GENERALIZATION BEYOND CODING: AN EMPIRICAL COMPARISON OF CODING AGENTS AND PURPOSE-BUILT AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in large language models (LLMs) have enabled both flexible coding agents and specialized, purpose-built agents to tackle complex tasks. General-purpose coding agents such as Codex show strong adaptability, while purpose-built agents rely on domain-specific structures to solve specialized problems. This prompts a key question: Are coding agents now advanced enough to make purpose-built engineering obsolete? This paper presents a comprehensive empirical comparison between Codex and comparable state-of-the-art purpose-built LLM agents across 20 non-coding benchmarks spanning finance, clinical reasoning, data analysis, information retrieval, and scientific research. For each benchmark, an AI-assisted procedure finds the comparable state-of-the-art purpose-built agent with released code, verifies its reported score, and re-runs it with the same language model as Codex; both agents are scored on the same instances by the same grader. Using the same underlying model and evaluation methods, Codex significantly outperforms on 5 benchmarks, shows no significant difference on 13, and underperforms on only 2. Many of the advantages previously attributed to purpose-built agents diminish when controlling for the backbone model and evaluation protocols. In certain cases, the additional complexity introduced by staged data processing even results in poorer performance compared to Codex's direct, end-to-end approach. Where purpose-built agents do win, much of their advantage is portable: their domain tools, handed to Codex, carry much of the benefit, and their decisive workflow steps encode procedural knowledge that Codex can execute but rarely finds on its own; once given them, it matches the purpose-built agents on most of the selected instances it had lost.

1 INTRODUCTION

Solving a non-coding task with an LLM now involves an engineering choice: build a purpose-built agent, or hand the task to an existing coding agent. The open problem is to know when a purpose-built agent's extra structure improves performance enough to justify the effort of building it, and when a general-purpose coding agent already suffices.

Purpose-built agents constructed atop Large Language Models (LLMs) have recently achieved remarkable success across a wide range of complex applications. Typically structured as modular pipelines, these agents decompose challenging workflows into sequential stages — such as retrieval, intermediate reasoning, planning, and verification — each powered by carefully designed LLM calls. This design paradigm leverages the adaptability and power of LLMs while preserving modularity: system builders can pinpoint and iteratively improve individual steps, target specific sources of error, and gain interpretability through transparent, stage-wise execution. As a result, LLM-based purpose-built agents have become the foundation for advanced solutions in domains from scientific literature review to complex decision support and data transformation (Pan et al., 2026).

Meanwhile, general-purpose coding agents have matured. Agents such as Codex from OpenAI, SWE-agent (Yang et al., 2024a), and OpenHands (Wang et al., 2025) solve tasks by editing files and running code in a workspace, and they are evaluated mainly on software engineering benchmarks such as SWE-bench (Jimenez et al., 2024), with a few general assistant benchmarks such

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.

as GAIA (Mialon et al., 2023). Writing and running code is also a general way to reason and act: programs can carry out the computation in math and financial questions (Gao et al., 2023; Chen et al., 2023), and executable code actions let an agent use tools more flexibly than fixed action formats (Wang et al., 2024). A coding agent might therefore handle many non-coding tasks without a task-specific pipeline.

This landscape raises a central question for both research and practice: How do these two paradigms compare when applied to difficult, real-world problems outside the domain of code? To address this, we focus on the following research questions:

RQ1: Rigorous Comparative Evaluation. How can a general-purpose coding agent be compared with purpose-built agents fairly and reproducibly, when benchmarks differ in their data, tools, metrics, and the conditions under which published scores were obtained?

RQ2: Comparative Performance. How does a general-purpose coding agent such as Codex perform across a broad suite of non-coding benchmarks, both against the purpose-built agents' published results and against the same agents run on a shared backbone?

RQ3: Task and Domain Dependence. Which characteristics of a task, such as its domain, task type, or complexity, are responsible for the performance differences between the two approaches?

RQ4: Strengths and Weaknesses of Purpose-Built Agents. Which components of a purpose-built agent, such as domain tools, intermediate representations, or verification steps, account for its successes, and where does decomposing the task lose information the answer depends on?

RQ5: Strengths and Weaknesses of Coding Agents. What lets a coding agent solve non-coding tasks without task-specific engineering, and at which stage (accessing the data, reasoning, or delivering the answer) do its failures arise?

To rigorously compare general-purpose coding agents and purpose-built agents in real-world, non-coding domains, we conducted an extensive empirical study spanning 20 challenging benchmarks across finance, clinical reasoning, data analysis, information retrieval, and scientific research. We define an *instance* as a single test case evaluated by a benchmark: typically a question, but in some cases a dataset (InsightBench), a task (EarthBench, Biomni-Eval1), or a search query (BRIGHT). Each benchmark typically consists of multiple instances; our head-to-head comparison covers 4,649 instances across the 20 benchmarks.

We created an AI-assisted meta-analysis pipeline enabling systematic, reproducible comparison by (1) Identifying, for each benchmark, the comparable state-of-the-art purpose-built agent (defined below) and verifying its reported results. (2) Translating heterogeneous benchmark protocols into unified, executable tasks, carefully aligning evaluation data, metrics, and model configurations. (3) Implementing both agent paradigms on a shared execution backbone, and capturing detailed execution traces to enable in-depth analysis of agent behaviors, decision processes, and outcomes.

Contributions

A reproducible, AI-assisted meta-analysis pipeline that enables systematic comparison by (1) Identifying, for each benchmark, the comparable state-of-the-art purpose-built agent and verifying its reported results; (2) Translating heterogeneous benchmark protocols into unified, executable tasks, carefully aligning evaluation data, metrics, and model configurations; and (3) Implementing both agent paradigms on a shared execution backbone, and capturing detailed execution traces to enable in-depth analysis of agent behaviors, decision processes, and outcomes.

An empirical comparison of Codex with purpose-built agents on 20 non-coding benchmarks. We report the results of a frontier general-purpose coding agent, Codex with `gpt-5.6-sol`, on 20 non-coding benchmarks, scored with each benchmark's own metric, and compare them with the purpose-built agents both as published (Figure 1) and on a shared backbone model, `gpt-5.6-sol`. Running head to head on the same backbone and instances, Codex significantly exceeds the purpose-built agent on 5 of the 20 benchmarks, shows no significant difference on 13, and falls significantly behind on only 2, EarthBench and OOLONG.

A detailed trace-level analysis of the performance differences with three headline findings: (1) fixed pipelines trade away information: purpose-built agents that filter or convert the input in fixed steps lose evidence that Codex keeps by having the full input available; (2) much of the purpose-

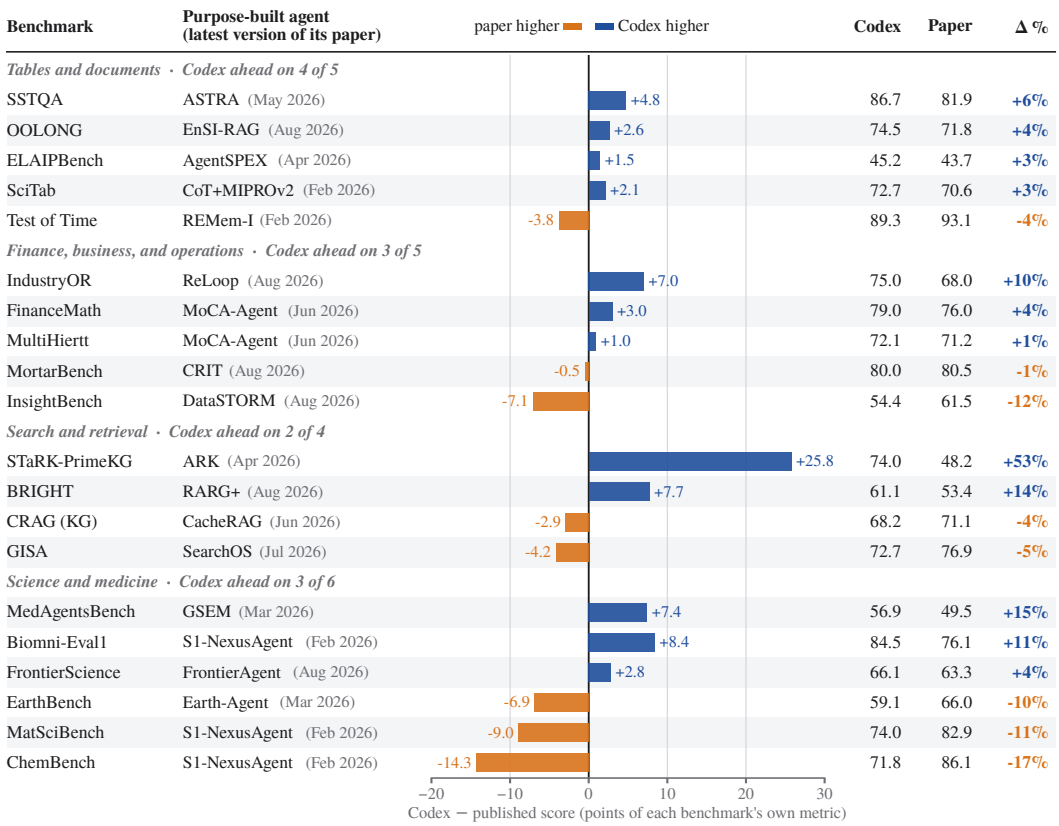

Figure 1: Comparison of Codex (`gpt-5.6-sol`, medium reasoning effort) against each benchmark's purpose-built SOTA agent as reported in its paper, grouped by benchmark type. Each row compares Codex's score against the purpose-built agent's published score with the date of the latest arXiv version of its paper. All numbers are reported in each benchmark's own metric. If a benchmark has several metrics or subsets, we use the same score and subset that the agent's paper reports and grade Codex on the same metric and subset. See Table 10 for details.

built agents' advantage is portable: their domain data and tools, handed to Codex as building blocks, carry much of the benefit without the fixed pipeline; and (3) their decisive workflow steps encode procedural knowledge rather than capability: Codex can execute these steps but rarely finds them on its own, and once given them it matches the purpose-built agents on most of the selected instances it had lost.

2 Related Work

Coding agents as general problem solvers. Code is an established medium for reasoning and acting: PAL and Program of Thoughts delegate computation to an interpreter (Gao et al., 2023; Chen et al., 2023), ReAct interleaves reasoning with actions (Yao et al., 2023), and CodeAct composes tools through executable Python actions (Wang et al., 2024). SWE-agent and OpenHands build general coding agents that navigate, edit, and run code in a workspace (Yang et al., 2024a; Wang et al., 2025). The study closest to ours, Bandel et al. (2026), runs Claude Code and four other general agents with five models on six agentic benchmarks (software engineering, customer service, deep research, and personal assistance) and finds the best general-agent configuration statistically indistinguishable from the top reported domain-specific agent on four of them. Its comparison has limits that our design removes: the domain-specific scores come from leaderboards, obtained with other models on the full benchmarks, while the general agents run on 100 sampled tasks, and its failure analysis covers only the general agents, so it cannot say where each design wins. We instead re-run each benchmark's comparable state-of-the-art purpose-built agent from its released code on Codex's

model and the same instances, across 20 benchmarks in scientific, medical, financial, retrieval, and tabular domains, and read the traces of both agents.

Purpose-built agents. Purpose-built agents add structure for a task: programmable multi-agent conversations (Wu et al., 2023), optimized language-model programs (Khattab et al., 2023), searched workflows (Zhang et al., 2025), expert discussion in medicine (Tang et al., 2024), or chemistry tools (Bran et al., 2023). The agents we compare against add, for example, behavioral verification to optimization modeling (Lian et al., 2026), claim decomposition and aggregation for numerical questions (Abdallah et al., 2026), and typed workflow steps (Wang et al., 2026b). Running them on Codex's model shows what this structure adds: in our results, steps fixed in advance can lose information that the answer needs, while domain tools and explicit checks help.

Benchmarks and evaluation beyond software. Agent benchmarks have moved from repository issues (Jimenez et al., 2024) to information gathering, tool use, and interactive environments (Mialon et al., 2023; Liu et al., 2025), data-driven scientific discovery (Chen et al., 2025), and machine-learning engineering (Chan et al., 2025). Shared frameworks standardize scenarios and scoring (Liang et al., 2023), and agent evaluation must also account for cost and reproducibility (Kapoor et al., 2024; 2025). Published comparisons remain fragile: they depend on released code (Pineau et al., 2020), on prompt format (Sclar et al., 2024), and on overfitting to a benchmark (Zhang et al., 2024). We therefore check each published score against its source and align the setting before comparing.

Workflow structure and execution traces. More structure does not guarantee better results. Agentless shows that a fixed localization–repair–validation workflow is competitive in software engineering (Xia et al., 2024), and controlled comparisons find that the benefit of coordinating agents depends on task structure and can be offset by overhead and error propagation (Kim et al., 2026). MAST sorts multi-agent failures into system design, inter-agent alignment, and verification (Cemri et al., 2025). We compare the traces of Codex and each purpose-built agent on the same instances to see where information is lost and which steps decide the answer.

3 Study design and evaluation pipeline

Published scores alone cannot tell us how a general-purpose coding agent compares with purpose-built agents, because each purpose-built agent reports its score on its own setup. We therefore built an AI-assisted pipeline that (1) selects benchmarks and verifies the purpose-built agents' published results, (2) turns each benchmark into an executable task under aligned conditions, and (3) runs both agents on the same model and instances and reads their execution traces.

3.1 Benchmark selection and verification of purpose-built agents

To study generalization beyond coding, we need non-coding benchmarks: tasks whose requested outcome is not code, such as a diagnosis, an optimization solution, retrieved evidence, or a scientific analysis, even if an agent writes code to reach it. From 562 candidate benchmarks with 14,215 published results, we shortlisted 87, favoring those that are not saturated; most of them have a published purpose-built agent to compare against. We grouped these by domain and, in choosing which to build into executable tasks with their own data and grader, generally favored the more recent or more widely cited benchmarks in each domain. Six more benchmarks, found through their purpose-built agents or added as replacements, bring the total built to 31. Building them showed that 11 lack a comparable state-of-the-art purpose-built agent, that is, one that can be reproduced from released code, can run with a different backbone model, and is evaluated under the benchmark's own setting: their agents' code, data, or trained state is unreleased; their published scores rest on a changed setting, an undisclosed subset, or a judge we cannot identify or access; or the agent does not fit our hardware. The remaining 20 are the benchmarks we compare on (Appendix A lists the reason for each exclusion).

3.2 Executable tasks and protocol alignment

Before we can compare two agents on a benchmark, both must run it under the same rules. We run every Codex trial in the open-source Harbor framework, which provides the container, the task format, the Codex adapter, and a network sidecar; on top of it we built a task adapter per benchmark that keeps gold answers out of the container and wraps the benchmark's own scorer, and a logging wrapper that runs the purpose-built agents under the same network rules (Appendix F).

To ensure that score differences reflect the agents rather than the setup, we write and audit a run specification for every benchmark, give Codex the data and, where the benchmark permits it, the network access that the benchmark gives every entrant, and leave each purpose-built agent's own indexes, caches, prompts, and tools in place. On EarthBench, the purpose-built agent uses its own analysis tools, and we analyze its results in light of this difference. To control answer leakage, we scan the traces of both agents for the benchmark's answers and re-run or remove every leaking instance (Section 4.2).

3.3 Comparison settings and scoring

Codex and purpose-built agents. Codex runs as Codex CLI on `gpt-5.6-sol` at medium reasoning effort, with one scored attempt per instance; it receives the task, the data and tools that the benchmark provides, and, where the benchmark allows it, the open web, but nothing from the purpose-built agent's workflow. We re-run each purpose-built agent from its released code on the same model and effort, with its own prompts, planners, and other components; ten of the 17 agents also use a model other than the shared backbone, such as a router or an embedding model (Table 8).

Two comparison settings. We compare Codex with each purpose-built agent in two settings: beside the score that the agent's paper reports on its own backbone and protocol, scored on that paper's metric and subset (Figure 1), and on Codex's backbone, with both agents scored on the same instances, the full set on five benchmarks and otherwise a seeded random sample, 25% of the benchmark in most cases (Table 1). The second holds the model and the instances fixed, so that a difference reflects the agents, and it gives us both agents' traces.

Scoring and cost. We score both agents with the metric of the purpose-built agent's paper, else its repository, else the benchmark's official metric (only CRAG and GISA depart from this rule), test each paired difference with an exact McNemar test or a sign-flip test for continuous metrics, and report 95% paired bootstrap intervals over clusters of instances that share a source. We price every recorded model call of both agents at list rates (Appendix I.1).

3.4 Paired trace analysis of agent trade-offs

Scores show which agent does better, but not why. To explain the differences, we used Claude Code (Opus 5.5) as annotators: they read the traces of both agents following a written codebook, and every label was confirmed by a human. For each requirement of a task, such as a table cell the answer depends on, the annotators check whether the agent saw it, used it, and carried it into the submitted answer. They label each instance that only one agent answers correctly with one primary mechanism (RQ4), and each failure of either agent with one of twelve failure patterns in five groups (RQ5); a second annotator independently labeled a stratified sample and a third adjudicated the disagreements (Appendix I.4).

4 Experiments

We first compare Codex with the purpose-built agents' published results and with the same agents re-run on Codex's backbone model (Section 4.1), and then answer RQ1–RQ5.

4.1 Published Results and Shared-Backbone Re-Runs

Figure 1 sets Codex beside the purpose-built agents' published results and gives a first view of how well a general-purpose coding agent carries over to these tasks. Without any benchmark-specific

Table 1: Codex and the purpose-built agent on the same backbone and instances. Scores are on a 0–100 scale in each benchmark's own metric; Δ is Codex minus the purpose-built agent with its 95% paired bootstrap interval, and n is the number of instances compared. **Bold**: $p < .05$; the sign of Δ shows which agent is better. Rows are grouped by the information each agent works from (Section 4.4). Appendix Table 11 adds each benchmark's metric, sample, per-instance counts, cost, and notes on individual rows.

Benchmark	Purpose-built agent	n	Codex	Purpose-built	Δ [95% CI]	p
Purpose-built agent converts or filters input that Codex reads in full						
SSTQA (zh) (Tang et al., 2025)	**ASTRA** (Guo et al., 2026)	**191**	**85.3**	**73.8**	**+11.5 [+4.7, +19.5]**	**<.001**
Test of Time (Fatemi et al., 2024)	**REMem-I** (Shu et al., 2026)	**280**	**89.3**	**82.5**	**+6.8 [+2.9, +11.0]**	**<.001**
MortarBench (Toles et al., 2026)	CRIT (Toles et al., 2026)	90	80.0	73.3	+6.7 [0.0, +13.0]	.15
STaRK-PrimeKG (Wu et al., 2024)	**ARK** (Polonuer et al., 2026)	**700**	**72.9**	**67.3**	**+5.6 [+2.7, +8.6]**	**<.001**
ELAIPBench (Dai et al., 2026)	AgentSPEX (Wang et al., 2026b)	403	45.2	45.7	−0.5 [−5.1, +4.0]	.91
Both agents have the same information						
FrontierScience (Wang et al., 2026a)	FrontierAgent (An et al., 2026)	15	73.3	60.0	+13.3 [0.0, +33.3]	.50
Biomni-Eval1 (Huang et al., 2025b)	**S1-NexusAgent** (NexusAgent Team, 2026)	**203**	**86.2**	**78.3**	**+7.9 [+3.0, +12.8]**	**.002**
CRAG (KG) (Yang et al., 2024b)	**CacheRAG** (Sun & Chen, 2026)	**578**	**68.2**	**63.1**	**+5.0 [+0.9, +9.3]**	**.02**
IndustryOR (Huang et al., 2025a)	ReLoop (Lian et al., 2026)	100	75.0	70.0	+5.0 [−1.0, +11.0]	.18
MultiHiertt (Zhao et al., 2022)	MoCA* (Abdallah et al., 2026)	261	77.4	75.9	+1.5 [−1.5, +4.6]	.45
MatSciBench (Zhang et al., 2026a)	S1-NexusAgent (NexusAgent Team, 2026)	246	78.9	77.6	+1.2 [−0.8, +4.9]	.65
MedAgentsBench (Shao et al., 2026)	GSEM (Han et al., 2026)	432	47.9	46.8	+1.2 [−1.9, +4.2]	.55
SciTab (Lu et al., 2023)	CoT+MIPROv2 (Du et al., 2026)	306	72.9	71.9	+1.0 [−2.3, +4.2]	.70
FinanceMath (Zhao et al., 2024)	MoCA* (Abdallah et al., 2026)	200	79.0	79.0	0.0 [−4.5, +4.5]	1.00
InsightBench (Sahu et al., 2025)	DataSTORM (Liu et al., 2026)	25	46.3	49.4	−3.0 [−7.4, +3.0]	.42
GISA (Zhu et al., 2026)	SearchOS (Zhang et al., 2026b)	50	60.8	66.9	−6.1 [−16.5, +3.5]	.24
OOLONG (Bertsch et al., 2025)	**EnSI-RAG** (Meng et al., 2026)	**79**	**74.6**	**82.3**	**−7.7 [−13.4, −2.4]**	**.03**
Purpose-built agent uses data or tools that Codex did not have						
BRIGHT (Su et al., 2025)	RARG (Li et al., 2026)	106	62.9	60.8	+2.1 [−1.6, +5.8]	.27
ChemBench (Mirza et al., 2024)	S1-NexusAgent (NexusAgent Team, 2026)	330	74.5	77.3	−2.7 [−5.8, +0.3]	.11
EarthBench (Feng et al., 2026)	**Earth-Agent** (Feng et al., 2026)	**54**	**61.1**	**81.5**	**−20.4 [−35.2, −5.6]**	**.02**

workflow, Codex scores at or above the published score on 12 of the 20 benchmarks. On each benchmark, the purpose-built agent is the comparable state of the art, and Codex runs on the instances that agent's paper reports (Section 3.1).

