clues. **(c) SWE:** The unguided teacher reverses the global sort to satisfy a flawed self-test. The skill-guided teacher checks the UTF-16 encoding, corrects the self-test input, and keeps the required ascending order.

expectation. The guided teacher checks the input encoding, corrects the self-test, and preserves the specified rule, passing 38 of 38 tests compared with 32 of 38. The useful behavior here is to establish what a failed check actually implies before changing the implementation.

Across these cases, the guided trajectories illustrate changes in how the teacher validates a numerical claim, redirects a search, or diagnoses a failed check. The first two cases show how skill guidance can address behavioral weaknesses *even when both executions succeed*, while the third illustrates an improvement in implementation correctness. These differences highlight the value of examining demonstrated behaviors alongside task outcomes to identify opportunities for improvement. AutoMaster identifies such opportunities through trajectory analysis and designs skills to address them. Together with the downstream results, these observations suggest that Skill-FT can improve the training value of the demonstrations produced by a fixed teacher.

5.2 Preserving Skill Context for Better Internalization

We investigate how to train the target model to better internalize skill-guided behaviors. Specifically, we examine whether the skill instructions used to guide teacher rollouts should be retained during fine-tuning. We compare **Keep**, which retains these instructions in the training input, with **Remove**, which omits them. Both conditions use the same trajectories and SFT recipe. All models are evaluated without generated skills, following the official evaluation protocols.

Figure 6 shows that Keep achieves higher mean performance than Remove across all six benchmark–model comparisons, with gains of 2.15–4.01 percentage points. Removing skill instructions reduces

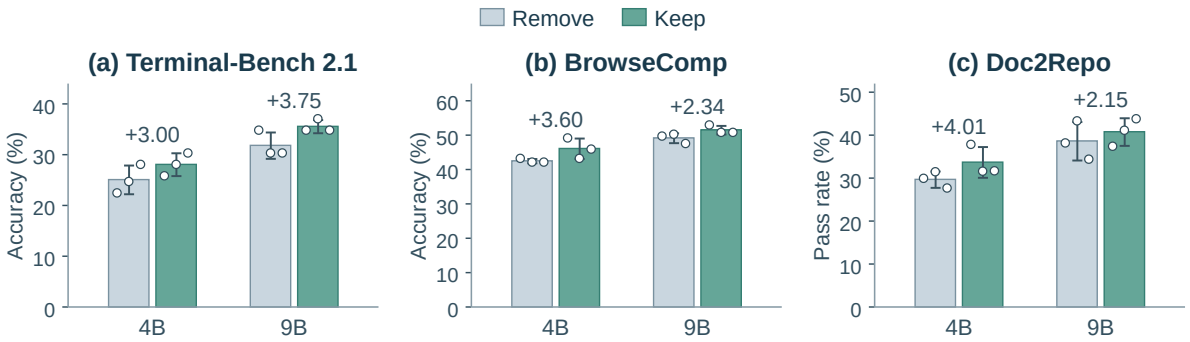


Figure 6. Retaining skill instructions during SFT improves performance without test-time skills. Both conditions use the same teacher trajectories and SFT recipe. Keep retains the skill instructions in the training input, whereas Remove omits them. All students are evaluated without skills under the official benchmark protocols. Bars show mean scores over three evaluation runs per checkpoint, circles show individual runs, and error bars indicate ±1 SD. Labels give the gain of Keep over Remove in percentage points.

