ELICITATION OR EXPANSION? DEMYSTIFYING GAINS FROM AGENT HARNESS OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent studies in harness optimization have shown substantial performance gains in language model agents without updating model weights. A well-designed harness more effectively leverages the underlying model's capabilities by shaping context, tool use, and execution workflows. However, naïve success rates alone do not fully characterize whether optimized harness yields more reliable success on tasks already within reach (*elicitation*) or a broader range of solvable tasks (*expansion*). This distinction is crucial for recursive self-improvement and evolutionary search, as expansion may provide a broader range of successful experiences to push the capability boundaries. Despite the growing significance, this distinction remains underexplored in harness optimization. We conduct a systematic analysis to study the effect of gains from harness optimization. Building on $\mathrm{pass@}k$ analyses in the reinforcement learning (RL) literature that compare models before and after RL training, we compare optimized harnesses with their pre-optimization counterparts. Specifically, we assess changes in reachable task coverage, compare harness optimization with model scaling and increased inference-time compute, and examine how gains vary with task difficulty. Across static prediction and dynamic tasks, we find that $\mathrm{pass@1}$ advantages of harness over stronger interventions can reverse at larger attempt budgets and aggregate gains conceal shortfalls on hard tasks, underscoring the need for a fine-grained diagnosis.

1 INTRODUCTION

The ability of a language model agent is influenced by the design of its *harness*, which governs prompting, context management, tool use, and execution logic (Young, 2025; OpenAI, 2026). Refining these components can substantially improve agent performance without updating model parameters, motivating growing interest in harness optimization. By broadening the scope of harness optimization beyond prompts (Agrawal et al., 2026) and context (Zhang et al., 2026d) to agent code and execution workflows (Hu et al., 2025; Lee et al., 2026b), recent methods have achieved huge performance gains across diverse tasks. However, these aggregate benchmark gains alone do not reveal which aspects of the agent system have improved. For instance, comparable gains arise either from simply reformatting otherwise correct answers to satisfy the evaluation protocol or from enabling the agent to more effectively leverage its environment and external feedback to deal with previously unsolved tasks. This distinction is especially important for approaches that leverage successful experience for subsequent improvement, including recursive self-improvement (Zhang et al., 2026b; Lee et al., 2026a). In these settings, even rare successes on previously unreachable tasks can provide a foundation for further search or learning.

Recent studies have begun to examine the nature of harness gains by relating prompt edits to task-dependent performance changes (Gong & Wen, 2026), attributing improvements to recoverable execution defects (Yang et al., 2026), and comparing harness evolution with simple retries (Wang et al., 2026). However, these analyses leave open how task coverage under the original and optimized harnesses differs as the attempt budget increases. Whether the task coverage of optimized harness persists, disappears, or reverses matters for assessing how far the optimized harness can extend an agent's practical problem-solving reach without further model training. A closely related debate in reinforcement learning (RL) provides a useful diagnostic perspective. Comparisons between RL-trained models and their pre-RL counterparts show that improvements in $\mathrm{pass@1}$ can diminish or reverse at larger sampling budgets, suggesting improved sampling efficiency rather than broader

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

task coverage (Yue et al., 2025). Other studies report broader coverage under the valid reasoning evaluation (Wen et al., 2026) or additional tool use (Zhai et al., 2026), while controlled experiments show that coverage gains depend on pre-training exposure and training-task difficulty (Zhang et al., 2026a). Together, these studies lay the ground for examining how improvements behave across sampling budgets and task conditions, rather than inferring their nature from a single aggregate score.

In this work, we investigate whether harness optimization makes success more frequent on tasks already solved by the baseline (*elicitation*) or broadens observed task coverage (*expansion*) through the lens of pass@k (Chen et al., 2021). We complement this analysis by comparing optimized harnesses with base harness using larger models from the same family or higher reasoning effort. These baselines provide empirical reference for assessing how far harness optimization alone enhances an agent's capability. Finally, we examine whether gains are concentrated on easier tasks or extend to harder ones. Figure 1 depicts why these multi-axis analyses matter. Here, overall pass@1 gain does not fully characterize harness gains; the optimized harness leaves the agent less capable of solving harder problems than under the original harness. Our analysis thus unveils concealed coverage loss on harder tasks.[1]

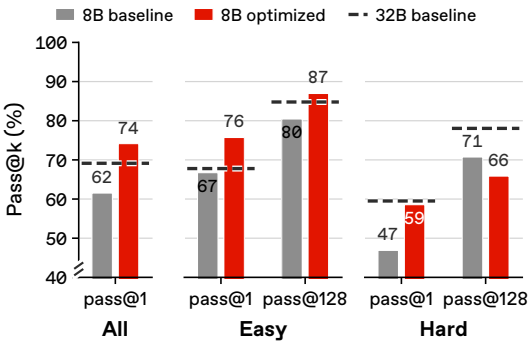


Figure 1: On HotpotQA, the optimized harness enables Qwen3-8B to outperform even 32B under base harness in overall pass@1. On easy questions, this advantage holds at both pass@1 and pass@128. On hard questions, however, the optimized harness outperforms the original 8B baseline at pass@1 but falls below it at pass@128.

The heterogeneous effects recur throughout our analyses on various benchmarks. First, advantages over stronger references can reverse as the attempt budget increases: on LawBench (Fei et al., 2024), the optimized GPT-OSS-20B harness exceeds the GPT-OSS-120B baseline by 16.05 percentage points (pp) at pass@1, but trails it by 4.00 pp at pass@64. Second, on HotpotQA (Yang et al., 2018) and AppWorld (Trivedi et al., 2024), the selected harnesses approach or exceed stronger references in aggregate but do not surpass them on hard subsets. Third, the transferred text-classification harness falls below its baseline at pass@64 on all out-of-distribution datasets, despite small pass@1 gains on two. This heterogeneity makes systematic analysis necessary: understanding harness gains requires distinguishing reliability from coverage and examining how both change across attempt budgets, task groups, reference systems.

2 Related Work

2.1 Harness Optimization

Harness optimization adapts agent's prompts, context, and execution logic through repeated evaluation and revision. TextGrad (Yuksekgonul et al., 2025) and GEPA (Agrawal et al., 2026) refine prompts through natural-language feedback, while ACE (Zhang et al., 2026d) incorporates lessons from execution into an evolving context playbook. Methods that search over agent code (Hu et al., 2025; Zhang et al., 2026b;e) use execution feedback to diagnose failures and guide revisions across harness components (Lin et al., 2026a; Zhang et al., 2026c; Park et al., 2026; Yue et al., 2026). Our study focuses on Meta-Harness (Lee et al., 2026b), which uses a coding-agent proposer selectively inspects prior harness code, evaluation scores, and execution traces to generate new candidates.

