WORKTHROUGH-BENCH: BENCHMARKING AGENTS ON LONG-HORIZON PRODUCTIVITY WORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon productivity work is becoming the proving ground for AI agents. Evaluating agents on such work remains challenging, as existing benchmarks often focus on minute-scale tasks, narrow work settings, or one-shot generation. We present WorkThrough-Bench, 86 tasks from professional requests across dozens of industries, spanning deep research, deliverable creation, and application building. Multi-artifact deliverables are independently graded through rule-based checks, LLM judgments, and behavioral probes. Across six frontier models and two agent harnesses, execution averages 122.8 minutes per run. Agents satisfy an average of 68.0% of the evaluation criteria per task, yet task pass rates remain below 11% in every model–harness group, and 54 of the 86 tasks are never completed in full by any group. Examining this gap between partial progress and reliable delivery, trajectory analysis reveals *self-contented completion*, where agents conclude that a task is complete based on checks of their own implementation and remembered requirements, without verifying the original task contract. We introduce feedback that redirects verification toward the original requirements and prompts corrections to overlooked discrepancies. Using fixed feedback, we improve the mean task pass rate by 26.7% relative to the no-feedback baseline, with trajectories showing renewed inspection and targeted repairs. These findings highlight the potential of further feedback design to improve agents' completion judgments and final delivery.

1 INTRODUCTION

Productivity work produces usable deliverables that support decisions and organizational operations. Long-horizon productivity work involves interdependent stages of investigation, creation, and verification, requiring hours to days of sustained expert effort (Mazeika et al., 2025). AI agents increasingly undertake such work (Handa et al., 2025), including compliance auditing (Vidgen et al., 2026), financial reconciliation (Patwardhan et al., 2025), research synthesis (Sun et al., 2026), and internal tool development (Miserendino et al., 2025). The value of such work depends on delivering a complete, verifiable product that satisfies the original requirements.

Assessing whether agents can sustain these workflows through complete delivery remains challenging, as existing benchmarks provide limited coverage of their duration and scope. Benchmarks for office and web agents operate at the minute scale (Zhou et al., 2024; Xie et al., 2024; Zhang et al., 2026). The suites that reach longer horizons each concentrate on one slice of the space: a single desktop or office-suite interface (Yuan et al., 2026; Zhou et al., 2026), a few expert professions (Vidgen et al., 2026), broad collections of expert project work that partly depend on specialized professional software (Sun et al., 2026), terminal-based engineering (Li et al., 2026b), or one-shot deliverable generation without agentic interaction (Patwardhan et al., 2025); the longest regimes are dominated by science and engineering under day-scale contracts (Chan et al., 2025; Starace et al., 2025; Zhu et al., 2026). What is missing is the long-horizon core of everyday office work: not operating a single interface or a specialized software package, but the recurring workflows that cut across industries and roles—investigating open questions, producing complete deliverables under professional rules, and building tools for repeated use.

To fill this gap, we present WorkThrough-Bench, a benchmark of 86 long-horizon tasks drawn from real daily office scenarios across dozens of industries. As illustrated in Figure 1(a), the benchmark

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.

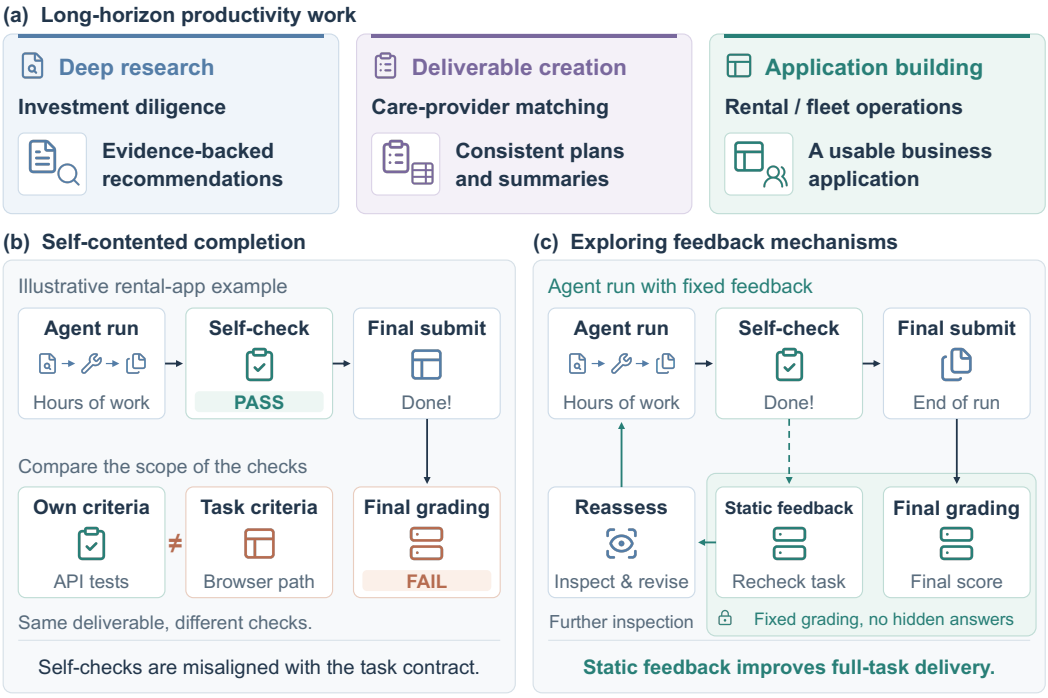

Figure 1: **WorkThrough-Bench: work scope, completion judgments, and feedback.** **(a)** Representative tasks and work products from three workflow families. **(b)** A rental-app example: API self-checks pass, but required browser acceptance fails. **(c)** A fixed feedback prompts agents to recheck requirements and revise their deliverables, improving full-task delivery under unchanged grading criteria and without revealing hidden grading signals.

spans three recurring workflow families, namely deep research, deliverable creation, and application building, with representative work products including evidence-backed recommendations, consistent plans and summaries, and usable business applications. Each task originates from an expert-provided professional request, which we develop with LLM assistance into a specification, input materials, and an execution environment. Task length arises from genuine dependencies among information gathering, judgment, computation, and verification, and mean agent execution time ranges from about 20 minutes to 6 hours across model–harness groups, with the longest retained run approaching 16 hours. Candidate tasks undergo expert self-review, independent cross-review, and agent-assisted audit, followed by inspection of sandbox trial trajectories and deliverables to identify defects and exclude unsuitable tasks.

