UNDERSTANDING AI · WRITTEN EDITION · 2026-09-10
Day 6 · typeset 2026-09-10 08:12Z

What the Application Adds

A trained model reads text and writes text. Everything a product appears to do besides that — searching, opening files, running commands, remembering a previous conversation — is ordinary software deciding what text to place in front of it and what to do with the text it returns. That division decides a great deal about what a system costs to run. It decides less than the industry's own marketing suggests about whether the system is right.

An experiment from 2020 that settles the question of principle

In May 2020 a group at Facebook AI Research, University College London and New York University published a paper introducing a system that consulted an external store of documents before producing an answer. The paper is remembered for coining a term. Its most instructive passage is a short experiment near the end, in section 4.7, headed "Hot-swapping indices".

The design is austere. Eighty-two questions, all of one shape — *"Who is the prime minister of the UK?"* — covering heads of state who had changed between 2016 and 2018. The system answered all eighty-two twice. Nothing was retrained; the model, the questions and the procedure were identical across both runs. One thing varied: which copy of Wikipedia the system was permitted to consult, an index dated 21 December 2016 or one dated 20 December 2018.

The failure case is the more useful half. Given the index from the wrong period, the system answered 4% of the questions correctly in one direction and 12% in the other. Given the index that matched, roughly seven in ten. And the paper's own word for what changed is *predictions*: only 21% of them were the same across the two runs.

One trained system, two document stores, eighty-two questions

21%
of answers identical across the two stores
79% of them changed

70% / 68%
accuracy when the store matched the period
2016 store on 2016 leaders; 2018 store on 2018 leaders

12% / 4%
accuracy when the store was mismatched
2018 store on 2016 leaders; 2016 store on 2018 leaders

Lewis, Perez, Piktus et al., Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, arXiv 2005.11401 version 1, 22 May 2020, section 4.7. Figures read from the explicit v1 PDF.

▼ **Table view**

One trained system, two document stores, eighty-two questions

Measure	Value
of answers identical across the two stores	21%
accuracy when the store matched the period	70% / 68%
accuracy when the store was mismatched	12% / 4%

The authors' reading of their own result, which is a separate matter from the result:

> Our result shows that we can effectively update RAG's behavior with new world knowledge by simply replacing its non-parametric memory.

Two things belong beside those numbers. The system tested was not the prompt-assembly pipeline that later inherited the paper's name: the abstract describes "a general-purpose fine-tuning recipe", and section 2.4 records that the authors "jointly train the retriever and generator components without any direct supervision on what document should be retrieved". And the experiment establishes a dependency rather than an improvement. Supplying the wrong material did not make the system decline to answer. It made it wrong.

Two readings that do not survive their own sources

The first: that consulting a store makes a system correct. The 4% figure is the refutation, and it is not an artefact of a research prototype from six years ago. In 2024 a Stanford-led team conducted what it describes as the first preregistered empirical evaluation of commercial legal research tools — three products from two companies, in a market where vendors had publicly described their systems as "eliminating" hallucinations, "avoid[ing]" them, or delivering "hallucination-free" legal citations.

> We demonstrate that the providers' claims are overstated.

The measured rate was 17% to 33%. The same sentence of the abstract carries the qualifier that must travel with it: hallucinations *were* reduced relative to a general-purpose chatbot. The products were better than the baseline and not what their pages claimed.

The paper also supplies a distinction the subject badly needs. Its coding scheme defines a response as **misgrounded** when "Key factual propositions are cited but the source does not support the claim", and observes that such a response "might be a technically 'grounded' response in the computer-science sense". Retrieval succeeded; a citation is attached; the citation does not support the sentence. The word carries two meanings and only one of them is the one a reader cares about.

The second: that the model barely matters and the application is what does. It is the reading this material most easily produces, and the best available evidence refuses it — set out below, under the question of how much the application actually adds.