2.2 Interpreting Capability Gains

Analyses of reinforcement learning show that higher pass@1 can accompany reduced coverage at large sampling budgets (Yue et al., 2025), while other studies find broader coverage under different training and evaluation conditions (Wen et al., 2026; Zhang et al., 2026a). Recent work further documents smaller gains on harder problems under standard reinforcement learning and shows that

[1]This example uses an optimized harness under relaxed per-attempt resource limits in Appendix C.3

adaptive sampling can improve learning on these problems (Noukhovitch et al., 2026). Studies of repeated sampling (Brown et al., 2024) and agent interaction budgets (Zhai et al., 2026) examine how additional attempts and interactions affect coverage, while test-time scaling analyses show that effective compute allocation depends on task difficulty (Snell et al., 2025). For harness optimization, analyses compare evolution with test-time search (Wang et al., 2026), trace gains to execution repairs (Yang et al., 2026), and examine the repeatability (Zhao et al., 2026) and model dependence (Lin et al., 2026b) of improvements. We analyze optimized harnesses across attempt budgets and difficulty levels, distinguishing coverage gains over the original baseline from advantages that persist against baselines with larger models or more inference compute.

3 Analysis Setup

3.1 Preliminary: Harness Optimization

Let M denote a solver model and H its harness, which controls model inputs, tool interactions, and execution state. On a task x, executing $(\mathcal{M}, H)$ yields a complete trajectory $\tau \sim \pi_{\mathcal{M},H}(\cdot \mid x)$, evaluated by a task-specific scoring function $r(\tau, x)$. Given a task distribution $\mathcal{X}$, harness optimization seeks to maximize the expected task score $J(\mathcal{M}, H)$ by revising H while keeping $\mathcal{M}$ fixed:

$$\max_{H} \; J(\mathcal{M}, H), \qquad J(\mathcal{M}, H) = \mathbb{E}_{x\sim\mathcal{X},\, \tau\sim\pi_{\mathcal{M},H}(\cdot|x)}\big[r(\tau, x)\big].$$

For harness optimization, we use Meta-Harness (Lee et al., 2026b), whose coding agent, called proposer, selectively inspects previous harness code, evaluation scores, and execution traces to diagnose failures and propose revised harnesses. Each candidate is evaluated, and its code and execution feedback are retained to guide subsequent proposals. Our analysis compares the initial harness H_0 with an optimized harness $H^\star$ selected based on validation performance. We estimate their pass@k from repeated executions on a common test set under the same task-specific scorer (Appendix A).

3.2 Motivation: Beyond Mean Accuracy

For a fixed set of tasks and number of attempts (k) per task, pass@1 determines the expected number of successful executions (*i.e.*, mean accuracy), whereas pass@k measures the expected fraction of tasks solved at least once in k complete executions (Chen et al., 2021) (*i.e.*, coverage). A harness can therefore achieve higher mean accuracy but low coverage, by repeatedly solving the same tasks without obtaining successes on additional ones. When generated solutions are used for self-training (Zelikman et al., 2022) or verifier-based selection (Cobbe et al., 2021), coverage is particularly important as it captures the breadth of tasks for which a successful example or solution is available.

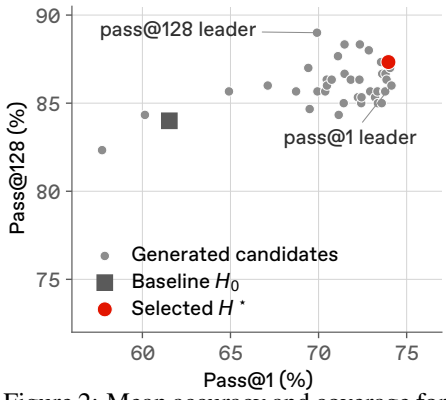


Figure 2: Mean accuracy and coverage for harnesses from one optimization run.

Figure 2 compares pass@1 and pass@128 for Qwen3-8B harness candidates from one optimization run, each evaluated on the same 300 HotpotQA tasks with $k = 128$ executions per task. Although the two metrics broadly track one another, the candidate with the highest estimated pass@1 differs from the one with the highest observed pass@128. Repeated evaluation can improve the precision of pass@1, but does not make it a substitute for measuring task coverage at larger budgets.

4 Evaluation Setup

We refer to $(\mathcal{M}, H_0)$ and $(\mathcal{M}, H^\star)$ as the baseline and optimized system, respectively. The reference $(\mathcal{M}_{\text{ref}}, H_0)$ uses the base harness with a larger solver model or the same solver model at higher reasoning effort. We compare each system on a large k attempts. More details are in Appendix B.

4.1 Static Tasks

Text classification. Following Lee et al. (2026b), we jointly optimize a single harness across three static prediction datasets: Symptom2Disease (Gretel AI, 2023) for diagnosis from symptom descriptions, LawBench (Fei et al., 2024) for charge-set prediction, and USPTO (Schneider et al., 2016) for reactant-SMILES set prediction. In these tasks, the harness constructs reference context to support prediction. We use a few-shot harness as H_0, since directly supplying labeled examples provides a simple way to construct such context. The optimized harness $H^\star$ prepares memory from the same pool and retrieves relevant information to construct context for each test input. During repeated evaluation, H_0 resamples demonstrations and $H^\star$ rebuilds its memory. For out-of-distribution (OOD) transfer, we reuse $H^\star$ obtained from in-distribution (ID) datasets to FiNER (Loukas et al., 2022), GoEmotions (Demszky et al., 2020), and Banking77 (Casanueva et al., 2020). We use GPT-OSS-20B (Agarwal et al., 2025) as $\mathcal{M}$ and GPT-OSS-120B as $\mathcal{M}_{\text{ref}}$ for comparison.

4.2 Dynamic Tasks

Multi-hop QA. On HotpotQA (Yang et al., 2018), the harness coordinates evidence retrieval and answer generation. The base harness H_0 follows multi-hop program of GEPA (Agrawal et al., 2026): it retrieves passages, generates a follow-up query from the retrieved evidence, and answers using summaries accumulated across the two retrieval rounds. We obtain $H^\star$ by optimizing the harness code governing query construction, evidence processing, and answer generation, while keeping the BM25 retriever fixed. Evaluation uses 300 test questions with their original difficulty labels. We use Qwen3-8B (Yang et al., 2025) as $\mathcal{M}$ and Qwen3-32B as $\mathcal{M}_{\text{ref}}$ for comparison.