We evaluate six frontier models under two agent frameworks on WorkThrough-Bench. Individual runs routinely span hours, and mean task completion ranges from 47.8% to 79.8%, yet the task pass rate stays between 0.4% and 10.4% for every model–harness group: **much of the work gets done, but a complete deliverable is rare**. Examining the failing runs, we observe one situation recur: the agent works at length, declares the job done with full confidence, and the deliverable still violates its contract, with numbers failing to reconcile across files, contracted interface elements missing, and required content silently absent. Figure 1(b) shows a representative case: the agent verified the rental application it had built by calling its own API, every self-test passed, and it declared completion, although the specification required verification through a browser-driven acceptance path, so external grading failed the run. The problem is not a lack of testing but a mismatch between what the agent checks and what the task requires; we call this pattern *self-contented completion*: the agent judges work to be finished against its own implementation and remembered requirements rather than reconciling the deliverable with the original task contract. To redirect verification from agents' own criteria to the original task requirements, we design feedback that asks agents to reassess their actual deliverables against those requirements. In our study, we use a fixed prompt as the feedback signal to encourage agents to identify and correct discrepancies before final submission

(Figure 1(c)). This feedback design yields a 26.7% relative improvement in mean task pass rate, with trajectories showing renewed inspection and repairs of overlooked requirements.

To summarize, our contributions are as follows.

- **Benchmark.** We present WorkThrough-Bench, a benchmark of 86 long-horizon tasks built from professional work requests, spanning deep research, deliverable creation, and application building across dozens of industries, with programmatic grading that verifies deliverables against the task contract rather than the agent's self-reports.
- **Finding.** Evaluating six frontier models under two agent frameworks, we find that agents sustain hours of work yet rarely deliver it complete. We identify trajectory-level instances of *self-contented completion*: agents judge work to be finished against their own implementation and memory of the requirements rather than the task contract.
- **Feedback.** We introduce a feedback mechanism that redirects agents to verify their deliverables against the original task requirements. A fixed-prompt implementation improves mean task pass rate by 26.7% relative to the no-feedback baseline, highlighting the potential of feedback mechanisms to improve task completion and handle *self-contented completion*.

2 Related Work

Agentic work and long-horizon benchmarks. Most office and web agent benchmarks operate at the minute scale, in web and desktop environments (Zhou et al., 2024; Xie et al., 2024) or in recent personal-assistant suites (Zhang et al., 2026), with per-task budgets below half an hour. Long-horizon suites have begun to appear, but each concentrates on one slice of the space, such as a single office interface (Yuan et al., 2026; Zhou et al., 2026), a few expert professions (Vidgen et al., 2026; Li et al., 2026a), or day-scale software engineering and ML (Chan et al., 2025; Zhu et al., 2026); simulated-time benchmarks compress days-to-months of virtual time without measuring real execution (Backlund & Petersson, 2025). Within this landscape, WorkThrough-Bench targets the long-horizon core of everyday office work: it covers the recurring workflows of dozens of industries and roles rather than a single interface or a few professions, and its agent runs routinely take hours, one to two orders of magnitude longer than the minute-scale suites.

Unreliable self-assessment and completion claims. A growing line of work studies how well LLMs assess their own outputs. Models can express calibrated uncertainty in short-form QA (Kadavath et al., 2022), but their self-verification is unreliable without external feedback (Huang et al., 2024; Gou et al., 2024), and as evaluators they favor their own generations (Panickssery et al., 2024); Kalai et al. (2025) further show that binary scoring rewards confident guessing over acknowledging uncertainty. In agentic settings, similar behavior has been observed qualitatively: TheAgentCompany (Xu et al., 2025) and AutomationBench (Shepard & Salimans, 2026) report that agents routinely claim success on work that is unfinished or incorrect, and in the latter 72–91% of failures involve false completion claims. We complement these observations with checks and trajectory evidence from hours-long work, examining *self-contented completion*: agents verify deliverables against their own implementation and memory of the requirements rather than the task contract.

3 The WorkThrough-Bench

This section describes the task construction and grading design of WorkThrough-Bench. Each task originates from a professional work request contributed by an industry expert and is delivered as four components: a task specification, the input materials of its scenario, an execution environment, and a grading package that stays isolated from the agent throughout the run. The construction pipeline is shown in Figure 2; the remainder of this section walks through this pipeline, presents the grading scheme, and summarizes the resulting benchmark.

3.1 Construction Pipeline

Data collection and preparation. Industry experts recruited through a crowdsourcing platform contribute work requests, the workflows used to address them, and key input materials. Grounding

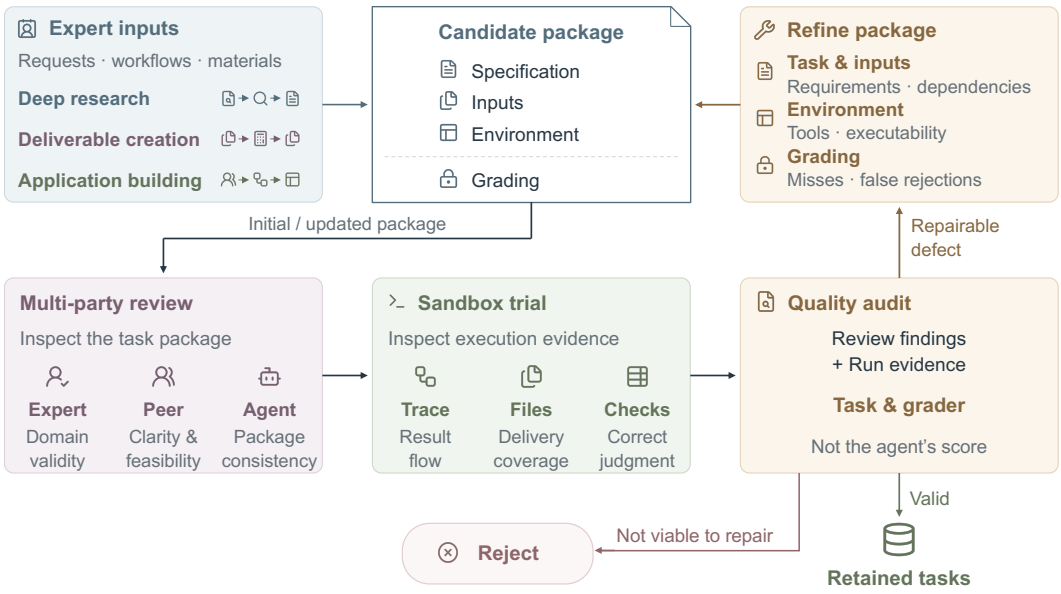

Figure 2: **Construction and iterative refinement of WorkThrough-Bench.** De-identified expert materials are formulated into task packages with isolated grading. Expert, peer, and agent-assisted reviews and sandbox trials guide iterative audits and repairs. Nonviable candidates are rejected; valid tasks are retained regardless of trial-agent success. Grading remains hidden from evaluated agents.