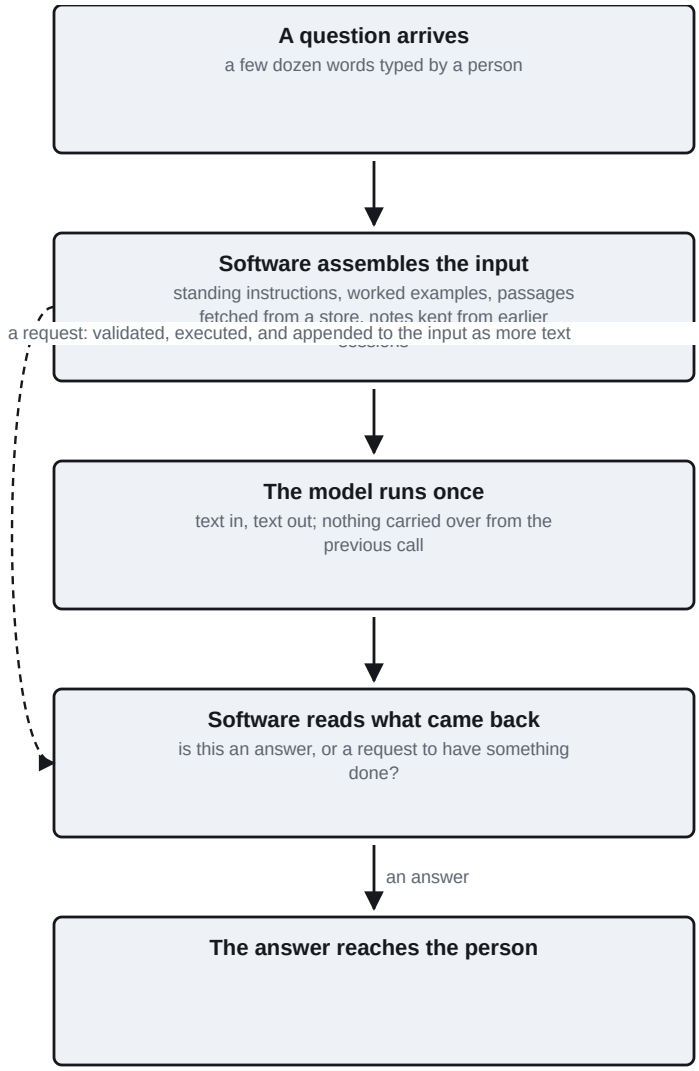
One boundary, and four ways of crossing it

The organising claim is narrow and it is the whole of the subject:

> **A model reads text and writes text. That is the whole of what it does.**

Everything else that a product appears to do is ordinary software on either side of that boundary: software choosing what text to place in front of the model, and software deciding what to do with the text it hands back. The model is a single call. It opens nothing, runs nothing, and retains nothing between calls.

Figure 1. One turn inside a product



Schematic. Every stage but the third is ordinary software, written and controlled by whoever ships the product. The lower edge is the loop that makes a chatbot look like an agent: it can run many times before anything reaches the person.

▼ **Table view**

Figure 1. One turn inside a product — stages

#	Stage	Note
1	A question arrives	a few dozen words typed by a person
2	Software assembles the input	standing instructions, worked examples, passages fetched from a store, notes kept from earlier sessions
3	The model runs once	text in, text out; nothing carried over from the previous call
4	Software reads what came back	is this an answer, or a request to have something done?
5	The answer reaches the person	

Figure 1. One turn inside a product — connections

From	To	Label

From	To	Label
A question arrives	Software assembles the input	
Software assembles the input	The model runs once	
The model runs once	Software reads what came back	
Software reads what came back	The answer reaches the person	an answer
Software assembles the input	Software reads what came back	a request: validated, executed, and appended to the input as more text

Four mechanisms are usually presented as four separate technologies. They are four ways of making the same two decisions.

Worked examples, and what their name concedes

Supplying examples inside a prompt was named "in-context learning" in the paper introducing GPT-3, submitted on 28 May 2020. The paper is careful in a way the term is not. It writes "learning" in its own scare quotes and notes that the curves "involve no gradient updates or fine-tuning". Most usefully, a footnote declines to settle what is happening:

> These terms are intended to remain agnostic on the question of whether the model learns new tasks from scratch at inference time or simply recognizes patterns seen during training

Any account asserting that the model learns from a reader's examples is stronger than the paper that named the phenomenon. The defensible statement is that examples in the current input change what the fixed model does next, and that the effect lasts exactly as long as the application keeps supplying them.

Two years later a team led by Sewon Min tested the point directly, and the result is the sharpest corrective available on the subject. Replacing the answers in the examples with random ones — deliberately showing the model *wrong* worked examples — cost very little:

> we show that ground truth demonstrations are in fact not required — randomly replacing labels in the demonstrations barely hurts performance on a range of classification and multi-choce tasks, consistently over 12 different models including GPT-3

What the examples do carry, on that account, is the shape of the task rather than its content: the paper names the label space, the distribution of the input text, and the overall format of the sequence. The scope of the finding is stated in the same sentence and travels with it — these were classification and multiple-choice tasks, not open-ended writing — and the misspelling of "multi-choce" is the source's own. A reader who takes one thing from this section should take that one: showing a model three examples with the wrong answers attached still mostly works, which is difficult to reconcile with the idea that it learned anything from them.