Agentic task. On AppWorld (Trivedi et al., 2024), the harness coordinates multi-turn interactions with simulated applications, where actions change the environment and execution feedback informs subsequent decisions. The baseline harness H_0 follows the benchmark's simplified ReAct implementation (Yao et al., 2023), alternating Python code generation and execution. Evaluation uses the 168 tasks in `test_normal` with their original difficulty labels. We use GPT-OSS-120B for both $\mathcal{M}$ and $\mathcal{M}_{\text{ref}}$. The solver's reasoning effort is fixed at medium during harness optimization and for the base and optimized systems; the reference system uses high reasoning effort.

5 Experimental Results

5.1 Do Optimized Harnesses Provide Gains Over Stronger References?

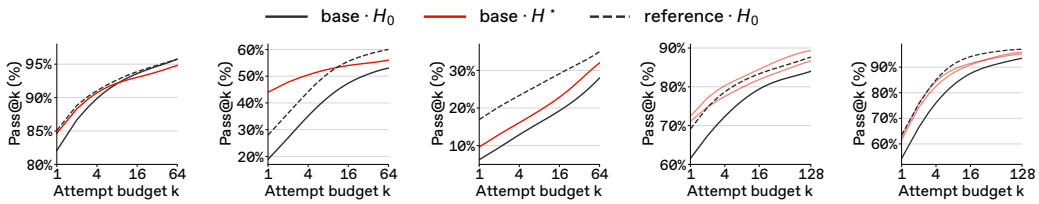
Observation

The optimized GPT-OSS-20B harness falls below the GPT-OSS-120B reference at pass@64 on all three static benchmarks, despite its varying relative performance at pass@1. As shown in Figure 3 (a-c), pass@1 of optimized harness nearly matches the 120B reference on Symptom2Disease, exceeds it on LawBench, and falls below it on USPTO. LawBench provides a particularly clear example: the 16.05 pp advantage at pass@1 reverses at $k = 11$, and the optimized harness reaches 56% at pass@64, compared with 60% for the reference. Thus, even a substantial single-attempt advantage does not necessarily translate into broader task coverage under repeated attempts.

On the dynamic benchmarks, the selected harnesses attain overall pass@1 and pass@128 close to those of their stronger references. On HotpotQA, both optimized Qwen3-8B harnesses exceed the 32B reference at pass@1, but only one remains above it at pass@128. On AppWorld, both optimized medium-effort harnesses fall below the high-effort reference at these endpoints, although one briefly exceeds it at $k = 2$. Comparisons within a single optimization run in Fig 8 (a) further illustrate that advantages in single-attempt success and finite-budget coverage do not coincide. Additional results on dynamic tasks are in Appendix C.2

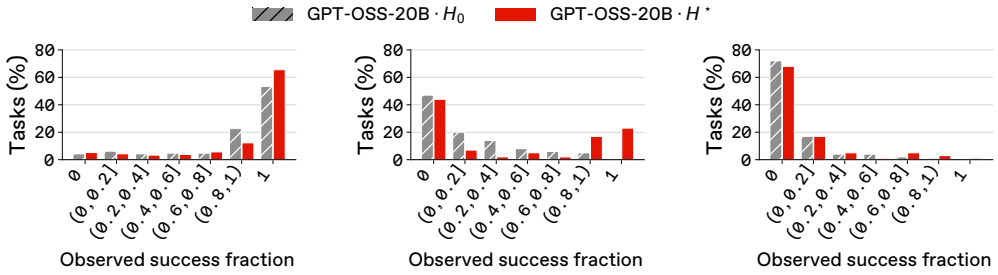
Discussion

Baseline success patterns help explain the heterogeneous gains from harness optimization. To interpret the text-classification results, we compare the task-level success distributions of the



(a) Symptom2Disease (b) LawBench (c) USPTO (d) HotpotQA (e) AppWorld

Figure 3: pass@k curves for the base solver with baseline (H_0) and selected optimized ($H^\star$) harnesses, compared with a stronger reference using H_0. "base" and "reference" denote GPT-OSS-20B and GPT-OSS-120B for text classification (a-c), Qwen3-8B and Qwen3-32B for HotpotQA (d), and GPT-OSS-120B at medium and high reasoning effort for AppWorld (e), respectively. Each red curve represents a harness selected from one optimization run.



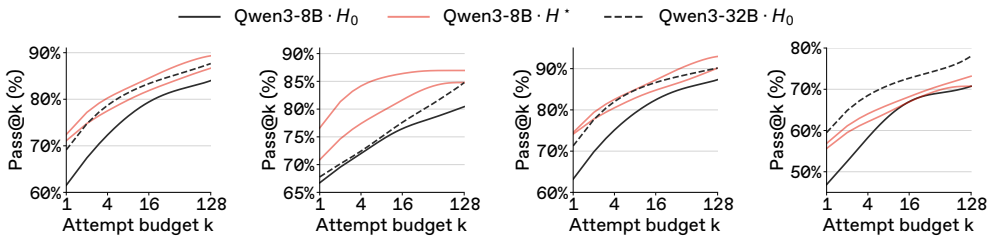
(a) Symptom2Disease (b) LawBench (c) USPTO

Figure 4: Distributions of task-level observed success fractions for GPT-OSS-20B with baseline (H_0) and optimized ($H^\star$) harnesses. Bars show the percentage of tasks within each dataset falling in each bin, computed separately for each harness.

Discussion (continued)