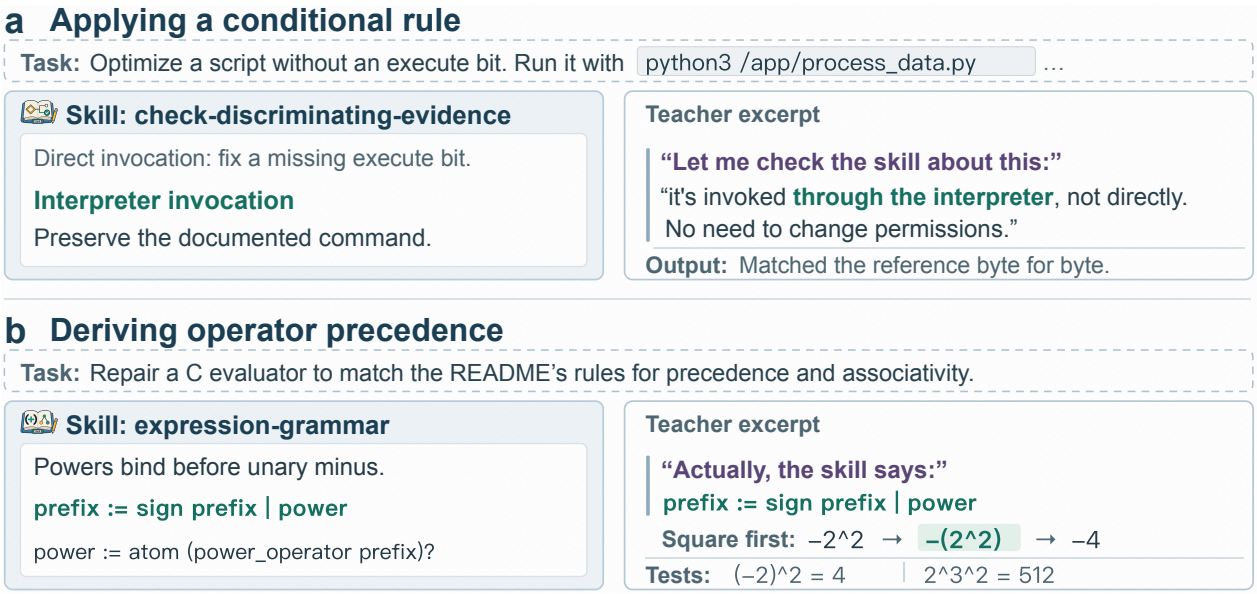


Figure 7. Explicit use of skill context in teacher demonstrations. Two examples from the SFT corpus illustrate how skill instructions inform the teacher's reasoning and actions. **(a)** The teacher applies a conditional rule to distinguish interpreter invocation from direct execution, avoiding an unnecessary permission change. **(b)** The teacher references the supplied grammar to derive an expression's grouping, then implements and tests the corresponding parser. Skill instructions and trajectory excerpts are condensed. Quotation marks indicate verbatim excerpts.

an explicit difference between training and evaluation prompts, yet lowers mean performance. These results show that *guidance can benefit learning even when it is absent at evaluation*.

The training trajectories offer a possible explanation for this pattern (Figure 7). In the first example, the teacher explicitly consults a conditional rule and determines that invoking a script through the interpreter requires no change to executable permissions. In the second, it references the supplied grammar to derive an expression's grouping, then implements and tests the corresponding parser.

Retaining skill instructions keeps behavioral rules and their conditions of application explicit in the

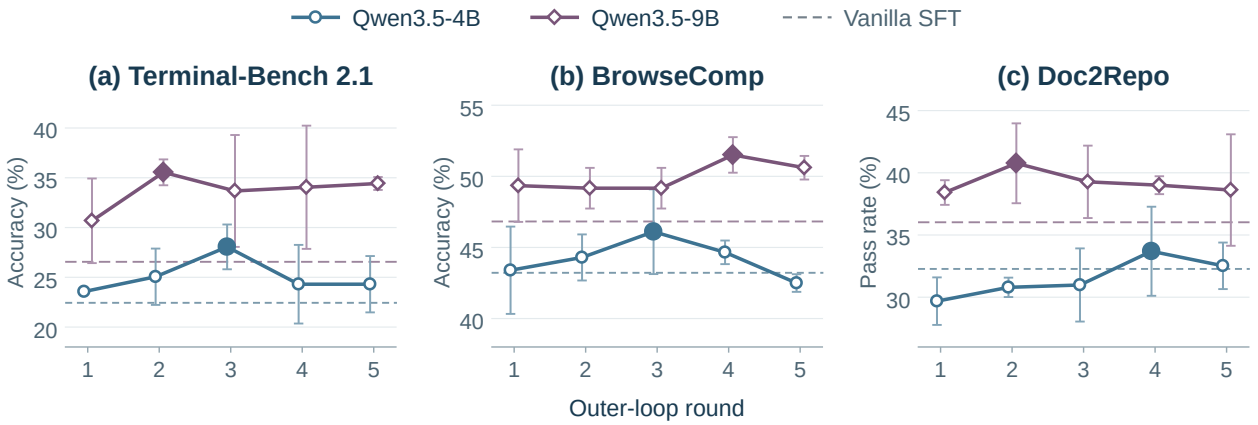


Figure 8. Performance across five rounds of autonomous experimentation. Each round starts from the corresponding initial target checkpoint. Points show individual-round means over three evaluation runs per checkpoint, and error bars indicate ±1 SD. Dashed lines show the corresponding Vanilla SFT baselines, and stars mark the highest mean in each series.

training input. Removing them preserves the same reasoning and actions as supervision, but may require the target model to infer more of these rules from the task and trajectory. For reasoning that explicitly references a skill, retaining the instructions also preserves the correspondence between the input and the supervised output. Together, these findings suggest that preserving skill context as part of a training demonstration can support *better behavioral internalization*.