Retrieval, and a problem named in 1945

The retrieval paper is six days older than the paper that named in-context learning: 22 May 2020 against 28 May 2020, both read from the arXiv submission-history blocks.

The problem it addresses is very much older. In July 1945 the engineer Vannevar Bush published an essay in *The Atlantic* arguing that stored records are hard to reach because filing is alphabetical and minds are not:

> Selection by association, rather than indexing, may yet be mechanized.

The sentences immediately after that one describe the machine he had in mind, and they cover two of the four mechanisms at once:

> A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.

Finding and remembering, one device, in an essay that presupposes no computing at all. Bush invented none of the technique: the lineage runs through Luhn's statistical indexing, Maron and Kuhns on probabilistic indexing in the *Journal of the ACM* in 1960, Spärck Jones on term specificity in 1972, the vector space model, and the BM25 family. What he contributed was the statement of the problem that this layer still exists to solve.

Fifteen years later J. C. R. Licklider put a number on the same complaint by timing his own working week, reporting that about 85% of his thinking time "was spent getting into a position to think, to make a decision, to learn something I needed to know" — one man measuring himself, as the sentence before it concedes: "Perhaps my spectrum is not typical--I hope it is not, but I fear it is." The activities he lists as having consumed it — "searching, calculating, plotting, transforming … preparing the way for a decision or an insight" — are, item for item, what the four mechanisms below automate.

Tools, and who executes

This is the mechanism most often described backwards. When OpenAI shipped function calling into its API on 13 June 2023, the announcement was precise about what the model does:

> have the model intelligently choose to output a JSON object containing arguments to call those functions

The model emits an object. It does not place the call. The clearest statement of that is in the announcement's own worked example, which labels its three steps by whose machine runs them: step one *OpenAI API*, step two **Third party API** — "Use the model response to call your API" — step three *OpenAI API* again. The party with the least interest in drawing that boundary drew it on the day the feature shipped.

Anthropic's current documentation states the same division for client-side tools: "Your code executes the operation and sends back a `tool_result`." The exception is worth naming rather than smoothing away, and the same page names it: some tools are hosted, and for those the provider's own infrastructure executes the operation.

The earliest of the commonly named systems is WebGPT, in December 2021. Its model could emit only commands drawn from a fixed list, and the paper is blunt about what happened otherwise:

> If a model generates any other text, it is considered to be an invalid action.

Whether some earlier language model was given a tool is not a question the record settles; the category is too loosely bounded for a first to be identifiable.

Memory, and where it lives

WebGPT also supplies the clearest early statement of what product memory is. Each of its steps began from a fresh context, so that, in a parenthesis in its own methods section, "the only memory of previous steps is what is recorded in the summary" — a note the surrounding software kept.

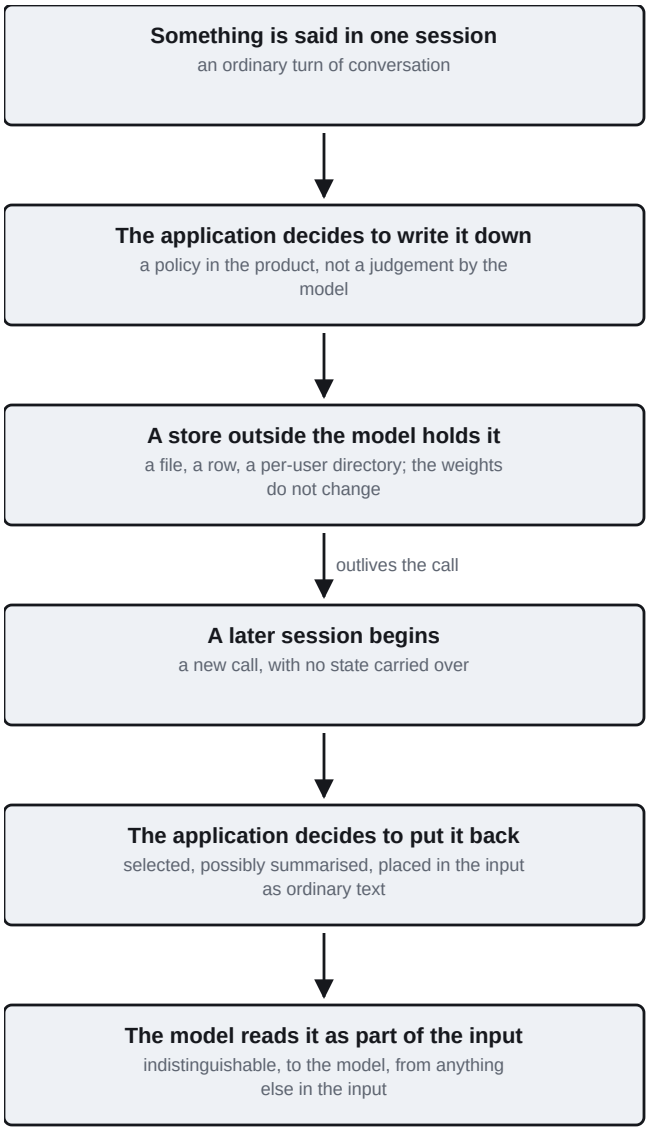
That is still the architecture. Anthropic's memory tool, versioned `memory_20250818`, is documented in these terms:

> Because the memory tool is client-side, Claude only requests memory operations. Your application executes each request against storage you control and returns the result in a `tool_result` block.

Requests, and executes: two verbs, two parties, and never the same party.

Four distinct things travel under the single word *memory*: the weights, fixed during an ordinary call; the current input, which lasts one call; product conversation state, stored but not necessarily shown to the model; and durable external storage with its own write, retrieval and deletion paths. Persistence is not visibility. A product may retain more than it shows the model, and show the model a summary rather than the original.

Figure 2. How a product "remembers" a person



Both decisions — whether to write, and whether to put back — belong to the application. Since retained state is information about a person, a memory feature is worth only as much as the ability to inspect, correct and delete what it holds.

▼ **Table view**

Figure 2. How a product "remembers" a person — stages

#	Stage	Note
1	Something is said in one session	an ordinary turn of conversation
2	The application decides to write it down	a policy in the product, not a judgement by the model
3	A store outside the model holds it	a file, a row, a per-user directory; the weights do not change
4	A later session begins	a new call, with no state carried over
5	The application decides to put it back	selected, possibly summarised, placed in the input as ordinary text
6	The model reads it as part of the input	indistinguishable, to the model, from anything else in the input

Figure 2. How a product "remembers" a person — connections

From	To	Label
Something is said in one session	The application decides to write it down	
The application decides to write it down	A store outside the model holds it	
A store outside the model holds it	A later session begins	outlives the call
A later session begins	The application decides to put it back	
The application decides to put it back	The model reads it as part of the input	

That is also where the subject stops being purely technical. Retained state is information about a person; a system that accumulates more of it is not thereby better.

A short history of choosing the text

Date	What happened
July 1945	Vannevar Bush names the retrieval problem in *The Atlantic* and sketches the memex
22 May 2020	Lewis et al. publish the retrieval-augmented generation paper — a fine-tuning recipe, retriever and generator trained together
28 May 2020	Brown et al. name "in-context learning", and decline in a footnote to say the model learns
17 December 2021	WebGPT: a model emits browser commands from a list of ten; anything else is an invalid action
25 February 2022	Min et al. show that randomising the answers in those examples barely hurts performance
6 October 2022	ReAct interleaves reasoning traces with actions
9 February 2023	Toolformer trains a model to decide when to call an API
13 June 2023	OpenAI ships typed function proposals in its API, with the execution step labelled *Third party API*
25 November 2024	Anthropic publishes the Model Context Protocol
1 September 2026	Anthropic withdraws the ability to compel a tool call on its two newest models

The Model Context Protocol is frequently misread as a capability launch. Its own announcement frames the problem as an integration cost — "Every new data source requires its own custom implementation, making truly connected systems difficult to scale" — rather than a new ability. A connector standard settles how a host discovers and reaches a tool. It does not choose an action and it does not authorise one.

How much does the application actually add?

Here the evidence is genuinely two-sided, and both halves are set out because both are real.

The large observational grid

In January 2026 a group led from Stanford and the Laude Institute published a benchmark of command-line agents together with its results: six products, sixteen frontier models, 32,155 trials, reported with 95% confidence intervals. The author list runs to roughly seventy names and includes researchers at Anthropic, Moonshot AI, Tencent and SambaNova — which is worth stating in a piece about whether vendors tune their scaffolds to their own models, because it is the same paper that says they do:

> Because Terminal-Bench is an interactive framework, agent and model performance are hard to decouple. Many agent scaffolds have been engineered to accommodate the tendencies of certain models, especially when the model and agent are developed by the same organization.