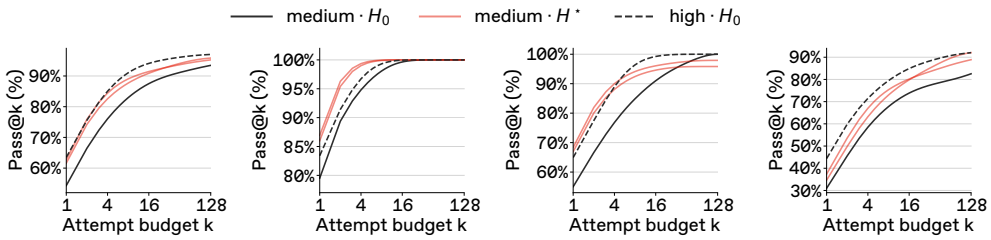
baseline and optimized harnesses with GPT-OSS-20B in Figure 4. For each task x, we compute its observed success fraction c/n separately under each harness, where $c = \sum_{j=1}^{n} r(\tau_j, x)$. We then construct a separate distribution for each harness by grouping tasks according to these fractions, distinguishing no observed success, intermittent success, and success on every attempt. On Symptom2Disease, frequent baseline success leaves limited pass@1 headroom (Figure 4 (a)). Optimization improves reliable successes but with the slight decrease in pass@64 coverage. On LawBench, optimization primarily makes existing successes more frequent (Figure 4 (b)). Of the 53 tasks solved by the baseline within 64 attempts, the optimized harness retains 49 and loses 4, while newly solving 7 others; the retained tasks account for 21.02 pp of the 25.14 pp pass@1 gain (Appendix C.1). Consequently, the gain over the baseline narrows to 3.00 pp at pass@64 (Figure 3 (b)). On USPTO, observed coverage increases at pass@64, but 68 of 100 tasks still yield no success, leaving substantial headroom unrealized (Figure 4 (c)). These patterns suggest that opportunities for harness improvement depend on both the frequency and breadth of baseline successes. These patterns motivate a possible analogy to the Matthew effect reported by Noukhovitch et al. (2026), where RL gains tend to favor tasks with greater initial model ability. Harness optimization similarly concentrates gains on tasks with existing baseline successes.

Environment interaction provides additional opportunities for harness optimization. On dynamic tasks, optimized harness also shapes actions and hence subsequent observations, creating opportunities to acquire additional information. For instance, in our HotpotQA experiment, the base harness restates the question as its second retrieval query, leaving the "George Litto" abstract unreachable in all 128 rollouts and ultimately reporting that the producer of *Blow Out* is not mentioned." By instead querying the newly resolved bridge entity, the optimized harness reaches the relevant abstract in 125 of 128 rollouts and answers correctly in 122 times, turning previously inaccessible external evidence into usable context. This aligns with prior work showing that structuring action loops and agent interfaces can improve performance without up-



(a) All tasks (b) Easy (c) Medium (d) Hard

Figure 5: HotpotQA pass@k curves for the Qwen3-8B baseline (H_0) and selected optimized harnesses ($H^\star$), with a Qwen3-32B baseline. Each red curve represents one selection from two optimization runs. Neither selected harness exceeds the Qwen3-32B baseline on the hard subset.



(a) All tasks (b) Easy (c) Medium (d) Hard

Figure 6: AppWorld pass@k curves for the GPT-OSS-120B baseline (H_0) and selected optimized harnesses ($H^\star$) at medium reasoning effort, with a high-effort baseline. Each red curve represents one selection from two optimization runs. Neither selected harness exceeds the high-effort baseline on the hard subset.

Discussion (continued)

dating model weights (Yao et al., 2023; Yang et al., 2024), while complementary evidence from tool-use RL suggests that improved sequential information gathering can broaden observed task coverage (Zhai et al., 2026). Such opportunities are absent or more limited in the static tasks in Figures 3 (a)-(c), where harness optimization mainly reshapes how the solver uses a fixed input. This contrast may help explain why, in the dynamic tasks in Figures 3 (d)-(e), the selected optimized harnesses produce pass@k curves that closely track those of the stronger references.

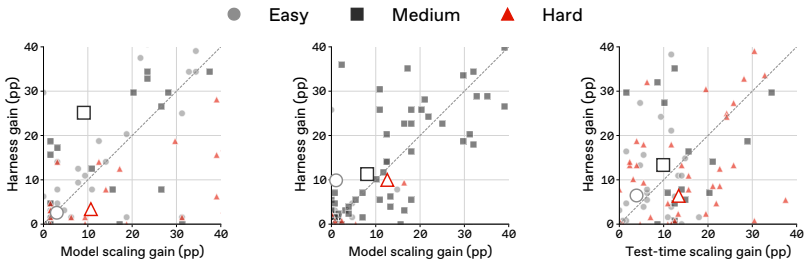
Takeaway

Harness optimization can yield substantial pass@1 gains without matching stronger references at larger k, especially on static tasks. On dynamic tasks, improved environment interaction can expand accessible evidence, helping optimized harnesses more closely track stronger references.

5.2 Do Optimized Harness Gains Differ by Task Difficulty?

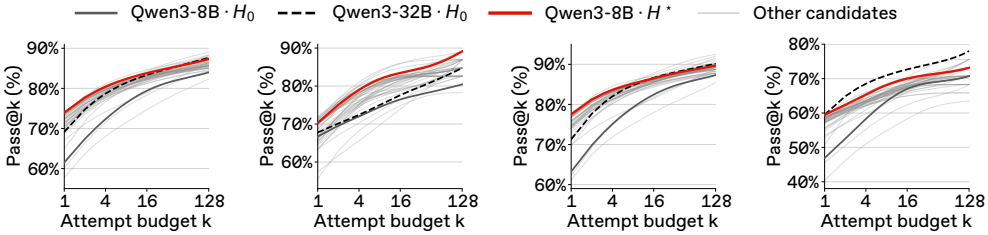
Observation

Aggregate curves can conceal shortfalls on hard tasks. As shown in Figures 5 (a) and 6 (a), the selected harnesses approach or exceed their stronger references when evaluated across all tasks. However, as shown in Figures 5 (d) and 6 (d), these pooled curves obscure a consistent pattern on the hard subsets: none of the selected harnesses exceeds its stronger reference at any evaluated attempt budget. Stratifying the curves by difficulty labels provided by benchmark makes this contrast explicit. On easy HotpotQA questions, the selected harnesses with 8B model match or exceed the stronger 32B reference throughout the evaluated budgets (Figure 5 (b)); on easy AppWorld tasks, all systems eventually reach complete observed coverage (Figure 6 (b)). On hard HotpotQA questions, the selected harnesses improve pass@1 over the 8B baseline but



(a) Text classification (b) HotpotQA (c) AppWorld

Figure 7: Task-level pass@1 gains (pp) from harness optimization versus model scaling (Text: GPT-OSS-20B to 120B; HotpotQA: Qwen3-8B to 32B) or reasoning effort (AppWorld: medium to high). Small markers denote tasks; outlined symbols show equal-task group means. The diagonal indicates equal gains.



(a) All tasks (b) Easy (c) Medium (d) Hard

Figure 8: HotpotQA pass@k curves for all 40 Qwen3-8B candidates generated in one optimization run, including the selected harness (red). On the hard subset, all candidate curves remain below the Qwen3-32B baseline.