5.3 Autonomous Experimentation Across Rounds

Having examined how skill context affects learning from teacher demonstrations, we next investigate how AutoMaster develops the interventions used to generate and curate those demonstrations. The agent draws on accumulated rollout and training feedback to explore alternative behavioral hypotheses and intervention strategies.

Figure 8 shows that continued experimentation identifies higher-performing candidates in all six benchmark–model settings. The highest mean scores occur in rounds two through four, exceeding first-round performance by 2.16–4.87 percentage points. Variation across rounds is consistent with the exploratory protocol: the agent continues to investigate alternative interventions after finding a strong candidate, without requiring each trial to improve on its predecessor. On BrowseComp-9B, mean accuracy remains near 49% for the first three rounds before reaching 51.53% in the fourth. This delayed improvement illustrates how continued experimentation can uncover further gains after several rounds of limited progress.

The consistency of gains also differs across model sizes. The 9B model exceeds Vanilla SFT in mean performance in every round across all three domains, whereas the 4B model does so throughout all five rounds only on Terminal-Bench 2.1. The component ablations show a similar pattern: both variants exceed Vanilla SFT in all three domains for the 9B model, but only on BrowseComp for the 4B model (Section 4.3). Together, these observations suggest that *greater target-model capacity may support more consistent gains from the interventions explored by AutoMaster*.

6 Conclusion

We introduced Skill-FT for shaping teacher behavior with natural-language skills and AutoMaster, an autonomous post-training system that designs, executes, and refines these interventions. Across three domains and two model sizes, AutoMaster achieves higher mean performance than Vanilla SFT. The results and trajectory analyses suggest that a fixed model can become a more effective teacher through behavioral shaping, even on tasks it already solves. Retaining skill instructions during SFT also yields higher mean performance than removing them, despite their absence at evaluation, suggesting that preserving skill context can support better behavioral internalization.

References

L. A. Agrawal, S. Tan, D. Soylu, N. Ziems, R. Khare, K. Opsahl-Ong, A. Singhvi, H. Shandilya, M. J. Ryan, M. Jiang, C. Potts, K. Sen, A. G. Dimakis, I. Stoica, D. Klein, M. Zaharia, and O. Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning, 2025. URL https://arxiv.org/abs/2507.19457.

Anthropic. Introducing the Model Context Protocol, Nov. 2024. URL https://www.anthropic.com/news/model-context-protocol.

AweAI Team. AweAgent: A unified, composable framework to build, evaluate, and train agents. https://github.com/AweAI-Team/AweAgent, 2026. GitHub repository.

B. Chen, C. Shu, E. Shareghi, N. Collier, K. Narasimhan, and S. Yao. FireAct: Toward language agent fine-tuning, 2023. URL https://arxiv.org/abs/2310.05915. arXiv:2310.05915.

G. Chen, M. Liao, C. Li, and K. Fan. Step-level value preference optimization for mathematical reasoning. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7889–7903, Miami, Florida, USA, Nov. 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.463. URL https://aclanthology.org/2024.findings-emnlp.463/.

G. Chen, M. Liao, C. Li, and K. Fan. Alphamath almost zero: Process supervision without process. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=VaXnxQ3UKo.

G. Chen, J. Chen, L. Chen, J. Zhao, F. Meng, W. X. Zhao, R. Song, C. Chen, J.-R. Wen, and K. Jia. Toward autonomous long-horizon engineering for ml research. *arXiv preprint arXiv:2604.13018*, 2026a.

G. Chen, F. Meng, J. Zhao, M. Li, D. Cheng, H. Song, J. Chen, Y. Lin, H. Chen, X. Zhao, R. Song, C. Liu, C. Chen, K. Jia, and J.-R. Wen. BeyondSWE: Can current code agent survive beyond single-repo bug fixing? *arXiv preprint arXiv:2603.03194*, 2026b. URL https://arxiv.org/abs/2603.03194.

W. Chen, X. Yang, X. Yang, T. Sha, Q. Li, Z. Wang, B. Xian, F. Kong, W. Liu, and J. Bian. Agent^2 RL-Bench: Can LLM agents engineer agentic RL post-training?, 2026c. URL https://arxiv.org/abs/2604.10547. arXiv:2604.10547.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

J. Deng, Y. Chen, X. Xiang, Z. Zeng, S. Tang, Z. Zhang, W. X. Zhao, F. Chang, C. Hao, Y. Wei, R. Tao, B. Dai, and J.-R. Wen. FORT-Searcher: Synthesizing shortcut-resistant search tasks for training

deep search agents. *arXiv preprint arXiv:2606.12087*, 2026. URL https://arxiv.org/abs/2606.12087.