Table 1: Task taxonomy of WorkThrough-Bench, with representative tasks.

Family	Sub-family	Representative tasks
Deep research	Exploratory research	Generative-AI writing's effects on learning
	Fact checking	Enterprise policy-match audit
	Decision support	AI-startup investment diligence
Deliverable creation	Content production	Semester-exam authoring; RPG design
	Plan design	Care-provider matching; Promotion planning
	Standards development	Inspection standards; Rating rubrics
Application building	Workflow management	Equipment repair-ticket system
	Planning & analysis	PC-build price configurator
	Transaction & settlement	Car-rental and fleet operations
	Content & knowledge	Evidence management; interactive reading

tasks in professional practice ties their requirements to an identifiable user and downstream use (Patwardhan et al., 2025). We de-identify sensitive information while preserving record associations and business rules, and distinguish simulated data from source records. The collected requests are organized into three families by the basic use of the work they produce: deep research, deliverable creation, and application building, each further divided into sub-families that recur across workplaces (Table 1). Classification follows the business problem and the bulk of the work rather than industry or deliverable format, so the families cut across dozens of industries.

Task formulation. The collected materials are then turned into tasks through an agent-based generation process, in which the agent's instructions specify the business objective, available inputs, domain rules, and expected deliverables. Their design centers on three considerations. *Result dependence*: earlier findings must determine later selections, calculations, or actions, consistent with compositional and dependency-based task design (Boisvert et al., 2024; Tang et al., 2026); provider eligibility, for example, constrains matching options, which in turn determine pricing and service

plans. *Family-specific structure*: the dependencies are instantiated differently for the three families, with evidence gaps and conflicting sources driving further investigation in deep research, per-entity decisions subject to professional rules, shared resource limits, and cross-output consistency in deliverable creation, and user roles, business rules, and state transitions defining the required behavior in application building. *Executability and verifiability*: each task must ship a usable execution environment and observable acceptance criteria, while ordinary scripting and batching remain allowed.

The long-horizon character of the tasks is likewise engineered into their structure. Deep-research tasks supply little prepared material, so the agent must gather evidence through open-ended search, and the research questions themselves are interdependent, with findings on one question determining what must be investigated next. Deliverable-creation tasks follow the workspace design of Workspace-Bench (Tang et al., 2026), spreading facts, rules, and records across large, interdependent files that the agent must locate, read, and reconcile as the work requires. Application-building tasks amount to implementing a requirements specification of realistic complexity, and fulfilling it demands multiple rounds of development, review, and interaction with the execution environment. Substantive ambiguities are resolved by domain experts before a task is finalized. The resulting package comprises the specification, prepared inputs, the environment, and an isolated grading package; the instruction templates, including the family-specific components, are provided in Appendix B.1. This stage yielded 182 candidate tasks: 77 in deep research, 41 in deliverable creation, and 64 in application building.

Review and filtering. The authoring expert checks professional validity, independent reviewers examine clarity and feasibility, and agent-assisted audits compare the specification, inputs, and proposed checks (Patwardhan et al., 2025). Grading-generation prompts map each requirement to an observable outcome, a verification method, and a failure the check should detect, and reviewer prompts probe both missed errors and valid solutions that might be rejected. Candidate tasks then run in a sandbox, where trajectories, final artifacts, and check-level outcomes are inspected together; for application-building tasks, a separate implementation session receives only agent-visible materials, and grading is tested against the observed functionality. Confirmed defects in the task, environment, or grader lead to revision and revalidation, whereas agent omissions and reasoning errors remain capability observations. Candidates with unclear purposes, insufficient inputs, weak dependencies, or unverifiable outcomes are returned for revision, and those that cannot be repaired within a reasonable scope are removed. This cycle continues until a candidate is retained or rejected. Filtering removed 46 of the 77 deep-research candidates, 15 of the 41 deliverable-creation candidates, and 35 of the 64 application-building candidates, leaving 86 tasks in the final benchmark. The grading-generation, review, and refinement templates appear in Appendices B.2 and B.3.

3.2 Grading and Metrics

Grading. We assess final artifacts and system states against task-specific criteria, allowing alternative valid ways to complete the work (Zhou et al., 2024; Starace et al., 2025); the grading package combines the following check types as the task requires and remains inaccessible to the evaluated agent during execution. Each check type matches the property it verifies. Executable rule checks cover deterministic conditions such as numerical correctness, entity coverage, and cross-file consistency, with expected values recomputed from trusted inputs wherever possible, so that agreement between two submitted files cannot conceal a shared error (Zhou et al., 2024; Boisvert et al., 2024). LLM-based assessment covers semantic requirements such as source support and content coverage, judging each criterion against explicit binary rubrics, citing evidence locations, and preserving the task's acceptance boundary for legitimate alternative conclusions (Starace et al., 2025; Vidgen et al., 2026; Zheng et al., 2023). Agent-based behavioral assessment covers requirements that depend on interaction: an evaluation agent operates the deployed application through a browser, establishes the required preconditions, and checks the resulting business state (Zhou et al., 2024). The corresponding prompts are provided in Appendices B.4.

Metrics. Following prior agentic benchmarks (Xu et al., 2025; Vidgen et al., 2026), we report two complementary metrics: *mean task completion* (**Completion**), the fraction of required work satisfied on an average task, and *mean task pass rate* (**Pass rate**), the fraction of tasks completed in full. For a single run of task i, let $s_i \in [0, 1]$ be the fraction of its scored checks that pass. These checks have binary outcomes and equal weight; mandatory prerequisite checks act as gates, with a failed gate

assigning $s_i = 0$. Grading infrastructure failures produce no valid score. Over N tasks, the two metrics are

$$\text{Completion} = \frac{100}{N}\sum_{i=1}^{N} s_i, \qquad \text{PassRate} = \frac{100}{N}\sum_{i=1}^{N} \mathbf{1}[s_i = 1]. \tag{1}$$

Both are percentages, with higher values better. Normalizing within each task prevents tasks with more checks from dominating. Completion gives partial credit, whereas Pass rate credits only full satisfaction of the grading contract; their gap distinguishes substantial progress from complete delivery. Repeated runs are averaged within each task before applying the task-level average, as detailed in Appendix A.1.

Table 2: Comparison with representative agentic-work benchmarks: PinchBench (Kilo Code, 2026), ClawBench (Zhang et al., 2026), WildClawBench (Ding et al., 2026), OmegaUse-OfficeVal (Zhou et al., 2026), and LHTB (Li et al., 2026b), ordered by per-task agent runtime. *Domain* is the work setting; *Task focus* summarizes the work types each benchmark targets. *Runtime / task* is the mean per-task agent execution time, pooling evaluated runs.

