Do You Copy? When Language Models Overwrite Observed Values with Remembered Defaults

Anonymous authors
Paper under double-blind review

Abstract

Coding agents copy values from tool output into the files they write. We show that when the output names a well-known service on a non-default port, such as `redis` on 6392, language models overwrite the observed port with the default they remember. We call this failure default capture. Both values are true, so counterfactual knowledge-conflict benchmarks cannot measure it. We isolate it by renaming the service: the name effect is the real name's rate of writing the remembered default minus the rate under a made-up name of the same length. Under made-up names six instruction-tuned models copy the port from command output almost without exception, so a default written under the real name is attributable to the name. Default capture concentrates where agents meet it: returning command output as a tool result instead of a chat message triples the average effect, while authored configuration files stay near zero. In the preregistered chat test a `docker ps` row exceeds a `.env` line, and three of six models pass the per-model criterion. A header naming the source of the text, as in retrieval-augmented generation, lowers the effect below chat delivery, and under tool delivery dropping the service name from the line being written removes most of it. In a sandboxed agent loop, Qwen3.5-9B with thinking on sets the documented default in 24 percent of real-name episodes and 1 percent under made-up names, DeepSeek V4.1-Flash in under 1 percent of either. Yang (2026) found the same pull in Python import aliases; the six models repeat it, as a preregistered test predicted. Training toward the model's own behaviour under made-up names cuts the effect on a 7B model from 28.8 to 1.0 points under tool delivery on held-out services and layouts, with recall of remembered defaults within 3 points. Every repair trained only toward the page, and every inference-time method we compared that lowers the effect, trades copying the observed value against a second request: what port the service normally uses. Supervising that request too, on prompts rotated through raw, chat and tool form, lets a 1.5B model answer it at 76 to 83 percent in every delivery, against 31 to 47 for the frozen model, with the name effect near zero (exploratory).

1 Introduction

A coding agent that runs `kubectl get svc` and then writes the application's configuration must copy the port it was shown, or the application cannot reach its service. Given output listing a `clickhouse` service on port 8136, gemma-4-31B-it wrote `8123"  # Default`, ClickHouse's remembered default, annotated as such. Both numbers are true: 8123 is the documented default and 8136 the observed value here, so the model must keep the general fact and act on the particular one. We call the failure default capture.

Conflicts between memory and context are mostly studied in question answering, where a passage is edited to contradict what the model knows (Longpre et al., 2021; Xu et al., 2024b; Section 5). That setting misses two properties of default capture that matter for agents. First, both numbers are true, so benchmarks built on counterfactual passages cannot measure it. Second, the value arrives as tool output, and pasting the same output into a chat message hides two thirds of the effect on average.

Renaming the service separates default capture from ordinary copying errors. Every prompt is paired with three made-up-name versions of itself, the service name replaced by a pronounceable string of the same length. The name effect Δ_{def} is how often the real-name prompt writes the remembered

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

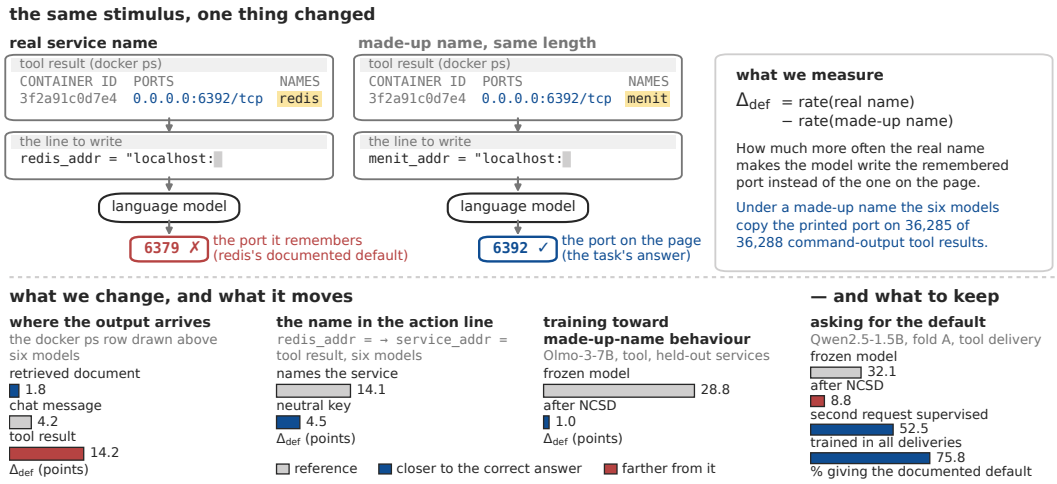


Figure 1: The experiment. **Top**, the measurement: the same tool result and line to write, run twice with only the service name changed. Under the real name the model sometimes writes the port it remembers; under a made-up name of the same length it copies the port on the page, and Δ_{def} is the difference between those two rates. **Bottom**, left to right: where the output arrives (Section 3.2), whether the action line names the service (Section 3.3), training toward the model's own behaviour under made-up names (Olmo-3-7B-Instruct, tool delivery, held-out services, Section 4.3), and asking for the default, where the repair gives up what the frozen model knew until training supervises the second request, most when trained in all deliveries (Qwen2.5-1.5B-Instruct, fold A, tool delivery, Table 7). Bottom-band bars are grey for each column's reference, blue when closer to that column's correct answer and red when farther.

default, minus how often its three made-up-name versions do (Figure 1). We measure it in a pre-registered study of six instruction-tuned models from the Qwen3.5, Gemma 4 and Olmo 3 families (Qwen Team, 2026; Gemma Team, 2026; Team Olmo et al., 2025), 48 services and 17 observation layouts. Under made-up names each model copies the port from command output on more than 99.7 percent of prompts, so a default written under the real name is attributable to the name. The name recalls its own service's default, not a familiar port in general: on command rows under tool delivery, a real name writes that default on 12.0 percent of prompts and one of the other 47 documented defaults on 0.37 percent (Section 3.4). Default capture concentrates where agents meet it, in command output returned by a tool. On command output the name effect triples when a tool returns the output instead of a chat message, from 3.2 to 9.7 points (Figure 2a). On the authored files that appear in both conditions it stays at or below 0.4 points in chat and 1.0 under tools. Each step toward the tool result adds to the effect, and a plain request instead of the preregistered one takes it to 21 and 27 points on `docker ps` and `kubectl` (Section 3). A header naming where the text came from lowers it. Under a retrieved-document header, as in retrieval-augmented generation, the effect falls 2.4 and 3.6 points below chat delivery on the `docker ps` and `kubectl` rows, with no model above it; on an API model's probability scale, every wrapper that names a source lowers it (Section 3.2, Appendix N). The name in the line being written carries most of the effect: replacing it with a neutral key removes 60 to 68 percent under tool delivery (Section 3.3). In a sandboxed agent loop where the model writes the whole file and only the observed port answers, Qwen3.5-9B with thinking on sets the documented default in 24 percent of real-name episodes and 1 percent of made-up-name ones; DeepSeek V4.1-Flash shows no effect (Section 3.4). Yang (2026) showed the same pull in code: when an import binds a conventional alias to another library, models continue with the conventional library's API. The six models repeat it: on unambiguous continuations the name effect is 49.5 points against a made-up alias (Section 3.5).