F. Ding, Y. Zhang, R. Liu, Y. Liao, Z. Zeng, and H. Yang. Scaffold-mediated post-training: Co-evolving model parameters and procedural scaffold graphs, 2026. URL https://arxiv.org/abs/2608.05156.

R. Dror, G. Baumer, S. Shlomov, and R. Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1128. URL https://aclanthology.org/P18-1128/.

Y. Du, X. Yang, Z. Zhou, Y. Ge, C. Wang, J. Wang, S. Chen, Z. Liu, Y. Zhang, W. Gu, J. Zhang, Z. Lei, and S. Chen. RSI-Master: Structuring experiments to guide autonomous model improvement, 2026. URL https://arxiv.org/abs/2609.35561.

D. Fu, K. He, Y. Wang, W. Hong, Z. GongQue, W. Zeng, W. Wang, J. Wang, X. Cai, and W. Xu. AgentRefine: Enhancing agent generalization through refinement tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=FDimWzmcWn.

Z. Ge, Y. Shao, Y. Yang, Y. Cai, J. Zhou, K. Chen, B. Zhang, Q. Chen, and L. He. SkillGym: Internalizing human skills into LLMs for real-world problem solving, 2026. URL https://arxiv.org/abs/2609.27717.

J. Jin, A. S. Paladugu, and C. Xiong. Beneficial reasoning behaviors in agentic search and effective training methods to obtain them. In M. Liakata, V. P. Moreira, J. Zhang, and D. Jurgens, editors, *Findings of the Association for Computational Linguistics: ACL 2026*, pages 28080–28097, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1400. URL https://aclanthology.org/2026.findings-acl.1400/.

A. Karpathy. autoresearch: AI agents running research on single-GPU nanochat training automatically. GitHub repository, 2026. URL https://github.com/karpathy/autoresearch. Accessed: 2026-10-06.

P. Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-3250/.

I. Kulikov, C. Whitehouse, T. Wu, Y. Nie, S. Saha, E. Helenowski, W. Yuan, O. Golovneva, J. Lanchantin, Y. Bachrach, J. Foerster, X. Li, H. Fang, S. Sukhbaatar, and J. Weston. Autodata: An agentic data scientist to create high quality synthetic data, 2026. URL https://arxiv.org/abs/2606.25996. arXiv:2606.25996.

Q. Li, Y. Zhang, X. Yang, X. Yang, Z. Wang, B. Xian, W. Liu, and J. Bian. FT-Dojo: Towards autonomous LLM fine-tuning with language agents. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 72523–72548. PMLR, 2026. URL https://proceedings.mlr.press/v306/li26jl.html.

J. Ma, L. Aster, H. Zhang, S. Song, W. Yuan, Z. Zhang, K. Jia, and Z. Sui. Traverse: Learning when to remember, reset, and redirect for long-horizon web search. *arXiv preprint arXiv:2609.37082*, 2026. URL https://arxiv.org/abs/2609.37082.

F. Meng, G. Chen, J. Zhao, S. Sun, Y. Lin, W. X. Zhao, R. Song, J.-R. Wen, and K. Jia. CalibForge: Adversarial solver calibration for scaling learnable terminal tasks. *arXiv preprint arXiv:2608.06352*, 2026. URL https://arxiv.org/abs/2608.06352.

M. Merrill, A. Shaw, N. Carlini, B. Li, H. Raj, I. Bercovich, L. Shi, J. Shin, T. Walshe, E. K. Buchanan, J. Shen, G. Ye, H. Lin, J. Poulos, M. Wang, M. Nezhurina, D. Lu, O. Menis Mastromichalakis, Z. Xu, Z. Chen, Y. Liu, R. Zhang, L. L. Chen, A. Kashyap, J.-L. Uslu, J. Li, J. Wu, M. Yan, S. Bian, V. Sharma, K. Sun, S. Dillmann, A. Anand, A. Lanpouthakoun, B. Koopah, C. Hu, E. Guha, G. Dreiman, J. Zhu, K. Krauth, L. Zhong, N. Muennighoff, R. Amanfu, S. Tan, S. Pimpalgaonkar, T. Aggarwal, X. Lin, X. Lan, X. Zhao, Y. Liang, Y. Wang, Z. R. Wang, C. Zhou, D. Heineman, H. Liu, H. Trivedi, J. Yang, J. Lin, M. Shetty, M. Yang, N. Omi, N. Raoof, S. Li, T. Y. Zhuo, W. Lin, Y. Dai, Y. Wang, W. Chai, S. Zhou, D. Wahdany, Z. She, J. Hu, Z. Dong, Y. Zhu, S. Cui, A. Saiyed, A. Kolbeinsson, C. Rytting, R. Marten, Y. Wang, J. Jitsev, A. Dimakis, A. Konwinski, and L. Schmidt. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 40903–40986, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/444a3737adaee10d86ad2ef5f74468e6-Paper-Conference.pdf.