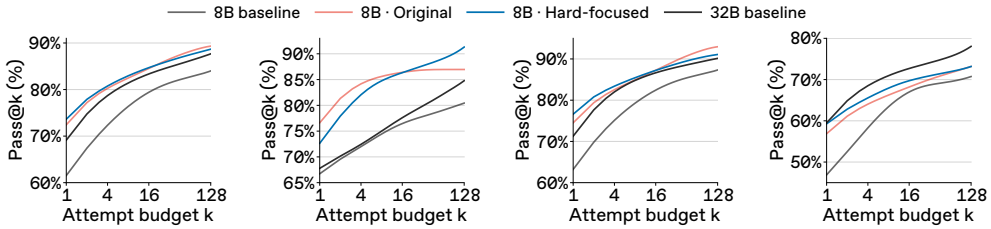
Observation (continued)

yield little additional coverage at pass@128, reaching 70.73-73.17%, compared with 70.73% for the baseline and 78.05% for the 32B reference (Figure 5 (d)). On hard AppWorld tasks, coverage gains over the medium-effort baseline are larger: the selected harnesses reach 88.89-92.06% at pass@128, compared with the baseline's 82.54%, but at best match the high-effort reference's 92.06% (Figure 6 (d)). Thus, viewing the curves only in aggregate would miss a consistent limitation: parity or gains over stronger references across all tasks do not translate into an advantage on hard tasks.

Discussion

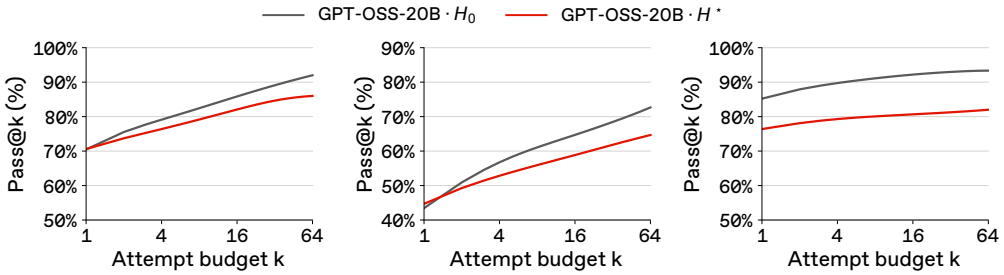
Across all three benchmark panels, scaling yields larger mean pass@1 gains on hard groups than harness optimization. In Figure 7, we compare task-level pass@1 gains from harness optimization and model or effort scaling relative to the same original baseline. Here, the text classification groups are dataset-level proxies informed by the success distributions in Figure 4, whereas HotpotQA and AppWorld use their original difficulty labels. Across all three benchmark panels, the hard group means consistently favor scaling, highlighting that aggregate performance can conceal differences in where the two interventions provide their gains.

The hard task shortfall is shared by all candidates during the evolution. We study all 40 Qwen3-8B candidate harnesses from one optimization run in Figure 8. On easy and medium questions, some candidates match or exceed the 32B reference, with their relative performance varying across attempt budgets. On hard questions, however, every candidate's pass@k curve remains below the 32B reference throughout k. Alternative candidates offer higher hard task coverage than the selected harness, but still fall short of the stronger reference. The selected harness achieves the highest hard task pass@1 among these candidates, yet does not attain the highest pass@128.



(a) All tasks (b) Easy (c) Medium (d) Hard

Figure 9: pass@k curves for Qwen3-8B harnesses optimized with the original and hard-focused task mixtures, alongside the 8B and 32B baselines, overall and by difficulty.



(a) FiNER (b) GoEmotions (c) Banking77

Figure 10: Harness transfer to new text-classification datasets. pass@k curves for GPT-OSS-20B with the baseline (H_0) and in-distribution selected harness ($H^\star$) on out-of-distribution benchmarks; FiNER, GoEmotions, and Banking77.

Discussion (continued)

The hard task shortfall persists even after optimization is redirected. One possible way to address the shared hard task shortfall is to give hard questions greater emphasis during optimization. To this end, we re-optimize the Qwen3-8B harness with a hard focused task mixture and examine whether this targeted intervention broadens coverage on hard test questions. As shown in Figure 9 (d), the resulting harness improves hard task pass@1 from the original optimized harness's 56.88% to 59.28%, nearly matching the 32B reference's 59.51%. At pass@128, however, coverage remains at 73.17%, unchanged from the original optimized harness and 4.88 percentage points below the reference's 78.05%. Thus, even after explicitly redirecting optimization toward hard questions, the gain amounts to more frequent success without an increase in the number of hard questions solved within 128 attempts.

Takeaway

When hard task coverage matters, evaluate whether harness optimization expands coverage on that subset at the intended attempt budget. Neither aggregate gains nor improved hard task pass@1 after hard-focused optimization establish broader coverage. Compare the optimized harness with feasible larger-model or higher-effort alternatives on the same subset.

5.3 Do Optimized Harnesses Transfer to OOD tasks?

Observation

The distinction between pass@1 and pass@64 also matters when the selected harness is transferred to new domains. We transfer the ID-selected GPT-OSS-20B harness used in Section 5.1 to OOD domains: FiNER, GoEmotions, and Banking77, with both base and optimized harnesses given labeled target-domain examples. As shown in Figure 10, the transferred harness's pass@64

(a) Base Harness

```
answer = solver(query, shuffle(labeled_examples))
return parse_final_answer(answer)
```

(b) Selected Harness

```
if average_label_length > 40:
    return use_anchor_prompt(query, context)
elif number_of_labels >= 30 and average_label_length < 40:
    return use_catalog_constrained_prompt(query, context)
else:
    return use_plain_classification_prompt(query, context)
```

Figure 11: Pseudocode derived from the base harness (a) and the subpart of the selected harness (b).

Observation (continued)

falls below the baseline by 6.00, 8.00, and 11.33 pp, respectively. Its small pass@1 advantages on FiNER and GoEmotions already reverse at $k = 2$. [2]

Discussion

To investigate possible mechanisms behind domain dependent harness behavior, we also review the selected harness. Since optimization modifies explicit prompts and executable code while keeping solver parameters fixed, its decision rules are directly inspectable. As illustrated in Figure 11, the base harness directly returns the prediction from few-shot context, whereas the optimized harness selects prompt templates using the number and average length of labels observed in the supplied examples. Its catalog template permits multiple labels separated by semicolons, a convention compatible with LawBench, where a case can involve multiple criminal charges. However, the same routing heuristic selects this template for Banking77, which requires exactly one intent. For the query "I need a new card and I live in the United States.", an OOD prediction was `order_physical_card;country_support`, while the gold label was `country_support`. This explains the curve pattern for OOD tasks.

Takeaway

For OOD deployment, validate the transferred harness against the target-domain baseline. Use code inspection to identify domain dependent assumptions for targeted testing.