Fifty-five of the ninety-six possible model-by-product cells are populated, and sixteen models were run through more than one product. The best-to-worst spread for a single model runs from 0.3 points to 16.9. Nine of those sixteen spreads are wider than the two confidence intervals combined; for the other seven the grid does not establish that the product mattered at all. The widest case:

Figure 3. One model, four products: Gemini 2.5 Pro on Terminal-Bench 2.0

arXiv 2601.11868 v1, 17 January 2026, Table 2, re-parsed from the explicit v1 PDF on 10 September 2026. Published 95% intervals: ±3.0, ±2.5, ±2.9, ±2.6. Terminus 2 is the benchmark team's own scaffold, which the paper describes as "a neutral testbed for comparing model performance".

▼ **Table view**

Figure 3. One model, four products: Gemini 2.5 Pro on Terminal-Bench 2.0

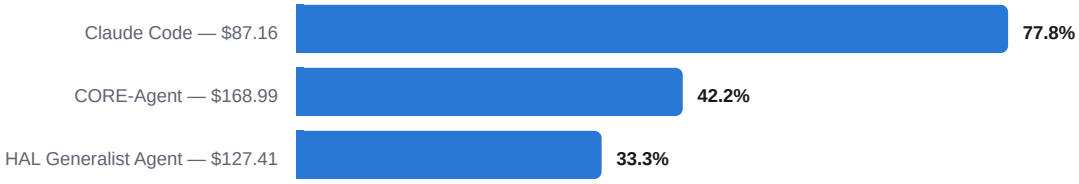
Product (scaffold)	Resolved
Terminus 2 (neutral testbed)	32.6%
Mini-SWE-Agent	26.1%
Gemini CLI (the model vendor's own)	19.6%
OpenHands	15.7%

The direction is not fixed, and reporting only this case would misrepresent the table. In the same grid GPT-5.2 scores 62.9% inside OpenAI's own Codex CLI against 54.0% in the neutral scaffold — the vendor's product ahead by 8.9 points. A vendor's own harness beating a neutral one and losing to it are both in the published results, four rows apart.

A second board, outside coding, where the vendor's product wins by 35 points

The Holistic Agent Leaderboard, run at Princeton, publishes a board for CORE-Bench Hard, a benchmark that "evaluates the ability of agents to computationally reproduce the results of published scientific papers" — not a coding benchmark. Read on 10 September 2026, its rows for one model:

Figure 4. Claude Opus 4.5 on CORE-Bench Hard, three scaffolds



hal.cs.princeton.edu/corebench_hard, read live 10 September 2026. All three rows are marked verified — reproduced by the HAL team. The top row is labelled "Claude Opus 4.5" and the other two "Claude Opus 4.5 (November 2025)"; the board does not state whether the labels denote different snapshots. The top row also carries a second figure, 95.5%, after manual re-grading; 77.78% is the automated score and the comparable one. Costs are the board's own totals for a full run.

▼ Table view

Figure 4. Claude Opus 4.5 on CORE-Bench Hard, three scaffolds

Scaffold	Accuracy
Claude Code — $87.16	77.8%
CORE-Agent — $168.99	42.2%
HAL Generalist Agent — $127.41	33.3%

A 44-point range on one model, and the cheapest configuration is also the best. The board's own note records what followed:

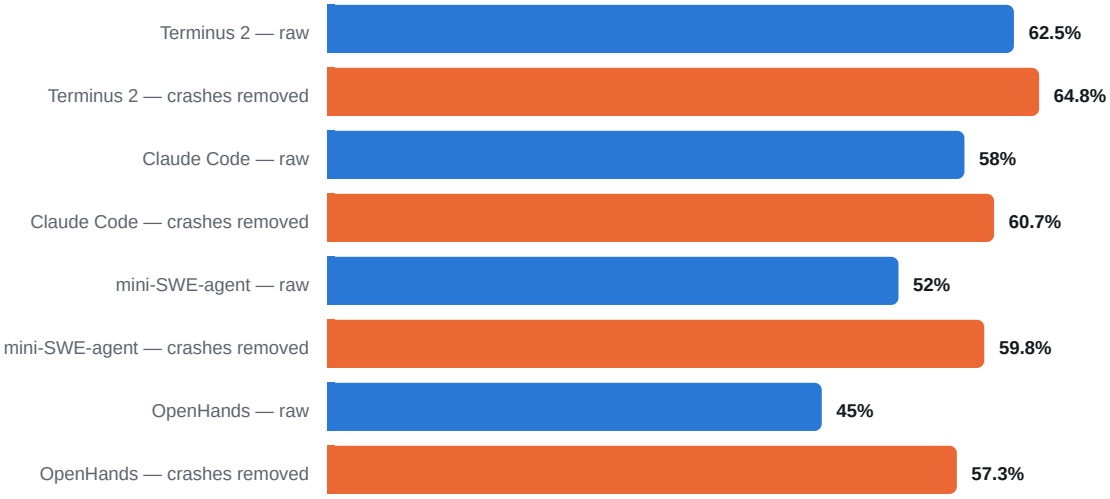
> Update: Running Opus 4.5 with an updated scaffold that uses Claude Code (submitted by Nicholas Carlini) drastically outperforms the CORE-Agent scaffold we used, especially after fixing a few grading errors via manual scoring. We have now declared CORE-Bench solved.