OpenAI. GPT-6 Astra: A new generation of intelligence, 2026. URL https://openai.com/index/gpt-6-astra/.

Qwen Team. Qwen3.5: Towards native multimodal agents, Feb. 2026. URL https://qwen.ai/blog?id=qwen3.5.

B. Rank, H. Bhatnagar, A. Prabhu, S. Eisenberg, K. Nguyen, M. Bethge, and M. Andriushchenko. PostTrainBench: Can LLM agents automate LLM post-training? In T. Zhang, M. Dudik, M. Jaggi, A. Agarwal, S. Li, D. Schuurmans, J. Zhu, F. Berkenkamp, H. Dong, and A. Bietti, editors, *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 103808–103823. PMLR, 06–11 Jul 2026. URL https://proceedings.mlr.press/v306/rank26a.html.

Q. Shi, Y. Dou, J. Zhu, Y. Tao, L. Jin, Z. Kang, Y. Zhou, and D. Weng. SKILL-KD: Contrastive skill distillation for LLM agents, 2026. URL https://arxiv.org/abs/2607.28048.

S. Sun, G. Chen, F. Meng, J. Deng, H. Song, J. Jiang, W. X. Zhao, H. Xu, and J.-R. Wen. Agent-Editing World Model: Rethinking world modeling for LLM agents. *arXiv preprint arXiv:2609.28416*, 2026. URL https://arxiv.org/abs/2609.28416.

A. Ta, J. Zhu, and S. Shayandeh. PROOF-Gen: From optimized data to better distillation, 2026. URL https://arxiv.org/abs/2608.23911. arXiv:2608.23911.

J. Wei, Z. Sun, S. Papay, S. McKinney, J. Han, I. Fulford, H. W. Chung, A. T. Passos, W. Fedus, and A. Glaese. BrowseComp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025. URL https://arxiv.org/abs/2504.12516.

W. Wei. Autonomous LLM post-training with Tunix on TPUs. Google Developers Blog, Sept. 2026. URL https://developers.googleblog.com/autonomous-llm-post-training-with-tunix-on-tpus/. Published September 11, 2026; accessed October 6, 2026.

P. Xia, J. Chen, H. Wang, J. Liu, K. Zeng, Y. Wang, S. Han, Y. Zhou, X. Zhao, H. Chen, Z. Zheng, C. Xie, and H. Yao. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning, 2026. URL https://arxiv.org/abs/2602.08234.

S. Yuan, Z. Chen, Z. Xi, J. Ye, Z. Du, and J. Chen. Agent-R: Training language model agents to reflect via iterative self-training, 2025. URL https://arxiv.org/abs/2501.11425.

A. Zeng, M. Liu, R. Lu, B. Wang, X. Liu, Y. Dong, and J. Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

T. Zhang and Z. Qi. Skill-to-LoRA: From using skills to learning behaviors for token-efficient LLM agents, 2026. URL https://arxiv.org/abs/2606.16769.

J. Zhao, G. Chen, F. Meng, W. X. Zhao, R. Song, J.-R. Wen, and K. Jia. DeNovoSWE: Scaling long-horizon environments for generating entire repositories from scratch. *arXiv preprint arXiv:2606.10728*, 2026. URL https://arxiv.org/abs/2606.10728.

L. Zheng, L. Yin, Z. Xie, C. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez, C. Barrett, and Y. Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pages 62557–62583, 2024. doi: 10.52202/079017-2000.

A Experimental Details

A.1 Tasks and Evaluation Harnesses

Teacher rollouts and evaluation use an agent execution framework built on AweAgent (AweAI Team, 2026). Table A1 summarizes the training sources, evaluation benchmarks, and domain-specific harness references. Each domain uses a fixed pool of 1,000 randomly sampled training tasks shared across methods. For BrowseComp, we evaluate on the Lite-185 subset released by Traverse (Ma et al., 2026). For SWE, we use the harness of Chen et al. (2026b) with search tools disabled.

Table A1. Training sources, evaluation benchmarks, and harness references. Task counts refer to the evaluation sets.

Domain	Training source	Evaluation benchmark	Tasks	Harness reference
Terminal	CalibForge (Meng et al., 2026)	Terminal-Bench 2.1 (Merrill et al., 2026)	89	Meng et al. (2026)
Search	FORT-Searcher (Deng et al., 2026)	BrowseComp-Lite (Wei et al., 2025; Ma et al., 2026)	185	Sun et al. (2026)
SWE	DeNovoSWE (Zhao et al., 2026)	Doc2Repo (Chen et al., 2026b)	50	Chen et al. (2026b)

A.2 Teacher Rollout Configuration

We use DeepSeek-V4-Pro-Preview (DeepSeek-AI, 2026) for teacher rollouts across all domains, with the settings listed in Table A2. Skill instructions, when applied, are appended to the base system prompt, and the teacher's reasoning content is retained in the interaction history.