Benchmark	**Domain**	**Task focus**	**#Tasks**	**Runtime / task**
PinchBench	Personal assist.	Light productivity workflows	148	1.3 min
WildClawBench	Personal assist.	Everyday personal tasks	60	8 min
ClawBench	Web	Everyday online tasks	281	12.1 min
OmegaUse-OfficeVal	Office suite	Office-suite deliverables	100	42.7 min
LHTB	Terminal	Long terminal workflows	46	88.9 min
Ours	**Industry Office**	**Long-term office work**	**86**	**122.8 min**

3.3 Benchmark Summary

The final WorkThrough-Bench contains 86 tasks, each shipping a specification, input materials, an execution environment, and an isolated grading package. Table 2 situates WorkThrough-Bench among representative agentic-work benchmarks.

4 Evaluation Results

4.1 Experiment Settings

We evaluate six frontier models: Kimi-K3 (Team, 2026), DeepSeek-V4.1-Flash (DeepSeek-AI, 2026), GLM-5.3 (GLM, 2026), Qwen-3.8-Flash (Qwen Team, 2026), GPT-5.6-Sol (OpenAI, 2026), and MiniMax-M3 (Lai et al., 2026). Each model is paired with two agent harnesses: Hermes Agent (Nous Research, 2026) and Codex CLI (OpenAI, 2025), yielding twelve model–harness groups. Both harnesses are configured with a context-length setting of 272,000 tokens and an automatic compaction threshold of 90%.

For each of the 86 tasks, the agent receives the specification and input materials in an isolated task environment and works without human intervention, under a 24-hour execution limit. The main experiment provides no in-run grading feedback. After execution ends, the isolated grader assesses the submitted artifacts and system state using the task's fixed checks (§3.2); the agent's own tests and completion claims do not determine its score. Runtime measures only the agent execution interval, excluding environment setup and final grading.

4.2 Main Results

As shown in Table 3, mean Completion reaches 68.0%, but mean Pass rate is only 6.9%, leaving a 61.1-percentage-point gap between partial progress and full acceptance. This gap also varies with task family: comparable Completion levels coexist with substantially lower Pass rates in deep research and deliverable creation than in application building. Aggregate progress therefore provides

Table 3: Main results on the 86-task WorkThrough-Bench. Completion and Pass rate are percentages. Deep research, Deliverable creation, and Application building contain 31, 26, and 29 tasks, respectively. For each model–harness combination, scores are averaged equally across tasks. Models are ordered by mean overall Completion across harnesses; the last row averages the twelve groups. Best values per metric column are bold. DeepSeek denotes the DeepSeek-V4.1-Flash model.

Model	Harness	Research		Deliverable		Application		Overall	
		Completion	Pass rate	Completion	Pass rate	Completion	Pass rate	Completion	Pass rate
Kimi-K3	Hermes	68.3	0.0	75.0	3.8	**83.8**	24.1	75.5	9.3
	Codex	**80.5**	1.1	**75.2**	2.6	83.2	**26.4**	**79.8**	10.3
DeepSeek	Hermes	70.8	1.2	74.6	3.8	81.2	25.3	75.6	**10.4**
	Codex	72.7	0.0	73.3	**6.4**	81.0	19.5	75.8	8.9
GLM-5.3	Hermes	69.8	0.0	69.1	0.0	82.2	21.8	73.8	7.4
	Codex	71.7	0.0	66.2	3.8	62.9	8.0	67.1	3.9
GPT-5.6-Sol	Hermes	72.6	**6.5**	69.3	3.8	62.8	9.2	68.3	6.6
	Codex	66.9	3.2	66.2	3.8	67.2	12.6	66.8	6.6
Qwen-3.8-Flash	Hermes	55.2	3.2	69.9	0.0	73.4	18.4	65.8	7.4
	Codex	59.7	2.4	68.9	3.8	78.9	23.0	69.3	10.0
MiniMax-M3	Hermes	49.4	0.0	59.1	0.0	43.1	1.1	50.2	0.4
	Codex	46.6	0.0	52.9	0.0	44.6	3.4	47.8	1.2
Average Score		65.4	1.5	68.3	2.7	70.4	16.1	68.0	6.9

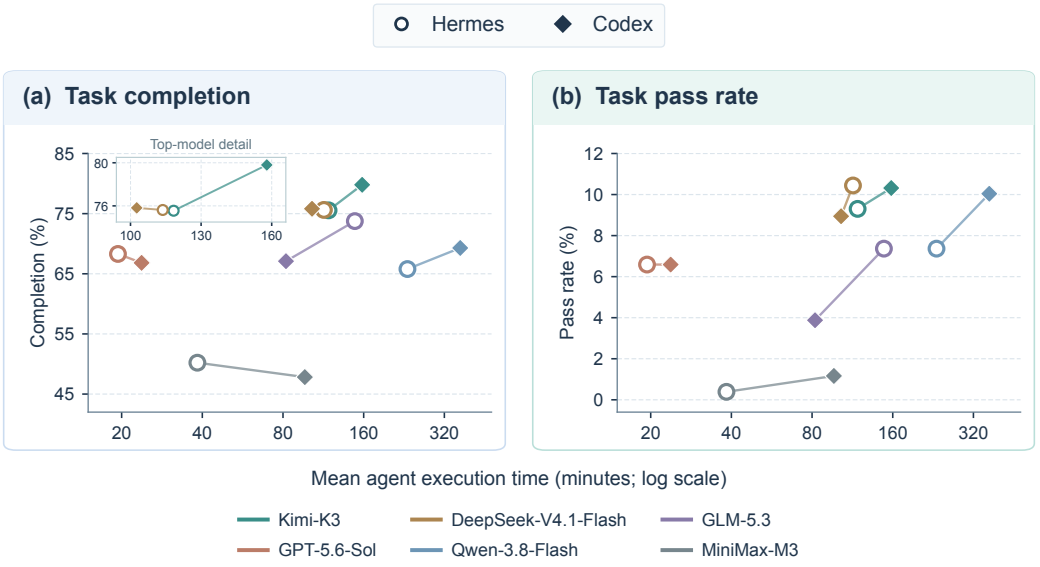

Figure 3: Completion and Pass rate versus mean agent execution time, averaged over the same retained runs as Table 3. Colors denote models; open circles denote Hermes and filled diamonds denote Codex. Lines connect paired harness results; the inset enlarges the Kimi and DeepSeek cluster. Setup and grading time are excluded.

an incomplete account of task difficulty, because satisfying much of a specification does not ensure an acceptable final product. Within this overall pattern, the leading and trailing model positions persist across harnesses and on the tasks shared by all configurations.

