Run It or It Didn't Happen

Reproduced-Win Benchmarks for Small-Model Tool Routing

Lewis Tham
Unbrowse AI
lewis@unbrowse.ai

Cayden Chik
Unbrowse AI
cayden@unbrowse.ai

June 2026

Abstract

We report a benchmark protocol with one unglamorous rule: a published number is admissible only if a command re-runs it green, honest negatives are recorded alongside wins, and no claim ships whose witness cannot be located and run. Applied to a 0.8B-parameter tool-routing agent, the protocol yields nine reproduced wins and five honest negatives. On the task classes the architecture targets — writing correct code, retrieving unknowable facts, following settled procedures, and routing to distilled specialists — accuracy lifts are large, seed-stable, and live-reproducible: served code-correctness $68\% \to 100\%$, knowledge-via-retrieval $0\% \to 95\%$, hard-reasoning-via-specialist $50\% \to 92\%$, skill-following $63\% \to 93\%$. We then make a narrow, falsifiable argument: a benchmark that scores a model's *own* multi-hop reasoning (e.g. BrowseComp, where this model scores 0/15) is the wrong instrument for a system whose thesis is to *offload* reasoning to tools, retrieval, and execution. The right instrument is answer-correctness *given tools*. We are explicit about the boundary: this holds for tool-completable tasks and **not** for genuinely reasoning-bound ones, where the model's ceiling is real and unmoved.

1 The protocol: a number is a runnable witness

The failure mode this protocol exists to prevent is the silent overclaim — a benchmark number that reads as proven but cannot be re-run. Three rules:

1. **Witness-or-it-didn't-happen.** Every published metric maps to a script that exits 0 exactly when the claim holds, scoring a disjoint held-out set through the *real* generation-and-execution loop (not a mocked stub).

2. **Honest negatives are first-class.** A measured failure is recorded with its real numbers and never hidden; a benchmark suite that only contains wins is suspect.

3. **No fabricated green; verify the harness.** A judge or scorer that fails silently (e.g. returning a tie on every exception) is indistinguishable from a real null until probed — so the harness itself is tested before its verdict is trusted.

This protocol caught a real defect in our own record while writing this note: four headline numbers cited witnesses that were absent from the primary repository. The protocol forced the question; the witnesses were located in a sibling repository, re-run live, and one (code-correctness) was additionally stress-tested for reliability (§3). The lesson is the protocol working, not failing: the discipline surfaced a real gap (witness locality) that prose alone had hidden.

2 Self-improvement results

Same served 0.8B model throughout; the lift comes from *routing* — to a tool, a retrieved fact, a distilled specialist — not from a larger model. Each row in Table 1 is a witness that spawns the server with the relevant adapter and scores a disjoint held-out set; all four were re-run live and green this cycle.

capability	mechanism	baseline → result	held-out
write correct code	distil correct decomposed code traces; execute the output	**68%→100%** (raw base → improved)†	28 items, 7 arithmetic families
know the unknowable	retrieve the formula, run it as code	**0%→95%** (19/20)	20 made-up formulas, 4 families
hard reasoning	a specialist teaches the generalist where self-distillation is blind	**50%→92%** (broad held 100%)‡	n-choose-k + broad families
follow a procedure	retrieve the right skill (100%) and apply it	**63%→93%** (28/30 vs 19/30)	30 tasks, 10 skill types

Table 1: Self-improvement on the task classes the architecture targets. Each row is a re-runnable witness over a disjoint held-out set.

† **Correction (honest baseline).** An earlier statement of the code-correctness row read "25%→100%". That 25% was measured against a *weak code-adapter* baseline, which — being a degraded artifact — scored *below* the raw instruct model. When the baseline is the **raw base model** (the true "before" state, and the only one a bare clone can reproduce without a fixed artifact), it scores **67.9**%, and the improved adapter reaches **100**%. The execute-don't-guess effect is real (+32 points to a perfect score, seed-stable), but the honest lift is 68→100, not 25→100; the dramatic version depended on an unfair baseline. This correction was forced by making the witness bare-clone reproducible — the discipline doing its job on our own number.

‡ **Resolved flag (hard reasoning).** The hard-reasoning row uses a *trained* r1 baseline (50%). We measured whether the raw base model beats it: on the hard families (n-choose-k, lcm) the raw base scores **4.2%** (1/24), far *below* the r1 baseline. So — unlike the code row, where a degraded trained baseline inflated the lift — the trained r1 baseline here is *conservative*: the true raw-base→unified lift is ≈ 4%→92%, larger than 50→92. The row understates rather than overclaims. Witness: `specialist_basebaseline.py`.

The unifying mechanism is **execute, don't guess**: the small model is not asked to *be* the oracle; it is asked to *write the program*, *retrieve the fact*, or *follow the procedure* that produces the answer, which a deterministic executor then verifies.

3 Reliability: a number you can stand on, with its scope

A single 100% on 28 items is not yet trustworthy. We hardened the code-correctness claim two ways. **Seed-stability:** across four seeds (42 items each) the result is 100.0% (± 0) vs a base of 32.7% (± 2) — the "25%" was a low-base seed; the lift is stable, not anecdotal. **Generalization scope (the honest boundary):** on three *unseen* families (gcd, digit-product, vowel-count — code-writable, not trained) the lift is only +11% (67% $\rightarrow$ 78%), below the in-distribution margin. So the large gain is **in-distribution / family-specific**; the base model already writes adequate code for novel families ($\approx$ 67%). The reliable restatement: *on the trained families, served code-correctness is* 100% *(from* $\approx$ 33%*), seed-stable; out of distribution it transfers only partially.* That sentence overclaims nothing and survives scrutiny.