A gap to a published score does not isolate the design of the agent, because each published agent used its own backbone model and evaluation protocol. We therefore re-run each purpose-built agent from its released code with Codex's backbone, `gpt-5.6-sol`, on the same instances as Codex, and call a difference significant when the paired test gives $p < .05$ (Appendix I.1).

4.2 RQ1: Rigorous Comparative Evaluation

How can a general-purpose coding agent be compared with purpose-built agents fairly and reproducibly, when benchmarks differ in their data, tools, metrics, and the conditions under which published scores were obtained?

A shared backbone removes differences in the model, but a comparison can still mislead in three ways, and addressing each changed our results. Published setups often do not match the benchmark: we judged 22 of 25 published comparisons misaligned with their benchmark, for example in protocol, metric, or tools. Agents with web access can find leaked answers: on the 256 MatSciBench instances we sampled, Codex had read the instance's own worked solution on 7 of its 17 wins, so we scanned every trace of both agents and re-ran or removed every leaking instance. And the scorer can miss an answer given in an unexpected place or format: reading only the answer file produced a significant AgentSPEX lead on ELAIPBench, where the agents do not differ, and one added sentence on answer format raises Codex on Test of Time from 89.3 to 98.6, more than its lead over REMem-I; Table 1 keeps the original prompt (Appendix I.5).

Takeaway. Published scores alone cannot support a fair comparison. Beyond a shared model and shared instances, auditing the published setup against the benchmark, scanning every trace for leaked answers, and reading each agent's answer wherever it gives it each changed our results, so all three should be standard practice when agents are compared.

4.3 RQ2: Comparative Performance

How does a general-purpose coding agent such as Codex perform across a broad suite of non-coding benchmarks, both against the purpose-built agents' published results and against the same agents run on a shared backbone?

In Table 1, Codex significantly exceeds the purpose-built agent on 5 benchmarks (SSTQA, Test of Time, STaRK-PrimeKG, CRAG, and Biomni-Eval1), shows no significant difference on 13, and falls significantly behind on 2 (EarthBench and OOLONG).

Sharing the backbone changes the purpose-built agents' own scores, by -10.6 to $+19.1$ points on the 6 benchmarks where our re-run covers the published setup, often by more than the gap to Codex. As a result, few published gaps survive: only 4 of the 20 keep a significant difference in the same direction, and Codex trails the published scores of REMem-I on Test of Time and CacheRAG on CRAG but beats both agents on the same model and instances. The purpose-built agents are also usually more expensive, with a higher model cost on 13 of the 19 benchmarks where we could measure it (Appendix I.5).

Takeaway. On the same model and instances, a general-purpose coding agent significantly exceeds purpose-built agents on 5 of 20 benchmarks, shows no significant difference on 13, and falls significantly behind on only 2. Only 4 published gaps keep a significant difference in the same direction once the model and instances are shared, so a published gap is a poor guide to what a purpose-built design adds.

4.4 RQ3: Task and Domain Dependence

Which characteristics of a task, such as its domain, task type, or complexity, are responsible for the performance differences between the two approaches?

The purpose-built agent is significantly better on two benchmarks: EarthBench, where it brought its own analysis tools, and OOLONG, where it labels each record and counts the labels exactly. Within a domain the winner varies, and features of an instance, such as a numeric answer or the length of the input, predict the winner only weakly.

After reading the traces, we sorted the benchmarks into three groups by where the needed information resides (Table 1). Where the purpose-built agent converts, extracts, or filters an input that Codex reads in full (five benchmarks), Codex answers 165 instances that the purpose-built agent misses, against 81. Where both have the same information (twelve), the pooled counts favor Codex, 175 to 130 ($p = .01$). Where the purpose-built agent uses a service, index, or tool of its own (three), they lean to the purpose-built agent without a significant difference, 64 to 48 ($p = .16$; Appendix I.5).

Takeaway. Purpose-built agents lead significantly on only two benchmarks: EarthBench, where they bring their own domain tools, and OOLONG, where they count record by record. Codex leads mainly when a purpose-built agent converts or filters input that Codex can read in full, and when both agents have the same information it is ahead in the pooled counts, significantly so on Biomni-Eval1 and CRAG. A coding agent is a reasonable default when the information can be handed to it as files, tables, documents, or graphs.

4.5 RQ4: Strengths and Weaknesses of Purpose-Built Agents

Which components of a purpose-built agent, such as domain tools, intermediate representations, or verification steps, account for its successes, and where does decomposing the task lose information the answer depends on?

Pooled over the 20 benchmarks, the purpose-built agent answers 275 instances correctly where Codex does not (Figure 2). On 49 it used its own data or tools, mostly tools such as RARG's document-ranking model; on 66 it took an explicit reasoning step on information both agents had; on 103 both had the same information and reached different answers; and 57 trace to a format or setup issue on Codex's side or a disputable reference answer.

Much of the purpose-built agents' advantage carries over to Codex. We re-ran Codex on 106 instances that the purpose-built agent had won on five benchmarks, with the resource that agent re-

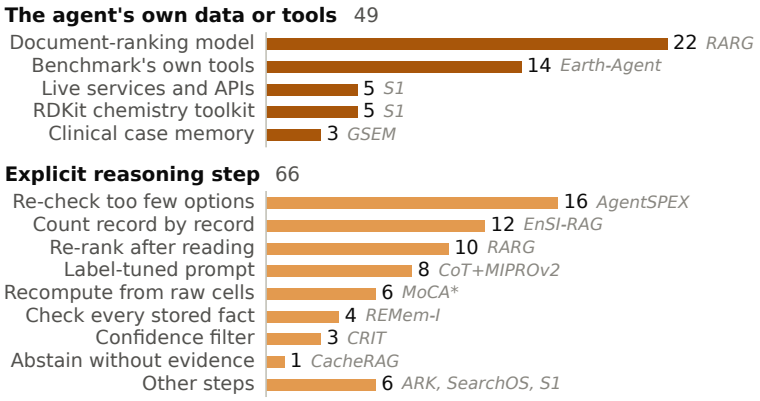

Figure 2: **What purpose-built agents had on the instances they won.** Of the 275 instances that only the purpose-built agent answered correctly, the bars show the 115 whose main reason is data or tools that only the purpose-built agent used, or an explicit reasoning step, with the agent after each count (S1 is S1-NexusAgent). The counts record what the agent had or did, not how much of the win it caused (Appendix I.3).

lies on installed in Codex's environment, such as a chemistry toolkit, a document-ranking model, or graph-query tools, and 61 of the 106 flipped to the purpose-built agent's result; since Codex's answers also vary between runs, a flip does not by itself show that the resource produced it (Appendix I.3).

The explicit steps encode procedural knowledge rather than capability, such as AgentSPEX's re-check on multi-answer questions (16 wins) and EnSI-RAG's record-by-record counting (12 wins). Codex had the data and tools to take the same step in 56 of these 66 instances, but its traces show the step in only 13; given a purpose-built agent's step prompts and procedure verbatim, it reached that agent's result on 22 of 27 selected instances that the agent had won (Appendix I.3).

The purpose-built agents also lose information: on 174 of the 388 instances that only Codex answers correctly (45%), the purpose-built agent had lost the answer inside its own pipeline, in a conversion step, a vote or filter, or a step limit. The tree that ASTRA's converter built from one SSTQA spreadsheet, for example, covered only 19% of the table.

Takeaway. Where purpose-built agents win through domain tools or explicit workflow steps, much of the advantage carries over to Codex, which can execute these steps but rarely finds them on its own. Their main weakness is that extraction and filtering steps fixed in advance can drop the information the answer depends on, so this loss of context should be analyzed whenever such a pipeline is designed.

4.6 RQ5: Strengths and Weaknesses of Coding Agents

What lets a coding agent solve non-coding tasks without task-specific engineering, and at which stage (accessing the data, reasoning, or delivering the answer) do its failures arise?

Codex's main advantage is that it works on the complete, original input and decides how to process it after reading each question, rather than following a representation fixed in advance; it wrote code to open every SSTQA workbook itself, answered almost every Test of Time question from the full list of facts, and followed the links of the STaRK-PrimeKG knowledge graph itself on most questions.

Both agents fail mostly at reasoning (72% of Codex's failures and 59% of the purpose-built agents'; Figure 3), and these errors overlap: when Codex is wrong, the purpose-built agent is also wrong on 83% of instances. The purpose-built agents more often lost information they already had (25% of their failures, against 12%). Few of Codex's failures come from information it could not access, and it rarely re-checks its work, revising its answer in only 1.4% of its runs (Appendix I.5).

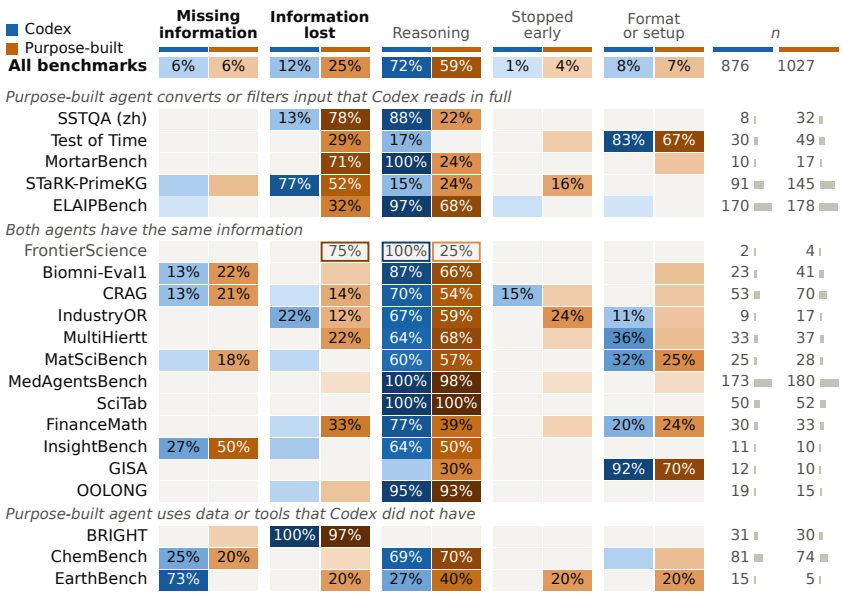

Figure 3: **Where each agent fails.** Each cell is the share of one agent's failures on a benchmark that fall into a group; Codex (blue) is on the left of each pair and the purpose-built agent (orange) on the right. The top row pools all 20 benchmarks; other rows print shares of at least 10%, and n is the number of failures (Appendix I.5 defines the groups).

Takeaway. Codex's main strength is working directly on the full input and deciding how to process it after seeing each question. Its failures are mostly reasoning errors that purpose-built agents share. Its own weaknesses are that it rarely re-checks its answers, finds a decisive workflow step, or reaches for the right domain tool on its own; given the purpose-built agents' tools and steps, it recovered many of the selected instances it had lost.

5 Conclusion

This paper assesses the generalization abilities of general-purpose coding agents by comparing Codex to a state-of-the-art purpose-built agent across 20 benchmarks. Using the same underlying model and evaluation methods, Codex significantly outperforms on 5 benchmarks, shows no significant difference on 13, and underperforms on only 2, EarthBench and OOLONG. Where the purpose-built agents excel, they bring specialized analysis tools or explicit workflow steps, and much of this advantage is portable: handed these tools or steps, Codex matches the purpose-built agents on most of the selected instances it had lost. Many of the advantages previously attributed to purpose-built agents diminish when controlling for the backbone model and evaluation protocols. In certain cases, the additional complexity introduced by staged data processing even results in poorer performance compared to Codex's direct, end-to-end approach.

In conclusion, general-purpose coding agents have progressed to the point that specialized solutions are needed mainly where a task depends on domain tools, or on procedures that a coding agent rarely finds on its own. These findings point to promising avenues for further improvement to coding agents: handing them the purpose-built agents' domain tools, and the procedural knowledge encoded in their decisive workflow steps, as reusable building blocks.

AI use statement

Generative AI tools assisted benchmark screening, source-result extraction, adapter and scorer implementation, alignment audits, and manuscript preparation. Quantitative claims derive from recorded evaluation artifacts and analysis scripts. AI-assisted audit findings require independent re-

view; agreement between model-based reviewers is not a measure of audit accuracy. The authors are responsible for the final claims and artifacts.

Ethics statement

The clinical and financial tasks are benchmark evaluations, not decisions about real patients or financial deployment. Measured task accuracy does not establish clinical safety or fitness for consequential use. The study uses existing benchmark resources under their access conditions.

Reproducibility statement

The recorded tasks specify inputs, evaluators, dataset versions, model settings, and departures from the published purpose-built agents. The paired per-instance results of both agents, the trace labels, and the analysis scripts reproduce every table and figure without additional model calls. Detailed operating conditions appear in the appendix.

References

Abdelrahman Abdallah, AbdelRahim A. Elmadany, Sameh Al Natour, Hasan Cavusoglu, Adam Jatowt, and Muhammad Abdul-Mageed. MoCA-Agent: A Market-of-Claims Code Agent for Financial and Numerical Reasoning, 2026. URL https://arxiv.org/abs/2606.11537.

B. An, B. Li, B. Wang, B. Zhang, B. L. Wang, C. Feng, C. Wei, C. Xue, C. Zhang, D. Ng, D. Ye, E. Min, F. Chen, F. Liu, F. Yang, F. Ye, G. Sun, H. Ji, H. Xu, H. Yang, H. Ye, H. Zhang, H. Zhao, J. Li, J. Lin, J. Xia, K. Jin, K. Wang, K. Yang, L. Bing, L. Lei, L. Su, Le. Wang, Lu. Wang, N. Wang, Q. Ren, Q. Yang, R. Li, S. Bai, S. Du, S. Li, S. Lin, S. Nie, S. Wang, S. Zhang, S. Z. Wang, T. Ge, Ta. Q. Fang, Ti. Q. Fang, W. Fang, W. Li, W. Zhang, X. Chen, X. Li, X. Tang, X. Wang, X. Xu, X. Zhang, X. Q. Wang, X. Y. Wang, Y. Deng, Y. Gao, Y. Hu, Y. Li, Y. Sui, Y. Wang, Y. Xiao, Y. Zhang, Y. Zhou, Z. Chen, Z. Cheng, Z. Feng, Z. Liang, Z. Liu, and Z. Zhang. Apodex 1.1: Scaling Agentic Intelligence for Complex Work, 2026. URL https://arxiv.org/abs/2608.23283.

Elron Bandel, Asaf Yehudai, Lilach Eden, Yehoshua Sagron, Yotam Perlitz, Elad Venezian, Natalia Razinkov, Natan Ergas, Shlomit Shachor Ifergan, Segev Shlomov, Michal Jacovi, Leshem Choshen, Liat Ein-Dor, Yoav Katz, and Michal Shmueli-Scheuer. General Agent Evaluation, 2026. URL https://arxiv.org/abs/2602.22953.

Amanda Bertsch, Adithya Pratapa, Teruko Mitamura, Graham Neubig, and Matthew R. Gormley. Oolong: Evaluating Long Context Reasoning and Aggregation Capabilities, 2025. URL https://arxiv.org/abs/2511.02817.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. ChemCrow: Augmenting large-language models with chemistry tools, 2023. URL https://arxiv.org/abs/2304.05376.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why Do Multi-Agent LLM Systems Fail?, 2025. URL https://arxiv.org/abs/2503.13657.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering, 2025. URL https://arxiv.org/abs/2410.07095.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks, 2023. URL https://arxiv.org/abs/2211.12588.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. ScienceAgentBench: Toward Rigorous Assessment of Language Agents for Data-Driven Scientific Discovery, 2025. URL https://arxiv.org/abs/2410.05080.

Alfred Clemedtson and Borun Shi. GraphRAFT: Retrieval Augmented Fine-Tuning for Knowledge Graphs on Graph Databases, 2025. URL https://arxiv.org/abs/2504.05478.

Xinbang Dai, Huikang Hu, Yongrui Chen, Jiaqi Li, Rihui Jin, Yuyang Zhang, Xiaoguang Li, Lifeng Shang, and Guilin Qi. ELAIPBench: A Benchmark for Expert-Level Artificial Intelligence Paper Understanding, 2026. URL https://arxiv.org/abs/2510.10549.

Xiaotang Du, Giwon Hong, Wai-Chung Kwan, Rohit Saxena, Ivan Titov, Pasquale Minervini, and Emily Allaway. Analyzing LLM Instruction Optimization for Tabular Fact Verification, 2026. URL https://arxiv.org/abs/2602.17937.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. Test of Time: A Benchmark for Evaluating LLMs on Temporal Reasoning, 2024. URL https://arxiv.org/abs/2406.09170.

Peilin Feng, Zhutao Lv, Junyan Ye, Xiaolei Wang, Xinjie Huo, Jinhua Yu, Wanghan Xu, Wenlong Zhang, Lei Bai, Conghui He, and Weijia Li. Earth-Agent: Unlocking the Full Landscape of Earth Observation with Agents, 2026. URL https://arxiv.org/abs/2509.23141.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided Language Models, 2023. URL https://arxiv.org/abs/2211.10435.

Xiaoke Guo, Songze Li, Zhiqiang Liu, Zhaoyan Gong, Yuanxiang Liu, Huajun Chen, and Wen Zhang. ASTRA: Adaptive Semantic Tree Reasoning Architecture for Complex Table Question Answering, 2026. URL https://arxiv.org/abs/2604.08999.

Xiao Han, Yuzheng Fan, Sendong Zhao, Haochun Wang, and Bing Qin. GSEM: Graph-based Self-Evolving Memory for Experience Augmented Clinical Reasoning, 2026. URL https://arxiv.org/abs/2603.22096.

Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. ORLM: A Customizable Framework in Training Large Models for Automated Optimization Modeling. *Operations Research*, 73(6):2986–3009, November 2025a. ISSN 1526-5463. doi: 10.1287/opre.2024.1233. URL http://dx.doi.org/10.1287/opre.2024.1233.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, Di Yin, Shruti Marwaha, Jennefer N. Carter, Xin Zhou, Matthew Wheeler, Jonathan A. Bernstein, Mengdi Wang, Peng He, Jingtian Zhou, Michael Snyder, Le Cong, Aviv Regev, and Jure Leskovec. Biomni: A General-Purpose Biomedical AI Agent. *bioRxiv*, 2025b. doi: 10.1101/2025.05.30.656746. URL https://www.biorxiv.org/content/early/2025/06/02/2025.05.30.656746.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?, 2024. URL https://arxiv.org/abs/2310.06770.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI Agents That Matter, 2024. URL https://arxiv.org/abs/2407.01502.

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, Franck Ndzomga, Dheeraj Oruganty, Sophie Luskin, Kangheng Liu, Botao Yu, Amit Arora, Dongyoon Hahm, Harsh Trivedi, Huan Sun, Juyong Lee, Tengjun Jin, Yifan Mai, Yifei Zhou, Yuxuan Zhu, Rishi Bommasani, Daniel Kang, Dawn Song, Peter Henderson, Yu Su, Percy Liang, and Arvind Narayanan. Holistic Agent Leaderboard: The Missing Infrastructure for AI Agent Evaluation, 2025. URL https://arxiv.org/abs/2510.11977.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines, 2023. URL https://arxiv.org/abs/2310.03714.

Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall, A. Ali Heydari, Yao Yan, Zhihan Zhang, Yuchen Zhuang, Yun Liu, Mark Malhotra, Paul Pu Liang, Hae Won Park, Yuzhe Yang, Xuhai Xu, Yilun Du, Shwetak Patel, Tim Althoff, Daniel McDuff, and Xin Liu. Towards a Science of Scaling Agent Systems, 2026. URL https://arxiv.org/abs/2512.08296.