We remove default capture by training an anchored LoRA adapter toward one of two targets: the frozen model's own predictions under made-up names (name-conditioned self-distillation, NCSD), the teacher Liu et al. (2024b) use to erase a named entity, which we anchor so that the knowledge stays; or the observed value itself (supervised fine-tuning, SFT). On Olmo-3-7B-Instruct, the diagnosed model with the largest effect under tools, NCSD takes the effect on held-out services and layouts from

28.8 to 1.0 points under tool delivery and from 24.6 to 1.7 on raw text, and recall of remembered defaults stays within 3 points on prompts that do not show the service's port (Figure 5). On a 1.5B model, in two folds against six inference-time methods, both targets remove the effect in raw, chat and tool deliveries, while context-aware decoding (Shi et al., 2024) only lowers it and loses 8 to 18 points of recall (Figure 3). Some prompts make a second request: what port the service normally uses. Every repair trained only toward the page, and every inference-time method we compared that lowers the effect, trades the two requests against each other. In exploratory runs on the 1.5B model, supervising the second request with all training prompts rotated through the three deliveries removes the trade-off except on some requests that name the default but ask for the page. The model then answers the second request at 76 to 83 percent in every delivery, against the frozen model's 31 to 47 (Figure 1, right).

Our contributions are:

- We define default capture, a conflict between two true values in which a model overwrites the one it observed with the one it remembers, and isolate it with a renaming control in a preregistered study of six models, 48 services and 17 layouts (Section 2).
- We locate it in tool delivery and in the name in the line being written (Section 3). It survives an agent loop in Qwen3.5-9B but not in DeepSeek V4.1-Flash, and the name's pull recurs in code, as Yang (2026) found.
- We nearly remove it on held-out services and layouts with label-free anchored training toward the model's own behaviour under made-up names (Section 4). Like every inference-time method that lowers the effect, a repair trained only toward the page trades copying against answering a request for the default; supervising that request as well largely removes the trade-off (exploratory, 1.5B model).

2 Measuring default capture

Task. Each stimulus pairs an observation with an action. The page is the observation as the model sees it. The observation states that a service listens on a port other than its documented default, in one of 17 layouts that range from authored files (a `.env` line, an INI section, YAML, JSON) to command output (`docker ps`, `kubectl get svc`, `ss -tlnp`; Appendix C). The action line is the start of the line the model must write, `redis_addr = "localhost:` or its shell `export` form, prefilled after an opening code fence. We classify each output as the observed value, the service's remembered default, or something else. The remembered default and the documented default are the same number: remembered wherever the model produces it from memory, in place of the observed value or on a page that does not show it, and documented where a request asks for it.

Services, names and models. The 48 services are 28 common development and infrastructure services and 20 added under a preregistered rule requiring one documented default port that no other service in the set shares. Each is observed at three values, one of them the default plus 13, and has three made-up names of the same length in characters, substituted for the name everywhere in the prompt (Appendix C). The models are Qwen3.5-9B and Qwen3.5-27B, gemma-4-12B-it and gemma-4-31B-it, and Olmo-3-7B-Instruct and Olmo-3.1-32B-Instruct.

Delivery. Chat delivery and tool delivery both use each model's chat template. In chat delivery one user turn holds "Here is the output I got:", the observation with its command line, and the request "Write ⟨file⟩ for the app. Reply with the file only." In tool delivery the user asks the model to inspect the environment first, an assistant turn calls a declared `run_shell` tool with the layout's command, and the tool result is the observation. We add no system message, leave each template's reasoning mode off, and decode greedily for at most 8 new tokens; Supplement S1 prints example prompts.

Name effect. For one service, one observed value and one action, Δ_{def} is an indicator — 1 if the real-name prompt writes the remembered default, 0 otherwise — minus the share of its three made-up-name prompts that write it. We average over values and actions, then over services, then over models, and report percentage points. Outputs that defer to a shell variable are dropped; the preregistration calls what is left denominator B. Intervals are 95% percentile intervals from 10,000 resamples of services, with the same resample shared across models.

Preregistration. The design, stimuli and analysis code were committed before any diagnosis data were collected. Chat delivery is the confirmatory condition, and its primary contrast passes: `docker ps` minus `.env` is +3.7 points (CI [2.3, 5.3]). The preregistered per-model criterion asks for at least 3 points with an interval above zero; three of the six models meet it, and a preregistered rule then confines the confirmatory claim to their families, Olmo 3 and Qwen3.5; we narrow Qwen3.5 to the 9B. Four of the nine B contrasts survive Holm correction, one against its predicted sign, and the C family, a rank test of prior strength, fails. Tool delivery is a preregistered secondary analysis, and the variants of Sections 3.2 and 3.3 are exploratory. In code, the preregistered alias tests K1 to K3 pass and K4 fails. In the repair of the 1.5B model, the preregistered tests P1, P3 and P4 pass and P2 fails; the 7B repair is exploratory (Appendix A). Table 1 gives the status of each claim family and the outcome of each preregistered test, and Appendix A the main deviations.

3 What drives default capture

3.1 Command output returned by a tool

Tool delivery triples the effect on command output and leaves authored files near zero, across the eleven layouts the diagnosis ran in both conditions (Figure 2a). The four authored layouts among them stay at or below 0.41 in chat and 1.04 under tools, below every command layout in both deliveries. Over the seven command layouts the effect averages 3.2 points in chat and 9.7 under tool delivery, and an exploratory regression with service-clustered standard errors puts the interaction of delivery and layout family at +6.1 points (CI [4.0, 8.2]; Appendix H).

The rise holds on two rows that match real tool output: `kubectl get svc` goes from 6.2 to 13.5 points and `D-MAP`, which publishes a port as `0.0.0.0:6392->6392/tcp`, from 1.9 to 8.1. The preregistered contrast used `DOCKER1` (below, the `docker ps` row), a single-port row printing a host address without that arrow; there the effect goes from 4.1 to 14.3. Five of the six models show the rise. Both Gemma 4 models score zero on `DOCKER1` in chat but 4.2 and 12.5 points under tools, a delivery the preregistered chat test does not measure.

A six-service listing raises the effect whether or not the neighbours sit on their own defaults: under tool delivery, 17.4 points with them on their defaults and 17.5 elsewhere, 3.2 and 3.3 above the bfloat16 regeneration's single-row 14.2 (CIs [0.2, 6.3] and [0.5, 6.1]).

Under tools, 93 percent of the real name's extra errors on command output are remembered numbers. Of 12,096 real-name prompts, 1,168 yield the default and 109 another recognisably remembered port (a second documented default of the same service, another service's default, or the default slightly altered), against 3 non-copies in 36,288 made-up-name prompts (Appendix E).

3.2 Which part of the delivery raises it

Each step toward the tool result raises the effect on both command rows (Figure 2b). Tool delivery adds a tool exchange (a tool schema, the request "Inspect the environment, then write…" and an assistant call) and moves the output into the tool result, which also drops the command line and the repeated request. Two further conditions, run for all six models beside chat and tool delivery in bfloat16 (Appendix I), separate these steps: the preregistered one places the whole tool exchange, with an empty tool result, before the chat message that holds the output, and the other is tool delivery with the chat request.