Harness choice further shapes these model differences and the runtime needed to achieve each result. For example, GLM ranks above Qwen under Hermes but below it under Codex. To examine how each model responds to the harness, we compare tasks shared by both configurations, avoiding differences due to missing coverage. Codex improves Kimi's Completion by 3.0 percentage points, driven by a 9.1-point gain in deep research, and improves Qwen by 1.5 points, driven by application

building. GLM instead benefits from Hermes: Completion is 6.7 points higher, including a 19.2-point advantage in application building. These gains also have different runtime costs (Figure 3). On shared tasks, Kimi's mean execution time rises from 113.1 minutes under Hermes to 157.9 under Codex; GLM's stronger Hermes configuration takes 148.3 minutes versus 82.1 under Codex. Additional time is not uniformly beneficial: MiniMax takes 96.5 minutes under Codex versus 38.3 under Hermes, yet Completion falls by 2.4 points. Its Pass rate nevertheless rises from 0.4% to 1.2%, illustrating that broader partial coverage and more frequent full success need not improve together. There is no universally preferable harness or monotonic relationship between runtime and delivery quality.

GPT-5.6-Sol provides a particularly clear illustration of this runtime–quality relationship: mean execution lasts only 19.4 minutes under Hermes and 23.7 under Codex, but Completion is 68.3% and 66.8%, and Pass rate is 6.6% under both. Its strongest result is in deep research under Hermes, where it leads in Pass rate; its relative weakness is application building. Retained GPT trajectories show an emphasis on concurrent execution: source retrieval is parallelized, and one run explicitly stops a slow retrieval pass before retrying with bounded timeouts. Such strategies can reduce elapsed time without ensuring that every requirement is checked. Conversely, Qwen spends 233.0–366.7 minutes without matching the leaders' Completion. These observations suggest examining how work is allocated and verified, rather than equating longer runtime with greater capability or treating GPT's short runtime itself as the cause of failure.

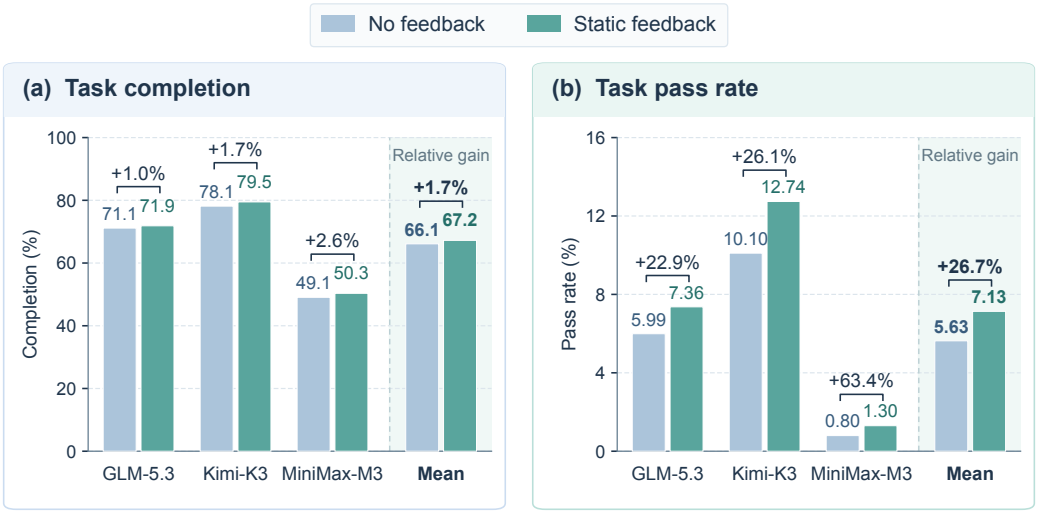

Figure 4: **Fixed feedback improves full-task delivery.** Paired bars compare no feedback and static feedback on shared tasks within each model–harness group. Values average repeats within tasks, then equally weight tasks and harnesses. The shaded Mean groups average the three models. Brackets report relative gains for each metric using unrounded values. Each Mean gain is computed from the two corresponding mean scores, rather than averaging model-specific relative gains: +1.7% for Completion and +26.7% for Pass rate.

4.3 Self-Contented Completion and Feedback

The preceding comparisons show that stronger partial performance and longer execution do not consistently lead to complete delivery. Indeed, 54 of 86 tasks never pass in any retained run. These results raise the question of why agents accept incomplete work as finished, and whether revisiting that judgment can help them satisfy requirements that remain unmet.

Agents' recorded plans and closing explanations reveal instances of *self-contented completion*, in which successful checks on the implementation are treated as evidence that the original request has been fulfilled. In one GPT-5.6-Sol investment-diligence run, the agent reports 365 passed checks but achieves only 15.9% Completion, with the required evidence ledger and five workbooks absent from their contracted paths. Such cases extend concerns about unreliable self-verification (Huang

et al., 2024; Gou et al., 2024) to multi-artifact work, where confidence in individual components can obscure omissions in the overall delivery.

The harness may also contribute to self-contented completion through repeated context compaction due to the long working time, which can gradually alter the agent's working representation of the task contract. Summaries may retain completed actions and successful checks while omitting unresolved obligations or acceptance conditions. When subsequent verification relies on these summaries, each compaction can carry forward a reduced set of requirements, progressively narrowing the scope of self-checks over a long task. The agent may consequently become more confident that its work is complete even as its checks diverge from the original delivery requirements. The task contract remains unchanged, but the criteria the agent uses to judge completion may drift across successive summaries.

To bring agents' verification back into alignment with the task contract, we introduce feedback that directs them to reread the original requirements and inspect their current deliverables. We use a fixed prompt to inform agents to correct discrepancies supported by explicit requirements while preserving verified work. File existence, earlier self-check reports, and completion claims are insufficient evidence; the prompt requires direct inspection and leaves unverified requirements unresolved. The full prompt appears in Appendix B.5. We evaluate this mechanism on GLM-5.3, Kimi-K3, and MiniMax-M3 under both harnesses, allowing up to three submissions within the original time limit. Final grading remains unchanged, and the prompt provides neither defect-specific advice nor hidden grading signals.

As Figure 4 shows, feedback raises mean Completion from 66.1% to 67.2% and mean Pass rate from 5.63% to 7.13%, corresponding to relative gains of 1.7% and **26.7%**. All three models improve on both metrics when averaged across harnesses. The inspected paired trajectories reveal a corresponding change in how agents approach work they have already declared complete. Agents return to the original requirements and inspect deliverable content, uncovering omissions missed by earlier self-checks. They correct missing boundary behavior, unsupported source claims, and inconsistencies across linked artifacts, followed by regression checks or artifact regeneration. Feedback thus brings previously unchecked requirements back into the verification process and prompts concrete revisions.