6 Conclusion

We provide a diagnostic lens for understanding what harness optimization improves, where its gains persist, and where performance shortfalls remain. By combining pass@k curves with stronger reference systems and analyses across task difficulty, we distinguish more reliable success from broader observed task coverage. These diagnostics help assess whether a harness suits an intended attempt budget and task group.

Limitations and future work. Achieving consistent capability expansion on hard tasks remains an important open challenge for harness optimization, especially when successful executions support recursive self-improvement or evolutionary search. This motivates search and selection methods that reward validated successes on previously uncovered hard tasks and retain candidates with complementary successes. Large-budget pass@k analysis provides a diagnostic basis. Our experimental results are bounded by the evaluated benchmarks, solver families, and optimization runs examined in this study.

[2] Although random guessing can inflate coverage at larger attempt budgets (Yue et al., 2025), both harnesses exceed uniform-guessing coverage on the four single-label datasets at smaller k, as detailed in Appendix C.1.

AI USE STATEMENT

We used generative AI tools to identify literature and assist with manuscript drafting and editing. The authors reviewed AI-assisted material, checking sources and revising the text for accuracy and clarity. We take responsibility for the content of this work, including all AI-assisted text and claims.

ETHICS STATEMENT

We study harness optimization using existing benchmark datasets and simulated application environments. The datasets and solver models may contain biases that task-success metrics do not capture. We therefore emphasize the need for domain-specific validation and appropriate human oversight before deployment.

REPRODUCIBILITY STATEMENT

We provide the rollouts from our text-classification experiments in the supplementary material. Upon acceptance, we plan to publicly release all rollouts from this study, together with the reproduction code. We document datasets and splits in Appendix B.1 and harnesses and optimization settings in Appendix B.2.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd workshop on natural language processing for conversational AI*, pp. 38–45, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. Goemotions: A dataset of fine-grained emotions. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4040–4054, 2020.

Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Alan Huang, Songyang Zhang, Kai Chen, Zhixin Yin, Zongwen Shen, Jidong Ge, and Vincent Ng. LawBench: Benchmarking legal knowledge of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7933–7962. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.452. URL https://aclanthology.org/2024.emnlp-main.452/.

Shuzhi Gong and Hechuan Wen. Why prompt optimization works, and why it sometimes doesn't: A causal-inspired edit-level analysis. *arXiv preprint arXiv:2605.26655*, 2026.

Gretel AI. Symptom to diagnosis dataset. Hugging Face dataset, 2023. URL https://huggingface.co/datasets/gretelai/symptom_to_diagnosis.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025.

Hyunin Lee, Jinglue Xu, Jeffrey Seely, Donghyun Lee, Matei Zaharia, and Yujin Tang. Recursive harness self-improvement. *arXiv preprint arXiv:2607.15524*, 2026a.

Yoonho Lee, Roshen Sanjay Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. In *Third Conference on Language Modeling*, 2026b. URL https://openreview.net/forum?id=tmbOUyFx3R.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026a. URL https://arxiv.org/abs/2604.25850v4.

Minhua Lin, Juncheng Wu, Zijun Wang, Zhan Shi, Yisi Sang, Bing He, Zewen Liu, Tianxin Wei, Zongyu Wu, Zhiwei Zhang, et al. Harness updating is not harness benefit: Disentangling evolution capabilities in self-evolving llm agents. *arXiv preprint arXiv:2605.30621*, 2026b.

Lefteris Loukas, Manos Fergadiotis, Ilias Chalkidis, Eirini Spyropoulou, Prodromos Malakasiotis, Ion Androutsopoulos, and Georgios Paliouras. Finer: Financial numeric entity recognition for xbrl tagging. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4419–4431, 2022.

Michael Noukhovitch, Hamish Ivison, Nathan Lambert, and Aaron Courville. Learning to solve hard problems in rl for llms by never giving up. *arXiv preprint arXiv:2609.13443*, 2026.

OpenAI. Harness engineering: leveraging Codex in an agent-first world. https://openai.com/index/harness-engineering/, February 2026. OpenAI Blog.

Sungho Park, Wonjoong Kim, Rongyuan Tan, Jue Zhang, Wook-Shin Han, Pengfei Gao, Chanyoung Park, Yongqiang Yao, Rao Fu, Elsie Nallipogu, et al. Autosaddler: Automatic harness optimization with durable updates from agent execution traces. *arXiv preprint arXiv:2608.23041*, 2026.

Nadine Schneider, Nikolaus Stiefl, and Gregory A. Landrum. What's what: The (nearly) definitive guide to reaction role assignment. *Journal of Chemical Information and Modeling*, 56(12):2336–2346, 2016. doi: 10.1021/acs.jcim.6b00564. URL https://doi.org/10.1021/acs.jcim.6b00564.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024.

Yike Wang, Huaisheng Zhu, Zhengyu Hu, Yige Yuan, Zhengyu Chen, Shakti Senthil, Hannaneh Hajishirzi, Yulia Tsvetkov, Pradeep Dasigi, and Teng Xiao. Rethinking the evaluation of harness evolution for agents. In *COLM 2026 The 2nd Workshop on Lifelong Agents: Learning, Aligning, and Evolving*, 2026.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jGbRWwIidy.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=mXpq6ut8J3.

Siqi Yang, Qianlan Yang, Yu-Xiong Wang, Saurabh Pujar, and Martin Hirzel. One recipe, many harnesses: What self-evolution encodes across languages and models. *arXiv preprint arXiv:2608.10178*, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Haoran Ye, Xuning He, Vincent Arak, Haonan Dong, and Guojie Song. Meta context engineering via agentic skill evolution. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=P1jHroBS5E.

Justin Young. Effective harnesses for long-running agents. https://anthropic.com/engineering/effective-harnesses-for-long-running-agents, November 2025. Anthropic Engineering Blog.

Ruiqing Yue, Yu Cui, Zhuoyu Sun, Sicheng Pan, Xianhong Xue, Tingyu Li, Ting Li, Wenzhuo Zhu, Yi Chen, Yifei Liu, et al. Ecdysis: Efficient and effective training of runtime harnesses for llm agents. *arXiv preprint arXiv:2609.11677*, 2026.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=4OsgYD7em5.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STar: Bootstrapping reasoning with reasoning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_3ELRdg2sgI.

Zhiyuan Zhai, Wenjing Yan, Xiaodan Shao, and Xin Wang. Does rl expand the capability boundary of llm agents? a pass@(k, t) analysis. *arXiv preprint arXiv:2604.14877*, 2026.