Changing the software around a fixed model did not merely move a number. It moved the conclusion the maintainers drew about whether the benchmark was finished. Two things belong beside that. The scaffold was submitted by Nicholas Carlini, who is also an author of the Terminal-Bench paper above. And this board accepts open community submissions, where the Terminal-Bench 2.1 board states that "Community submissions are currently closed for Terminal-Bench 2.1. Only submissions run by the maintainers will be added to the leaderboard at this time" — two defensible policies with opposite failure modes, one exposed to vendor tuning and one to maintainer bandwidth.

The correction that shrinks the effect by more than half

A team from University College London, Nanjing University and Tencent built a benchmark of 1,530 terminal tasks reverse-engineered from 80,870 real recorded sessions, and reported one model across four applications twice: once as a raw pass rate, and once with the tasks removed on which the evaluation harness failed before the agent got a turn at all.

Figure 5. Claude Opus 4.7 across four applications, before and after removing harness crashes



arXiv 2605.22535 v1, 21 May 2026, Table 2, read from the v1 PDF. The paper's dagger footnote defines the second figure: "resolved rate = pass / (total - errors), where errors are tasks on which the Harbor evaluation harness failed before the agent could attempt the task". The raw spread is 17.5 points; the adjusted spread is 7.5.

▼ **Table view**

Figure 5. Claude Opus 4.7 across four applications, before and after removing harness crashes

Application	Pass rate
Terminus 2 — raw	62.5%
Terminus 2 — crashes removed	64.8%
Claude Code — raw	58%
Claude Code — crashes removed	60.7%
mini-SWE-agent — raw	52%
mini-SWE-agent — crashes removed	59.8%
OpenHands — raw	45%
OpenHands — crashes removed	57.3%

The raw spread across four applications is 17.5 points. Once the tasks are removed on which the evaluation harness never got the agent started, it is 7.5. More than half of the apparent difference between these applications is therefore not the model performing differently on tasks it attempted; it is runs that never began, and they were not evenly distributed across the four. The failures are recorded against Terminal-

Bench's Harbor harness, which ran all four, rather than against the four applications themselves — which is a fact about where the errors were counted, not an account of what caused them. What did not shrink is the bill: cost per solved task across those four rows runs from $0.51 to $4.12, a factor of eight.

The authors' own conclusion, with their own hedge:

> The results suggest that agent frameworks drive cost-effectiveness rather than shifting the model's capability ceiling.

The same paper measures how well the earlier benchmark predicts its own: Terminal-Bench scores correlate with performance on the new set at a Pearson r of 0.20. That does not make Terminal-Bench wrong; it makes it narrow, and a reader weighing the grid above should weigh that alongside it.

The controlled study

Naman Vats and Oleg Golev of Sentient Labs held the model fixed and — the decisive move — held the system prompt template largely fixed as well, so that what varied was the harness: three harnesses, two models, fifty tasks, 300 trials.

> Harness choice induces up to a 40× difference in tokens per solved task, while paired within-model pass-rate differences remain 0–8 percentage points (95% paired-task bootstrap CIs include zero except for the largest gap).

Their section 4.1 puts it more starkly: "At n = 50, most pairwise pass-rate differences are not statistically distinguishable from zero." What they did not standardise is stated in the same section and is the phenomenon under study: each harness's native tool interface, its automatic context pre-loading, and its internal retry logic. The work is a preprint, and its own arXiv comments describe it as "Preliminary work; under review".