4 The reframe: what to measure for a tool-routing system

A tool-routing agent's product value is: *given a user intent, does the right answer come out, cheaply?* That is what §2 measures. It is **not** the same quantity as *can the model, with no tools, multi-hop reason its way to the answer?* — which is what raw web-agent reasoning benchmarks score.

Our honest score on the latter is a **negative**: 0/15 on BrowseComp. We do not paper over it. But the negative is a property of the *model's* unaided reasoning, and our architecture's entire thesis is to not rely on that — to route to retrieval and execution instead. Scoring such a system by its unaided reasoning is like scoring a calculator by the operator's mental arithmetic: it measures the part the design exists to remove.

So the claim is a reframe, deliberately narrow:

> **For a system whose thesis is execute-don't-guess, the load-bearing benchmark is answer-correctness given tools, not the model's unaided reasoning.** On the former we win decisively and reproducibly; on the latter we report an honest negative, because it scores a capability we offload by design.

4.1 "Don't need to beat," not "don't matter": the tool layer is not the agent

The sharper statement is not that raw-reasoning benchmarks stopped mattering — it is that **they are not our benchmark to beat.** BrowseComp scores an *agent* solving multi-hop web tasks end to end. Unbrowse is not that agent; it is the **tool layer the agent calls** — `resolve` (intent $\rightarrow$ a ranked route), `execute` (route $\rightarrow$ real data), `capture` (browser when no route exists). The benchmark for a tool layer is **tool-reliability**: when the agent routes to a tool, does the correct answer come out? That is exactly the execute-don't-guess suite (§2) and the route-ranking results (§6), all of which Unbrowse wins.

So the division of labor is the whole answer to *"do we need to beat it?"*: **the reasoning is the agent's job; the reliable tool-call is Unbrowse's.** A capable agent that reasons well *plus* Unbrowse's reliable tools is what clears BrowseComp — and the 0.8B experiments show that even a *small* agent clears the tool-completable subset once the tools are reliable. The 0/15 is the small model's reasoning score; it is not a measure of whether the tool layer delivers, and the tool layer is what Unbrowse ships.

This is a claim we have to *earn*, not just assert: it holds only if Unbrowse is honestly positioned and measured as a tool layer. Where a product instead promises end-to-end research from the tiny model alone, the agent-reasoning score is fair game and the honest path is to pair Unbrowse with a

capable agent — never to pretend the 0.8B did the reasoning. With that honesty kept, the answer to *"why don't those benchmarks need to be beaten for us?"* is: **because they grade the caller, and we are the tool the caller holds.**

5 The boundary: where the reframe is FALSE

The reframe is *not* "reasoning benchmarks don't matter." They matter exactly as the measure of the **residual** that tools cannot cover. Two limits hold:

- **Reasoning-bound tasks.** When a task genuinely requires multi-hop inference that no tool or retrieval can shortcut, the small model's ceiling is real and unmoved — distillation cannot transfer a capability the teacher itself lacks, and retrieval cannot fetch a fact that must be *derived*. There, the raw-reasoning score is the right one, and ours is low.
- **Out-of-distribution generation.** §3 shows the code-correctness gain is in-distribution; a benchmark of *novel* task families is a legitimate, harder yardstick we only partly pass.

A reader should leave with the bounded version, never the dismissal: *the benchmarks that measure raw model reasoning are the wrong yardstick for the value a tool-routing system delivers, but the right yardstick for the capability it does not.*

6 Reproduce

Every number above maps to a command. The mechanism wins (route ranking, intent-type classification, energy-based ranker gates, sealed-cache reuse) need no model serving:

```
# discrete-structure / mechanism wins (route ranking, type-class,
# energy-based gates, content-addressed cache reuse)
bash bench/energy/reproduce-all.sh
```

The four execute-don't-guess wins are vendored in this repository. Each spawns the served 0.8B with the relevant adapter and scores a disjoint held-out set through the real generation-and-execution loop:

```
bash   bench/execute-dont-guess/setup.sh         # resolve adapters
python3 bench/execute-dont-guess/codebench_witness.py   # code        68% -> 100%
python3 bench/execute-dont-guess/farformula_witness.py  # knowledge    0% ->  95%
python3 bench/execute-dont-guess/specialist_witness.py  # reasoning   50% ->  92%
python3 bench/execute-dont-guess/skillfollow_witness.py # skill       63% ->  93%
python3 bench/execute-dont-guess/codebench_reliability.py # seed-stability + scope
```

The honest negative is reported by its own gate; we run it to show the limit, not to hide it:

```
# honest negative: the model's unaided multi-hop reasoning ceiling
bash bench/browsecomp/browsecomp-gate.sh          # 0/15
```

A machine-readable ledger of record records the full set — nine reproduced wins, five honest negatives. Every win re-runs green; every negative is witnessed.

7 Conclusion

The contribution is two things, neither of which is a leaderboard victory. First, a benchmark *protocol* that makes a number mean "re-runs green," and that — applied to our own record — caught a real witness-locality defect rather than rubber-stamping it. Second, a *reframe*: for an execute-don't-guess system, measure the answer it delivers given tools (where a 0.8B reaches 100% / 95% / 92% / 93% on its target task classes, seed-stable), and record honestly the raw-reasoning score it does not chase (0/15). We do not claim the older benchmarks stopped mattering. We claim they measure a different quantity than the one this architecture optimizes — and we are willing to be judged by the witnesses, not the prose. The one previously-open item is now resolved by measurement: the hard-reasoning row's baseline is a trained r1 specialist, and the raw base scores only 4.2% on those families — below the baseline — so the row is conservative, not inflated (§2, ‡).