Rodney Kinney, Chloe Anastasiades, Russell Authur, Iz Beltagy, Jonathan Bragg, Alexandra Buraczynski, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Arman Cohan, Miles Crawford, Doug Downey, Jason Dunkelberger, Oren Etzioni, Rob Evans, Sergey Feldman, Joseph Gorney, David Graham, Fangzhou Hu, Regan Huff, Daniel King, Sebastian Kohlmeier, Bailey Kuehl, Michael Langan, Daniel Lin, Haokun Liu, Kyle Lo, Jaron Lochner, Kelsey MacMillan, Tyler Murray, Chris Newell, Smita Rao, Shaurya Rohatgi, Paul Sayre, Zejiang Shen, Amanpreet Singh, Luca Soldaini, Shivashankar Subramanian, Amber Tanaka, Alex D. Wade, Linda Wagner, Lucy Lu Wang, Chris Wilhelm, Caroline Wu, Jiangjiang Yang, Angele Zamarron, Madeleine Van Zuylen, and Daniel S. Weld. The Semantic Scholar Open Data Platform, 2025. URL https://arxiv.org/abs/2301.10140.

Jiangnan Li, Yuqing Li, Mo Yu, Jinchao Zhang, and Jie Zhou. A New Role for Relevance: Guiding Corpus Interaction in Agentic Search, 2026. URL https://arxiv.org/abs/2607.24223.

Junbo Jacob Lian, Yujun Sun, Huiling Chen, Chaoyu Zhang, Hanzhang Qin, and Chung-Piaw Teo. ReLoop: Structured Modeling and Behavioral Verification for Reliable LLM-Based Optimization, 2026. URL https://arxiv.org/abs/2602.15983.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic Evaluation of Language Models, 2023. URL https://arxiv.org/abs/2211.09110.

Shicheng Liu, Yucheng Jiang, Sajid Farook, Camila Nicollier Sanchez, David Fernando Castro Pena, and Monica S. Lam. DataSTORM: Deep Research on Large-Scale Databases using Exploratory Data Analysis and Data Storytelling, 2026. URL https://arxiv.org/abs/2604.06474.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents, 2025. URL https://arxiv.org/abs/2308.03688.

Xinyuan Lu, Liangming Pan, Qian Liu, Preslav Nakov, and Min-Yen Kan. SCITAB: A Challenging Benchmark for Compositional Reasoning and Claim Verification on Scientific Tables, 2023. URL https://arxiv.org/abs/2305.13186.

Xuanyu Meng, Jiashuo Sun, Jash Rajesh Parekh, and Jiawei Han. EnSI-RAG: Entity-Structure-Indexed Retrieval-Augmented Generation for Long-Document Question Answering, 2026. URL https://arxiv.org/abs/2608.21252.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for General AI Assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Adrian Mirza, Nawaf Alampara, Sreekanth Kunchapu, Martiño Ríos-García, Benedict Emoekabu, Aswanth Krishnan, Tanya Gupta, Mara Schilling-Wilhelmi, Macjonathan Okereke, Anagha Aneesh, Amir Mohammad Elahi, Mehrdad Asgari, Juliane Eberhardt, Hani M. Elbeheiry, María Victoria Gil, Maximilian Greiner, Caroline T. Holick, Christina Glaubitz, Tim Hoffmann, Abdelrahman Ibrahim, Lea C. Klepsch, Yannik Köster, Fabian Alexander Kreth, Jakob Meyer, Santiago Miret, Jan Matthias Peschel, Michael Ringleb, Nicole Roesner, Johanna Schreiber, Ulrich S. Schubert, Leanne M. Stafast, Dinga Wonanke, Michael Pieler, Philippe Schwaller, and Kevin Maik Jablonka. Are large language models superhuman chemists?, 2024. URL https://arxiv.org/abs/2404.01475.

NexusAgent Team. S1-NexusAgent: a Self-Evolving Agent Framework for Multidisciplinary Scientific Research, 2026. URL https://arxiv.org/abs/2602.01550.

Melissa Z. Pan, Negar Arabzadeh, Riccardo Cogo, Yuxuan Zhu, Alexander Xiong, Lakshya A Agrawal, Huanzhi Mao, Emma Shen, Sid Pallerla, Liana Patel, Shu Liu, Tianneng Shi, Xiaoyuan Liu, Jared Quincy Davis, Emmanuele Lacavalla, Alessandro Basile, Shuyi Yang, Paul Castro, Daniel Kang, Koushik Sen, Dawn Song, Joseph E. Gonzalez, Ion Stoica, Matei Zaharia, and Marquita Ellis. Measuring Agents in Production, 2026. URL https://arxiv.org/abs/2512.04123.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Hugo Larochelle. Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program), 2020. URL https://arxiv.org/abs/2003.12206.

Joaquín Polonuer, Lucas Vittor, Iñaki Arango, Ayush Noori, David A. Clifton, Luciano Del Corro, and Marinka Zitnik. Autonomous Knowledge Graph Exploration with Adaptive Breadth-Depth Retrieval, 2026. URL https://arxiv.org/abs/2601.13969.

Jason Priem, Heather Piwowar, and Richard Orr. OpenAlex: A fully-open index of scholarly works, authors, venues, institutions, and concepts, 2022. URL https://arxiv.org/abs/2205.01833.

Rudra Ranajee Saha, Laks V. S. Lakshmanan, and Raymond T. Ng. A Multi-Agent Approach for Claim Verification from Tabular Data Documents, 2026. URL https://arxiv.org/abs/2604.17225.

Gaurav Sahu, Abhay Puri, Juan Rodriguez, Amirhossein Abaskohi, Mohammad Chegini, Alexandre Drouin, Perouz Taslakian, Valentina Zantedeschi, Alexandre Lacoste, David Vazquez, Nicolas Chapados, Christopher Pal, Sai Rajeswar Mudumba, and Issam Hadj Laradji. InsightBench: Evaluating Business Analytics Agents Through Multi-Step Insight Generation, 2025. URL https://arxiv.org/abs/2407.06423.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting, 2024. URL https://arxiv.org/abs/2310.11324.

Yanjun Shao, Xiangru Tang, Jiwoong Sohn, Jiapeng Chen, Yuxuan Liao, Jiayi Zhang, Jinyu Xiang, Fang Wu, Yilun Zhao, Chenglin Wu, Wenqi Shi, Arman Cohan, and Mark Gerstein. MedicalAgentsBench for Complex Medical Reasoning: Comparing Internalized Reasoning Models versus Externalized Agent-based Frameworks, 2026. URL https://arxiv.org/abs/2503.07459.

Yiheng Shu, Saisri Padmaja Jonnalagedda, Xiang Gao, Bernal Jiménez Gutiérrez, Weijian Qi, Kamalika Das, Huan Sun, and Yu Su. REMem: Reasoning with Episodic Memory in Language Agent, 2026. URL https://arxiv.org/abs/2602.13530.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. BRIGHT: A Realistic and Challenging Benchmark for Reasoning-Intensive Retrieval, 2025. URL https://arxiv.org/abs/2407.12883.

Yushi Sun and Lei Chen. CacheRAG: A Semantic Caching System for Retrieval-Augmented Generation in Knowledge Graph Question Answering, 2026. URL https://arxiv.org/abs/2604.26176.

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. MedAgents: Large Language Models as Collaborators for Zero-shot Medical Reasoning, 2024. URL https://arxiv.org/abs/2311.10537.

Zirui Tang, Boyu Niu, Xuanhe Zhou, Boxiu Li, Wei Zhou, Jiannan Wang, Guoliang Li, Xinyi Zhang, and Fan Wu. ST-Raptor: LLM-Powered Semi-Structured Table Question Answering, 2025. URL https://arxiv.org/abs/2508.18190.

Yicheng Tao, Yiqun Wang, Xiangchen Song, Xin Luo, Kai Liu, and Jie Liu. GRASP: Plan-Guided Graph Retrieval with Adaptive Fusion and Reranking on Semi-Structured Knowledge Bases, 2026. URL https://arxiv.org/abs/2605.30237.

Matthew Toles, Yunan Lu, Manav Munjal, Bojun Liu, Yuanhao Deng, Stephanie Selig, Derek Rindner, Cheng Li, and Zhou Yu. MortarBench: Evaluating Mortgage Loan Origination Agents, 2026. URL https://arxiv.org/abs/2606.19416.

Miles Wang, Robi Lin, Kat Hu, Joy Jiao, Neil Chowdhury, Ethan Chang, and Tejal Patwardhan. FrontierScience: Evaluating AI's Ability to Perform Expert-Level Scientific Tasks, 2026a. URL https://arxiv.org/abs/2601.21165.

Pengcheng Wang, Jerry Huang, Jiarui Yao, Rui Pan, Peizhi Niu, Yaowenqi Liu, Ruida Wang, Renhao Lu, Yuwei Guo, and Tong Zhang. AgentSPEX: An Agent SPecification and EXecution Language, 2026b. URL https://arxiv.org/abs/2604.13346.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable Code Actions Elicit Better LLM Agents, 2024. URL https://arxiv.org/abs/2402.01030.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An Open Platform for AI Software Developers as Generalist Agents, 2025. URL https://arxiv.org/abs/2407.16741.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation, 2023. URL https://arxiv.org/abs/2308.08155.

Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N. Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. STaRK: Benchmarking LLM Retrieval on Textual and Relational Knowledge Bases, 2024. URL https://arxiv.org/abs/2404.13207.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying LLM-based Software Engineering Agents, 2024. URL https://arxiv.org/abs/2407.01489.

Xin Xiao, Jiang Zhong, Junnan Zhu, Yingchao Feng, Peijin Wang, Yidan Zhang, and Kaiwen Wei. GeoForge: Non-Parametric Self-Evolving Agents for Earth-Observation Reasoning, 2026. URL https://arxiv.org/abs/2608.10494.

Guangrui Xie. ORPilot: A Production-Oriented Agentic LLM-for-OR Tool for Optimization Modeling, 2026. URL https://arxiv.org/abs/2605.02728.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering, 2024a. URL https://arxiv.org/abs/2405.15793.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen-tau Yih, and Xin Luna Dong. CRAG – Comprehensive RAG Benchmark, 2024b. URL https://arxiv.org/abs/2406.04744.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, 2023. URL https://arxiv.org/abs/2210.03629.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Charlotte Zhuang, Dylan Slack, Qin Lyu, Sean Hendryx, Russell Kaplan, Michele Lunati, and Summer Yue. A Careful Examination of Large Language Model Performance on Grade School Arithmetic, 2024. URL https://arxiv.org/abs/2405.00332.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating Agentic Workflow Generation, 2025. URL https://arxiv.org/abs/2410.10762.

Junkai Zhang, Jingru Gan, Xiaoxuan Wang, Zian Jia, Changquan Gu, Jianpeng Chen, Yanqiao Zhu, Mingyu Derek Ma, Dawei Zhou, Ling Li, and Wei Wang. MatSciBench: Benchmarking the Reasoning Ability of Large Language Models in Materials Science, 2026a. URL https://arxiv.org/abs/2510.12171.

Yuyao Zhang, Junjie Gao, Zhengxian Wu, Jiaming Fan, Jin Zhang, Shihan Ma, Yao Yao, Weiran Qi, Chuyan Jin, Guiyu Ma, Xingzhong Xu, Kai Yang, Ji-Rong Wen, and Zhicheng Dou. SearchOS-V1: Towards Robust Open-Domain Information-Seeking Agent Collaboration, 2026b. URL https://arxiv.org/abs/2607.15257.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. MultiHiertt: Numerical Reasoning over Multi Hierarchical Tabular and Textual Data, 2022. URL https://arxiv.org/abs/2206.01347.

Yilun Zhao, Hongjun Liu, Yitao Long, Rui Zhang, Chen Zhao, and Arman Cohan. FinanceMath: Knowledge-Intensive Math Reasoning in Finance Domains, 2024. URL https://arxiv.org/abs/2311.09797.

Yutao Zhu, Xingshuo Zhang, Maosen Zhang, Jiajie Jin, Liancheng Zhang, Xiaoshuai Song, Kangzhi Zhao, Wencong Zeng, Ruiming Tang, Han Li, Ji-Rong Wen, and Zhicheng Dou. GISA: A Benchmark for General Information-Seeking Assistant, 2026. URL https://arxiv.org/abs/2602.08543.

A Detailed benchmark selection and SOTA identification

This appendix details how we selected the benchmarks and identified the comparable state of the art summarized in Section 3.1. We will release the evaluation code and task definitions.

Selection stages. Table 2 gives the counts. The audit corpus holds 562 candidate benchmarks with 14,215 extracted result records: 556 from our automated collection and six added by our five reviewers. The reviewers selected 68 benchmarks after reading their extracted results, together with the purpose-built agents to compare against, and proposed 19 more in comments. The resulting shortlist of 87 holds 115 pairs of a benchmark and a purpose-built agent; eight of its benchmarks had none yet. We built 31 benchmarks as executable tasks: an adapter over the released data, the benchmark's own grader, and a run specification (Appendix D). We admitted a task only after its grader gave the gold answer full credit and a do-nothing agent the benchmark's null floor; ArXivLean, FigureBench, and WritingBench ship no gold answer, CritPt's gold is held by its grading service, and PPTEval's judge gives the reference deck 0.72, so these five were admitted on an evaluator self-test instead. Twenty-five of the 31 come from the shortlist. Six were added outside it: Biomni-Eval1 and MatSciBench,

Stage	n	What admits a benchmark
Audit corpus	562	Extracted published results exist (14,215 records); 556 from the automated collection, 6 added by the reviewers
Reviewer shortlist	87	Selected by the five reviewers after reading its results (68) or proposed in their comments (19); 115 pairs of a benchmark and a purpose-built agent, 8 benchmarks without one
Executable tasks	31	Released data, the benchmark's own grader and a run specification; gold scores correct (an evaluator self-test for five without a usable gold), a do-nothing agent scores the null floor. 25 from the shortlist, 6 added
Paired suite	20	The purpose-built agent runs from its released code on the shared backbone over Codex's instances
Not paired	2	RareBench, LLM-SRBench: purpose-built agent cannot be run as published
Supplementary	1	LongBench v2: purpose-built agent's code for it is unreleased
Excluded	8	Table 3

Table 2: Benchmark selection. Stages are not strictly nested: six executable tasks entered outside the shortlist through their purpose-built agents or as replacements. A paired run is not a reproduction of the published score.

Benchmark	Status	Reason
RareBench	not paired	DeepRare released 1 of 6 case banks (14.8% of cases)
LLM-SRBench	not paired	FunctionEvolve stops on a threshold calibrated on the ground-truth equation, not described in its paper
LongBench v2	supplementary	DeLM released no LongBench v2 loader, runner or scorer
HiTab	replaced by SSTQA	ASTRA's HiTab score is GPT-5-judged accuracy on 1,412 of the 1,584 test questions, not HiTab's execution accuracy
CritPt	excluded	Purpose-built agent evolved on all 70 test problems with official-grader feedback; evolved state unreleased
WritingBench	excluded	The purpose-built agent's judge cannot be identified
PPTEval	excluded	Purpose-built agent's score has no stated evaluation set; pinned judge unavailable to us
FigureBench	excluded	Pinned judge unavailable; published denominators inconsistent
RareBench-HMS	excluded	Published bar on 86 of 88 cases; same case-bank gap as RareBench
S2-Bench	excluded	Purpose-built agent's two local chemistry models do not fit on our single GPU alongside the other agents' local models
ArXivLean	excluded	The candidate purpose-built agent's published score is on PutnamBench; the leading ArXivLean system (AlephProver) is not publicly available

Table 3: The eleven executable benchmarks outside the paired suite.

which S1-NexusAgent, our purpose-built agent for ChemBench, also reports; InsightBench, with DataSTORM; SSTQA, which replaced HiTab under the same purpose-built agent (ASTRA); ArXivLean from the PutnamBench entry; and MortarBench from a later search for purpose-built agents with released code and a fixed-backbone baseline. For 20 of the 31 we could pair Codex with the purpose-built agent on the same instances by running its released code on the shared backbone; these 20 form the paired suite (DataSTORM's re-run scores below its published `gpt-5` predictions on InsightBench, so the comparison with those predictions is kept as a sensitivity check, Appendix I). Running a released purpose-built agent on our backbone is not a reproduction of its published score. Some runs need more than a change of backbone: MoCA, on FinanceMath and MultiHiertt, runs without its unreleased baseline proposer (MoCA*); BRIGHT, CRAG, and InsightBench substitute a component that could not be obtained; and on MedAgentsBench, Biomni-Eval1, ChemBench, and MatSciBench the purpose-built agent needs our own benchmark loader or scorer. Table 8, Appendix I.1, and the notes to Table 11 give the departures that bear on each verdict. Table 3 gives the reason for each of the other eleven. RareBench and LLM-SRBench are not paired, because their purpose-built agents cannot be run as published. LongBench v2 is supplementary because DeLM released no code for it. The other eight are excluded.

Screening criteria. The screen records whether a benchmark's requested outcome is something other than code, whether a usable purpose-built agent has been published for it, whether papers other than the benchmark's own have evaluated it, and whether it leaves headroom above the best published score. A purpose-built agent reported only by the benchmark's own authors is kept and

marked as such rather than discarded. Three conditions rule a benchmark out: grading that needs human raters at scale, simulations whose wall-clock time we cannot support, and proprietary access we do not have. Paid APIs, live web access, GPU compute, and token demand are recorded as resource requirements, not treated as grounds for exclusion. These rules do not remove every practical selection effect from the final suite.

Evidence validation. We distinguish papers that define a benchmark from papers that only use it. Benchmark identity is resolved through curated links, citation graphs, full-text search, and web confirmation; identities that remain unresolved go to human review. A model-extracted label that could exclude a candidate must be backed by a verbatim quotation, which we validate against the retrieved source. Without a valid quotation the field is recorded as unknown, and unknown evidence cannot exclude a benchmark.

Search and extraction. Citation retrieval combines Semantic Scholar paper identifiers (Kinney et al., 2025), an OpenAlex citation sweep (Priem et al., 2022), and full-text benchmark-name and alias search. DOI or paper identifiers deduplicate records. Influence scores prioritize reading without excluding papers. Alias filtering precedes model extraction; rejected passages can be sampled to evaluate recall. Low-confidence results and candidate SOTA claims receive a stronger-model review. Full source evidence is retained for judgment.

Score provenance and attribution. Every leaderboard row records method, backbone, frame, metric, split, evaluated instance count, setting, date, quotation, and source location. Mechanical quotation checks do not guarantee correct attribution: an automatic check of the 1,406 rows extracted for the highest-priority benchmarks judged 56 to belong to another method, 33 to another benchmark, 6 to another metric and 2 not to be scores, and could not decide 12, although every one had passed the mechanical quotation check. Candidate SOTA claims therefore require source-table attribution and an adversarial search for a stronger claim. Official leaderboards are checked for denominator and protocol consistency. A later claim replaces the reference only if it is stronger under the same scoring rule (Table 4). The records separate the best score on the full benchmark from the best score in any setting (a subset, split, or other protocol), and they keep agent, bare-model, and classical-system reference scores apart.

Comparable state of the art. We treat a published agentic system as comparable when (i) its released implementation can be instantiated with Codex's backbone model, and we declare every component we had to restore, substitute, or run without (notes to Table 11); (ii) its paper reports a score on the benchmark's own instances and answer key, on the full set or on a stated subset or split, and we score Codex on the same instances; (iii) its gains come from its agentic framework design: systems whose advantage comes from parameter-level changes, such as a model trained or fine-tuned by their authors, are not comparable, whereas components fitted to labeled benchmark data without changing model parameters, such as the optimized prompt and demonstrations of SciTab's CoT+MIPROv2, are; we re-fit such components on Codex's backbone and report their overlap with the scored instances; (iv) it is an agent or pipeline, not a bare-model baseline; and (v) its score is checked against the table that prints it, or against the printed cells or released per-instance results it is computed from. Among comparable systems we take the highest score. On BRIGHT, Figure 1 shows RARG+, the strongest configuration in RARG's paper, whose entry-point module was never released; Table 1 re-runs RARG.

Stronger published results we did not use. Table 4 lists the higher figures that other methods report on the same benchmark version and metric, and why we did not use them. We do not list figures on a changed benchmark (rewritten or re-annotated instances, derived tasks, or other metrics), other configurations of the purpose-built agent itself, or single-subset cells. For the other 15 benchmarks we found no higher figure of this kind.