The tool exchange in front of the chat message adds 3.5 points on the `docker ps` row (CI [1.4, 5.5]) and 3.0 on `kubectl` ([1.4, 4.7]), in four of the six models on each row. Moving the output into the tool result adds 6.5 points ([4.5, 8.9]) on the `docker ps` row, in six of six models, and 4.1 points ([2.3, 6.0]) on `kubectl`, in four. The request matters too: replacing "Inspect the environment, then write…" with the plain chat request adds a further 7.2 points on the `docker ps` row, to 21.4, and 13.4 on `kubectl`, to 26.9. On the `.env` file no step moves it by more than 1.2 points. The move into the tool result on `docker ps` and the request swap on `kubectl` survive resampling models too (Appendix I). On DeepSeek V4.1-Flash's probability scale (Section 3.4), dropping the command line alone, one of the three things the move changes, raises it.

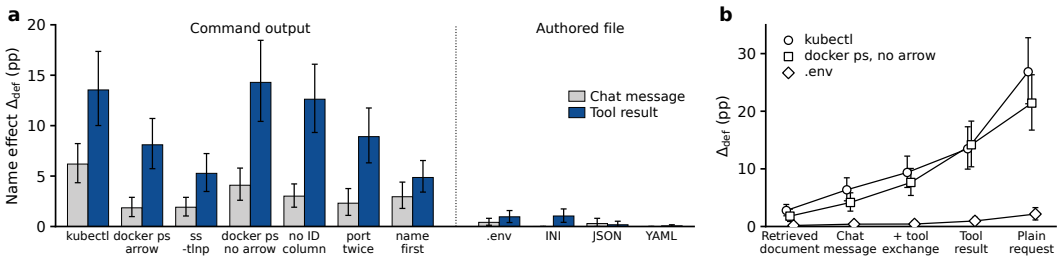


Figure 2: The name effect Δ_{def}: the rate of writing the remembered default with the real service name minus the mean rate over its three made-up names, averaged over 48 services and six models, in percentage points, with 95% service-cluster intervals. (a) Every layout the diagnosis ran in both delivery conditions; command output on the left, authored files on the right; grey bars are chat delivery and dark blue bars tool delivery. (b) The same stimuli under five delivery conditions: the observation as a retrieved document; the chat message of (a); that message with the tool exchange in front of it; the output moved into the tool result, which is tool delivery with the preregistered request; and tool delivery with the plain chat request. The three series are the `kubectl` row, the `docker ps` row without the arrow and the `.env` file. Panel (b) is a bfloat16 regeneration, whose chat and tool cells match panel (a) within 0.2 points.

A header naming the source of the output lowers the effect below chat delivery. Replacing the header of the chat message, "Here is the output I got:", with "Retrieved document 1 of 1 (source: ops-notes.md):" and changing nothing else leaves 1.8 points on the `docker ps` row and 2.8 on `kubectl`. That is 2.4 points ([-3.4, -1.4]) and 3.6 points ([-4.9, -2.4]) below chat delivery, and above it in no model. It is 12.4 points ([9.0, 16.1]) and 10.7 points ([7.8, 14.0]) below the tool result, in six of six. All four gaps survive resampling models too (Appendix I). Made-up names still copy the observed value on 15,550 of this condition's 15,552 command-output prompts, so the drop is a shift from default to observed value. On DeepSeek V4.1-Flash's probability scale for the `kubectl` row, the three wrappers that only restyle the output agree within 1.4 points, and the three that name a source sit 4.4 to 6.6 below the chat header (Appendix N).

3.3 The name in the action line, and the order of port and name

Under tool delivery, the name in the action line carries most of the effect on the `docker ps` and `kubectl` rows. Edits to frozen stimuli in that delivery separate two candidate causes of the gap between command rows and authored files: the name in the prefilled action sits next to the answer, and command rows print the port before the name (`docker ps`) or far from it (`kubectl`) where authored files print it just after.

Replacing the name in the action with a neutral key, `service_addr`, while leaving the observation unchanged byte for byte, lowers Δ_{def} from 14.1 to 4.5 points on the `docker ps` row and from 13.6 to 5.5 on `kubectl`. The name in the observation alone still yields 4.2 points more on the `kubectl` row than on `.env`, and 3.2 on the `docker ps` row.

Under tool delivery, printing the port first raises the effect, and a command row raises it further than a JSON object in the same order. Inside a JSON object, printing the port before the name takes it from 0.2 points to 4.3, while moving the name 47 characters from the port there, as in `kubectl` rows, does not detectably change it. Format and order interact. Against a JSON object in the same order, the `docker ps` name and port cells (name first) are 3.3 points higher and the full row (port first) 9.8 higher; `kubectl`, also name first, reaches 13.6 (Appendix J).

3.4 What the name contributes, and an agent loop

Under tools, 41 of the 48 services have a positive name effect on command output, and the ten largest carry about half of it. Token count does not produce the effect: computed with only the made-up names that leave the prompt's token count unchanged, it stays within a point of the all-name effect

on the same services in all six layout-delivery cells. Fitted jointly with the name's prior strength and token count, only its familiarity to the model predicts the `docker ps` rate (Appendix E).

What the name contributes is the remembered default of the service it names, not a pull toward familiar ports in general. The 48 documented defaults are distinct, so the two are separable in the recorded generations. On the command layouts of the delivery comparison (Appendix I), plus `D-MAP` under tools, a real name under tool delivery writes its own service's default on 12.0 percent of prompts and one of the other 47 defaults on 0.37 percent; in chat the two rates are 5.3 and 0.03. Made-up names write no documented default on 15,552 tool and 10,368 chat prompts. The twenty cross-service answers mostly name a related service's default, none printed on the page (Appendix E).

On DeepSeek V4.1-Flash, an API model, the name effect under greedy decoding is 3.1 points on `kubectl` rows in chat (CI [1.0, 5.9]). On a two-candidate probability scale it is positive in all six cells measured, from 0.7 on the `docker ps` row to 14.0 on `kubectl` (CI [10.4, 18.0]; Appendix N).

An instruction to use exactly the value in the output still leaves 29 to 39 percent of the command-row effect under tool delivery (Appendix I). In Qwen3.5-9B the effect survives an agent loop with nothing prefilled (Appendices M and U). Given the same tool result and twelve turns on a sandboxed machine where the service answers only on the observed port, Qwen3.5-9B with thinking on configures the service with its documented default in 24.2 percent of real-name episodes and 0.7 percent of made-up-name ones (Δ_{def} 23.6 points, CI [18.1, 29.5]); it checks the port in 1.9 percent. DeepSeek V4.1-Flash checks it in 74.5 percent, and its effect is 0.2 points (CI $[-0.8, 1.0]$).

3.5 The same substitution in code

A familiar name pulls a remembered default outside configuration too. Yang (2026) reports this for Python import aliases, and we repeat it in the six models (Appendix P). With a conventional alias bound to another library (`import pandas as np`), the six models write the conventional library's API in 40 of 79 unambiguous continuations, against 1 of 236 under a made-up alias: 49.5 points over twelve library pairs (CI [27.3, 69.9]), as the preregistered test K1 predicted. The page holds no value to copy there, so the task shows the name's pull alone.