Figure 6. The same question, measured four ways

0.3 – 16.9
points between best and worst product, same model
16 models on Terminal-Bench 2.0; 9 spreads exceed the combined intervals

44
points across three scaffolds, same model
CORE-Bench Hard, and the best was the cheapest

17.5 → 7.5
points, before and after removing harness crashes
TerminalWorld, four applications, one model

0 – 8
points under experimental control
intervals include zero except for the largest gap

Sources in order: arXiv 2601.11868 v1; hal.cs.princeton.edu/corebench_hard read 10 Sep 2026; arXiv 2605.22535 v1; Vats and Golev, arXiv 2607.22585 v1. The cost figures attached to each — 8×, 40×, and a run costing half as much as a worse one — move in one direction where the accuracy figures do not.

▼ **Table view**

Figure 6. The same question, measured four ways

Measure	Value
points between best and worst product, same model	0.3 – 16.9
points across three scaffolds, same model	44
points, before and after removing harness crashes	17.5 → 7.5
points under experimental control	0 – 8

What the four together support

The application layer reliably changes what a task costs, by up to one or two orders of magnitude. It changes whether the task succeeds by anywhere between nothing measurable and more than doubling, depending on the pairing — and in the one case where somebody subtracted the harness crashes, more than half of the apparent capability difference turned out to be the harness failing to start rather than the model doing worse. Three separate teams reach a version of that; none of them claims to have separated model from harness cleanly, and the team running the largest comparison says in its methods section that they are hard to decouple.

There is one more thing that column of numbers does not contain. Every measurement above is of coding or terminal work. No comparable published experiment holding a model fixed across several retrieval pipelines, on one corpus and one question set, was found for this edition — which means the mechanism most readers actually meet is the one for which the credit-split question has not been answered.

A threshold is worth applying to these numbers rather than only quoting them. Anthropic's own measurement of infrastructure noise, published 5 February 2026, recommended treating leaderboard gaps under about three percentage points with scepticism until the two setups are known to have matched. Five of the sixteen spreads in the grid above fall under it: Claude Sonnet 4.5 (2.7), Kimi K2 Instruct (2.2), Gemini 2.5 Flash (1.7), Qwen 3 Coder 480B (0.4) and GPT-OSS-20B (0.3). A recommendation that exempts the evidence in front of it is not a recommendation.

What is checkable, and when

A promise, and its quiet withdrawal. A LexisNexis post sits at a URL that still spells out `hallucination-free`. Its capture history is public, and both the headline and the sentence beneath it changed in a single edit that can be dated to a window seven weeks wide:

Capture	Title	The claim sentence
22 May 2024	"...Delivers Hallucination-Free Linked Legal Citations"	"all linked legal citations are **hallucination-free** ... Lexis+ AI delivers **100% hallucination-free** linked legal citations"
24 Mar 2025	"...Delivers \"Hallucination-Free\" Linked Legal Citations"	unchanged
21 Jun 2025	"...Delivers \"Hallucination-Free\" Linked Legal Citations"	unchanged — the last capture of the old wording
10 Aug 2025	"...Delivers Trustworthy Linked Legal Citations"	"all linked legal citations are **verified and reliable** ... Lexis+ AI delivers **validated** legal citations"
live, 10 Sep 2026	"...Delivers Trustworthy Linked Legal Citations"	unchanged since 10 Aug 2025

The preregistered study landed on 30 May 2024 and its journal version on 23 April 2025, both before the edit; the two happened in that order, and nothing on the page says one caused the other. The sentence that never changed in any version is the modest one: "No Gen AI tool today can deliver 100% accuracy, regardless of who the provider is."

An affordance withdrawn. Anthropic's release notes for 1 September 2026 record that on its two newest models, `tool_choice` types `any` and `tool` "aren't supported and return a 400 error", while `auto` and `none` are unchanged. Compelling a model to emit a tool request is no longer available on those models; letting it choose, or offering it nothing, is what remains. The same entry adds a turn-scoped system message — `clear_at: "next_user_message"` — which "renders for the current turn only, then stays in the history at no token cost". Both are the subject of this piece appearing in a changelog: one is a control over whether the model may ask, the other a control over which single turn a piece of text is in front of it.

The idea to keep

A model reads text and writes text. Everything else is software: choosing what goes in front of it, and deciding what to do with what comes back. Worked examples, passages fetched from a store, a result returned by a tool, a note kept from a previous session — four names for one operation performed in two directions.

That division is not pedantic. Once an action can read private data, move money or send a message, the boundary that matters is not the object the model produced but the policy that decided to honour it. It is also why two products sharing a model can differ so much, and why the same model can be either better or worse inside the product built by the company that trained it, depending on the pairing.