Finder validation. Every published score we use is checked against the table that prints it, or against the printed cells or released per-instance results it is computed from (criterion (v) above). Agreement between two model agents does not measure the finder's accuracy; that requires a human-adjudicated sample that checks benchmark identity, eligible-system retrieval, score and method attribution, and metric and frame correctness separately.

Table 4: Higher figures that other methods report on the same benchmark, and why each is not our purpose-built agent. ‡: our average of published per-subset cells over the subsets our row uses.

Benchmark	Our purpose-built agent	Higher figure	Why not used
STaRK-PrimeKG	ARK 48.2	GRASP 67.8 (Tao et al., 2026); GraphRAFT 63.7 (Clemedtson & Shi, 2025)	Trained on STaRK
BRIGHT	RARG+ 53.4 (Fig. 1); RARG 51.8, the configuration Table 1 re-runs	Mira-Reasoning-Retrieval 73.3‡	Our average of its per-subset cells; no code or paper. RARG+'s entry-point module is unreleased
SciTab	CoT+MIPROv2 70.6 (70.5 macro-F1, on its 429-claim split)	Qwen-235B with CoT 73 macro-F1; MACE 71 (Saha et al., 2026)	Qwen-235B is a bare model; MACE has no public code (both on all 1,224 claims)
EarthBench	Earth-Agent 66.0	GeoForge 77.1 (Xiao et al., 2026)	No public code
IndustryOR	ReLoop 68.0	ORPilot 75.0 (Xie, 2026)	Counts an answer within 0.1% of the reference as correct, 1000× looser than our 10^{-6}; under that rule Codex scores 82

B BENCHMARKS AND PURPOSE-BUILT AGENTS

Table 5 describes the 20 benchmarks of Table 1, grouped by task type as in Figure 1, and Table 6 gives the core design of the purpose-built agent paired with each.

Table 5: The 20 benchmarks. The metric is the one used in Table 1.

Benchmark	Domain	What it tests	Metric
Tables and documents			
SSTQA (zh) (Tang et al., 2025)	Semi-structured tables (Chinese split)	Answering questions over real-world spreadsheets whose layouts include hierarchical headers and merged cells (764 questions over 102 tables).	Accuracy, GPT-5 judge
OOLONG (Bertsch et al., 2025)	Long-context reasoning	Classifying each of many short text instances in a 262,144-token context and aggregating these judgments (counts, temporal and user relations) to answer distributional questions (OOLONG-synth).	Official partial-credit score
SciTab (Lu et al., 2023)	Scientific tables	Verifying expert-verified claims from scientific papers against a table from the paper, as supports, refutes, or not enough info; the claims require compositional reasoning (1,224 claims).	Accuracy
ELAIPBench (Dai et al., 2026)	AI research papers	Answering expert-written single- and multi-answer multiple-choice questions about a full AI research paper that require reasoning rather than shallow retrieval (403 questions over 137 papers).	Exact match on the selected options
Test of Time (Fatemi et al., 2024)	Temporal reasoning	Reasoning over given synthetic temporal facts with anonymized entities and relations: ordering events, finding what held at a time, counting events in an interval, and computing durations (ToT-Semantic, 2,800 questions).	Exact match
Finance, business, and operations			
IndustryOR (Huang et al., 2025a)	Operations research	Turning 100 industrial optimization problems stated in natural language (linear, integer, mixed-integer, nonlinear, and other programs, at three difficulty levels) into models and solver code that reach the correct optimal value.	Execution accuracy (relative tolerance 10^{-6})
FinanceMath (Zhao et al., 2024)	Finance	Knowledge-intensive math problems that require college-level finance knowledge, some with tables; the answer is a number (200-problem validation split).	Accuracy (0.15% relative tolerance)
MultiHiertt (Zhao et al., 2022)	Financial reports	Multi-step numerical reasoning across several, mostly hierarchical, tables and longer text of a financial report; the answer is a number or a text span.	Accuracy (1% relative tolerance)
InsightBench (Sahu et al., 2025)	Business data analytics	End-to-end analysis of one of 100 business datasets toward a stated goal: formulating questions, interpreting the answers, and summarizing insights, graded against the insights planted in the data.	Insight-level G-Eval score, `gpt-4o` judge

Table 5, continued

Benchmark	Domain	What it tests	Metric
MortarBench (Toles et al., 2026)	Mortgage lending	Answering loan-origination underwriting questions from synthetic bank statements and a loan application; answers are yes/no, a list of transaction or account IDs, or a dollar amount.	Exact match
Search and retrieval			
STaRK-PrimeKG (Wu et al., 2024)	Biomedical knowledge-graph retrieval	Retrieving the entities that satisfy a query combining relational constraints and textual properties, from a precision-medicine knowledge base of about 129K nodes and 8.1M relations.	Hit@1
BRIGHT (Su et al., 2025)	Reasoning-intensive retrieval	Ranking the documents relevant to real-world queries when relevance requires in-depth reasoning rather than keyword or semantic matching (biology, earth science, economics, and robotics subsets; 423 queries).	nDCG@10
CRAG (KG) (Yang et al., 2024b)	Retrieval-augmented factual QA	Answering short factual questions in five domains (finance, sports, music, movie, open) from the benchmark's mock knowledge-graph search API, where a wrong answer scores below a missing one (the 578 knowledge-graph questions of the public test split).	Truthfulness (accurate minus hallucinated share)
GISA (Zhu et al., 2026)	Web information seeking	Meeting realistic, human-written information needs from the web in tasks that combine deep reasoning with broad aggregation; the answer is a structured item, set, list, or table (373 queries).	Official per-format score (F1 or exact match)
Science and medicine			
EarthBench (Feng et al., 2026)	Earth observation	Multi-step, quantitative analysis of Earth-observation images across spectral, product, and RGB modalities, posed as multiple-choice questions (248 expert-curated tasks with 13,729 images).	Accuracy
MatSciBench (Zhang et al., 2026a)	Materials science	College-level textbook problems that require domain knowledge, calculation precision, and multi-step reasoning; the answer is a number or a formula (the 1,025 text-only of 1,340 questions).	Accuracy (5% numeric tolerance)
ChemBench (Mirza et al., 2024)	Chemistry	Chemical knowledge and reasoning across nine topics, from analytical, organic, and physical chemistry to materials science, chemical preference, and toxicity and safety (2,785 multiple-choice and numeric questions).	Accuracy (exact match)
FrontierScience (Wang et al., 2026a)	Scientific research (physics, chemistry, biology)	Solving PhD-level, open-ended sub-problems of scientific research, written and verified by PhD scientists (Research track, 60 problems); a GPT-5 judge grades each answer against a 10-point expert rubric that also credits intermediate reasoning.	Rubric pass rate (at least 7 of 10 points), open book
Biomni-Eval1 (Huang et al., 2025b)	Biomedical research	Ten biological research tasks, including GWAS causal-gene identification, variant prioritization, rare-disease diagnosis, lab-bench database and sequence questions, and CRISPR delivery selection (433 instances).	Per-task 0/1 score
MedAgentsBench (Shao et al., 2026)	Medicine	Complex multiple-choice medical questions from eight medical QA datasets, kept when fewer than half of the tested models answered them correctly, screened for contamination, and validated by medical professionals (862 questions).	Accuracy

Table 6: Purpose-built agents and their core design, grouped as in Table 5.

Agent	Benchmark(s)	Core design
Tables and documents		
ASTRA (Guo et al., 2026)	SSTQA (zh)	Uses an LLM to rebuild each table as a Logical Semantic Tree, adapting the construction to table size, and answers with a dual-mode reasoner that combines tree-search navigation over retrieved tree paths with symbolic code execution. Its headline variant adds a Qwen3-8B selector between the textual and symbolic answers; we score the textual answer.
EnSI-RAG (Meng et al., 2026)	OOLONG	Indexes documents as entity-centered records linked to their source passages, separating evidence localization from answer synthesis. On OOLONG it parses the context into instances, has a smaller extractor model label every instance, computes label, user, and date statistics in Python, and answers from these statistics only.
CoT+MIPROv2 (Du et al., 2026)	SciTab	One chain-of-thought call per claim, with no tools, that labels the claim from the table and its caption. The instruction is tuned by DSPy's MIPROv2 optimizer on training data sampled from TabFact, PubHealthTab, and SciTab, with no few-shot demonstrations.
AgentSPEX (Wang et al., 2026b)	ELAIPBench	A language for agent workflows (typed steps, branching, loops, explicit state) that run in a harness providing tools and a sandbox. Its ELAIPBench workflow uses no tools: it skims the paper's structure, analyzes the question, extracts up to five verbatim evidence passages, judges each option, and re-checks a multi-answer question on which fewer than two options were selected.
REMem-I (Shu et al., 2026)	Test of Time	Indexes the given facts into a hybrid memory graph that links time-aware gists and extracted facts. Its iterative variant answers with an agentic retriever that, for up to five steps, chooses among lexical retrieval, semantic retrieval, entity- and gist-context exploration over the graph, and answering.
Finance, business, and operations		

Table 6, continued

Agent	Benchmark(s)	Core design
ReLoop (Lian et al., 2026)	IndustryOR	One model extracts the problem's parameters to JSON, writes Gurobi code through a four-stage chain (understand, formalize, synthesize, verify), and checks the executed model at two levels: execution errors, and constraints or objective terms whose perturbation barely changes the optimum and are therefore likely missing. Failures trigger up to three rounds of targeted repair, and a repair that shifts the objective by more than 4% is rolled back.
MoCA* (Abdallah et al., 2026)	MultiHiertt, FinanceMath	MoCA-Agent decomposes a question into at most ten typed atomic claims, lets four trader roles (extractor, formula, accountant, skeptic) buy or sell them, clears the orders into confidence-weighted accept/reject decisions, and synthesizes a Python program from the accepted claims, which is executed and checked by a verifier with at most one repair round. MoCA* is the released code, which lacks the baseline proposer and therefore the selector committee and conflict arbiter of the full system.
DataSTORM (Liu et al., 2026)	InsightBench	A deep-research agent over databases and the web. It runs a tree search in which each node poses a sub-question, answers it with generated SQL (web search can be mixed in), judges whether the result is interesting, and expands promising branches; the findings are then re-ranked into a set of global insights.
CRIT (Toles et al., 2026)	MortarBench	A confidence-calibration layer on the benchmark authors' baseline, which makes one answering call and one answer-cleaning call per question. On transaction-list questions the model lists every plausibly related transaction with a 1–5 confidence rating and only those rated 5 are kept; other answer types run the baseline unchanged, with no retrieval or tools.
Search and retrieval		
ARK (Polonuer et al., 2026)	STaRK-PrimeKG	Gives the model two tools over the knowledge graph, BM25 search over node descriptions and one-hop neighborhood expansion that composes into multi-hop traversal, and lets it alternate between breadth and depth without seed selection, a preset hop depth, or retrieval training. Three agents run in parallel for up to 20 steps each, and nodes are ranked by how many agents selected them.
RARG (Li et al., 2026)	BRIGHT	Explores the raw corpus with ripgrep, while a dense retriever orders the documents that ripgrep traverses so that matches from more relevant documents surface first. The RARG+ and RARG++ variants add query-relevant entry-point paragraphs and re-ranking of matches; both steps are off in RARG.
CacheRAG (Sun & Chen, 2026)	CRAG (KG)	An LLM planner writes a plan of knowledge-graph API calls, using cached plans of past questions, chosen for structural diversity, as examples, and widens retrieval by bounded depth and breadth expansion when it is insufficient. A local Llama-3.1-8B-Instruct model routes each question to its domain, and the released CRAG script substitutes knowledge-graph results retrieved offline by the authors when a check finds the retrieved content insufficient.
SearchOS (Zhang et al., 2026b)	GISA	Casts information seeking as filling entity-by-attribute tables with cited values: an orchestrator dispatches up to eight parallel search sub-agents at the empty cells, while a shared search state (task queue, evidence graph, coverage map, failure memory) tracks progress. An extraction step writes sourced records into the evidence graph, and strategy and site-access skills guide the search.
Science and medicine		
Earth-Agent (Feng et al., 2026)	EarthBench	A ReAct tool-calling agent over 104 Earth-observation tools in five kits (index, inversion, perception, analysis, statistics) served through MCP. Its perception tools return the outputs of expert vision models, which the release ships precomputed.
S1-NexusAgent (NexusAgent Team, 2026)	MatSciBench, ChemBench, Biomni-Eval1	A hierarchical Plan-and-CodeAct agent: intent recognition, tool retrieval from a 101-tool scientific registry (filtered by domain, then chosen by an LLM), and a planner that dispatches sub-tasks to a code agent that writes and runs Python calling those tools in a sandbox, with web search always available. The chemistry tools include RDKit and PubChem tools and the biology tools include clients for databases such as UniProt, Ensembl, the GWAS Catalog, and Open Targets; its paper adds a critic agent that distills successful trajectories into reusable skills.
FrontierAgent (An et al., 2026)	FrontierScience	The released scaffold of Apodex 1.1 in Agent Team mode: a coordinator writes its decomposition of the problem to a task board, spawns parallel sub-agents that search the web, read pages, and run code in a sandbox, and synthesizes their reports; in its FrontierScience mode, literature agents first locate the paper the question comes from. The published score uses Apodex 1.1's own model, which is not released.
GSEM (Han et al., 2026)	MedAgentsBench	Stores clinical experiences as indications and contraindications in a dual-layer graph, recalls seed experiences by BM25 and embedding similarity, and has an LLM traverse the graph from the seeds to select a few experiences. These are injected into a ReAct-style physician prompt; the shipped memory holds 1,465 experiences, used without updates (static mode).

C CODEX CONFIGURATION AND EXECUTION PROTOCOL

Table 7 lists the Codex configuration used for every run. Codex denotes this specific configuration, rather than every possible coding-agent configuration.

C.1 CONTAINER EXECUTION AND CHECKS

Each trial runs inside a Docker container that holds the instance workspace. The agent phase has no access to the answer key, the dataset adapter, or other trials' outputs; grading runs separately with access to the answer key. A purpose-built agent's code runs inside the same container boundary, so that the task's environment and network restrictions apply to it. On five benchmarks (CRAG,

Field	Configuration or required record
Agent	Codex CLI, version 0.149.1
Model	`gpt-5.6-sol`
Reasoning effort	`medium`
Attempts	One attempt per instance ($k = 1$, pass@1)
Scaffold	No domain-specific scaffold or per-benchmark tuning; task instructions are checked for strategy hints
Task and grader	The benchmark's task and its own scorer, fixed in the run specification
Environment	The data, tools, and network access the benchmark provides or permits, each named in the prompt; exceptions in Appendix D
Agent visibility	The instance workspace; gold answers and other trials' outputs are excluded
Network	Only the model endpoint is reachable on closed-book and fixed-corpus tasks; open egress where the benchmark permits web access
Built-in web search	Enabled where the benchmark permits web access; set separately from container egress
Sampling	No temperature, top-p, or seed is set; the provider's defaults apply
Resource limits	Per-benchmark turn and compute budgets in the run specification

Table 7: Codex configuration. Endpoint defaults are not assumed to equal a purpose-built agent's own configuration.

ELAIPBench, FinanceMath, IndustryOR, and MortarBench), the purpose-built agent ran on the host instead, behind a proxy that admits only the model endpoint, and only its answer entered the container for grading; we checked the CRAG, ELAIPBench, FinanceMath, and IndustryOR outputs for answer leakage. A logging proxy records every model call a purpose-built agent makes, and Codex's token usage comes from its own session records. We price every recorded call of both agents at list rates, with cached input at 10% of the input rate.

Before any agent runs, each task passes the following checks: the emitted tasks match their recorded checksums; the instructions specify the required output; the staged inputs reach the container; a known-correct answer receives full credit; taking no action receives no unintended credit; and the model is reachable through the configured network path. After a run, we separate valid grades from sandbox or evaluator failures, check that the agent's behavior follows the task, and compute the benchmark's own headline metric.

The known-correct and no-action checks test the grader, not the agent, and they do not replace checking for evaluator errors: a grader that cannot run yields no measurement, not an incorrect answer. A valid execution also does not rule out that the model saw the benchmark during training. We record whether each benchmark was released after the model's training cutoff, but we cannot verify that it was not memorized.

Null baseline and task fidelity. A constant-answer baseline measures each task's scoring floor, for example 0.02 on FinanceMath (4 of 200 answers within the numeric tolerance). These floors characterize the scorer, not agent capability. Each task states its search space, environment, outputs, and scope. FinanceMath's task, for example, asks for the program as well as the answer, because the benchmark's official metrics include the program's execution rate as well as accuracy.

Instances and metrics. We check each metric against the exact column of the published table, because a benchmark's own scorer can compute a different statistic from the one a paper reports, such as item-level F1 instead of exact match. Each agent makes one attempt per instance (pass@1). Where a source paper does not state its sampling settings, our records leave them unstated rather than assume them.

Network and tool access. Container egress, the model provider's built-in search, and the task instructions are set and checked separately. Restricting the container's HTTP requests does not disable search performed by the model provider, which needs its own setting; conversely, a task with open egress can suppress all browsing if its instructions say that no network is available. Neither restriction removes information stored in the model's parameters.

D PURPOSE-BUILT AGENT ADAPTATION AND ALIGNMENT DECLARATIONS

Method versus environment. Codex receives the resources that the benchmark's own paper or release provides to every entrant. CRAG provides its mock API; Biomni-Eval1 provides its E1 data lake with open web access; corpus-based benchmarks provide their staged corpus. Where public web search is permitted, Codex's built-in search tool is enabled and its prompt states the permission. Both agents work from the output of CRAG's mock API. In the open-web ChemBench run of Figure 1, Codex's built-in search tool failed on every call, so Codex reached the web only through shell commands.

Method-specific components remain available to their own purpose-built agent: learned embeddings, indexes, caches, fitted prompts, evolved populations, or trained checkpoints. They are not removed to make the purpose-built agent resemble Codex or copied into the baseline. Runtime resources provided to every entrant belong to both agents. Where a purpose-built agent uses data or tools that Codex lacks, we still report the paired verdict in either direction, name the asymmetry, and analyze the difference in light of it (Section 4.4). For fitted components, we record the data they were fitted on and its overlap with the evaluation instances (Appendix I.1 reports it for SciTab).

Models other than the shared backbone. Table 8 lists, for each purpose-built agent, every model other than the shared backbone `gpt-5.6-sol` that its released pipeline uses or configures, and what our re-run did with it. We keep open-weight routers and embedding models at their released weights; we run on `gpt-5.6-luna` the auxiliary LLM steps that the release or its documentation gives to a separate model, and REMem-I's fact extraction, which its release runs on its question-answering model; and we leave components that are off in the released configuration off. GSEM's embedding model, DashScope's hosted text-embedding-v4, is the one replacement: we use NV-Embed-v2 and re-embed GSEM's shipped experience memory with it, since vectors from different embedding models cannot be compared. Judges that grade both agents belong to the benchmark, not to an agent, and are listed at the foot of the table.

Exposure checks. We scan the citations, fetched URLs, and response text of every open-web trace of both agents for copies of the benchmark or its answer key and for pages that give the instance's own answer. Where the leaking sites can be blocked, a leaking instance is re-run with them blocked. Trace screening removes instances from both agents, including every instance that leaks again or whose leak cannot be blocked (Appendix I.2). The model provider's built-in search is not bound by container egress, so the scan, not the network restriction, is the safeguard.

Run specification and alignment declaration. For each benchmark, a run specification records the evaluated split and size, the metric and its exact scoring implementation, the selected published reference and its source, tool access, network mode, sampling, turn budget, and the purpose-built agent's repository and pinned commit. For each purpose-built agent, an alignment declaration covers ten variables: instance set, answer key, grader, backbone, reasoning effort, sampling, execution environment, network policy, tool surface, and compute budget. For each variable it gives both agents' values, the evidence, and whether they are identical, equivalent, different, or not applicable; an omitted variable counts as undeclared. A deliberate method difference is marked as such, and any other difference is corrected or reported. An automatic check verifies the schema, missing variables, undeclared deviations, and selected cross-field constraints, but it cannot establish that a declaration describes the executed code; the code audit below addresses that.

Code audit. A separate audit reads each run specification against the adapter code, the upstream code, our patches, and the saved runs, and a second agent tries to refute each finding. It covered 25 pairings of a benchmark with a published purpose-built agent; Section 4.2 and Appendix G give how they were selected and what the audit found. Because the audited pairings were selected, the audit does not estimate an error rate for all candidate benchmarks or for the final suite.

Backbone substitution and patches. The shared backbone's endpoint rejects an explicit sampling temperature, so purpose-built agents that request one run with the endpoint's default sampling. This changes a method's operating conditions, especially when it votes or repeats attempts, and we did not measure its effect on the score. Every patch we apply to a purpose-built agent records the upstream behavior, the change, its reason, and any expected direction of effect. A shared backbone

Table 8: Models other than the shared backbone `gpt-5.6-sol` in each purpose-built agent, as released and in our re-run. Every role not listed runs on the backbone.