4 Removing default capture

4.1 Anchored fine-tuning toward two targets

The made-up-name control doubles as a label-free training target: with the name gone the frozen model already does the right thing, so its behaviour there is a correct target for the real-name prompt. Name-conditioned self-distillation (NCSD) trains a LoRA adapter (Hu et al., 2022) under the real name toward the frozen model θ_0 under made-up names. Let x be a prompt and $\tilde{x}_j$, $j = 1, 2, 3$, the same prompt with every surface form of the target name replaced by a made-up name. A span y of at most eight tokens is sampled from θ_0 on one $\tilde{x}_j$, cut before any made-up name and at the end of the first line, and every prompt is scored teacher-forced on it:

$$q_t = \frac{1}{3}\sum_{j=1}^{3} p_{\theta_0}(\cdot \mid \tilde{x}_j, y_{<t}), \qquad \mathcal{L}_{\text{bind}} = \frac{1}{|y|}\sum_{t=1}^{|y|} \mathrm{KL}(q_t \,\|\, p_\theta(\cdot \mid x, y_{<t})). \tag{1}$$

No ground-truth value enters this objective. Liu et al. (2024b), building on Eldan & Russinovich (2023), use a teacher of this form to unlearn a named target. Distilling a model's predictions on an edited input into the unedited input is context distillation (Askell et al., 2021; Snell et al., 2022), and consistency training when the edit removes a spurious cue (Chua et al., 2024; Irpan et al., 2025). Our cue, the name, also carries knowledge, so NCSD reverses the unlearning use: the adapter answers from the observed value and keeps the remembered default. The second target needs no teacher, because the observed value is printed in the prompt: supervised fine-tuning (SFT) replaces $\mathcal{L}_{\text{bind}}$ with the negative log-likelihood of the observed value's digits, and the anchor and every other setting stay as they are. Where the value is printed, SFT trains in about half the time; NCSD needs no label.

Trained alone, either target admits a degenerate solution that copies any number on the page. An anchor of two KL terms toward θ_0 penalises it. $\mathcal{L}_A$ is computed on continuations of anchor prompts

in four families: the action alone, other services' rows without the target's, prose naming the target's default without a number, and a non-port number on the target's row. $\mathcal{L}_G$ is computed on 128-token chunks of public code. The loss is $\mathcal{L}_{\text{bind}} + \lambda_A \mathcal{L}_A + \lambda_G \mathcal{L}_G$ with $\lambda_A = 1$ and $\lambda_G = 0.2$. Those weights and the LoRA rank were fixed in the design document before any adapter was trained, and the learning rate was fixed without tuning (Appendix A). Nothing was selected on the evaluation prompts, and Appendix K ablates the anchor weights, the rank, the training length and the teacher's made-up names one at a time.

4.2 Setup

We train two models. Olmo-3-7B-Instruct is one of the six diagnosed models, in one fold; Qwen2.5-1.5B-Instruct carries both folds, a second seed, the anchor ablation and the comparison with six inference-time methods. Both use rank-16 LoRA on every attention and MLP projection, 300 steps and raw-text prompts. The 48 services are split into two halves of 24, stratified by the 1.5B model's prior strength and by whether it ever writes the default; the 7B adapter uses that same split. An adapter trained on one half is scored on the other, in five command layouts disjoint from its nine training layouts. Each fold has 316 conflict prompts, 84 congruent prompts that show the default, 414 preservation prompts that show an observation without the target's value, and 240 asked-for-default prompts; the default is the correct answer on the last two. Every prompt is scored as raw text with no chat template and, without retraining, as a chat message and as a tool result (Appendices B and K.1). Paired 1.5B numbers are folds A and B unless stated otherwise.

4.3 Results

On the diagnosed 7B model, label-free NCSD nearly removes the effect. Under tool delivery the frozen Olmo-3-7B-Instruct writes the default on 98 of the 316 conflict prompts and Δ_{def} is 28.8 points; NCSD leaves 1.0 points (CI [0.0, 2.3]). On raw text NCSD takes 24.6 points to 1.7 ([0.2, 3.4]) and SFT to 2.6 ([0.8, 4.9]). Under NCSD the share of conflict prompts answered with the observed value rises from 64 to 97 percent on raw text and from 61 to 95 under tool delivery, congruent prompts stay copied, and recall of remembered defaults changes by at most 2.9 points in any delivery (Figure 5).

The repair generalises at two levels. Its battery holds out the 24 services and the layouts and reuses the training action templates; the diagnosis stimuli also change the value schemes and one of the two action lines, and there both 7B adapters leave 0.00 points in all eight layout-delivery cells, six of them on layouts new to the adapters, where the frozen model reaches 27.1 on the `docker ps` row under tool delivery (Appendix L).

On raw text the frozen 1.5B model writes the default on 13 and 14 of the 316 conflict prompts per fold, and both targets take it to none; in chat and tool delivery they and a second NCSD seed leave at most one such prompt per fold and delivery.

Recall of the remembered default holds up because of the anchor. Recall is measured on the preservation prompts: rows of other services without the target's (row-absent), and prose naming the default without a number. On raw text, paired with the frozen model by prompt, NCSD changes it by -0.2 points (CI [-2.2, 1.8]) in fold A and $+3.4$ ([1.3, 5.4]) in fold B, and SFT by $+0.7$ and $+3.4$. With no observation, each 1.5B NCSD or SFT adapter changes its answer on one or two of 21 held-out services per fold, all of them services the frozen model had answered wrongly, and fails the pre-registered no-change test (Appendix B). Without the anchor, either target removes the effect and the memory with it. NCSD in fold A has 3 percent row-absent recall against 66 with the anchor, and SFT has none (Appendix K).

Every inference-time method applied to the frozen 1.5B model leaves part of the effect, costs recall, or needs the target's name at inference time (Figure 3; Appendix O). An instruction to copy the observed value lowers the effect to 0.6 and 2.8 points on raw text and costs 3.6 to 8.2 points of recall in chat and tool delivery. On raw text, context-aware decoding (CAD) (Shi et al., 2024) lowers it to 0.9 and 2.2 points at either weight α and loses 8 to 18 points of recall. Constrained copying and anonymisation, which replaces the name with a made-up one, remove the effect and lose 39 to 70. Conditional anonymisation, which acts only where a rule finds the target's row, costs nothing here. But it needs the target's surface forms at inference time and hides the name the model needs. On all

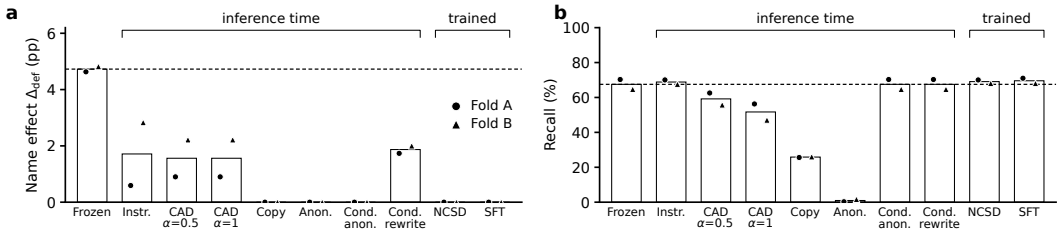


Figure 3: The 1.5B repair against inference-time methods on raw text, on the services each adapter never trained on. Bars are fold means, white for raw text as in Figure 5, markers the folds, and the dashed line the frozen model. CAD is context-aware decoding at contrast weight α; Copy is constrained copying and Anon. anonymisation. Cond. anon. and Cond. rewrite act only where a rule finds the target's row, anonymising the prompt or rewriting its rows as key=value lines; the other methods act on every prompt. (a) Δ_{def} on the conflict prompts, against one made-up name per prompt. (b) Recall, where the default is the correct answer: of the 414 prompts per fold whose page shows an observation without the target's value, the share answered with the remembered default.