Charlie Zhang, Graham Neubig, and Xiang Yue. On the interplay of pre-training, mid-training, and RL on reasoning language models. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=TBaUfO9znF.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=pUpzQZTvGY.

Luan Zhang, Ruochen Zhou, Dandan Song, Zhengyu Chen, Yuhang Tian, Jun Yang, Huipeng Ma, Chenhao Li, Guangyuan Feng, Xudong Li, et al. Harnesscompass: Guiding automatic harness evolution toward generalizable and effective agent harnesses. *arXiv preprint arXiv:2608.01918*, 2026c.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026d. URL https://openreview.net/forum?id=eC4ygDs02R.

Yifan Zhang, Yutong Dai, Juntao Tan, Luyu Yang, Rishi Mullur, Thai Hoang, Zhiyuan Hu, James Zhu, Phil Mui, Silvio Savarese, et al. Darwinx: Evolving agent harnesses through natural selection. *arXiv preprint arXiv:2608.07545*, 2026e.

Cen Mia Zhao, Haibo Ruan, Wenjie Chen, Pei fen Tu, Usman Abbasi, and Joel Hesch. Beyond prompts: Measuring and optimizing llm tool-agent harnesses, 2026. URL https://arxiv.org/abs/2609.05736.

A Repeated Evaluation and pass@k

We repeat complete executions of each fixed solver–harness pair on the same N test tasks, including all internal model calls and tool interactions. Let n be the number of rollouts and c_i be the number of successes of task x_i. We estimate coverage within k attempts following Chen et al. (2021):

$$\hat{\text{pass}}@k(M, H) = \frac{1}{N} \sum_{i=1}^{N} \left[1 - \frac{\binom{n-c_i}{k}}{\binom{n}{k}} \right], \qquad 1 \le k \le n, \tag{1}$$

where the numerator is zero when $n - c_i < k$. Tasks are weighted equally within each dataset or benchmark; text macro averages weight datasets equally. At $k = 1$, this is mean task success; at $k = n$ for equal-sized pools, it is the fraction solved at least once. Equation 1 is unbiased under independent, identically distributed attempts from a fixed policy.

B Experimental Details

B.1 Datasets and Splits

We follow prior split protocols, with sample sizes and adaptations in Table 1. Symptom2Disease, LawBench, USPTO, and FiNER use the MCE splits (Ye et al., 2026) distributed with Meta-Harness (Lee et al., 2026b); GoEmotions (Demszky et al., 2020) and Banking77 (Casanueva et al., 2020) derive from their public releases. We subsample only examples with one annotated label from official splits for GoEmotions to ease the evaluation. HotpotQA follows GEPA's split sizes (Agrawal et al., 2026) using disjoint subsets of the official fullwiki training set. AppWorld retains the official train, development, and `test_normal` partitions (Trivedi et al., 2024). Both dynamic benchmarks retain their original difficulty labels.

Table 1: Dataset sizes and split roles. Prep. supplies examples or tasks to the harness; search/validation supplies optimization feedback. Reused selection shares the validation set.

Dataset	Prep.	Search/val.	Select.	Test
Symptom2Disease (ID)	200	50	reused	212
LawBench (ID)	200	50	reused	100
USPTO (ID)	50	30	reused	100
FiNER (OOD)	200	—	—	100
GoEmotions (OOD)	200	—	—	150
Banking77 (OOD)	200	—	—	150
HotpotQA	150	300	300	300
AppWorld	30	60	57	168

B.2 Baselines and Harness Optimization

We use Meta-Harness (Lee et al., 2026b) with Claude Opus 5 as the coding-agent proposer, keeping solver weights fixed. Each optimization allows 20 iterations with up to two proposals per iteration.

Text classification. We use GPT-OSS-20B for the baseline and optimized harness, with GPT-OSS-120B as the stronger reference; all use medium reasoning effort and temperature 0. Both baselines adapt the few-shot harness of Lee et al. (2026b), resampling demonstrations for each query and repetition within a pre-defined character budget. We select by equal-dataset mean validation accuracy. The selected harness rebuilds memory from preparation examples for each repetition. OOD evaluation uses GPT-OSS-20B with the same preparation policy on target-domain examples, without further code optimization.

HotpotQA. We use Qwen3-8B with Qwen3-32B as the stronger reference, both at temperature 0.6. The baseline follows GEPA's multihop seed program (Agrawal et al., 2026) with BM25 retrieval. The optimized harnesses are capped at four solver calls, two retrievals, and seven passages

per retrieval as well as the baseline. We shortlist candidates using validation performance and select by exact match (EM) on a separate selection set.

AppWorld. We use GPT-OSS-120B at medium reasoning effort, with a high-effort baseline as the stronger reference. Repeated evaluation uses temperature 1. The baseline ports AppWorld's simplified ReAct agent (Trivedi et al., 2024). Each attempt permits 50 solver calls and 50 environment steps, with a 2,400-second timeout extended to 7,200 seconds for high effort. Search and selection use temperature zero and original TGC, with final selection on the development set.

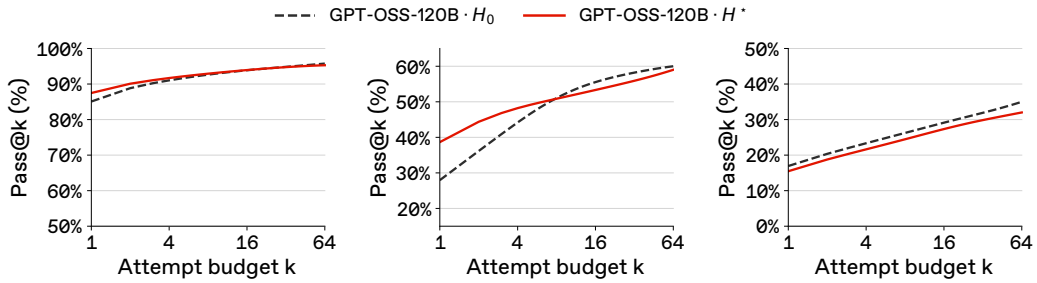
B.3 Success Scorers

We apply the same scorer to all systems within each benchmark, relaxing format requirements where necessary to better distinguish task completion from format compliance. In text classification, we use binary correctness after dataset-specific surface normalization. In HotpotQA, to accept correct answers embedded in longer responses, we use $\mathrm{Judge} = (\mathrm{Contain} \wedge \mathrm{CORRECT})$: answer containment with an LLM judgment confirming correctness. With 128 planned attempts per task, EM gives the Qwen3-32B lower $\mathrm{pass@1}$ than Qwen3-8B, which is counterintuitive, whereas Judge reverses this ordering. In AppWorld, we use arity-fixed TGC: original successes remain successful, and an action attempt is additionally accepted only if all state checks pass, no execution error is recorded, and its sole failed check concerns an unrequested answer. With 128 planned attempts per task, original TGC ranks the GPT-OSS-120B below GPT-OSS-20B at $\mathrm{pass@1}$; arity-fixed TGC reverses this ordering.