Agent (benchmark)	Component	As released	In our re-run
AgentSPEX (ELAIPBench)	None	One model for every step; no tools on ELAIPBench	Backbone only
ARK (STaRK)	None; global search is BM25	Optional embedding search and a fine-tuned Qwen3-8B variant, both off by default	Off
ASTRA (SSTQA)	Path-retrieval embedding model Answer selector (Adaptive Selection)	Multilingual-E5-Large Qwen3-8B	Same model, run locally Not run; the textual answer is scored
CacheRAG (CRAG)	Per-question router Plan caches, pre-retrieved KG results (offline)	Llama-3.1-8B-Instruct, local Shipped with the code	Same weights, bf16 on our GPU Used as shipped
CoT+MIPROv2 (SciTab)	None; the optimizer's instruction proposer is the task model	GPT-4o for both; no optimized program released	Backbone for both; we ran the released optimizer
CRIT (MortarBench)	None	Every call, including the benchmark's answer-cleaning pass, uses the model under test	Backbone only
DataSTORM (InsightBench)	Per-insight rewrite before grading	gpt-4o	`gpt-5.6-luna`
	Location linking, warm start, article retrieval, SUQL	gpt-4o, text-embedding-3-small, MiniLM, and others	Not reached with the released InsightBench flags
Earth-Agent (EarthBench)	Answer extraction before scoring	GPT-4o	`gpt-5.6-luna`
	Eight perception expert models	Precomputed outputs in a shipped table (live weights on another branch)	Same table, with the SM3Det look-up the published runs used; Strip-R-CNN and ChangeOS return no output, as released
EnSI-RAG (OOLONG)	Record extractor	gpt-4.1-mini	`gpt-5.6-luna`
	Embedding retrieval	Off (all records are used)	Off
FrontierAgent (FrontierScience)	Web-page condenser	Unset, so the primary model; the documentation recommends a small model	`gpt-5.6-luna`
	Image reader	Unset	Unset
GSEM (MedAgentsBench)	Experience and query embedding model	text-embedding-v4 (DashScope, hosted); vectors shipped	NV-Embed-v2; memory re-embedded
	Answer judge for test-time learning	Backbone; learning mode only	Not run (static mode, as in the paper's MedAgentsBench results)
MoCA (FinanceMath, MultiHiertt)	Baseline proposer, selector committee, arbiter	Backbone; the proposer's code is not released	Absent; selector and arbiter never reached
RARG (BRIGHT)	Dense retriever (corpus index, query encoder)	llama-nv-embed-reasoning-3b; index not shipped	Same model; index built with the authors' script
	Paragraph and match re-rankers	Qwen3-4B-Instruct, Qwen3-Embedding-4B; off in RARG	Off
ReLoop (IndustryOR)	None	One model generates, verifies, and repairs	Backbone only
REMem-I (Test of Time)	Fact extraction	GPT-4.1-mini, also the question-answering model	`gpt-5.6-luna`; question answering on the backbone
	Retrieval embedding model	NV-Embed-v2	Same weights, served by us
S1-NexusAgent (MatSciBench, ChemBench, Biomni-Eval1)	Tool-retrieval embedding model, image-description model, EVO2, ESM3, MassSpec	Embedding store never queried; model endpoints unset; expert tools not registered	Never called
SearchOS (GISA)	Evidence extraction, alias resolution, skill runtime	Qwen3.5-35B-A3B	`gpt-5.6-luna`

Benchmark scorers (grade both agents; not part of any agent). CRAG: `claude-sonnet-4.5` judge. SSTQA: `gpt-5` with ASTRA's released judge prompt. FrontierScience: GPT-5 rubric judge. InsightBench: G-Eval with `gpt-4o`. MortarBench's scoring includes the benchmark's answer-cleaning pass, run on each agent's own model. The other 15 benchmarks use deterministic scorers.

also does not mean that every call uses it: six agents run an auxiliary step on `gpt-5.6-luna`, and the benchmark judges that grade both agents use their own models (Table 8).

Reasoning effort. Both agents run at medium reasoning effort. Our patch to ReLoop sets it explicitly; MoCA-Agent sends no effort setting, so it runs at the endpoint's default, which is documented as medium.

Paired samples. Appendix I.1 and Table 12 give the paired samples: for most benchmarks, a seeded draw of about a quarter of the instances, stratified by the benchmark's own categories where it has them. Both agents run on the same instances, and each benchmark's score follows its own weighting.

Interpretation of reproduction accuracy. A faithful reconstruction on the original configuration and an adapted run on the shared backbone are different quantities. A difference between the adapted purpose-built agent and its published score can reflect changes in backbone, data, tools, sampling, or implementation, and does not by itself measure reproduction error. Appendix I.2 compares published and shared-backbone scores on the six benchmarks where the instance population, metric, and scorer match.

E Review interface and trace analysis

Reviewers read the extracted records in a web interface that shows each benchmark's and method's records, the source evidence, and comments. Comments are anchored to the benchmark, the method, and the source paper rather than to a table position or a score, so they stay attached when records are corrected.

Trace analysis. Section 3.4 describes the paired trace analysis, and Appendices I.3 and I.4 give its labels and agreement checks. A paired case requires the same instance, valid grades, and the execution traces of both agents. Answers in the wrong place or format and faults in our deployment are labeled as failure patterns of their own, apart from the agents' reasoning.

F Further details of the study design

Task execution. Before we can compare two agents on a benchmark, both must run it under the same rules. We run every Codex trial in Harbor (Open-source framework in github), which provides the container, the task format, the Codex adapter, and a sidecar that enforces each task's network allowlist. On top of it, we built what makes the two agents comparable: a task adapter per benchmark that states the inputs, tools, and answer file, keeps gold answers out of the container, and wraps the benchmark's own scorer; checks that every trial ends in a graded answer or a recorded infrastructure failure; and a logging wrapper that records every model call of the purpose-built agents and runs their code under the same network rules. On five benchmarks, the purpose-built agent ran on the host instead, behind a proxy that admits only the model endpoint (Appendix C.1).

Protocol alignment. To ensure that score differences reflect the agents rather than experimental setup variations, we write a run specification for every benchmark: instances, answer key, scorer, backbone, reasoning effort, sampling, environment, network, tools, and budget. We audit each specification against the adapters, the upstream code, and the saved runs, and we keep each repair and ruling with its evidence. To give Codex what the benchmark gives every entrant, we stage the benchmark's own data and, where the benchmark permits it, open the container's network. On EarthBench, the purpose-built agent uses its own analysis tools, and we analyze its results in light of this difference. To keep each purpose-built agent intact, we leave its indexes, caches, fitted prompts, and search tools in place. When these give it data or tools of its own, we still count the paired result in either direction and name the asymmetry. To control answer leakage, we scan the traces of both agents for the benchmark's answers and, where the leaking sites can be blocked, re-run a leaking instance with them blocked; trace screening removes instances from both agents, including every instance that leaks again or whose leak cannot be blocked (Section 4.2).

Codex and purpose-built agents. Codex runs as Codex CLI on `gpt-5.6-sol` at medium reasoning effort, with one scored attempt per instance. Within an attempt, it may call the model many times, run shell commands, and write and run programs. It receives the task, the data and tools that the benchmark provides, and, where the benchmark allows it, the open web, but nothing from the purpose-built agent's workflow. We re-run each purpose-built agent from its released code on the same model and effort, and it keeps its own prompts, planners, and other components. Some released agents needed changes to run. The most common is that six agents run one auxiliary step on the smaller `gpt-5.6-luna`, as most of their releases or documentation give that step to a separate model. Counting these, ten of the 17 agents use a model other than the shared backbone, such as a router or an embedding model (Table 8 lists every such model, and Appendix I.1 the changes we made).

Two comparison settings. We compare Codex with each purpose-built agent in two settings. First, we set Codex beside the score that the agent's paper reports on its own backbone and protocol, scoring Codex on that paper's metric and, where the paper reports one, on its subset or split (Figure 1). Second, we re-run the agent on Codex's backbone and score both agents on the same instances: the full set on five benchmarks, and otherwise a seeded random sample, 25% of the benchmark in most cases (Table 1). The first places Codex against the published state of the art. The second holds the model and the instances fixed, so that a difference reflects the agents rather than their backbones or protocols, and it gives us the traces of both agents.

Scoring, cost, and time. We take each metric from existing sources by a fixed rule: we score both agents with the metric of the purpose-built agent's paper, else its repository, else the benchmark's official metric; only CRAG and GISA depart from this rule. We test each paired difference with an exact McNemar test, or a sign-flip test for continuous metrics, and we report 95% paired bootstrap intervals that resample clusters of instances sharing a source, such as a paper, table, or textbook. We price every recorded model call of both agents, including auxiliary calls, at list rates, and we time each agent run without setup and grading (Appendix I.1).

Trace annotation. Scores show which agent does better, but not why. To explain the differences, Claude Code (Opus 5.5) acting as annotators read the traces of both agents, Codex's session logs and the purpose-built agents' stage records, following a written codebook. For each requirement of a task, such as a table cell the answer depends on or a condition it must satisfy, the annotators check whether the agent saw it, used it, and carried it into the submitted answer. On all 20 benchmarks, they label each instance that only one agent answers correctly with one primary mechanism (RQ4), and each failure of either agent with one of twelve failure patterns in five groups (RQ5). We count a gold answer or scorer as disputable only when the benchmark's own materials support it. To check the failure patterns, a second annotator independently labeled a stratified sample and a third adjudicated the disagreements; another annotator re-read a sample of the mechanism labels (Appendices I.4 and I.3 report the agreement).

G ALIGNMENT AUDIT FINDINGS

We audited the 25 pairings of a benchmark with a published purpose-built agent that no earlier check had verified. For each pairing, an LLM agent checked the run specification (Section 3.2) against our adapters, the purpose-built agent's released code, and the saved runs, and a second LLM agent tried to refute each finding. Of the 308 findings raised, 303 survived this check (Table 9). The audit judged 22 of the 25 pairings misaligned; two (HiTab and LongBench v2) had no runnable purpose-built agent, and one (CritPt) could not be assessed. Five pairings that an earlier check had verified (STaRK-PrimeKG, BRIGHT, SciTab, LLM-SRBench, and IndustryOR) were not audited again, so the 22 of 25 describes the audited set, not every published comparison. A finding records a mismatch in the setup, not a measured change in score.

Some findings concern the benchmarks' own scorers. FinanceMath's official matcher, which grades both agents in Table 1, compares absolute values and accepts any exact power-of-ten ratio to the gold answer, so it accepts 8184 for a gold answer of -8184.

Table 9: The 303 findings of the alignment audit of 25 benchmark–agent pairings, by severity, axis, and direction. A critical finding prevents a valid comparison until it is corrected. Direction is the agent that a finding favors: correcting the finding would move the comparison against that agent. The last column restricts the direction split to the 62 critical findings.

by severity		*by axis*		*by direction*	all 303	of 62 critical
critical	62	protocol	50	favors the purpose-built agent	95	19
material	145	metric	38	favors Codex	65	15
minor	96	outdated result	35	neutral or unknown	140	27
		tool access	33	both directions	3	1
		undeclared change	27			
		other	26			
		backbone	21			
		denominator	20			
		judge	16			
		dataset version	13			
		contamination	13			
		instances	8			
		embedding model	3			

H PER-BENCHMARK COMPARISON WITH PUBLISHED SCORES

Table 10 gives every row of Figure 1.

Published backbones. Each published score in Table 10 was obtained with its agent's own backbone model: ARK with GPT-4.1 (STaRK-PrimeKG), RARG+ with GPT-5.4-mini (BRIGHT), ASTRA with DeepSeek-V3 (SSTQA), CacheRAG with DeepSeek-V3 (CRAG), EnSI-RAG with GPT-4.1 and GPT-4.1-mini (OOLONG), SearchOS with GLM-5 and Qwen3.5-35B-A3B (GISA), CoT+MIPROv2 with GPT-4o (SciTab), Earth-Agent with GPT-5 (EarthBench), S1-NexusAgent with Claude-4.5-Sonnet (MatSciBench, ChemBench, and Biomni-Eval1), FrontierAgent with Apodex 1.1, whose weights are not released (FrontierScience), AgentSPEX with GPT-5 at high reasoning effort (ELAIPBench), GSEM with Qwen3.5-35B (MedAgentsBench), ReLoop with Claude Opus 4.6 (IndustryOR), MoCA-Agent with Qwen3.6-27B (FinanceMath and MultiHiertt), DataSTORM with GPT-5 (InsightBench), CRIT with Gemini 3.1 Pro (MortarBench), and REMem-I with GPT-4.1-mini (Test of Time). Codex runs on `gpt-5.6-sol` at medium reasoning effort on every row.

Notes on individual rows. *SSTQA.* ASTRA's paper does not name the language of the split it reports; its released code reads only the Chinese split, on which Codex is scored. Codex's answers are judged by GPT-5 with ASTRA's released judge prompt, and ASTRA's paper also uses a GPT-5 judge. *OOLONG.* Codex answers the first 192 questions of the synthetic test split at context length 262,144, the instances EnSI-RAG reports, with the context in its prompt. OOLONG's official partial-credit scorer grades Codex; EnSI-RAG's published score uses an LLM judge for non-numeric answers. *SciTab.* CoT+MIPROv2's published score uses prompts optimized on SciTab's labeled training data. *Test of Time.* Codex answers 280 of the 2,800 questions, one for each combination of question type, graph generator, and ordering. REMem-I's 93.1 is its main-table result with GPT-4.1-mini; the 99.0 that its appendix text reports with GPT-5-chat has no table entry and is not used. *ELAIPBench.* AgentSPEX's published score uses high reasoning effort; Codex, and AgentSPEX in Table 1, use medium. *IndustryOR.* Answers are scored at ReLoop's relative tolerance of 10^{-6} against the benchmark's original answer key, which ReLoop's published score uses. *MultiHiertt.* The metric is MoCA-Agent's answer accuracy on the 1,044 questions of the development split, whose answers are public, rather than the benchmark's exact match and F1. *CRAG (KG).* Codex works from the benchmark's mock knowledge-graph API, as CacheRAG does, on all 578 knowledge-graph questions. `claude-sonnet-4.5` judges Codex's answers, since CRAG's official judges are retired; CacheRAG's published score was judged by Llama-3.1-70B-Instruct. *GISA.* SearchOS's published score takes the best of three runs on each question; Codex's score comes from one run. *FrontierScience.* Codex is scored on 59 of the 60 research problems; the other was removed because Codex's trace reached a leaked answer (Section 4.2). *EarthBench.* Earth-Agent's published

Table 10: Figure 1 in full: Codex against each benchmark's purpose-built agent as reported in that agent's paper. Codex (n): Codex's score and the number of instances it is scored on. Δ: Codex minus the published score, in points; $\Delta\%$: the same gap relative to the published score.

Benchmark	Purpose-built agent	Latest arXiv version	Metric	Codex (n)	Published	Δ	$\Delta\%$
Tables and documents (Codex at or above the published score on 4 of 5)							
SSTQA (Tang et al., 2025)	ASTRA (Guo et al., 2026)	2026-05-26	judge accuracy (Chinese split)	86.7 (764)	81.9	+4.8	+6%
OOLONG (Bertsch et al., 2025)	EnSI-RAG (Meng et al., 2026)	2026-08-21	official score	74.5 (192)	71.8	+2.6	+4%
ELAIPBench (Dai et al., 2026)	AgentSPEX (Wang et al., 2026b)	2026-04-14	exact match	45.2 (403)	43.7	+1.5	+3%
SciTab (Lu et al., 2023)	CoT+MIPROv2 (Du et al., 2026)	2026-02-20	accuracy (paper's 429-question test split)	72.7 (429)	70.6	+2.1	+3%
Test of Time (Fatemi et al., 2024)	REMem-I (Shu et al., 2026)	2026-02-28	accuracy	89.3 (280)	93.1	−3.8	−4%
Finance, business, and operations (Codex at or above the published score on 3 of 5)							
IndustryOR (Huang et al., 2025a)	ReLoop (Lian et al., 2026)	2026-08-16	accuracy	75.0 (100)	68.0	+7.0	+10%
FinanceMath (Zhao et al., 2024)	MoCA-Agent (Abdallah et al., 2026)	2026-06-17	accuracy	79.0 (200)	76.0	+3.0	+4%
MultiHiertt (Zhao et al., 2022)	MoCA-Agent (Abdallah et al., 2026)	2026-06-17	answer accuracy	72.1 (1044)	71.2	+1.0	+1%
MortarBench* (Toles et al., 2026)	CRIT (Toles et al., 2026)	2026-08-25	exact match (released 90 cases; paper used 188)	80.0 (90)	80.5	−0.5	−1%
InsightBench* (Sahu et al., 2025)	DataSTORM (Liu et al., 2026)	2026-08-26	insight score (69 of 100 datasets; authors' released per-dataset scores)	54.4 (69)	61.5	−7.1	−12%
Search and retrieval (Codex at or above the published score on 2 of 4)							
STaRK-PrimeKG (Wu et al., 2024)	ARK (Polonuer et al., 2026)	2026-04-28	Hit@1	74.0 (2801)	48.2	+25.8	+53%
BRIGHT (Su et al., 2025)	RARG+ (Li et al., 2026)	2026-08-03	nDCG@10	61.1 (423)	53.4	+7.7	+14%
CRAG (KG)† (Yang et al., 2024b)	CacheRAG (Sun & Chen, 2026)	2026-06-08	truthfulness $\times 100$	68.2 (578)	71.1	−2.9	−4%
GISA* (Zhu et al., 2026)	SearchOS (Zhang et al., 2026b)	2026-07-16	Table Item F1 (table questions of gisa_100)	72.7 (59)	76.9	−4.2	−5%
Science and medicine (Codex at or above the published score on 3 of 6)							
MedAgentsBench* (Shao et al., 2026)	GSEM (Han et al., 2026)	2026-03-23	mean of the 5 subsets GSEM reports	56.9 (489)	49.5	+7.4	+15%
Biomni-Eval1 (Huang et al., 2025b)	S1-NexusAgent (NexusAgent Team, 2026)	2026-02-10	task macro	84.5 (419)	76.1	+8.4	+11%
FrontierScience (Wang et al., 2026a)	FrontierAgent (An et al., 2026)	2026-08-25	rubric pass rate (open book)	66.1 (59)	63.3	+2.8	+4%
EarthBench (Feng et al., 2026)	Earth-Agent (Feng et al., 2026)	2026-03-03	answer accuracy	59.1 (247)	66.0	−6.9	−10%
MatSciBench (Zhang et al., 2026a)	S1-NexusAgent (NexusAgent Team, 2026)	2026-02-10	accuracy (text-only)	74.0 (1025)	82.9	−9.0	−11%
ChemBench (Mirza et al., 2024)	S1-NexusAgent (NexusAgent Team, 2026)	2026-02-10	overall score (open web, stratified 25% sample)	71.8 (699)	86.1	−14.3	−17%

A benchmark is placed in the first section that fits: a finance, business, or operations problem; otherwise, a table, document, or set of facts to answer from; a large collection or the web to search; or a science or medical question with no document, table, or collection to answer from. Sections are ordered by the share of their benchmarks on which Codex reaches the published score, ties by the mean relative gap; with four to six benchmarks per section, one benchmark can change the order, so the order describes these 20 benchmarks only. Codex's largest deficits are on ChemBench, InsightBench, and MatSciBench. Three rows do not use the paper's main result: on BRIGHT the published agent is RARG+, the strongest configuration in RARG's paper; on MedAgentsBench the published score is 49.5, the mean of GSEM's five MedAgentsBench cells (GSEM's printed averages, 69.24 and 70.90, also cover MedR-Bench); on InsightBench it is 61.5, recomputed on the 69 datasets with valid tables from the authors' per-dataset scores (the paper prints 61.9 on all 100). On BRIGHT, Table 1 runs RARG, because RARG+'s entry-point module was never released (Appendix I.1). *Codex is scored on the purpose-built agent's subset or released cases (the metric column says which). †Truthfulness: the share of correct answers minus the share of hallucinated answers, $\times 100$. Notes on individual rows follow the table.

score, 66.0, is the GPT-5 result in the latest arXiv version of its paper (v3); versions 1 and 2 print 65.6 in the same cell, which Earth-Agent's released outputs reproduce (162 of 247). *Biomni-Eval1.* Codex runs in the benchmark's official setting, with the E1 data lake and open web access. It is

scored on 419 of the 433 tasks: screening its traces removed 14, including every task whose trace reached a leaked answer (Section 4.2). The metric is the macro average over the benchmark's ten task types. *ChemBench.* Codex, with open web access, is scored on a stratified 25% sample of the 2,785 questions, less one question whose trace reached a leaked answer; S1-NexusAgent's paper does not name the questions its score covers.