240 asked-for-default prompts the rule fires and the model answers with the observed value. On the 7B model the instruction and CAD leave 10.8 to 23.4 of its 24.6 points, and constrained copying and both anonymisations reach zero, only conditional anonymisation keeping recall.

None of these methods does well on both requests. The frozen 1.5B model answers raw asked-for-default prompts with the documented default on 47.1 and 42.1 percent; constrained copying and the two anonymisations answer with it on none, context-aware decoding on 11.7 to 26.2, and the anchored adapters on 19.6 to 35.8 in fold A. Among the methods that lower the effect, only the conditional rewrite is above the frozen model, at 62.5 and 63.8, and it costs accuracy on the first request: on 7.4 and 9.7 percent of prompts the model answers with the name of the key the rewrite introduced (Appendix Q). The trade-off holds under tool delivery and on the 7B model, where the frozen model answers 57 percent of tool-delivered asked-for-default prompts with the documented default and NCSD 18 (Figure 5).

On the 1.5B model, a fifth anchor family that asks for the documented default wins back about half of NCSD's loss (Appendix K). An anchor pulls toward θ_0 and cannot pass the frozen model, which is not the limit here: on raw text half its asked-for-default answers are the value on the page, so most of what it lacks is the habit of reading the request, not the number. Supervising the fifth family instead, with the documented default the generator records, teaches the request. On raw text, in three runs over two folds, the supervised arm answers 75.8 to 76.7 percent of asked-for-default prompts with the documented default, against the frozen model's 47.1 and 42.1, with Δ_{def} 0.00 and every conflict prompt answered from the page. The arm's rate is 3.0 to 5.4 points below the frozen model's prose recall. Under chat and tool delivery the arm stays above the frozen model but falls further below that recall. Rotating the training prompts through all three deliveries closes most of that gap. In three runs over two folds this arm reaches 75.8 to 82.9 percent in every delivery, with Δ_{def} at most 0.25 and prose recall at most 3.8 points lower. It writes the default on 5 to 60 percent of requests naming the default yet asking for the page. On the 7B model this arm answers 63.8 to 72.5 percent, against the frozen model's 46.2 to 64.2. It leaves Δ_{def} at 4.41 on raw text and 3.82 under tools and loses 4.5 to 12.0 points of prose recall. All fifth-family arms are exploratory (Appendix R).

5 Related work

In question answering, memory can override a passage that contradicts it (Longpre et al., 2021), in forms that surveys map (Xu et al., 2024b), a benchmark catalogues at scale (Su et al., 2024) and methods set out to resolve (Jin et al., 2024a). It persists with real documents rather than edited ones (Kortukov et al., 2024). It is stronger for popular entities (Xie et al., 2024; Mallen et al., 2023), for confidently held facts (Kim et al., 2026; Wu et al., 2024a) and for contextual values far from what the model believes (Wu et al., 2024a). Scale does not reliably help (Ming et al., 2025; McKenzie et al., 2023), and context is used less faithfully than benchmarks imply (Tao et al., 2024; Hagström et al., 2025; Liu et al., 2024a; Modarressi et al., 2025). Outside question answering, models misuse

swapped Python builtins (Miceli Barone et al., 2023), degrade on counterfactual variants of familiar tasks (Wu et al., 2024b), call deprecated library versions (Wang et al., 2025a; Bae et al., 2025; Liu et al., 2024c) and invent package names (Spracklen et al., 2025). Yang (2026) measures the pull of an import alias; Appendix P repeats it in the six models and adds a trained repair. Agents apply updated state less reliably than they recognise it and leave found information unused (Chao et al., 2026; Engländer et al., 2026).

Where models have been studied against tool output, that output is itself the answer and may be wrong; here it is correct and must be copied into an action, where it competes with the remembered default tied to the name. When such output contradicts what a model knows, the model mostly defers to it: a correct closed-book answer survives a wrong tool return in 6.5–17.1% of cases (Varlamov et al., 2026), an agent adopts a graph network's labels whether or not they are right (Wang & Vemuri, 2026), and models err both ways when the two disagree (Cheng et al., 2025). Other failures trace to misjudging whether the tools at hand can solve a task (Zhang et al., 2024), mismanaged calls (Barkley & van der Merwe, 2024) or selection errors (Xu et al., 2024a).

Default capture is more specific than ignoring a result: the model overwrites the observed value with one the name recalls, and the made-up names separate that from copying error. Around it, tool-use reliability is charted as taxonomies and benchmarks (Soni, 2026; Hamad et al., 2025; Hu et al., 2026a; Cheng et al., 2026), and what a tool returns changes behaviour even when the call is right (Fatahi Bayat et al., 2026; Xuan et al., 2026; Zhang et al., 2026a), down to how a structured result is read (Kate et al., 2026). Doshi et al. (2026) name overwriting a record among the hazards for enterprise agents. Mechanistic work locates binding, copying and recall in circuits (Feng & Steinhardt, 2024; Olsson et al., 2022; Meng et al., 2022; Geva et al., 2022) and memory-context competition in heads and layers (Yu et al., 2023; Jin et al., 2024b; Ortu et al., 2024; Li et al., 2025).

Most training-time repairs supervise the answer (Longpre et al., 2021; Li et al., 2023; Neeman et al., 2023; Henkel et al., 2022; Bi et al., 2025; Zhang et al., 2025; Lin et al., 2026; Huang et al., 2025; Sun et al., 2026; Goyal et al., 2025). NCSD has the form of consistency training, which trains a model on a prompt with a spurious cue toward its own output or internal states without the cue (Chua et al., 2024; Irpan et al., 2025; Han et al., 2026; Gautam et al., 2026; Africa & Mani, 2026; Imran et al., 2026). Our cue, the name, is also what the second request needs. We replace it with made-up names rather than delete it, and anchor the student to the frozen model on prompts where the name should still recall the default (Section 4.1). Unlearning by name replacement uses a teacher of this form (Eldan & Russinovich, 2023; Liu et al., 2024b), and unlearning can widen what is removed (Wang et al., 2025c). Others distil a model from a privileged copy of itself (Ye et al., 2026; Zhao et al., 2026; Zhang et al., 2026b; Padmanabhan et al., 2023; Wang et al., 2026). Supervising the second request applies the principle of Minder et al. (2025), who label one question about a false context under an instruction to use the context and under one to ignore it. At inference, context-aware decoding contrasts predictions with and without the context (Shi et al., 2024), and later work weighs the context per step or steers hidden states (Wang et al., 2025b; Yuan et al., 2024; Bi et al., 2026; Li et al., 2025; Zhou et al., 2023; Ravulapalli et al., 2026). Appendix Q scores one of them, Appendix S the rest.

6 Conclusion

Six instruction-tuned models from three families overwrite observed values with remembered defaults, most when a tool returns the output and the line being written names the service. Pasting that output into a chat message understates the failure in five of the six models. A header naming the source lowers it. The failure survives an agent loop in Qwen3.5-9B but not in DeepSeek V4.1-Flash. The pull that Yang (2026) found in code recurs in the six models. Training toward the model's own behaviour under made-up names nearly removes default capture on held-out services, from 28.8 to 1.0 points on a 7B model under tool delivery. A repair trained only toward the page answers a request for the documented default less often than the frozen model. Supervising that request too lifts the rate above the frozen model's in every delivery, and the supervised 1.5B model still copies every raw-text conflict prompt. Appendix V gives the scope of these results.