And the finding that cuts against the enthusiasm: on the available evidence, that software changes what work costs far more reliably than it changes whether the work succeeds — and part of what looks like a capability difference is one harness crashing more often than another.

One consequence follows immediately and is not developed here. The same route that brings useful text into the input can bring text written by someone other than the user — a retrieved document, a web page, a tool result — and the protection is not the model's judgement but the surrounding software declining to act on instructions it finds in data. A single tool call is also not an agent: one request, honoured once, is not a system that decides how long to keep going.

The limits of the evidence

Every credit-split measurement above is of coding, terminal or scientific-reproduction work by developer-facing harnesses; no measured, documented comparison of one model inside two consumer chat products was found, and the figures circulating for such comparisons trace to secondary write-ups without a resolvable primary source. The spreads in those tables are observed differences rather than causal effect estimates: rows were produced by different teams on different dates, and a clean attribution would need a preregistered matrix of fixed model snapshots against fixed scaffolds, on one task set, at matched inference budgets. Terminal-Bench's input-token column cannot support a cost claim — the same product appears at 0.2M input tokens on one model and 256.9M on another, and the paper does not state how cached input was counted — so the cost figures above come from the output column, the dollar totals and the published cost-per-solved-task. The Terminal-Bench paper states 89 tasks in its abstract and 74 in the caption of the table used here, so no task count for it is given above. The claim that function calling in June 2023 changed practice would need adoption data that was not obtained; what is said above is only that it productised typed proposals in a mainstream API on a documented date. WebGPT is the earliest of the systems named here rather than the first language model given a tool, a title the record is too loosely bounded to award. And Min et al.'s random-label result was measured on classification and multiple-choice tasks, which is not the same population as the open-ended work most readers will put examples into.

Sources

Source	Date	Identifier
Lewis, Perez, Piktus et al., *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*	22 May 2020 (v1)	arXiv 2005.11401
Brown et al., *Language Models are Few-Shot Learners*	28 May 2020 (v1)	arXiv 2005.14165
Min, Lyu, Holtzman, Artetxe, Lewis, Hajishirzi, Zettlemoyer, *Rethinking the Role of Demonstrations*	25 Feb 2022 (v1)	arXiv 2202.12837
Vannevar Bush, *As We May Think*, *The Atlantic*	July 1945	digitised original issue, re-fetched 10 Sep 2026
J. C. R. Licklider, *Man-Computer Symbiosis*, IRE Transactions on Human Factors in Electronics HFE-1	March 1960	pages 4–11
Nakano, Hilton et al., *WebGPT*	17 Dec 2021 (v1)	arXiv 2112.09332

Source	Date	Identifier
Yao et al., *ReAct*	6 Oct 2022 (v1)	arXiv 2210.03629
Schick et al., *Toolformer*	9 Feb 2023 (v1)	arXiv 2302.04761
OpenAI, *Function calling and other API updates*	13 June 2023	Wayback capture web/20230615
Anthropic, *Introducing the Model Context Protocol*	25 Nov 2024	anthropic.com/news
Anthropic, tool-use and memory documentation (`memory_20250818`)	read 10 Sep 2026	platform.claude.com
Anthropic, Claude platform release notes	entry of 1 Sep 2026, read 10 Sep 2026	platform.claude.com
Anthropic (Gian Segato), *Quantifying infrastructure noise in agentic coding evals*	5 Feb 2026	anthropic.com/engineering
Magesh, Surani, Dahl, Suzgun, Manning, Ho, *Hallucination-Free?*	30 May 2024 (v1); 23 Apr 2025 online	arXiv 2405.20362; *J. Empirical Legal Studies* 22(2):216–242, DOI 10.1111/jels.12413
Merrill, Shaw, Carlini et al., *Terminal-Bench*	17 Jan 2026 (v1)	arXiv 2601.11868
Terminal-Bench 2.1 leaderboard	read 10 Sep 2026	commit `7131e437` , 20 submissions
Chu, Hu, Jiang, O'Hearn, Barr, Harman, Sarro, Ye et al., *TerminalWorld*	21 May 2026 (v1)	arXiv 2605.22535
Vats and Golev, *The Scaffold Effect in Coding Agents*	submitted 8 June 2026 (v1)	arXiv 2607.22585; under review, 5th DL4C Workshop @ ICML 2026
Holistic Agent Leaderboard, CORE-Bench Hard	read live 10 Sep 2026	hal.cs.princeton.edu/corebench_hard
LexisNexis, *How Lexis+ AI Delivers Trustworthy Linked Legal Citations*	live page + 10 archive captures	read 10 Sep 2026