Table A2. Teacher rollout settings across domains. Max completion tokens specifies the output budget for a single model response.

Parameter	Terminal	Search	SWE
Reasoning effort	`high`	`high`	`high`
Temperature	1.0	1.0	1.0
Top-p	1.0	1.0	1.0
Max completion tokens	16,384	8,192	16,384
Maximum execution steps	500	300	400
Per-task time budget	1 hour	No limit	3 hours

A.3 Supervised Fine-Tuning

Table A3 summarizes the common SFT recipe used across methods, model sizes, and domains. Each training run starts from the corresponding original target-model checkpoint, including successive outer-loop rounds of AutoMaster.

We compute the autoregressive cross-entropy loss over assistant outputs, including reasoning, responses, and tool calls, and normalize it over supervised tokens. System, user, and tool messages serve as context and are excluded from the loss. By default, we retain each trajectory's original system prompt, including any skill instructions used during generation. In the **Remove** ablation (Section 5.2), we omit only the skill instructions, leaving the remaining input context and supervised outputs unchanged.

Table A3. Shared SFT configuration across methods, target-model sizes, and domains.

Parameter	Value
Training method	Full-parameter SFT
Training trajectories	512
Epochs	3
Global batch size	32
Maximum sequence length	131,072 tokens
Optimizer	Megatron Adam
Learning rate	10^{-5}
Minimum learning rate	10^{-6}
Learning-rate schedule	Cosine decay
Warmup ratio	0.05
Adam β_1, β_2	0.9, 0.999
Weight decay	0
Gradient clipping	1.0
Training precision	`bfloat16`

A.4 Model Evaluation

For each SFT run, we evaluate the checkpoint saved at the end of the third epoch without generated skills. We follow each benchmark's official evaluation protocol using the domain-specific harnesses listed in Table A1. Inference and execution settings are identical across methods within each domain. These benchmark evaluations provide scores and execution trajectories for subsequent experimentation.

Across all domains, we use a temperature of 1.0, a top-p of 0.95, a maximum of 65,500 tokens per model response, and a limit of 600 execution steps per task. The per-task agent time limit is three hours for Terminal-Bench 2.1 and four hours for Doc2Repo. No overall task time limit is imposed for BrowseComp.

At the end of autonomous experimentation, AutoMaster selects a final checkpoint based on accumulated evaluation feedback and records its choice in a final report. We then evaluate the selected checkpoint in three new runs to obtain the reported results. For all methods, we report mean $\pm$ SD over three evaluation runs per checkpoint, using task accuracy for Terminal-Bench 2.1 and BrowseComp and mean test pass rate across tasks for Doc2Repo. The SD summarizes evaluation variability for a fixed checkpoint.

A.5 Detailed Ablation Results

Table A4 reports the complete numerical results for the component ablations and Task-Matched SFT comparison in Section 4.3.

We construct Task-Matched SFT from the initial unguided teacher rollouts corresponding to AutoMaster's final training tasks. Task IDs and per-task sample counts are preserved. We retain both successful and unsuccessful unguided trajectories to preserve the matched task composition. Each trajectory retains its original unguided prompt, without generated skill instructions.

Table A5 characterizes the paired teacher demonstrations in these training corpora. Each row contains 512 pairs for the corresponding target model. For Terminal and Search, we compare

Table A4. Component ablations and task-matched SFT results. Values are mean ± SD over three evaluation runs per checkpoint, in percent. We report accuracy for Terminal-Bench 2.1 and BrowseComp, and mean test pass rate for Doc2Repo. Bold marks the highest mean in each column.

	Terminal-Bench 2.1		BrowseComp		Doc2Repo	
Method	**4B**	**9B**	**4B**	**9B**	**4B**	**9B**
AutoMaster	**28.09** ± 2.25	**35.58** ± 1.30	**46.13** ± 2.98	**51.53** ± 1.25	**33.72** ± 3.58	**40.79** ± 3.21
w/o Skill Guidance	17.60 ± 3.43	28.46 ± 1.72	45.77 ± 3.43	49.19 ± 1.43	32.03 ± 1.24	36.97 ± 1.88
w/o Data Selection	20.22 ± 2.25	34.08 ± 3.43	43.78 ± 2.48	50.27 ± 0.94	31.43 ± 2.53	38.29 ± 3.58
Task-Matched SFT	17.98 ± 1.95	27.72 ± 0.65	44.32 ± 1.87	50.45 ± 1.25	32.37 ± 2.01	36.93 ± 3.53
Vanilla SFT	22.47 ± 1.12	26.59 ± 1.30	43.24 ± 0.54	46.85 ± 2.98	32.30 ± 4.00	36.05 ± 3.89