Ethics statement

The study uses public open-weight models, one commercial API and synthetic stimuli built from public documentation of widely used software services; it involves no human subjects and no personal data. The failure it measures is a reliability risk for agents that write configuration and code, and the repair's training code is released with this paper to reduce it.

AI use statement

We used generative AI tools to brainstorm and critique designs, to search and summarise related work, to write and debug the experiment and analysis code, to draft and revise the text, and to check the manuscript against the result files. They produced no experimental data: every reported number was computed by code from the released model outputs, and the research question, the preregistered designs and the decisions about what to claim are ours. We reviewed all AI-assisted work. We tested the code, including invariant checks against reference outputs and a bit-identical check of resumed training, and a script rechecks more than 200 printed numbers against the result files. We checked that every citation exists, and we read every sentence that attributes a method or a result to another paper against that paper. We take responsibility for the final content of this work.

Reproducibility statement

Appendices A to C give the preregistrations and deviations, the objective and prompt construction, and every layout and name; the supplementary material holds the stimulus generators, all six models' raw outputs, the analysis scripts, the training and evaluation code with each adapter's configuration and training log, and example rendered prompts. The adapters themselves (2.4 GB) are released with the de-anonymised paper.

References

David Demitri Africa and Arathi Mani. Consistency training can entrench misalignment. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2606.03810.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

Nikhil Anand, Shwetha Somasundaram, Anirudh Phukan, Apoorv Saxena, and Koyel Mukherjee. ContextFocus: Activation steering for contextual faithfulness in large language models, 2026. URL https://arxiv.org/abs/2601.04131.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021. URL https://arxiv.org/abs/2112.00861.

Jaesung Bae, Cameron Churchwell, Mitchell Hermon, Tsun-An Hsieh, Jocelyn Xu, Yekaterina Yegorova, Mark Hasegawa-Johnson, and Heng Ji. That's deprecated! understanding, detecting, and steering knowledge conflicts in language models for code generation, 2025. URL https://arxiv.org/abs/2510.19116.

Liam Barkley and Brink van der Merwe. Investigating the role of prompting and external tools in hallucination rates of large language models, 2024. URL https://arxiv.org/abs/2410.19385.

Baolong Bi, Shaohan Huang, Yiwei Wang, Tianchi Yang, Zihan Zhang, Haizhen Huang, Lingrui Mei, Junfeng Fang, Zehao Li, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, and Shenghua Liu. Context-DPO: Aligning language models for context-faithfulness. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10280–10300, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.536. URL https://aclanthology.org/2025.findings-acl.536/.

Baolong Bi, Shenghua Liu, Yiwei Wang, Yilong Xu, Junfeng Fang, Lingrui Mei, and Xueqi Cheng. Parameters vs. context: Fine-grained control of knowledge reliance in language models. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TJ3DqFiGau.

Hanxiang Chao, Yihan Bai, Rui Sheng, Tianle Li, and Yushi Sun. STALE: Can LLM agents know when their memories are no longer valid?, 2026. URL https://arxiv.org/abs/2605.06527.

Baicheng Chen, Zheyuan Liu, Jingyu Zhang, Kaize Ding, Ningshan Ma, Yue Huang, and Meng Jiang. Forgotten in weights, recovered by tools: Agentic tool unlearning for LLM agents, 2026. URL https://arxiv.org/abs/2608.21544.

Jiali Cheng, Rui Pan, and Hadi Amiri. Investigating tool-memory conflicts in tool-augmented LLMs. In *The 2nd Workshop on Reliable and Responsible Foundation Models at ICML 2025*, 2025. URL https://openreview.net/forum?id=VTfDMX7UC6.

Yize Cheng, Chenrui Fan, Mahdi JafariRaviz, Keivan Rezaei, and Soheil Feizi. Model-adaptive tool necessity reveals the knowing-doing gap in LLM tool use, 2026. URL https://arxiv.org/abs/2605.14038.

James Chua, Edward Rees, Hunar Batra, Samuel R. Bowman, Julian Michael, Ethan Perez, and Miles Turpin. Bias-augmented consistency training reduces biased reasoning in chain-of-thought, 2024. URL https://arxiv.org/abs/2403.05518.

Roi Cohen, Yvan Carré, Nick Lechtenbörger, Hendrik Droste, Lucas Kerschke, Russa Biswas, Gerard de Melo, and Jan Buys. Knowledgeless language models: Suppressing parametric recall for evidence-grounded language modeling, 2026. URL https://arxiv.org/abs/2607.12831.

Aarya Doshi, Yining Hong, Congying Xu, Eunsuk Kang, Alexandros Kapravelos, and Christian Kästner. Towards verifiably safe tool use for LLM agents, 2026. URL https://arxiv.org/abs/2601.08012. ICSE 2026, New Ideas and Emerging Results.

Ronen Eldan and Mark Russinovich. Who's Harry Potter? approximate unlearning in LLMs, 2023. URL https://arxiv.org/abs/2310.02238.

Leon Engländer, Sophia Althammer, Ahmet Üstün, Matthias Gallé, and Tom Sherborne. Agents explore but agents ignore: LLMs lack environmental curiosity, 2026. URL https://arxiv.org/abs/2604.17609.

Farima Fatahi Bayat, Pouya Pezeshkpour, and Estevam Hruschka. From proof to program: Characterizing tool-induced reasoning hallucinations in large language models, 2026. URL https://arxiv.org/abs/2511.10899. ACL 2026.

Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? In *International Conference on Learning Representations*, 2024.

Sukrati Gautam, Neil Shah, Arav Dhoot, Bryan Maruyama, Caroline Wei, Rohan Kapoor, Robert Sidey, Prakhar Gupta, Zi Cheng Huang, and David Demitri Africa. Consistency training along the transformer stack, 2026. URL https://arxiv.org/abs/2606.05817.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11): 665–673, 2020. doi: 10.1038/s42256-020-00257-z.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. URL https://arxiv.org/abs/2607.02770.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 30–45. Association for Computational Linguistics, 2022. URL https://aclanthology.org/2022.emnlp-main.3.

Sachin Goyal, Christina Baek, J. Zico Kolter, and Aditi Raghunathan. Context-parametric inversion: Why instruction finetuning may not actually improve context reliance. In *International Conference on Learning Representations*, pp. 68067–68099, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/aa27ac7aca4e462da1439b43ceebc04c-Paper-Conference.pdf.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8ac015d409635f196f9e3e9dcfb9a94e-Abstract-Conference.html.

Prakhar Gupta and Vaibhav Gupta. Context-grounding gains are mediated by pre-existing machinery: Auditing GRPO, SFT, and DPO, 2026. URL https://arxiv.org/abs/2609.00925.

Lovisa Hagström, Sara Vera Marjanovic, Haeun Yu, Arnav Arora, Christina Lioma, Maria Maistro, Pepa Atanasova, and Isabelle Augenstein. A reality check on context utilisation for retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19691–19730, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.968. URL https://aclanthology.org/2025.acl-long.968/.