These revisions can be decisive for deliverables that already satisfy most requirements. Cases with improved Pass rate have a median baseline Completion of approximately 89%, compared with 67% for the remainder, consistent with feedback recovering omissions in substantially completed work. Correcting the remaining discrepancies adds little partial credit but can determine whether the entire deliverable passes acceptance. This connects the modest increase in Completion to the larger relative gain in Pass rate. The trajectories suggest that some premature completion judgments leave agents' existing capabilities insufficiently applied to the full task contract. Redirecting verification to the original requirements can bring these capabilities to bear on overlooked obligations, highlighting feedback's potential to help agents recognize and correct shortcomings in their own verification.

5 Conclusion

We introduced WorkThrough-Bench, 86 long-horizon knowledge-work tasks with independently verifiable deliverables. Evaluation reveals substantial progress but incomplete delivery, accompanied by self-contented completion judgments. We introduce feedback that directs agents to reassess their deliverables against the original task requirements. A fixed-prompt implementation improves mean task pass rate by 26.7% relative, with trajectories showing renewed verification and repairs of overlooked requirements. These results highlight the potential of feedback to help agents recognize and correct shortcomings in their own verification, suggesting that further advances in feedback design could improve completion judgments and final delivery. By distinguishing partial progress from full acceptance, WorkThrough-Bench provides a testbed for assessing how such advances contribute to reliable long-horizon work.

AI USE STATEMENT

We used generative AI tools for drafting assistance and, as described in §3.1, as part of the benchmark's grading-package construction pipeline. All AI-generated grading artifacts were verified through the validation and audit procedures of §3.1. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The benchmark contains no human-subjects data. Tasks are derived from professional work scenarios and use public or de-identified materials; tasks involving realistic evaluation of harmful or sensitive content are reviewed during task authoring. The failure analyses report model behavior only.

REFERENCES

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

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault Le Sellier de Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. Mlebench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. *CoRR*, abs/2606.19348, 2026. doi: 10.48550/ARXIV.2606.19348. URL https://doi.org/10.48550/arXiv.2606.19348.

Shuangrui Ding, Xuanlang Dai, Long Xing, Shengyuan Ding, Ziyu Liu, Yang JingYi, Penghui Yang, Zhixiong Zhang, Xilin Wei, Xinyu Fang, Yubo Ma, Haodong Duan, Jing Shao, Jiaqi Wang, Dahua Lin, Kai Chen, and Yuhang Zang. Wildclawbench: A benchmark for real-world, long-horizon agent evaluation. *CoRR*, abs/2605.10912, 2026.

GLM. GLM-5: from vibe coding to agentic engineering. *CoRR*, abs/2602.15763, 2026. doi: 10.48550/ARXIV.2602.15763. URL https://doi.org/10.48550/arXiv.2602.15763.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Kunal Handa, Alex Tamkin, Miles McCain, Saffron Huang, Esin Durmus, Sarah Heck, Jared Mueller, Jerry Hong, Stuart J. Ritchie, Tim Belonax, Kevin K. Troy, Dario Amodei, Jared Kaplan, Jack Clark, and Deep Ganguli. Which economic tasks are performed with ai? evidence from millions of claude conversations. *CoRR*, abs/2503.04761, 2025.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *CoRR*, abs/2207.05221, 2022.

Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Why language models hallucinate. *CoRR*, abs/2509.04664, 2025.

Kilo Code. PinchBench: Real-world benchmarks for ai coding agents. `https://github.com/pinchbench/skill`, 2026.

Xunhao Lai, Weiqi Xu, Yufeng Yang, Qiaorui Chen, Yang Xu, Lunbin Zeng, Xiaolong Li, Haohai Sun, Haichao Zhu, Vito Zhang, Jinkai Hu, Jiayao Li, Rui Gao, Zekun Li, Songquan Zhu, Jingkai Zhou, and Pengyu Zhao. Minimax sparse attention. *CoRR*, abs/2606.13392, 2026. doi: 10.48550/ARXIV.2606.13392. URL `https://doi.org/10.48550/arXiv.2606.13392`.

Yuetai Li, Yichen Feng, Zhangchen Xu, Zixian Ma, Kaiyuan Zheng, Fengqing Jiang, Xinghua Sun, Rulin Shao, Zichen Chen, Yue Huang, Xinyang Han, Brian Lee, Kayla Xu, Shenglai Zeng, Hang Hua, Xiangliang Zhang, Basel Alomair, Ranjay Krishna, Luke Zettlemoyer, Pang Wei Koh, Bhaskar Ramasubramanian, Luyao Niu, Xiang Yue, and Radha Poovendran. Jobbench: Aligning agent work with human will. *CoRR*, abs/2605.26329, 2026a.

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, Lichao Sun, Haotao Mi, and Leowei Liang. Long-horizon-terminal-bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading. *CoRR*, abs/2607.08964, 2026b.

Mantas Mazeika, Alice Gatti, Cristina Menghini, Udari Madhushani Sehwag, Shivam Singhal, Yury Orlovskiy, Steven Basart, Manasi Sharma, Denis Peskoff, Elaine Lau, Jaehyuk Lim, Lachlan Carroll, Alice Blair, Vinaya Sivakumar, Sumana Basu, Brad Kenstler, Yuntao Ma, Julian Michael, Xiaoke Li, Oliver Ingebretsen, Aditya Mehta, Jean Mottola, John Teichmann, Kevin Yu, Zaina Shaik, Adam Khoja, Richard Ren, Jason Hausenloy, Long Phan, Ye Htet, Ankit Aich, Tahseen Rabbani, Vivswan Shah, Andriy Novykov, Felix Binder, Kirill Chugunov, Luis Ramirez, Matias Geralnik, Hernán Mesura, Dean Lee, Ed-Yeremai Hernandez-Cardona, Annette Diamond, Summer Yue, Alexandr Wang, Bing Liu, Ernesto Hernandez, and Dan Hendrycks. Remote labor index: Measuring AI automation of remote work. *CoRR*, abs/2510.26787, 2025.

Samuel Miserendino, Michele Wang, Tejal Patwardhan, and Johannes Heidecke. Swe-lancer: Can frontier llms earn $1 million from real-world freelance software engineering? In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

Nous Research. Hermes Agent. `https://github.com/NousResearch/hermes-agent`, 2026.

OpenAI. Codex CLI: Lightweight coding agent that runs in your terminal. `https://github.com/openai/codex`, 2025.