H.1 Benchmarks outside the paired suite

Two further benchmarks have a Codex result but no entry in Figure 1 or Table 1, because their purpose-built agents cannot be run as published (Table 3).

LLM-SRBench. On all 129 problems, Codex meets the benchmark's $\text{Acc}_{0.1}$ criterion (relative error at most 0.1 at every test point) on 116 (89.9%), against FunctionEvolve's published 87.6% (113 of 129) with Claude Opus 4.6; FunctionEvolve's search stops on a threshold calibrated on the ground-truth equation.

LongBench v2. DeLM, its purpose-built agent, released no code for LongBench v2, which is therefore not in the paired suite. On the 125 multi-document QA questions that DeLM reports, Codex scores 61.5, the unweighted mean of its accuracies on the five sub-domains, against DeLM's published 60.1, a mean over three runs with GPT-5.4 and a DeepSeek-V4-Flash summarizer. Codex receives each document as a file, with an instruction that suggests a strategy for searching it; LongBench v2's official prompt contains no such strategy.

I Details for Section 4

This appendix holds the procedural details behind Section 4: how the paired instances were drawn, the evaluation checks behind RQ1, how we labeled the reasons for each win, and the failure patterns behind Figure 3. We abbreviate S1-NexusAgent as S1.

I.1 Paired samples

Instances. An *instance* is one test case that a benchmark scores: a question on most benchmarks, a dataset on InsightBench, a task on EarthBench and Biomni-Eval1, and a search query on BRIGHT. Both agents answer the same instances on every benchmark. On CRAG's knowledge-graph subset, ELAIPBench, FinanceMath, IndustryOR, and MortarBench, both agents ran every instance, so we compare them on the full set. On the other fifteen benchmarks, we drew a seeded sample before running the purpose-built agent, stratified by the benchmark's own categories where it has them (seed 20260917 unless Table 12 says otherwise). EarthBench keeps the tasks on which its two answer keys agree, MatSciBench drops the ten questions voided for answer leakage, and trace screening removes 13 Biomni-Eval1 instances from both agents, including every instance whose trace reached a leaked answer. SearchOS evaluates GISA on its own 100-question subset (gisa_100), so we pair on 50 of those questions: 15 from our stratified GISA draw that fall in gisa_100, and 35 others drawn by answer type (seed 20260924). Biomni-Eval1 and MedAgentsBench use two disjoint 25% draws. Test of Time pairs on 280 questions, one for each combination of question type, graph generator, and ordering (10%). ChemBench pairs on 330 questions (12%): a stratified 280, 150 of them chosen without a recorded seed, and a disjoint seeded 50. InsightBench pairs on a seeded 25% sample of its 100 datasets.

Draws. Table 12 lists, for every row of Table 1, the population, the paired sample, and the seeds of every draw. Each draw except STaRK's (simple random) is stratified by the benchmark's own categories, allocates instances to categories in proportion with largest remainders, and reproduces exactly from its seeds. A second seed shuffles the order in which instances are run. Two exceptions: 150 of ChemBench's 280 were chosen without a recorded seed, and only the other 130 are seeded; OOLONG's 79 are drawn at random within its context-window strata, though not exactly in proportion to them. Every draw in Table 12 was fixed before the purpose-built agent's run that uses it. Codex's results on a sample come either from a full-set run restricted to the sampled instances or from a run on exactly those instances.

Table 11: Full version of Table 1: Codex and the purpose-built agent on the same instances and backbone. Scores are on a 0–100 scale in each benchmark's own metric. Δ is Codex minus the purpose-built agent, with its 95% paired bootstrap interval. **Bold** rows differ at $p < .05$, and the sign of Δ shows which agent is better. n: instances compared, with their share of the benchmark ($2 \times 25\%$: two disjoint 25% samples). C/P: instances that only Codex, or only the purpose-built agent, answers correctly (for continuous metrics, scores higher on). p: exact McNemar test, or a paired sign-flip test for continuous metrics. Cost×: the purpose-built agent's model cost divided by Codex's, on instances both answer correctly (†: whole-run totals over all instances of the run, since MoCA's logs do not tie model calls to instances). Rows are grouped by the information each agent works from (Section 4.4).

Benchmark	Purpose-built agent	n (sample)	Metric	Codex	Purpose-built	Δ [95% CI]	C/P	p	Cost×
Purpose-built agent converts or filters input that Codex reads in full									
SSTQA (zh)[a] (Tang et al., 2025)	**ASTRA** (Guo et al., 2026)	**191 (25%)**	**acc.**	**85.3**	**73.8**	**+11.5 [+4.7, +19.5]**	**25/3**	**<.001**	**1.56**
Test of Time (Fatemi et al., 2024)	**REMem-I** (Shu et al., 2026)	**280 (10%)**	**EM**	**89.3**	**82.5**	**+6.8 [+2.9, +11.0]**	**23/4**	**<.001**	**0.65**
MortarBench (Toles et al., 2026)	CRIT (Toles et al., 2026)	90 (100%)	EM	80.0	73.3	+6.7 [0.0, +13.0]	9/3	.15	0.50
STaRK-PrimeKG (Wu et al., 2024)	**ARK** (Polonuer et al., 2026)	**700 (25%)**	**Hit@1**	**72.9**	**67.3**	**+5.6 [+2.7, +8.6]**	**68/29**	**<.001**	**1.10**
ELAIPBench (Dai et al., 2026)	AgentSPEX (Wang et al., 2026b)	403 (100%)	EM	45.2	45.7	−0.5 [−5.1, +4.0]	40/42	.91	1.24
Both agents have the same information									
FrontierScience (Wang et al., 2026a)	FrontierAgent (An et al., 2026)	15 (25%)	rubric pass	73.3	60.0	+13.3 [0.0, +33.3]	2/0	.50	4.92
Biomni-Eval1 (Huang et al., 2025b)	**S1-NexusAgent** (NexusAgent Team, 2026)	**203 (47%)**	**EM**	**86.2**	**78.3**	**+7.9 [+3.0, +12.8]**	**21/5**	**.002**	**4.51**
CRAG (KG)[c] (Yang et al., 2024b)	**CacheRAG** (Sun & Chen, 2026)	**578 (100%)**	**truthfulness**	**68.2**	**63.1**	**+5.0 [+0.9, +9.3]**	**34/21**	**.02**	**1.63**
IndustryOR (Huang et al., 2025a)	ReLoop (Lian et al., 2026)	100 (100%)	acc.	75.0	70.0	+5.0 [−1.0, +11.0]	7/2	.18	1.18
MultiHiertt (Zhao et al., 2022)	MoCA* (Abdallah et al., 2026)	261 (25%)	acc.	77.4	75.9	+1.5 [−1.5, +4.6]	10/6	.45	2.73†
MatSciBench[e] (Zhang et al., 2026a)	S1-NexusAgent (NexusAgent Team, 2026)	246 (24%)	acc.	78.9	77.6	+1.2 [−0.8, +4.9]	11/8	.65	2.41
MedAgentsBench (Shao et al., 2026)	GSEM (Han et al., 2026)	432 (2×25%)	acc.	47.9	46.8	+1.2 [−1.9, +4.2]	25/20	.55	1.78
SciTab (Lu et al., 2023)	CoT+MIPROv2 (Du et al., 2026)	306 (25%)	acc.	72.9	71.9	+1.0 [−2.3, +4.2]	15/12	.70	0.14
FinanceMath (Zhao et al., 2024)	MoCA* (Abdallah et al., 2026)	200 (100%)	acc.	79.0	79.0	0.0 [−4.5, +4.5]	10/10	1.00	1.95†
InsightBench[g] (Sahu et al., 2025)	DataSTORM (Liu et al., 2026)	25 (25%)	G-Eval	46.3	49.4	−3.0 [−7.4, +3.0]	11/14	.42	–
GISA[f] (Zhu et al., 2026)	SearchOS (Zhang et al., 2026b)	50 (50% of gisa_100)	per-format	60.8	66.9	−6.1 [−16.5, +3.5]	22/15	.24	17.82
OOLONG (Bertsch et al., 2025)	**EnSI-RAG** (Meng et al., 2026)	**79 (41%)**	**partial credit**	**74.6**	**82.3**	**−7.7 [−13.4, −2.4]**	**7/17**	**.03**	**0.51**
Purpose-built agent uses data or tools that Codex did not have									
BRIGHT (Su et al., 2025)	RARG (Li et al., 2026)	106 (25%)	nDCG@10	62.9	60.8	+2.1 [−1.6, +5.8]	36/32	.27	0.48
ChemBench[b] (Mirza et al., 2024)	S1-NexusAgent (NexusAgent Team, 2026)	330 (12%)	acc.	74.5	77.3	−2.7 [−5.8, +0.3]	8/17	.11	3.38
EarthBench[d] (Feng et al., 2026)	**Earth-Agent** (Feng et al., 2026)	**54 (22%)**	**acc.**	**61.1**	**81.5**	**−20.4 [−35.2, −5.6]**	**4/15**	**.02**	**0.78**

Six purpose-built agents (REMem-I, EnSI-RAG, SearchOS, Earth-Agent, FrontierAgent, and DataSTORM) run one helper step on the smaller `gpt-5.6-luna`. FrontierAgent is the released scaffold of Apodex 1.1, the system behind the published score in Figure 1, whose model is not released. MoCA* is the released MoCA-Agent, which lacks one unreleased component. [a]ASTRA is scored on its textual answer, without the selector of its Adaptive Selection configuration. [b]S1-NexusAgent used web search, which the benchmark allows. [c]The share is of CRAG's 578 knowledge-graph questions, not of the 1,335 in Table 12. Both agents work from the benchmark's official mock knowledge-graph API (KDD Cup 2024). A departure from our scoring rule: CRAG's official judges are retired, so `claude-sonnet-4.5` judges both agents. [d]The 54 of 61 sampled tasks on which the benchmark's two released answer keys agree. Scored against the second key on all 61, the difference is not significant (Codex 62.3, Earth-Agent 73.8, $p = .21$). [e]The seeded 25% sample (256 questions) without the ten questions voided for answer leakage (Section 4.2). [f]The 50 questions of SearchOS's own 100-question evaluation subset (gisa_100) that both agents ran. A departure from our scoring rule: each question is scored with the official scorer's metric for its answer type. [g]Our shared-backbone DataSTORM run on the seeded 25% sample (25 datasets; no cost record). Against DataSTORM's published `gpt-5` predictions on the 69 datasets whose tables are valid, Codex is significantly behind (54.4 vs. 58.4, $\Delta = -4.0$ [−7.5, −0.4], $p = .04$). With our run, the two are not distinguishable ($\Delta = -3.0$, $p = .42$; see "Sensitivity").

Table 12: Paired instances for the 20 rows of Table 1.

Benchmark	Population	n	Sample	Seeds
SSTQA (zh)	764	191	25%, question type × difficulty	20260920
STaRK-PrimeKG	2,801	700	25%, simple random	20260917
Test of Time	2,800	280	one question per question type × graph generator × ordering	20260822
MortarBench	90	90	full set (both agents ran all 90)	–
ELAIPBench	403	403	full set (both agents ran all 403)	–
IndustryOR	100	100	full set (both agents ran all 100)	–
FinanceMath	200	200	full set (both agents ran all 200)	–
CRAG (KG)	1,335	578	all 578 knowledge-graph questions (both agents ran all of them)	–
EarthBench	247	54	25% by modality (61); the 54 on which both released keys agree	20260917
BRIGHT	423	106	25% by domain	20260917
ChemBench	2,785	330	280 by topic × kind (150 chosen without a recorded seed, 130 seeded) and a disjoint 50	20260917, 20260921
Biomni-Eval1	433	203	two disjoint 25% draws by task (216), minus 13 removed in trace screening	20260917, 20260921
MatSciBench	1,025	246	25% by answer type (256), minus 10 leak voids	20260917
MultiHiertt	1,044	261	25% by answer type	20260917
SciTab	1,224	306	25% by gold label	20260917
MedAgentsBench	862	432	two disjoint 25% draws by subset	20260917, 20260921
OOLONG	192	79	drawn at random within context-window strata	20260917, 20260925, 20260926
FrontierScience	60	15	25% by subject; one leaked question replaced by a seeded draw	20260917, 20260923
GISA	100	50	50% of SearchOS's gisa_100: 15 from our 25% draw by question type × answer type, 35 by answer type	20260917, 20260924
InsightBench	100	25	25% by domain; published-prediction comparison on the 69 datasets with valid tables	20260917

Changes to the released agents. Six purpose-built agents (REMem-I, EnSI-RAG, SearchOS, Earth-Agent, FrontierAgent, and DataSTORM) run one helper step on the smaller `gpt-5.6-luna`. For five of them, the release or its documentation gives that step to a separate model: EnSI-RAG and SearchOS name a smaller model, Earth-Agent and DataSTORM run the step on GPT-4o, and FrontierAgent's documentation recommends a small model for it. REMem-I's paper uses one model for both roles, so its split is ours. GSEM's release embeds its memory with DashScope's hosted text-embedding-v4; we re-embed the shipped memory with NV-Embed-v2. Appendix Table 8 lists every model other than the backbone that each agent uses. CoT+MIPROv2 releases no optimized program, so we run its released optimizer with the repository defaults on the paper's training data. MoCA* is the released MoCA, which lacks the unreleased proposer and therefore its selector committee and arbiter. On SSTQA we score ASTRA's textual answer. Its paper's headline configuration, Adaptive Selection, adds a Qwen3-8B selector that chooses between the textual and symbolic answers, worth 2.1 points in the paper (79.8 to 81.9), which is well below Codex's 11.5-point lead in Table 1. On BRIGHT we run RARG rather than RARG+, the stronger configuration in the same paper that Figure 1 shows. RARG+ gives the agent a few query-relevant paragraphs as an entry point, but the module that selects them is not in the public repository, and the paper does not specify it (paragraph count, splitting rule, scoring model). Without the module, the released RARG+ configuration runs as RARG, so it cannot reproduce RARG+'s 53.4.

Width of the intervals. Among the 13 benchmarks that are not distinguishable, the interval stays within about five points on FinanceMath, MatSciBench, MedAgentsBench, MultiHiertt, and SciTab, and just over five on ELAIPBench (-5.1 to $+4.0$). On FrontierScience, MortarBench, IndustryOR, and GISA it cannot rule out a difference of ten points or more.

Sensitivity. Four verdicts depend on a reporting choice: EarthBench's on the answer key, CRAG's and Biomni-Eval1's on the sample, and InsightBench's on the comparison (notes c, d, and g of Table 11). Scored against EarthBench's first released key on all 61 sampled tasks, its verdict is unchanged ($p = .007$), but against its second key the agents are not distinguishable (Codex 62.3, Earth-Agent 73.8, $p = .21$). On seeded 25% samples of the five benchmarks compared on their full sets, every verdict is the same as on the full set except CRAG's, which is not distinguishable on its 145-instance sample ($\Delta = +3.4$ [-4.8, $+12.4$], $p = .54$). On all 578 questions, Codex stays ahead without the 19 questions for which the knowledge-graph results that CacheRAG retrieves in advance are empty ($\Delta = +5.0$, $p = .03$). For RQ3 we classified all 578 CRAG questions by whether these results contain the answer: 471 do, 59 do not, and 48 rest on a false premise. On the false-premise questions, Codex scores higher on 3 and CacheRAG on 5 ($p = 1.0$). On one of Biomni-Eval1's two 25% draws alone, the agents are not distinguishable (87 vs. 84 of 101, $p = .55$). InsightBench's

verdict depends on the comparison. Against our shared-backbone DataSTORM run (Table 1) the two are not distinguishable. Against DataSTORM's published predictions Codex is significantly behind on the 69 datasets with valid tables ($\Delta = -4.0$, $p = .04$; Table 11, note g), and the verdict holds on all 100 datasets ($\Delta = -9.0$, $p = .02$) and when the 31 null-table datasets count as one cluster (-4.2, $p = .03$). The 69-dataset gap is borderline, since some remaining datasets also carry weaker gold answers. Without datasets 95–100 it is -2.6 ($p = .12$), and on datasets 1–46 alone it is -1.9 ($p = .34$). Other readings leave verdicts unchanged. On GISA, SearchOS's own scorer gives 64.8 vs. 68.1 ($\Delta = -3.3$, $p = .46$). On one of MedAgentsBench's two 25% draws alone, the verdict is the same (51.4 vs. 48.6, $p = .31$). SciTab's 306 include 21 instances that also appear in CoT+MIPROv2's optimization data. On the other 285 the difference is $+0.4$ points ($p = 1.0$).

Cost and time. Per-instance cost ratios cover the instances of Table 1 that both agents solve, for example 183 on MatSciBench, 238 on ChemBench, and 154 on Biomni-Eval1. GISA's cost ratio covers all 50 paired instances. Where an agent's usage records do not attribute model calls to instances, we assign each call to its instance from the agent's own logs (by time window, or by matching token counts or request text); MoCA's logs do not allow this, so its two ratios use whole-run totals. Run times (without setup and grading) are available for 14 benchmarks, and the purpose-built agent takes longer on 12 of them. We also compared cost by outcome on five benchmarks: Biomni-Eval1, ChemBench, EarthBench, IndustryOR, and MatSciBench. There the purpose-built agent spends 2.38 times as much per correct answer (95% CI 1.03–3.79). Figure 4 plots each benchmark's cost ratio beside its paired difference.

I.2 Evaluation checks behind RQ1

Audit and reproduction. We audited 25 pairings of a benchmark with a published purpose-built agent that no earlier check had verified. After an adversarial check of every finding, the audit kept 303 findings and judged 22 of the 25 pairings misaligned, for example in protocol, metric, outdated results, tool access, or undeclared changes. Of the other three, two had no runnable purpose-built agent, and one could not be assessed (Appendix G).

Re-running released code gave mixed results. CoT+MIPROv2, re-run on our backbone, scores 70.6 on the 102 sampled SciTab instances that fall in the paper's test split, the same as its published GPT-4o score; with 102 instances this shows agreement within about nine points, not an exact reproduction. Our ELAIPBench answer parser agrees with AgentSPEX's own verdicts on all 403 instances. DataSTORM, run from its released code and settings on our backbone, scores 9.6 points below the predictions its authors released for the same 25 InsightBench datasets ($p = .005$). With the released settings the code emits at most 52 insights per dataset (median 33 in our run), whereas every released list has 55–93 (median 90). InsightBench's scorer matches up to 90 insights against the reference, so longer lists score higher. The released predictions also came from a different backbone model (GPT-5).