Hassan Hamad, Yingru Xu, Liang Zhao, Wenbo Yan, and Narendra Gyanchandani. ToolCritic: Detecting and correcting tool-use errors in dialogue systems, 2025. URL https://arxiv.org/abs/2510.17052.

Andy Han, Kristina Fujimoto, Avidan Shah, Kiet Nguyen, Kai Xu, Yueh-Han Chen, Ilia Sucholutsky, and Rico Angell. On-policy consistency training improves LLM safety with minimal capability degradation, 2026. URL https://arxiv.org/abs/2605.21834.

Jordan Henkel, Goutham Ramakrishnan, Zi Wang, Aws Albarghouthi, Somesh Jha, and Thomas Reps. Semantic robustness of models of source code. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 526–537, 2022. URL https://pages.cs.wisc.edu/~aws/papers/saner22.pdf.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jinchao Hu, Meizhi Zhong, Kehai Chen, Xuefeng Bai, and Min Zhang. Agentic tool use in large language models, 2026a. URL https://arxiv.org/abs/2604.00835.

Xiaoyang Hu, Mike Angstadt, Shane Storks, Zan Huang, Aman Taxali, Alex Weigard, Richard L. Lewis, and Chandra Sripada. Conflict and congruency effects in large language models: In-weight and in-context competition in a verbal conflict task, 2026b. URL https://arxiv.org/abs/2608.11510.

Yukun Huang, Sanxing Chen, Hongyi Cai, and Bhuwan Dhingra. To trust or not to trust? enhancing Large Language Models' situated faithfulness to external contexts. In *International Conference on Learning Representations*, pp. 8625–8654, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/186a213d720568b31f9b59c085a23e5a-Paper-Conference.pdf. arXiv:2410.14675.

Sohaib Imran, Prakhar Gupta, Jannes Elstner, and David Demitri Africa. Consistency training while mitigating obfuscation via rate matching, 2026. URL https://arxiv.org/abs/2606.02211.

Alex Irpan, Alexander Matt Turner, Mark Kurzeja, David K. Elson, and Rohin Shah. Consistency training helps stop sycophancy and jailbreaks, 2025. URL https://arxiv.org/abs/2510.27062.

Runze Jiang, Taiqiang Wu, Yan Wang, Bingyu Zhu, and Longtao Huang. From context-aware to conflict-aware: Generalizing contrastive decoding for knowledge conflict in LLMs, 2026. URL https://arxiv.org/abs/2606.10298.

Zhuoran Jin, Pengfei Cao, Yubo Chen, Kang Liu, Xiaojian Jiang, Jiexin Xu, Qiuxia Li, and Jun Zhao. Tug-of-war between knowledge: Exploring and resolving knowledge conflicts in retrieval-augmented language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 16867–16878, Torino, Italia, 2024a. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.1466/.

Zhuoran Jin, Pengfei Cao, Hongbang Yuan, Yubo Chen, Jiexin Xu, Huaijun Li, Xiaojian Jiang, Kang Liu, and Jun Zhao. Cutting off the head ends the conflict: A mechanism for interpreting and mitigating knowledge conflicts in language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1193–1215, Bangkok, Thailand, 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.70. URL https://aclanthology.org/2024.findings-acl.70/.

Josip Jukić and Ivan Titov. Geometric self-distillation for reasoning generalization, 2026. URL https://arxiv.org/abs/2607.06855.

Kiran Kate, Yara Rizk, Poulami Ghosh, Ashu Gulati, Tathagata Chakraborti, Zidane Wright, and Mayank Agarwal. How good are LLMs at processing tool outputs? In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 2918–2941, 2026. URL https://aclanthology.org/2026.eacl-long.134/.

Divyansh Kaushik, Eduard Hovy, and Zachary C. Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1909.12434.

Minsung Kim, Dong-Kyum Kim, Jea Kwon, Nakyeong Yang, Kyomin Jung, and Meeyoung Cha. How training data shapes the use of parametric and in-context knowledge in language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23242–23257, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1064. URL https://aclanthology.org/2026.acl-long.1064/.

Evgenii Kortukov, Alexander Rubinstein, Elisa Nguyen, and Seong Joon Oh. Studying large language model behaviors under context-memory conflicts with real documents, 2024. URL https://arxiv.org/abs/2404.16032.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. Large language models with controllable working memory. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 1774–1793, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.112. URL https://aclanthology.org/2023.findings-acl.112/.

Gaotang Li, Yuzhong Chen, and Hanghang Tong. Taming knowledge conflicts in language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 34074–34104. PMLR, 2025. URL https://proceedings.mlr.press/v267/li25c.html.

Chenyu Lin, Yilin Wen, Du Su, Hexiang Tan, Fei Sun, Muhan Chen, Chenfu Bao, and Zhonghou Lv. Resisting contextual interference in RAG via parametric-knowledge reinforcement. In *International Conference on Learning Representations*, pp. 46721–46742, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4dbcd4ba82866fdefc7ea0d15e9a72f2-Paper-Conference.pdf.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a. URL https://arxiv.org/abs/2307.03172.

Yujian Liu, Yang Zhang, Tommi Jaakkola, and Shiyu Chang. Revisiting who's Harry Potter: Towards targeted unlearning from a causal intervention perspective. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8708–8731, Miami, Florida, USA, 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.495. URL https://aclanthology.org/2024.emnlp-main.495/.

Zeyu Leo Liu, Shrey Pandit, Xi Ye, Eunsol Choi, and Greg Durrett. CodeUpdateArena: Benchmarking knowledge editing on API updates, 2024c. URL https://arxiv.org/abs/2407.06249.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7052–7063, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.565. URL https://aclanthology.org/2021.emnlp-main.565/.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

Ian R. McKenzie, Alexander Lyzhov, Michael Martin Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Xudong Shen, Joe Cavanagh, Andrew George Gritsevskiy, Derik Kauffman, Aaron T. Kirtland, Zhengping Zhou, Yuhui Zhang, Sicong Huang, Daniel Wurgaft, Max Weiss, Alexis Ross, Gabriel Recchia, Alisa Liu, Jiacheng Liu, Tom Tseng, Tomasz Korbak, Najoung Kim, Samuel R. Bowman, and Ethan Perez. Inverse scaling: When bigger isn't better. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=DwgRm72GQF. Featured Certification.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Antonio Valerio Miceli Barone, Fazl Barez, Shay B. Cohen, and Ioannis Konstas. The larger they are, the harder they fail: Language models do not recognize identifier swaps in python. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 272–292, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.19. URL https://aclanthology.org/2023.findings-acl.19/.

Julian Minder, Kevin Du, Niklas Stoehr, Giovanni Monea, Chris Wendler, Robert West, and Ryan Cotterell. Controllable context sensitivity and the knob behind it. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Igm9bbkzHC.

Yifei Ming, Senthil Purushwalkam, Shrey Pandit, Zixuan Ke, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. FaithEval: Can your language model stay faithful to context, even if "the moon is made of marshmallows". In *International Conference on Learning Representations*, pp. 29430–29456, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/48404cd9ce03946c6b7177691f3267a1-Paper-Conference.pdf.

Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schütze. NoLiMa: Long-context evaluation beyond literal matching. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2502.05167.