Table A5. Paired outcomes of teacher training demonstrations. N denotes the number of paired training examples. The target-model column identifies the model trained on the corresponding corpus. **(a)** Transitions are from the unguided teacher rollout to the AutoMaster demonstration. **(b)** SWE categories compare task scores: improved, unchanged, and decreased indicate higher, equal, and lower scores under AutoMaster, respectively.

(a) Binary task outcomes in Terminal and Search

Domain	Target model	N	Failure → success	Both succeed	Success → failure	Both fail
Terminal	4B	512	106	406	0	0
Terminal	9B	512	100	412	0	0
Search	4B	512	45	467	0	0
Search	9B	512	64	448	0	0

(b) Task-score changes in SWE

Target model	N	Improved	Unchanged	Decreased
4B	512	43	433	36
9B	512	24	483	5

binary task outcomes. Across these four corpora, 8.79–20.70% of the pairs contain a successful AutoMaster demonstration and an unsuccessful unguided rollout. The remaining pairs succeed in both conditions. For SWE, we report whether the task score increases, remains unchanged, or decreases under AutoMaster, allowing partial improvements and regressions to be represented. These statistics describe the final selected training corpora and complement the downstream student results in Table A4.

A.6 Paired Analysis of Performance Gains

We quantify uncertainty in the gains over Vanilla SFT using paired bootstrap resampling (Koehn, 2004; Dror et al., 2018). This estimates uncertainty in the performance difference while preserving the correspondence between the two methods' scores on each evaluation task. For each task, we first average its scores over the three evaluation runs for each method. We then resample task IDs with replacement, retaining both methods' scores for each sampled task, and recompute the mean performance difference. We use 10,000 resamples with random seed 0 to obtain pointwise 95% percentile bootstrap confidence intervals. For Doc2Repo, resampling is performed over whole

repositories, each corresponding to one task in the 50-repository evaluation set. All tasks and their recorded scores are retained. The intervals summarize variation across tasks or repositories for the reported checkpoints and recorded evaluations.

Table A6. Paired performance gains over Vanilla SFT. Δ denotes the mean gain of AutoMaster over Vanilla SFT. Gains and pointwise 95% bootstrap confidence intervals are reported in percentage points and computed from unrounded scores. N denotes the number of evaluation tasks, or repositories for Doc2Repo.

Benchmark	Model	N	Δ	95% CI
Terminal-Bench 2.1	4B	89	+5.62	[1.12, 10.49]
	9B	89	+8.99	[2.25, 16.10]
BrowseComp	4B	185	+2.88	[−1.08, 6.85]
	9B	185	+4.68	[0.72, 8.83]
Doc2Repo	4B	50	+1.42	[−1.94, 5.33]
	9B	50	+4.74	[0.65, 8.93]

Table A6 shows positive mean gains across all six benchmark–model settings. For the 4B model, the confidence interval lies entirely above zero on Terminal-Bench 2.1, while the smaller mean gains on BrowseComp and Doc2Repo have intervals that include zero. For the 9B model, mean gains range from 4.68 to 8.99 percentage points, and the lower bounds of all three 95% confidence intervals are positive, providing consistent paired evidence of improvements across domains.

B Skills Generated by AutoMaster

This section presents four skills generated by AutoMaster and used in the teacher trajectories analyzed in Sections 5.1 and 5.2. They illustrate how behavioral objectives are translated into instructions specifying conditions of application, concrete procedures, and domain-specific knowledge.

Skill instructions are reproduced verbatim, with formatting adapted and metadata and enclosing tags omitted. The numerical and grammar instructions are shown in full. `[...]` marks omitted passages in the search and contract instructions.

Evidence–claim consistency. This skill links completion claims to executable numerical checks and requires the specified acceptance criterion to be preserved during repair (Figure 5(a)).

Skill B.1 `execute-numeric-decision` Complete instruction

Execute the numerical decision

When correctness depends on numerical accuracy or convergence, put the relevant acceptance condition into the check you run. Adapt this finite-scalar example to the task's actual error convention:

```
error = abs(actual - reference)
passed = math.isfinite(error) and error <= allowed_error
print(actual, reference, error, allowed_error, "PASS" if passed else "FAIL")
assert passed, "numerical requirement remains unresolved"
```

Define the reference and bound before running this block. Use the specified absolute, relative, or norm error and strictness; the example does not authorize substituting absolute error for a relative requirement. Keep prescribed input scale and iteration budget. If no bound is specified, justify a diagnostic bound and label it as such. An existing test that already makes the decision suffices. Returning a comparison from a test function does not enforce it: its result must reach the assertion or failure branch.