Purpose-built scores on the shared backbone. RQ2 compares each purpose-built agent's published score with our re-run of it. We make this comparison only where our re-run covers a random sample of the same instance population, uses the same metric and scorer, and runs the same released agent. Six benchmarks qualify (Table 13). The change combines the backbone with the setup our protocol fixes (medium reasoning effort, the sampling temperature the backbone allows, and the model endpoint we use), so it measures model and setup, not the backbone alone. Two changes exceed what the papers' own backbone comparisons suggest. ARK's paper reports 50.6 Hit@1 with GPT-5.2, 2.4 points above GPT-4.1, so our $+19.1$ is a measurement under our setup, not a general model effect. REMem-I's drop comes with our split of its two roles (fact extraction on the smaller `gpt-5.6-luna`); its paper reports 99.0 with GPT-5-chat. IndustryOR and ELAIPBench use our full-set re-runs, which cover the published set itself. SciTab uses the 102 sampled instances that fall in the paper's test split. BRIGHT uses RARG's own answer extraction (60.6); Table 1's extraction differs on one query and gives 60.8. Figure 1 shows RARG+ (53.4), which cannot be re-run, so this comparison uses RARG's own published 51.8. The other fourteen benchmarks do not qualify. The published score used a different scorer or judge (CRAG, MedAgentsBench, OOLONG), a different agent configuration and judge prompt (SSTQA), a different protocol or instance set (GISA: best of three runs; MortarBench: an unreleased set, three runs), or an agent that cannot be re-run as published (FrontierScience, InsightBench, and MoCA* on FinanceMath and MultiHiertt); or the paper

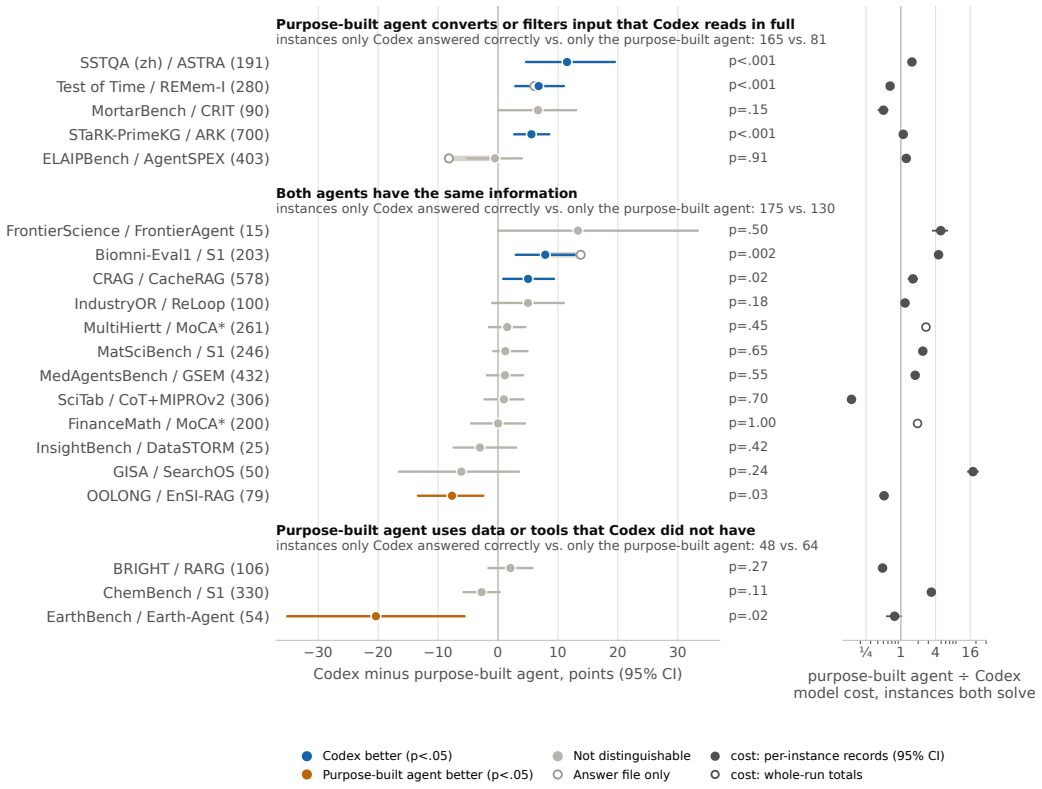

Figure 4: **Paired differences and model cost on the 20 benchmarks.** Left: Codex minus the purpose-built agent with 95% CI, grouped as in Table 1. Each group header gives the pooled number of instances won by only one agent. Hollow markers show the difference when only each agent's answer file is read. Right: the purpose-built agent's model cost relative to Codex. Filled markers use per-instance cost records on instances both solve, with 95% CIs. Hollow markers (MultiHiertt, FinanceMath) use whole-run totals over all instances. The six cheaper agents hand work to a smaller model (REMem-I, EnSI-RAG), a precomputed index (RARG), a single prompt (CoT+MIPROv2), or a few calls (CRIT, Earth-Agent). InsightBench has no cost record for our DataSTORM run.

does not state details that could move the score by more than a point, or our setup could not match them (Biomni-Eval1, ChemBench, MatSciBench, and EarthBench).

Table 13: Purpose-built agents' published scores and our re-runs of the same released agents on `gpt-5.6-sol`, on the six benchmarks where the instance population, metric, and scorer match. 95% bootstrap intervals; the published backbone is in parentheses.

Benchmark	Purpose-built agent	Published	Our re-run [95% CI]	Change
STaRK-PrimeKG	ARK	48.2 (GPT-4.1)	67.3 [62.9, 71.6]	+19.1
BRIGHT	RARG	51.8 (GPT-5.4-mini)	60.6 [54.7, 66.6]	+8.9
IndustryOR	ReLoop	68.0 (Claude Opus 4.6)	70.0 [61.0, 79.0]	+2.0
ELAIPBench	AgentSPEX	43.7 (GPT-5, high effort)	45.7 [39.8, 51.4]	+2.0
SciTab	CoT+MIPROv2	70.6 (GPT-4o)	70.6 [61.8, 79.4]	0.0
Test of Time	REMem-I	93.1 (GPT-4.1-mini)	82.5 [77.9, 86.8]	−10.6

Where answers are read. On ELAIPBench, 71 Codex answers never reached the answer file. Reading Codex's final message recovers 31 correct ones. Reading only the answer file would give AgentSPEX a significant lead (34 vs. 67 instances won by only one agent, $p = .001$), whereas with these answers the agents do not differ (40 vs. 42). On Biomni-Eval1, reading only the answer file would widen Codex's lead (32 vs. 4 instances won by only one agent, instead of 21 vs. 5), because 13 correct S1 answers appeared only in its report. Such errors strike both agents, so aggregate scores

do not reveal them. They concentrate in a few benchmarks. On the Codex side they are answers left outside ELAIPBench's answer file, and verbatim "I don't know" answers on CRAG. On the purpose-built side they are report-only answers on Biomni-Eval1, REMem-I's "N years" answers on Test of Time, and CacheRAG's rejections of a false premise in words on CRAG. MoCA also rounded beyond FinanceMath's tolerance, and ReLoop failed to extract the objective value that its own code printed on IndustryOR. After alignment, a format or setup issue on Codex's side remains the main reason for 36 of the 275 purpose-built-only wins (Appendix I.5).

Answer-format instructions. The added sentence on Test of Time comes from the prompt of TISER, a baseline in REMem-I's paper. With it, Codex answers 276 of the same 280 questions (98.6, against 89.3 without it); the 9.3-point gain is larger than Codex's 6.8-point lead over REMem-I. In the other direction, our MultiHiertt task adds two rules absent from the benchmark (signed answers and a single value). They cause four Codex failures, which count against Codex; the verdict does not change.

Leakage control. We treat two kinds of material as leaks: a copy of the benchmark or its answer key, and a page that gives this instance's own answer, such as a textbook's worked solution or a homework-help answer. Reading a task's inputs is allowed, including the source paper on the open-book FrontierScience benchmark.

On benchmarks without web access, Codex runs in a container that holds the task's inputs but not its gold answers, and a scan of every Codex trace on these benchmarks found no read of a gold answer. Some Codex runs tried to reach the web anyway, for example with a question's own text on MedAgentsBench, and none received any content.

On benchmarks that permit web access, we scan the trajectories of both agents. On FrontierScience, one trajectory of each agent read a mirrored copy of the rubric. The FrontierAgent question was voided and replaced by a seeded draw, and the Codex question lies outside the paired set. On the 256 sampled MatSciBench instances, Codex read the instance's own worked solution on 16 (and possibly on 4 more), 3 of which it still answered wrongly, and S1 used one on 3 (and read 2 more without giving an answer). On ChemBench, S1 read a homework-help page with one question's answer and still answered it wrongly. On Biomni-Eval1, both agents reached source papers or databases for some tasks, in a few cases the supplementary tables of the instance's own source paper, and Codex's built-in search found a copy of the benchmark.

A leaking instance is either voided from both agents or re-run with the leaking sites blocked, in the container's network and in the purpose-built agents' search tools, and voided from both agents if the new trajectory leaks again. Sites that a purpose-built agent's own tools need stay reachable (NCBI and Europe PMC on Biomni-Eval1), so on those re-runs the scan alone decides. Codex's built-in web search runs on the model provider's servers and cannot be blocked, so these re-runs close only its shell channel, and the scan remains the safeguard. Where a re-run could not block the route of a Codex leak, through its built-in search or through a site that stays reachable, we voided the instance without a re-run. On MatSciBench, 7 of the 15 instances re-run for Codex leaked again, and we voided 3 more without a re-run, because Codex had read their worked solutions through its built-in search. Voiding these 10 from both agents leaves 194 vs. 191 correct answers on 246 ($p = .65$). If we also void the 6 remaining instances on which Codex's trajectory may have read the answer, the agents tie at 188 vs. 188 on 240 ($p = 1.0$). On Biomni-Eval1, Codex's leaks came through its built-in search or through Europe PMC, so we voided these instances from both agents without a re-run. Of the 7 S1 trajectories re-run under this rule (3 on MatSciBench, 3 on Biomni-Eval1, and 1 on ChemBench), one leaked again, and we voided that instance from both agents. In all, trace screening removed 13 of the 216 sampled Biomni-Eval1 instances from both agents, including every instance whose trace reached a leaked answer. Neither agent's trajectory on the replacement FrontierScience question reached a copy of the benchmark or its rubric.

Benchmark defects. Thirty-one InsightBench datasets ship one byte-identical null table upstream, although their goals and gold insights differ. We exclude them from the headline and report all 100 as a check. EarthBench releases two answer keys that disagree on 7 of our 61 sampled tasks, and its key is the output of Earth-Agent's own tool chain. MedAgentsBench's speech-pathology file has its key column shifted by one row, so those 19 sampled questions are wrong for both agents. A

disputable gold answer or scorer, supported by the benchmark's own materials, underlies 33% of Codex's scored failures and 27% of the purpose-built agents' (Appendix I.4).

I.3 Reasons for each agent's wins

Task features and information groups. Instance features such as a numeric answer, a need for computation, or input length predict which agent wins only weakly: a classifier trained on them and tested on a benchmark it has not seen reaches an AUC of 0.54, where 0.5 is chance. The information groups of Table 1 separate the benchmarks more clearly. Codex wins more instances where the purpose-built agent converts or filters the input (165 to 81, $p < .001$) and where both agents have the same information (175 to 130, $p = .01$). In the data-and-tools group the purpose-built agents win 64 instances to 48 ($p = .16$), from EarthBench and ChemBench (net 11 and 9 instances), while BRIGHT nets 4 instances to Codex. On CRAG, both agents work from the benchmark's official mock knowledge-graph API. CacheRAG does not lead where the knowledge-graph results that it retrieves in advance, and falls back on, lack the answer. Of all 578 questions, on the 59 where these results lack the answer Codex scores higher on 6 and CacheRAG on 3 ($p = .45$), and on the 471 where they contain it Codex scores higher on 25 and CacheRAG on 13 ($p = .03$).

Labels. For 201 of the 275 purpose-built-only wins and 169 of the 388 Codex-only wins, the label comes from a direct read of both traces. The remaining labels come from per-instance annotations, some of which assign Codex-only wins by rule from structural fields in the purpose-built agent's records. An independent check re-read a stratified sample of the purpose-built-only labels, 56 of the 66 labels that credit an explicit step (two independent annotators made the other ten), and a random sample of 30 Codex-only labels, 6 of which were wrong. All 275 purpose-built-only wins and all 388 Codex-only wins in Table 1 are annotated.

How much of Codex's lead comes from pipeline losses. Pipeline losses make up 45% of the 388 Codex-only wins (174; 95% CI 25–62%, resampling benchmarks). The share falls to 34% (21–45%) when unsubmitted branches and low-confidence losses between stages count as reasoning errors (Figure 5a). These losses mark the instances Codex wins. They make up 52% of the purpose-built agent's failures on instances only Codex solves, but 19% on instances both agents miss (within-benchmark difference 27 points, Mantel–Haenszel 95% CI 12–38).

Resources behind purpose-built wins. Of the 49 wins with data or tools that only the purpose-built agent used, 8 used data and 41 a tool. On the three benchmarks of the data-and-tools group, 72% of the purpose-built agents' wins (46 of 64; 95% CI 59–93%) fall on instances where they used such data or tools; on the other 17 benchmarks the share is 1% (3 of 211). The data are S1's web search on ChemBench (all five of S1's data wins) and a curated clinical memory (GSEM, 3 wins). The tools are a trained document-ranking model (RARG, 22 wins), the benchmark authors' own answer-generating tools (Earth-Agent, 14 of its 15 wins), and the RDKit chemistry toolkit (S1, 5 wins). Besides AgentSPEX's re-check, the explicit reasoning steps are a prompt tuned on labeled data (CoT+MIPROv2), re-ranking after reading the documents (RARG), counting record by record (EnSI-RAG), recomputing from raw table cells (MoCA*), checking every stored fact (REMem-I), filtering low-confidence answers (CRIT), abstaining when evidence is missing (CacheRAG), and steps of S1, ARK, and SearchOS that account for six more wins (Figure 2).

Same-resource ablation. To test whether a purpose-built agent's resource helps Codex on instances that agent won, we took such instances, made the resource that agent relies on available in Codex's environment, and re-ran Codex (Table 14). Pooled over five benchmarks, 61 of the 106 instances flipped with the resource installed: Codex answered them correctly or, on BRIGHT, reached or exceeded RARG's nDCG@10 on that query. Codex's answers also vary between runs, so a flip does not by itself show that the resource produced it. Earth-Agent's own tools flipped 14 of 15 EarthBench instances; EarthBench's answer key was itself produced with those tools. The ablation tests whether a resource is sufficient for Codex, not whether it was necessary for the purpose-built agent, which would require withholding it from that agent.

Step-transfer ablation. To test whether the purpose-built agents' steps carry over to Codex, we gave Codex a purpose-built agent's step prompts and procedure verbatim, with a command that

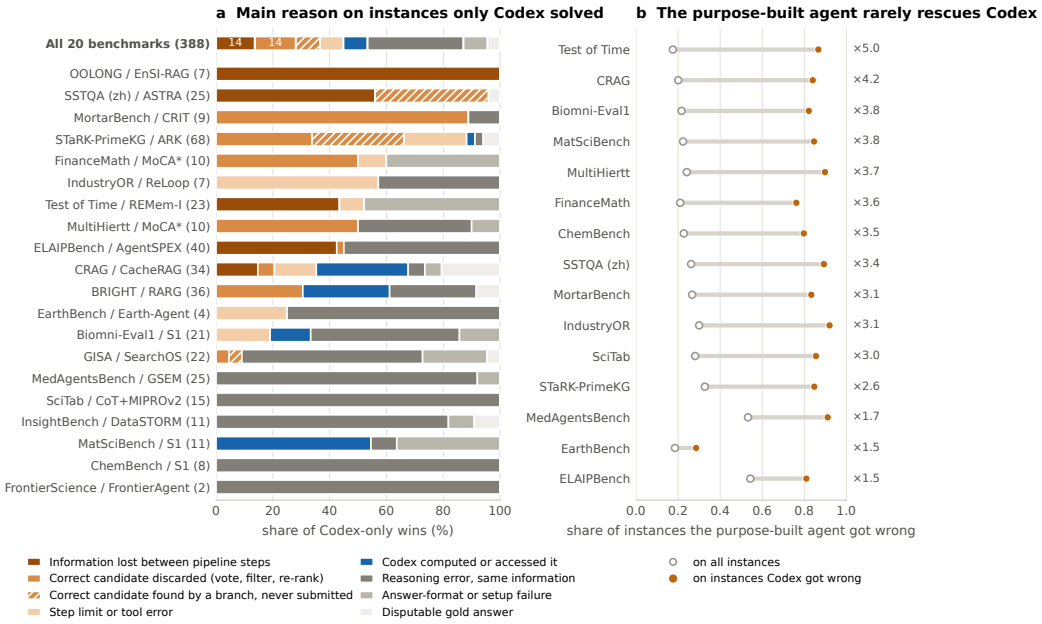

Figure 5: **Where purpose-built agents lose.** (a) The main reason behind each instance that only Codex solved, pooled over all 20 benchmarks and per benchmark, ordered by the share lost inside the purpose-built agent's pipeline (orange). The share exceeds half on seven benchmarks, among them SSTQA (24 of 25), STaRK-PrimeKG (60 of 68), and MortarBench (8 of 9). It is zero for single-call agents over a retrieved memory or tuned prompt (GSEM, CoT+MIPROv2), for S1 on MatSciBench and ChemBench, and for FrontierAgent and DataSTORM. Pooled, it is 45% (174 of 388). Under the strictest labeling rule, which counts unsubmitted branches and low-confidence losses between stages as reasoning errors, it is 34% (95% CI 21–45%). Typical cases: the tree that ASTRA's converter built from one workbook covered 19% of the table, and its symbolic branch found the correct answer on 10 questions it never submitted. ARK ranked the correct entity first in one of its branches on 30 questions that it then lost in the vote, and Codex solved 24 of them (an optimistic comparison that assumes the best branch is known). CRIT's confidence filter removed gold transactions that its own model had rated 4 of 5. (b) The purpose-built agent's error rate on all instances (hollow) and on instances Codex got wrong (filled), with their ratio: the purpose-built agent seldom rescues an instance Codex misses.

Table 14: Same-resource ablation on instances the purpose-built agent won: Codex re-run with that agent's resource installed.

Benchmark	Resource installed for Codex	Flipped / targets
ChemBench	RDKit chemistry toolkit (S1-NexusAgent)	9 / 15
EarthBench	Earth-Agent's own analysis tools	14 / 15
STaRK-PrimeKG	ARK's graph-query tools	16 / 29
MedAgentsBench	GSEM's case-memory recall	10 / 20
BRIGHT	RARG's document-ranking model	12 / 27
Pooled		61 / 106

makes each of the agent's model calls as a separate call with a fresh context, on the model the agent uses for that call. For GISA, the steps are SearchOS's query cleaning and its reformatting of the finished answer into the shape the question asks for, which trims columns the question did not ask for. The targets are a selection of 27 of the purpose-built agents' wins over Codex (instances that the agent answered correctly and Codex did not or, on a continuous metric, on which the agent scored higher), mostly ones where a reading of both agents' traces pointed to one of the agent's steps; none of them came from data or tools that only the purpose-built agent used. An instance flips when

Codex answers it correctly or, on a continuous metric, reaches or exceeds the purpose-built agent's score on it. As for the main results, each answer is read from the answer file or else the final message. Of the 27 targets, 22 flipped, and none of these runs reached a copy of the benchmark or its answers. On one FinanceMath instance, Codex had answered 6.2517, a return in percentage points, where the reference is the decimal fraction 0.063; MoCA's synthesizer prompt asks for the decimal fraction on percentage questions, and with the procedure Codex answered 0.062517. On another, the procedure printed 0.0675, but Codex submitted 6.75. SearchOS's query cleaning restates the question without its output-format demands and lists the required columns, filters, and sort order separately; with it, Codex scored 0.80 on one GISA instance, where Codex had scored 0.10 and SearchOS 0.70. A continuous flip need not be a correct answer: on one OOLONG instance, Codex answered 388 and EnSI-RAG 390, where the reference is 387. Codex's answers also vary between runs, so a flip does not by itself show that the step produced it. Table 15 gives the counts for each benchmark.

Table 15: Step-transfer ablation on instances the purpose-built agent won: Codex given that agent's steps verbatim.

Benchmark	Purpose-built agent	Steps given to Codex	Flipped / targets
FinanceMath	MoCA*	claim catalog, four traders, market clearing, synthesizer	3 / 4
MultiHiertt	MoCA*	claim catalog, four traders, market clearing, synthesizer	4 / 4
OOLONG	EnSI-RAG	per-record labels (one call per batch), exact counts in code	8 / 9
GISA	SearchOS	query cleaning; answer reformatting with column trimming	7 / 10
Pooled			22 / 27

Explicit steps. AgentSPEX's re-check is the main reason for 16 of its wins over Codex. When AgentSPEX selects fewer than two options on a multi-answer question, it re-checks its answer. On all 403 ELAIPBench instances the step runs on 101, turns 40 wrong answers right, and turns 2 right answers wrong. Without it, AgentSPEX would answer 146 instances correctly instead of 184, below Codex's 182. General-purpose additions helped less: ARK's multi-branch search and vote are the main reason for only 2 of its 29 wins on STaRK-PrimeKG, and ReLoop's repair stages add one correct answer in 100.

Added steps that help and hurt. On ELAIPBench, 18 of Codex's 40 wins are answers that AgentSPEX lost in its own pipeline, while its re-check turns 40 wrong answers right; the benchmark shows no significant difference. CRIT adds a rewritten instruction and a confidence filter to MortarBench's released baseline agent. Removing both recovers 8 of the 9 instances Codex won and none of the 3 CRIT won, so the additions come out even (66 vs. 67 of 90, $p = 1.0$). Raising AgentSPEX's reasoning effort from medium to high does not significantly help on ELAIPBench: it answers 190 of the 403 instances correctly instead of 184 (21 vs. 15 won by only one setting, $p = .41$).

Computation and answer revisions. Writing code does not separate Codex's wins from its losses. On the ten benchmarks where it computes, it did so before 121 of its 202 wins and before 82 of its 144 losses (within-benchmark odds ratio 1.4, 95% CI 0.8–2.6, $p = .33$). Codex also answers once: only 1.4% of its runs (60 of the 4,149 whose steps we could parse) rewrite an answer, and none of the 42 changed answers turns a right answer wrong. On ELAIPBench the task states that multi-answer questions have two or three correct options, yet Codex submitted one option or none on 25 of the 315 such instances, 24 of them wrongly; this is the error that AgentSPEX's re-check catches.