OpenAI. Gpt-5.6 system card, 2026. URL `https://deploymentsafety.openai.com/gpt-5-6`.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Miserendino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. Gdpval: Evaluating AI model performance on real-world economically valuable tasks. *CoRR*, abs/2510.04374, 2025.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Daniel Shepard and Robin Salimans. Automationbench. *CoRR*, abs/2604.18934, 2026.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. Paperbench: Evaluating ai's ability to replicate AI research. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

Yiyou Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, Yuhan Cao, Yixiao Huang, Chris Duroiu, Haoyun Zhang, Jeffrey Lin, Weishu Zhang, Tyler Zeng, Ying Yan, Bo Liu, Hanson Wen, Mingyang Xu, Xiaoyuan Liu, Zimeng Chen, Weiyan Shi, Amanda Dsouza, Vincent Sunn Chen, Patrick Bryant, Carl Boettiger, Yamini Rangan, Bradley Rothenberg, Kyle Steinfeld, Arvind Rao, Tapio Schneider, Georgios Yannakakis, Laure Zanna, Kaan Ozbay, Ida Sim, Tarek Zohdi, George Em Karniadakis, Jack Gallant, Teresa Head-Gordon, Yushan Li, Wenxi Deng, Tao Sun, Huiqi Wang, Zhun Wang, Justin Xu, Chris Yuhao Liu, Yafei Cheng, Rongwang Hu, Aras Bacho, Shengcao Cao, Zengyi Qin, Yixiong Chen, Hengduan Fan, Hao Liu, Lin Zeng, Shashank Muralidhar Bharadwaj, Litian Gong, Yingxuan Yang, Maojia Song, Ruheng Wang, Zongzheng Zhang, Honglin Bao, Shuo Lu, Jianhong Tu, Zhonghua Wang, Zheng Zhang, Zijiao Chen, Yanqiong Jiang, Zhendong Li, Bohan Lyu, Chang Ma, Peiran Xu, Benran Zhang, Shangding Gu, Haoyue Hua, Haoyang Li, Wanzhe Liao, Chengzhi Liu, Junbo Peng, Haoran Sun, Zechen Xu, Bo Chen, Jiayi Cheng, Yi Jiang, Keying Kuang, Yuan Li, Youbang Pan, Ziyan Rao, Alexander Schubert, Yifan Shen, Vincent Siu, Xiatao Sun, Kangqi Zhang, Xiaopan Zhang, Yuchen Zhu, Ishaan Singh Chandok, Lei Ding, Jingxuan Fan, Andrew Glover, Jiaming Hu, Yiran Hu, Wenbo Huang, and et al. Agents' last exam. *CoRR*, abs/2606.05405, 2026. doi: 10.48550/ARXIV.2606.05405. URL https://doi.org/10.48550/arXiv.2606.05405.

Zirui Tang, Xuanhe Zhou, Yumou Liu, Linchun Li, Yukai Wu, Weizheng Wang, Hongzhang Huang, Wei Zhou, Jun Zhou, Jiachen Song, Shaoli Yu, Jinqi Wang, Zihang Zhou, Hongyi Zhou, Yuting Lv, Jinyang Li, Jiashuo Liu, Ruoyu Chen, Chunwei Liu, Guoliang Li, Jihua Kang, and Fan Wu. Workspace-bench 1.0: Benchmarking AI agents on workspace tasks with large-scale file dependencies. *CoRR*, abs/2605.03596, 2026.

Kimi Team. Kimi K3: open frontier intelligence. *CoRR*, abs/2607.24653, 2026. doi: 10.48550/ARXIV.2607.24653. URL https://doi.org/10.48550/arXiv.2607.24653.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, Neel Venugopal, Alannah Hsia, Isaac Robinson, Calix Huang, Olivia Varones, Daniyal Khan, Michael Haines, Zach Richards, Chirag Mahapatra, Brendan Foody, and Osvald Nitski. Apex-agents. *CoRR*, abs/2601.14242, 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Frank Fangzheng Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Zhiruo Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu,

Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking LLM agents on consequential real world tasks. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, Dunjie Lu, Haikong Lu, Qi Zhen, Xinyuan Wang, Jiaqi Deng, Yuhao Yang, Cheng Chen, Boyuan Zheng, Alex Su, Xiao Yu, Hao Zou, Saaket Agashe, Xing Han Lù, Manpreet Kaur, Zhengyang Qi, Vincent Sunn Chen, Frederic Sala, Dayiheng Liu, Junyang Lin, Zhou Yu, Yu Su, Siva Reddy, Xin Eric Wang, Peng Qi, Tianbao Xie, and Tao Yu. Osworld2.0: Benchmarking computer use agents on long-horizon real-world tasks. *CoRR*, abs/2606.29537, 2026.

Yuxuan Zhang, Yubo Wang, Yipeng Zhu, Penghui Du, Junwen Miao, Xuan Lu, Wendong Xu, Yunzhuo Hao, Songcheng Cai, Xiaochen Wang, Huaisong Zhang, Xian Wu, Yi Lu, Minyi Lei, Kai Zou, Huifeng Yin, Ping Nie, Liang Chen, Dongfu Jiang, Wenhu Chen, and Kelsey R. Allen. Clawbench: Can AI agents complete everyday online tasks? *CoRR*, abs/2604.08523, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Jingbo Zhou, Yusai Zhao, Qi Bao, Jingjia Cao, Zhenghai Chen, Chang Gao, Kaiqi Guo, Muxin Guo, Mingxuan Li, Xinjiang Lu, Yanru Ma, Yixiong Xiao, Zenghui Zhang, Le Zhang, and Hua Wu. Omegause-officeval: Benchmarking LLM agents on long-horizon office-suite tasks with economic grounding. *CoRR*, abs/2607.27155, 2026.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Deyao Zhu, Xin Zhou, Shengling Qin, Xuekai Zhu, Hangliang Ding, Shu Zhong, Zixin Wen, Zhonglin Xie, Chenhui Gou, Linxuan Ren, Yueyang Wang, Junfeng Zhong, Rui Liu, Tian Gao, Yangguang Lin, Jingyuan Zhang, Maojia Song, Xuan Qi, Jinhong Wu, Chenyang Zhang, Yinzhu Piao, Ziru Niu, Hongbin Lin, Lingxiang Meng, Peng Tang, Chengyao Tang, Shanyu Wu, Huanyu Zheng, Yu Liu, Liya Zhu, He Wang, Ming Ding, Ziyu Wan, Hao Liu, Sibo Wang, Haotian Zhu, Xintian Zhang, Nan Chai, Yipeng Liu, Panhao Lai, Sihang Yuan, Zixin Su, Ge Zhang, Wangchunshu Zhou, Yantao Du, Wenhao Huang, and Guang Shi. Edgebench: Unveiling scaling laws of learning from real-world environments. *CoRR*, abs/2607.05155, 2026.

A Appendix

We provide supplementary execution-resource statistics and settings, followed by the construction, grading, and feedback prompts.

A.1 Execution Resources and Settings

Token totals sum response-level usage records for Hermes and use cumulative session counters for Codex, including auxiliary calls where recorded. Input counts repeated context across calls without a cache discount; output uses the reported totals, with reasoning tokens included where provided. GPT–Hermes usage is recovered from response records because its exported summary omits the token fields. These statistics describe logged inference usage under each model's tokenizer, rather than billed cost or distinct context size.