C Additional Experimental Results

C.1 Static Tasks

Additional results on GPT-OSS-120B. Single-attempt gains also do not improve pass@64 for GPT-OSS-120B. In Figure 12, LawBench pass@1 increases from 27.95% to 38.66%, while pass@64 decreases from 60% to 59%. The optimized harness also has lower pass@64 on Symptom2Disease, and its USPTO curve remains below the baseline throughout.



(a) Symptom2Disease (b) LawBench (c) USPTO

Figure 12: Text-classification pass@k curves for the GPT-OSS-120b baseline (H_0) and selected optimized harnesses ($H^\star$).

Task-level changes in observed success sets. Net coverage changes on the static tasks combine newly solved and lost tasks across each dataset, as shown in Table 2.

Table 2: Changes in observed success sets from the GPT-OSS-20B base harness (H_0) to the selected harness ($H^\star$) in Figure 3 (a)-(c), with $n = k = 64$. S_0 and $S_\star$ are the tasks with at least one success under H_0 and $H^\star$.

		Tasks			pass@1 gain (pp)			
Dataset	$\lvert S_0\rvert \to \lvert S_\star\rvert$	$S_\star \setminus S_0$	$S_0 \setminus S_\star$	$S_0 \cap S_\star$	Total	$S_0 \cap S_\star$	$S_\star \setminus S_0$	$S_0 \setminus S_\star$
Symptom2Disease	203 → 201	3	5	198	+2.57	+2.63	+0.38	−0.43
LawBench	53 → 56	7	4	49	+25.14	+21.02	+4.64	−0.52
USPTO	28 → 32	7	3	25	+3.39	+3.25	+0.19	−0.05

Frozen memory. Reusing one prepared memory across repetitions, instead of rebuilding it for each repetition, leaves the pattern in Figure 3 (a)-(c) unchanged, as shown in Table 3. With either memory policy, the selected GPT-OSS-20B harness improves pass@1 over H_0 on all three datasets, while at pass@64 it falls below H_0 on Symptom2Disease, its LawBench gain narrows, and it remains below the GPT-OSS-120B reference on every dataset.

Table 3: pass@1 and pass@64 (%) for the text-classification in Figure 3 (a)-(c), with $n = 64$. Rebuilt and Frozen denote the selected GPT-OSS-20B harness ($H^\star$) with memory rebuilt for each repetition, as in the figure, or prepared once per dataset and reused.

	pass@1				pass@64			
Dataset	20B H_0	Rebuilt	Frozen	120B H_0	20B H_0	Rebuilt	Frozen	120B H_0
Symptom2Disease	82.08	84.66	86.34	85.08	95.75	94.81	92.45	95.75
LawBench	18.86	44.00	43.53	27.95	53.00	56.00	53.00	60.00
USPTO	6.25	9.64	10.03	16.94	28.00	32.00	30.00	35.00

Random guessing on static tasks. Both harnesses exceed uniform-guessing coverage on the four single-label datasets at smaller k, as shown in Table 4. Independent uniform guesses over L labels yield expected coverage $1 - (1 - 1/L)^k$.

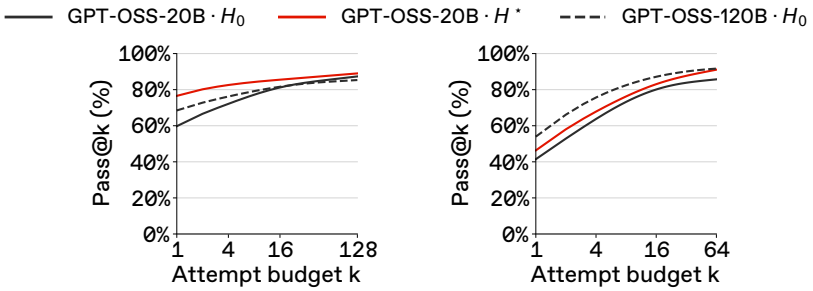
Table 4: Coverage (%) for uniform-label guessing and the GPT-OSS-20B baseline (H_0) and selected harness ($H^\star$) in Figures 3 and 10. Guessing values are analytical; harness estimates use $n = 64$.

	pass@4			pass@8		
Dataset	Guessing	H_0	$H^\star$	Guessing	H_0	$H^\star$
In-distribution						
Symptom2Disease	16.98	89.86	90.72	31.08	92.08	92.16
LawBench	—	35.12	50.51	—	42.15	52.59
USPTO	—	12.90	16.10	—	16.08	19.30
Out-of-distribution						
FiNER	29.39	79.11	76.39	50.15	82.44	79.17
GoEmotions	13.54	56.74	52.84	25.24	61.08	55.92
Banking77	5.09	89.73	79.29	9.93	91.10	80.07

Guessing samples the complete label inventory uniformly with replacement. This single-label reference is not applicable to LawBench and USPTO, which require open-ended structured outputs.

C.2 Dynamic Tasks

We also optimize a harness for GPT-OSS-20B on each dynamic benchmark, as shown in Figure 13. The optimized system exceeds the 120B baseline on HotpotQA at pass@128 (89.00% versus 85.33%), while remaining slightly below it on AppWorld at pass@64 (91.07% versus 91.67%).



(a) HotpotQA (b) AppWorld

Figure 13: pass@k curves for the GPT-OSS-20B baseline (H_0) and selected optimized harnesses ($H^\star$), with a GPT-OSS-120B baseline.

C.3 Optimization Conditions

Relaxed resource limits do not yield higher mean coverage at $k = 128$ in these runs. We compare three Qwen3-8B optimization runs per setting, with caps of 4 solver calls, 2 retrievals, and 7 passages per retrieval (tight) versus 12/12/20 (relaxed). Each run proposes 40 candidates on the same optimization splits; the caps apply during both optimization and evaluation.

Table 5: HotpotQA Judge scores (%) on 300 test questions with $n = 128$, reported as mean $\pm$ sample SD across three runs per cap setting.

Caps	pass@1	pass@128
Relaxed	73.39 ± 1.60	87.22 ± 1.54
Tight	72.50 ± 1.43	87.78 ± 1.39