Web access on FrontierScience. Two Codex runs on the same 59 FrontierScience questions that differ only in web access go from 26 to 39 correct (15 vs. 2, $p = .002$). In 13 of the 15 gains, Codex retrieved the paper the question was drawn from, which this open-book benchmark permits.

I.4 Failure patterns

Scope. We annotated the traces of all 20 benchmarks on the same instance sets as Table 1 (4,649 instances). Codex fails 1,301 of them and the purpose-built agents 1,408. Binary metrics count

wrong answers, CRAG counts truthfulness below 1, and BRIGHT, OOLONG, and GISA count scores below 0.5. Each failing agent–instance pair is labeled once, from that agent's trace.

Codebook. There are twelve patterns in five groups, plus a separate category for disputable gold answers. *Missing information*: the information was not accessible to the agent, or it was accessible but never found. *Information lost*: content was lost or distorted by a processing step (conversion, extraction, or snippet or node selection), or a correct candidate the agent had committed to was dropped by its own selection, vote, filter, or re-ranking. *Reasoning*: the agent misread the question or a condition, drew a wrong conclusion with the information in view, or made a computation error. *Stopped early*: the agent hit its own step limit, its own tool error left no answer, or it declined to answer although the question was answerable. *Format or setup*: the right content was in the wrong answer location or format, or our deployment failed. The separate category, *disputable gold or scorer*, requires evidence from the benchmark's own materials: a later release, the question's source contradicting its key, information absent from the input, a literal ambiguity in the prompt, or a scorer rejecting an equivalent answer. It is excluded from the agents' own failures. The label is the earliest cause in the agent's run without which the instance would have been answered correctly.

Reliability. For most failure labels (2,409 of 2,709), a second annotator independently labeled a stratified sample of 391 failures and agreed on 85% ($\kappa = 0.79$, 95% CI 0.74–0.84); by agent, κ is 0.75 for Codex and 0.82 for the purpose-built agents, and at the group level it is 0.80. A third annotator adjudicated the 60 disagreements from the traces, and the label matched the adjudicated one on 375 of the 391 (96%). Every disputable-gold label among them was checked against the benchmark's own materials. The other labels were each made by two independent LLM-agent annotators, who agreed before adjudication with $\kappa = 0.74$ on 300 failures, 0.65 on 63 purpose-built-only wins, and 0.69 on 78 Codex-only wins; a third annotator adjudicated the disagreements.

Table 16: Share of each agent's own failures by pattern, pooled over 20 benchmarks (%, 95% CI from resampling benchmarks), and the difference Codex minus purpose-built agent in points.

Pattern	Codex (876)	Purpose-built (1,027)	Difference
Not accessible	4.5 [0, 11]	1.5 [0, 4]	+3.0 [−1, 9]
Accessible but not found	1.6 [0, 4]	4.6 [1, 9]	−3.0 [−7, −1]
Lost in processing	0.2 [0, 1]	10.7 [2, 21]	−10.5 [−21, −2]
Correct candidate dropped	12.2 [1, 32]	13.8 [3, 30]	−1.6 [−7, 4]
Task misread	11.9 [5, 20]	9.9 [5, 17]	+1.9 [−2, 6]
Wrong conclusion	58.9 [31, 77]	47.6 [23, 68]	+11.3 [5, 17]
Computation	1.6 [0, 5]	1.0 [0, 3]	+0.6 [−1, 3]
Own step limit	0.0 [0, 0]	2.3 [0, 6]	−2.3 [−6, 0]
Own tool error	0.0 [0, 0]	0.7 [0, 2]	−0.7 [−2, 0]
Declined to answer	1.3 [0, 4]	0.5 [0, 1]	+0.8 [−1, 3]
Answer format	6.3 [1, 17]	6.2 [2, 16]	0.0 [−2, 3]
Our deployment	1.6 [0, 5]	1.2 [0, 3]	+0.4 [−1, 3]

Groups as shares and as rates. Figure 3 shows the groups as shares of each agent's own failures, Table 16 shows the patterns, and Figure 6 breaks them down by benchmark. The group differences, Codex minus purpose-built agent, are 0.0 points for missing information (95% CI −3.1 to 4.1), −12.1 for information lost (−23.6 to −1.2), and +13.9 for reasoning (4.8 to 22.9). Shares of a whole move together, so we also give rates on a common denominator: failures per 100 paired instances (4,649 instances, resampling benchmarks). Codex and the purpose-built agents fail by missing information 1.1 and 1.3 times per 100 instances, by information lost 2.3 and 5.4 times, by reasoning 13.6 and 12.9 times, and by stopping early or format and setup 1.7 and 2.4 times. The differences, purpose-built agent minus Codex, are +0.2 for missing information (95% CI −0.7 to +0.8), +3.1 for information lost (+1.1 to +5.9), −0.7 for reasoning (−3.2 to +1.1), and +0.7 for stopping early or format and setup (−0.3 to +1.7). For information lost, both views separate the agents at the 95% level; for missing information, neither does. For reasoning, only the share view separates them: its larger share for Codex follows from the purpose-built agents' larger share of information lost, and the rates do not differ.

Shared and exclusive failures. When Codex is wrong, the purpose-built agent is also wrong on 83% of instances (1,081 of 1,301), 2.7 times its base error rate, and when both fail, they give the

same wrong answer on 72% of the instances whose answers can be compared (568 of 787). All 39 of Codex's failures for information it could not access fall on ChemBench (20), EarthBench (11), CRAG (7), and ELAIPBench (1). On the first two, the purpose-built agent used network access or tools of its own. The seven CRAG questions ask for values that the benchmark's knowledge graph does not hold, such as a price on a day with no record, and CacheRAG fails them for the same reason. The ELAIPBench question asks for knowledge beyond its paper, which neither agent could look up, and AgentSPEX also fails it. Of the 658 instances that both agents fail on their own, 524 (80%) carry the same pattern for both agents and 549 (83%) the same group. On instances that only one agent fails, missing information accounts for 10% (1–23%) of Codex's failures and 6% of the purpose-built agents'. Information lost accounts for 19% and 43% (24–57%). Some disputable gold answers rest on a question's source contradicting its key. Counting these as the agents' own reasoning errors leaves the group contrasts almost unchanged: missing information 0 points (95% CI -3 to $+4$) and information lost -12 points (-22 to -2).

I.5 Further details of the results

Shared backbone. A gap to a published score does not isolate the design of the agent. Each published agent used its own backbone model (the LLM that drives the agent) and its own evaluation protocol, and models differ in how much of a benchmark's public material they saw during training, a problem known as knowledge contamination. We therefore re-run each purpose-built agent from its released code with Codex's backbone, `gpt-5.6-sol`, on the same instances as Codex: all instances on the five benchmarks that both agents ran in full, and otherwise a seeded random sample fixed before the purpose-built agent ran, 25% of the benchmark in most cases. Both agents then draw on the same model, and the re-runs give us the traces of both agents, which RQ4 and RQ5 analyze. We call a difference significant when the paired test (exact McNemar, or sign-flip for continuous metrics) gives $p < .05$, and call the other benchmarks not distinguishable (Appendix I.1).

Evaluation checks (RQ1). A shared backbone removes differences in the model, but a comparison can still mislead in three ways. The published setup may not match the benchmark, or the released code may not reproduce the published score. An agent with web access may find the answer online instead of working it out. And the scorer may miss a correct answer that the agent gave in an unexpected place or format. We addressed each concern, and each changed our results (Appendix I.2 gives the details).

Published setups often do not match the benchmark they report on, and re-running the released code does not always reproduce the published score. We judged 22 of 25 published comparisons misaligned with their benchmark, for example in protocol, metric, or tools (Appendix G).

Leaked answers are the second concern. Where a task needs the web, agents found copies of the benchmark on Hugging Face and GitHub, and answers on sites such as homework-help forums and online textbooks. We scanned every trace of both agents in Table 1 and re-ran a leaking instance with those sites blocked wherever a re-run could block them. Trace screening removed instances from both agents, including every instance that leaked again or whose leak could not be blocked. Such pages matter: on the 256 MatSciBench instances we sampled, Codex had read the instance's own worked solution on 7 of its 17 wins (and possibly on 2 more) before we removed the leaks. Without web access, Codex ran in containers that held no gold answers, and no trace shows it reading one.

The third concern is where each answer is read. Reading only the answer file, and not the final message or report of an agent that wrote none, produced a significant AgentSPEX lead on ELAIPBench, where the agents do not differ, and widened Codex's lead on Biomni-Eval1 ($+13.8$ instead of $+7.9$ points). Answer format matters as well. One added sentence on answer format raises Codex on Test of Time from 89.3 to 98.6, more than its lead over REMem-I; Table 1 keeps the original prompt.

Comparative performance (RQ2). Sharing the backbone changes the purpose-built agents' own scores. On the 6 benchmarks where our re-run covers the same instances, metric, and released agent as the published result, the purpose-built agent's score changed by -10.6 to $+19.1$ points. ARK rose from 48.2 to 67.3 on STaRK-PrimeKG and RARG from 51.8 to 60.6 on BRIGHT, while REMem-I fell from 93.1 to 82.5 on Test of Time; each change is larger than the gap between Codex and the same agent on the shared backbone. On SciTab, IndustryOR, and ELAIPBench, the score moved by 2 points or less (Appendix I.2).

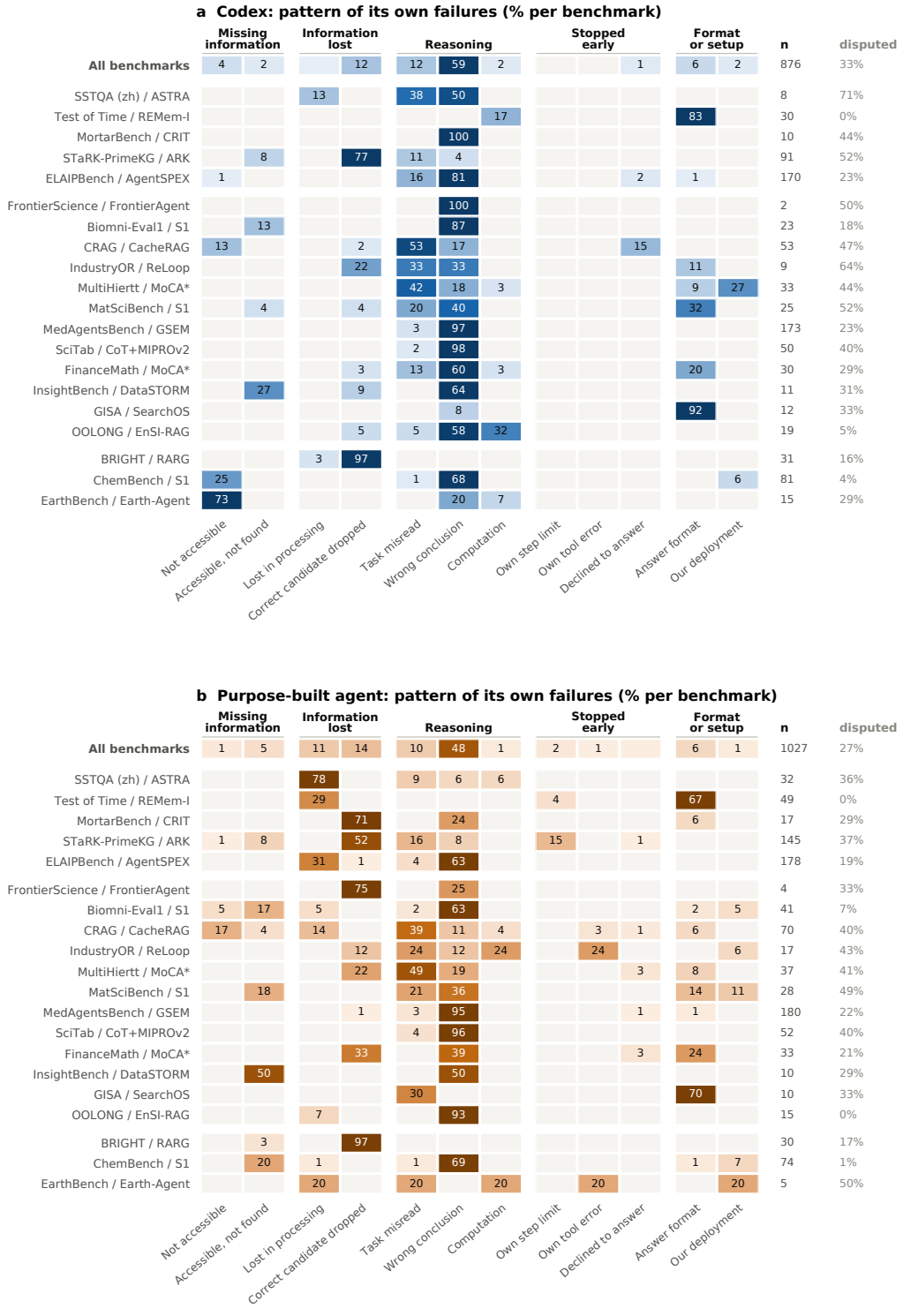

Figure 6: Every failure of Codex (a) and of the purpose-built agent (b) by pattern and benchmark, with the five groups of Figure 3. Cells give the share of that agent's own failures on that benchmark, with disputable gold answers excluded. The right-hand columns give the number of the agent's own failures (n) and the share of its scored failures set aside as a disputable gold answer or scorer (disputed).

As a result, few published gaps survive a shared backbone: only 4 of the 20 keep a significant difference in the same direction. Of the 12 benchmarks on which Codex reaches the published score in Figure 1, 3 remain significant Codex wins in Table 1 (SSTQA, STaRK-PrimeKG, and Biomni-Eval1), 8 become not distinguishable, and 1 turns into a significant loss (OOLONG). Of the 8 on which Codex falls short, 1 remains a significant deficit (EarthBench), 5 become not distinguishable, and 2 reverse: Codex trails the published scores of REMem-I on Test of Time and CacheRAG on CRAG but beats both agents on the same model and instances. A published comparison therefore mixes the design of an agent with its model and setup, which can move a score as much as the design does. The purpose-built agents are also usually more expensive, with a higher model cost on 13 of the 19 benchmarks where we could measure it (Appendix I.1).

Task and domain dependence (RQ3). The purpose-built agent is significantly better on two benchmarks: EarthBench, a scientific benchmark, and OOLONG, a long-context aggregation benchmark. On EarthBench it brought its own analysis tools; on OOLONG it labels each record and counts the labels exactly, the step behind 12 of its 17 wins. Within a domain, the winner varies. Among the six science and medicine benchmarks, Codex leads significantly on Biomni-Eval1, the purpose-built agent on EarthBench, and the other four are not distinguishable; the search and retrieval benchmarks include clear Codex wins (STaRK-PrimeKG and CRAG) and ties (BRIGHT and GISA). Features of an instance, such as a numeric answer, a need for computation, or the length of the input, predict the winner only weakly (Appendix I.3).

After reading the traces, we sorted the benchmarks into three groups by where the information needed for the answer resides and whether each agent can reach it (Table 1). The groups describe the results rather than predict them. On five benchmarks, the purpose-built agent converts, extracts, or filters an input that Codex reads in full. Codex answers 165 instances there that the purpose-built agent misses, and the reverse happens on 81. On twelve, both agents have the same information, and the pooled counts favor Codex (175 instances to 130, $p = .01$); Biomni-Eval1 and CRAG differ significantly in Codex's favor and OOLONG in the purpose-built agent's. These twelve include four science and medicine benchmarks: Biomni-Eval1, FrontierScience, MatSciBench, and MedAgentsBench. On the other three, the purpose-built agent uses a live service, a search index, or a tool of its own, and the pooled counts lean to the purpose-built agent without a significant difference (64 instances to 48, $p = .16$).

Reasons for the purpose-built agents' wins (RQ4). We now pool the instances of all 20 benchmarks in Table 1. The purpose-built agent answers 275 of them correctly where Codex does not, and we recorded the main reason for each from the traces (Figure 2). On 49 of the 275, the purpose-built agent used data or tools that only it had: mostly tools, such as RARG's trained document-ranking model and Earth-Agent's own analysis tools, and a few data sources, such as S1-NexusAgent's web search on ChemBench. On 66, it took an explicit reasoning step on information that both agents had. On 103, both had the same information and reached different answers. The remaining 57 trace to a format or setup issue on Codex's side (36) or a disputable reference answer (21) (Appendix I.3).

The explicit steps encode procedural knowledge rather than capability. The most frequent is AgentSPEX's re-check on ELAIPBench when it selects fewer than two options on a multi-answer question (16 wins); without this step it would score below Codex. Next are EnSI-RAG's record-by-record counting on OOLONG (12 wins) and RARG's re-ranking after it reads the retrieved documents (10 wins). Codex had the data and tools to take the same step in 56 of these 66 instances, but its traces show the step in only 13. Given a purpose-built agent's step prompts and procedure verbatim, Codex reached that agent's result on 22 of 27 selected instances that the agent had won on FinanceMath, MultiHiertt, OOLONG, and GISA (Appendix I.3).

The purpose-built agents also lose information. On 174 of the 388 instances that only Codex answers correctly (45%), the purpose-built agent had lost the answer inside its own pipeline: a conversion step dropped it, a vote or filter discarded it, or a step limit cut the run short. The tree that ASTRA's converter built from one SSTQA spreadsheet, for example, covered only 19% of the table.

Failure groups. In Figure 3, each cell is the share (%) of one agent's failures on a benchmark that fall into a group. Codex (blue) is on the left of each pair, and the purpose-built agent (orange) on the right. *Missing information*: the agent could not access the information it needed, or did not find it. *Information lost*: the agent's own processing lost the information, or a selection step dropped a

correct candidate answer. *Reasoning*: a misread task, a wrong conclusion, or a computation error. *Stopped early*: the agent hit its own step limit, a tool error ended its run, or it declined to answer. *Format or setup*: a correct answer in the wrong place or format, or a fault in our deployment. The top row pools all 20 benchmarks of Table 1 and prints every share. Other rows print shares of at least 10%. n is the number of failures. Disputable gold answers are excluded, and hollow cells rest on fewer than five failures (Appendix I.4).

Codex's strengths and failures (RQ5). Codex's main advantage is that it works on the complete, original input and decides how to process it after reading each question, rather than following a representation fixed in advance. It still breaks long inputs into pieces, but it chooses the pieces for each question. It wrote code to open every SSTQA workbook itself, answered almost every Test of Time question from the full list of facts, and followed the links of the STaRK-PrimeKG knowledge graph itself on most questions.

Figure 3 assigns every failure of each agent to one of five groups. Both agents fail mostly at reasoning (72% of Codex's failures and 59% of the purpose-built agents'), and these errors overlap: when Codex is wrong, the purpose-built agent is also wrong on 83% of instances. The two differ mainly in how they keep information: the purpose-built agents more often lost information they already had (25% of their failures, against 12%), in conversion, extraction, and selection steps, which is also why reasoning makes up a larger share of Codex's failures. Each agent could not access or did not find the information on 6% of its failures. All but 8 of Codex's 39 failures for information it could not access fall on ChemBench and EarthBench, where the purpose-built agent used network access or tools of its own; the other 8, seven CRAG questions and one ELAIPBench question, ask for information that the benchmark itself does not hold, and the purpose-built agent fails them too. Codex also rarely re-checks its work: it revised its answer in only 1.4% of its runs (Appendices I.3 and I.4).

I.6 Implications for agent design

Codex does well when it can work on the full input. Purpose-built agents do well when they bring domain resources or explicit workflow steps. Three lessons follow. First, fixed pipelines trade away information: long inputs often have to be split, but a pipeline that fixes its extraction and filtering steps in advance can drop information the answer depends on, while Codex breaks the full input down as each question requires. Builders of purpose-built agents should analyze this loss of context against what each step adds. Second, much of the purpose-built agents' advantage is portable: domain data and tools, such as a ranking model or a chemistry library, can be given to a general-purpose coding agent as building blocks, and in our ablation they carried much of the benefit without the fixed pipeline. Third, the decisive workflow steps, such as counting records one by one, encode procedural knowledge rather than capability: Codex can execute them but rarely finds them on its own, and once given them it matched the purpose-built agents on most of the selected instances it had lost.

Together, these results suggest a design direction that our ablations test only on instances the purpose-built agents had won. Rather than building ever more complex end-to-end agents, one could expose their strongest components, their tools and their decisive steps, as modular building blocks. A general-purpose coding agent would use them when a task needs them. It would keep its advantage of working on the full input and gain the resources and procedures that the purpose-built agents bring.