Table 4: Execution time and recorded token usage for the main evaluation. P90 denotes the 90th percentile. Runtime excludes setup and grading; token counts describe cumulative agent-side model usage, with input in millions and output in thousands. DeepSeek denotes the DeepSeek-V4.1-Flash model.

Model	Harness	Runtime (min)			Mean tokens	
		Mean	P90	Max	Input (M)	Output (k)
Kimi-K3	Hermes	118.3	214.8	416.3	24.23	181.4
	Codex	157.9	297.6	706.1	20.00	199.2
DeepSeek	Hermes	113.7	192.8	295.9	20.84	207.9
	Codex	102.7	158.6	250.4	10.45	127.5
GLM-5.3	Hermes	148.3	260.3	538.0	37.00	320.7
	Codex	82.1	158.6	545.5	15.69	104.1
GPT-5.6-Sol	Hermes	19.4	37.1	104.2	5.28	36.9
	Codex	23.7	39.6	107.2	3.49	44.4
Qwen-3.8-Flash	Hermes	233.0	411.2	955.9	39.91	420.8
	Codex	366.7	687.3	903.5	55.82	465.1
MiniMax-M3	Hermes	38.3	65.1	155.4	25.16	131.8
	Codex	96.5	161.0	394.8	22.81	143.3

Execution configuration. The recorded versions are Codex CLI 0.146.1 and Hermes Agent 0.20.0, connected through model adapters. Both use a 272,000-token context setting and 90% auto-compaction setting. DeepSeek uses temperature 0 and top-k 1; the others use temperature 1 and top-k 50. Top-p is 0.95 for Kimi, GLM, and MiniMax and 1.0 otherwise. These settings describe the evaluated configurations, not a controlled decoding comparison.

B Construction and Evaluation Prompts

The following boxes organize the construction and evaluation prompts by inputs, procedure, and expected output. The construction and judge templates are condensed English presentations of project guidelines and stored instructions, with editorial headings added for readability. Task-specific materials, rubrics, and environment instructions are supplied at use time. The static-feedback box reproduces the deployed wording, with paragraph breaks added only for presentation.

B.1 Task Authoring

Task Authoring

Inputs

An expert's work request, workflow description, source materials, and intended use of the deliverable.

Task construction

1. Develop an executable benchmark task that preserves the request's professional purpose. Identify the user, decisions to be made, available evidence, applicable rules, and required outputs. Surface ambiguities or missing assumptions for the author to resolve. Preserve relationships among records during de-identification and identify simulated inputs.
2. Specify how intermediate results affect later work. For each essential dependency, explain which later selection, calculation, action, or conclusion would change if its upstream result changed. Include the materials needed to resolve these dependencies, while allowing scripting and batch processing.

Workflow focus

– **Deep research.** Connect source verification and conflict resolution to further investigation and synthesis.
– **Deliverable creation.** Connect entity-specific decisions to shared constraints and consistent outputs.
– **Application building.** Specify roles, operations, preconditions, and expected state transitions.

Expected output

A task specification, prepared inputs, an execution environment, and acceptance criteria for a separate grading package. Keep grading references inaccessible to the evaluated agent.

B.2 Grading Generation and Independent Review

Grading Generation

Inputs

The task specification and referenced materials, with answer keys or trusted source snapshots where available.

Check design

1. For each requirement, identify the deliverable or behavior to inspect, its acceptance condition, the verification method, and an error the check would detect.
2. Prefer executable checks for deterministic properties, file-based judgment for semantic requirements, and interactive assessment for behavior that requires operating the application.
3. Derive expected values from trusted inputs and allow valid alternative solutions. Check coverage of required outputs, substantive correctness, and consistency between sources and downstream results.

Expected output

Produce the grading configuration, executable checks, and necessary reference material.

Independent Grading Review

Inputs

The specification, inputs, grading configuration, checking code, references, and available validation evidence.

Review procedure

1. Inspect the files directly. Compare what each criterion claims to verify with what its implementation actually checks.
2. Identify incomplete or fabricated work that could pass, and valid work that could fail because of unstated requirements, fragile parsing, or overly strict matching.
3. Check for omitted requirements, duplicate penalties, and grading failures incorrectly attributed to the evaluated agent.

Expected output

A verdict (`BLOCK`, `WARN`, or `PASS`), issues tied to check identifiers or file locations, coverage gaps, and proposed repairs. Human authors adjudicate the findings and submit affected components for revalidation.

B.3 Independent Implementation and Refinement

Independent Implementation and Refinement

Stage 1: Independent implementation

Inputs. Only the task specification, inputs, and execution environment available to the evaluated agent.

For an application-building trial, develop, test, and deploy a solution using these materials. Preserve the resulting implementation and execution records for a separate grading check.

Stage 2: Evidence-based refinement

Inputs. The task package, trial trajectory, frozen deliverables, and check-level grading outcomes.

1. Determine whether required information affects downstream results and whether each verdict agrees with the actual artifact or system behavior.
2. Separate specification defects, missing inputs, environment failures, and grading errors from agent omissions or reasoning mistakes.
3. Propose changes only where the evidence supports a task or grading defect. Recheck affected components after revision; retain valid requirements even when the trial agent failed to satisfy them.

B.4 LLM Judge

LLM Judge

Inputs

The task instruction, submitted response, relevant file contents or attachments, and a list of rubrics with identifiers.

Evaluation procedure

1. Evaluate every rubric against the actual submitted content. Use only the stated criteria; do not add or omit requirements. Assess each relevant file or object separately.
2. Distinguish content requirements from formatting requirements, and accept equivalent representations when the rubric permits them. Additional correct, relevant information is acceptable unless it contradicts the task or violates an explicit restriction.
3. Verify numerical values and formula results when they determine the verdict.

Output contract

For each rubric, provide its original identifier, the criterion, a binary pass/fail decision, and a reason citing concrete evidence, such as a file, page, passage, or cell. Cover all rubrics without merging or renumbering them.

Return a JSON object with a `results` list and an `all_pass` field that is true only when every criterion passes.

B.5 Static Feedback

Static Feedback

Reread the task and the requirements documents it explicitly references, then check them against the actual current deliverables. Do not assume a requirement is satisfied merely because the files exist, a previous self-check report says it passed, or you have declared the task complete; inspect the relevant deliverable content directly.

If you find an omission, error, or inconsistency supported by a clear task requirement, correct only the affected parts, preserve content already verified as compliant, and submit again after making those corrections.

If your review finds no issue that requires correction, do not make changes merely to submit again; you may finish the task. Do not treat anything you could not verify as having passed.