Base the reference on an independent analytic case, trusted implementation, or mathematical identity. Preserve exact comparisons for discrete results. Treat zero references and expected NaN/Inf according to the interface; apply finite-value checks only where finite output is required.

If the check fails, retain that case and condition while investigating. Save a random failing input or its reproducible seed. A common scale error suggests units, normalization or weights; an error near the bound suggests stopping criteria or error estimates. Inspect the responsible computation, make a justified correction, and rerun the same condition. Correct a mistaken reference only from independent evidence. Do not widen a required tolerance or replace a failing input with a passing one.

Connect completion to the observed decision, or state the unresolved requirement. More suites and repeated unchanged runs are unnecessary.

Evidence-guided search. When an identification search stalls, this skill ties the next search action to the unsupported assumption or unresolved relation (Figure 5(b)).

Skill B.2 `recover-search-anchor` Instruction excerpts

Recover a stalled search anchor

Use this when searches keep returning the same weak matches, or a promising candidate fits one clue but conflicts with another. The aim is to obtain evidence that can change the candidate, not to accumulate more mentions of it. If a search is yielding new relevant evidence, continue it.

`[...]` Choose the next action from the observed problem:

- **The candidate or scope is still a guess.** Drop that guessed restriction from the next discovery query. Search a different distinctive relation from the question that can identify an entity independently: an unusual office and succession, an event and venue, an original name and later renaming, or a study's measurable design. Use the clue's concrete terms, not every clause of the puzzle. If a new candidate appears, test the relation that the old candidate could not explain.

`[...]` At a consequential change, briefly connect the observation to the next action: what the results failed to establish, which assumption you are relaxing, and what the next query can distinguish. The actual query should implement that change; saying "try another approach" while retaining the same unsupported restriction is not a pivot. `[...]`

Diagnosing verification failures. This skill pairs a check derived from the specification with a contrasting input that distinguishes plausible implementations. It requires diagnosing failures before changing the implementation or the expected result (Figure 5(c)).

Skill B.3 `contract-checkpoint` Instruction excerpts

Keep a runnable public contract through construction

Apply this when parsing, conversion, serialization, configuration, or stateful APIs compose several operations into a visible result. For a small independent function, use a direct input/output check; do not invent an architecture or a large test suite.

`[...]` **Diagnose and preserve the failing example**

Read the actual failing assertion and its input before changing code. Reduce that case and inspect the type and value at the first boundary that could explain the mismatch. Check whether the error belongs to parsing, conversion, state, or output only as the observations require. Fix the shared cause and rerun the original public assertion plus its contrasting case.

If your expected value is wrong, establish why from the specification or a small independent calculation before changing it. When an example contradicts a general rule, record the conflict, test a second discriminating input, and choose a consistent interpretation; do not invent a new global algorithm merely to fit that example or claim the contradiction is resolved by a passing self-test.

Keep the same runnable script across repairs. Invoke it again after changes to shared helpers, imports,

constructors, or state transitions. Add a small regression for a newly found mechanism instead of repeatedly rewriting a broad inline suite. Run it after the last relevant edit, alongside the task's required verification, and distinguish exercised behavior from paths that remain unverified.

Explicit rules and their scope. This skill supplies a grammar together with compatibility constraints and discriminating tests. The teacher explicitly consults this grammar when deriving expression grouping (Figure 7(b)).

Skill B.4 `expression-grammar` Complete instruction

When implementing arithmetic with unary signs and powers, derive the grouping rules from the specification before changing parser calls. For conventional mathematical binding, recursive descent can use:

```
sum     := product (("+" | "-") product)*
product := prefix (("*" | "/" | "%") prefix)*
prefix  := sign prefix | power
power   := atom (power_operator prefix)?
```

Here the power base excludes an ungrouped sign, but its right operand admits signs and further powers. In a Pratt or precedence-climbing parser, keep prefix parsing available at every operand entry; use infix binding thresholds for associativity. Adapt the existing parser, including only supported operators. For binary compatibility, preserve the reference's grouping instead of imposing mathematical conventions.

Check the actual evaluator with a signed base versus a grouped signed base, a signed expression in the right operand, and repeated powers. Derive expected values first; include a positive exponent formed with repeated signs when integer powers are required. Build-only repairs need no grammar change.

These instructions guide teacher execution and, when retained during SFT, form part of the input accompanying the supervised reasoning and actions. Their role as training context is examined in Section 5.2.