Ella Neeman, Roee Aharoni, Or Honovich, Leshem Choshen, Idan Szpektor, and Omri Abend. DisentQA: Disentangling parametric and contextual knowledge with counterfactual question answering. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10056–10070, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.559. URL https://aclanthology.org/2023.acl-long.559/.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Francesco Ortu, Zhijing Jin, Diego Doimo, Mrinmaya Sachan, Alberto Cazzaniga, and Bernhard Schölkopf. Competition of mechanisms: Tracing how language models handle facts and counterfactuals. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8420–8436, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.458. URL https://aclanthology.org/2024.acl-long.458/.

Shankar Padmanabhan, Yasumasa Onoe, Michael J.Q. Zhang, Greg Durrett, and Eunsol Choi. Propagating knowledge updates to LMs through distillation. In *Advances in Neural Information Processing Systems 36*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/932147114c48f8b04d41aebc0c631158-Abstract-Conference.html.

Qwen Team. Qwen3.5: Towards native multimodal agents, 2026. URL https://qwen.ai/blog?id=qwen3.5. Blog post, February 2026.

Manas Venkata Sai Ravulapalli, Samrath Singh Chadha, and Abhinav M. Hari. Legible failures: Detecting and repairing in-context binding errors, 2026. URL https://arxiv.org/abs/2609.11216.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 783–791, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-short.69. URL https://aclanthology.org/2024.naacl-short.69/.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022. URL https://arxiv.org/abs/2209.15189.

Harsh Soni. ToolFailBench: Diagnosing tool-use failures in LLM agents, 2026. URL https://arxiv.org/abs/2607.04686.

Joseph Spracklen, Raveen Wijewickrama, A H M Nazmus Sakib, Anindya Maiti, Bimal Viswanath, and Murtuza Jadliwala. We have a package for you! a comprehensive analysis of package hallucinations by code generating LLMs. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 3687–3706, Seattle, WA, 2025. USENIX Association. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/spracklen.

Zhaochen Su, Jun Zhang, Xiaoye Qu, Tong Zhu, Yanshu Li, Jiashuo Sun, Juntao Li, Min Zhang, and Yu Cheng. ConflictBank: A benchmark for evaluating the influence of knowledge conflicts in LLMs. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024) Datasets and Benchmarks Track*, volume 37, pp. 103242–103268. Curran Associates, Inc., 2024. doi: 10.52202/079017-3280. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/baf4b960d118f838ad0b2c08247a9ebe-Abstract-Datasets_and_Benchmarks_Track.html.

Xian Sun, Wei Chow, Yingshuo Wang, Junhao Liu, Wei Gao, Qing Wu, and Lingdong Kong. Learning when to trust via selective context preference optimization. *arXiv preprint arXiv:2608.06377*, 2026. URL https://arxiv.org/abs/2608.06377.

Yufei Tao, Adam Hiatt, Erik Haake, Antonie J. Jetter, and Ameeta Agrawal. When context leads but parametric memory follows in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4034–4058. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.234.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025. URL https://arxiv.org/abs/2512.13961.

Arseniy Varlamov, Rishat Zinnatullin, Elisei Rykov, Alexander Panchenko, and Ilseyar Alimova. MemToC: Benchmarking memory-tool conflict resolution in large language models, 2026. URL https://arxiv.org/abs/2608.26295.

Chong Wang, Kaifeng Huang, Jian Zhang, Yebo Feng, Lyuye Zhang, Yang Liu, and Xin Peng. LLMs meet library evolution: Evaluating deprecated API usage in LLM-based code completion. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pp. 885–897. IEEE, 2025a. doi: 10.1109/ICSE55347.2025.00245.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. AdaCAD: Adaptively decoding to balance conflicts between contextual and parametric knowledge. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 11636–11652, Albuquerque, New Mexico, 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-long.581. URL https://aclanthology.org/2025.naacl-long.581/.

Wenyu Wang, Mengqi Zhang, Xiaotian Ye, Zhaochun Ren, Zhumin Chen, and Pengjie Ren. UIPE: Enhancing LLM unlearning by removing knowledge related to forgetting targets, 2025c.

Xun Wang, Ruishuo Chen, Zhuoran Li, Yu Chen, and Longbo Huang. When context returns: Toward robust internalization in on-policy distillation, 2026. URL https://arxiv.org/abs/2606.11627.

Zhongyuan Wang and Pratyusha Vemuri. When the tool decides: LLM agents defer blindly to graph neural network tools, and stronger backbones defer more, 2026. URL https://arxiv.org/abs/2606.14476.

Nadya Yuki Wangsajaya, Haeun Yu, and Isabelle Augenstein. Emergence of context characteristics sensitivity in large language models, 2026. URL https://arxiv.org/abs/2606.09525.

Yuqiao Wen, Zichao Li, Wenyu Du, and Lili Mou. f-divergence minimization for sequence-level knowledge distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10817–10834, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.605. URL https://aclanthology.org/2023.acl-long.605/.

Kevin Wu, Eric Wu, and James Zou. ClashEval: Quantifying the tug-of-war between an LLM's internal prior and external evidence. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024a.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1819–1862, Mexico City, Mexico, June 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.102. URL https://aclanthology.org/2024.naacl-long.102/.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *International Conference on Learning Representations*, pp. 35623–35646, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/99261adc8a6356b38bcf999bba9a26dc-Paper-Conference.pdf.

Hongshen Xu, Zichen Zhu, Lei Pan, Zihan Wang, Su Zhu, Da Ma, Ruisheng Cao, Lu Chen, and Kai Yu. Reducing tool hallucination via reliability alignment, 2024a.

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. Knowledge conflicts for LLMs: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8541–8565, 2024b. URL https://aclanthology.org/2024.emnlp-main.486/.

Weihao Xuan, Qingcheng Zeng, Heli Qi, Yunze Xiao, Junjue Wang, and Naoto Yokoya. The confidence dichotomy: Analyzing and mitigating miscalibration in tool-use agents, 2026. URL https://arxiv.org/abs/2601.07264. ACL 2026.

Haotian Yang. import pandas as np: the numpy strikes back. In *Proceedings of the 9th Workshop on Analyzing and Interpreting Neural Networks for NLP (BlackboxNLP 2026)*, 2026. doi: 10.5281/zenodo.22943845. To appear.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models, 2026. URL https://arxiv.org/abs/2602.12275.

Qinan Yu, Jack Merullo, and Ellie Pavlick. Characterizing mechanisms for factual recall in language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9924–9959, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.615. URL https://aclanthology.org/2023.emnlp-main.615/.

Xiaowei Yuan, Zhao Yang, Yequan Wang, Shengping Liu, Jun Zhao, and Kang Liu. Discerning and resolving knowledge conflicts through adaptive decoding with contextual information-entropy constraint. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3903–3922, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.234. URL https://aclanthology.org/2024.findings-acl.234/.

Kaituo Zhang, Zhen Xiong, Mingyu Zhong, Zhimeng Jiang, Zhouyuan Yuan, Zhecheng Li, and Ying Lin. Are tools all we need? unveiling the tool-use tax in LLM agents, 2026a. URL https://arxiv.org/abs/2605.00136.

Ruizhe Zhang, Yongxin Xu, Yuzhen Xiao, Runchuan Zhu, Xinke Jiang, Xu Chu, Junfeng Zhao, and Yasha Wang. KnowPO: Knowledge-aware preference optimization for controllable knowledge selection in retrieval-augmented language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25895–25903, 2025. doi: 10.1609/aaai.v39i24.34783.

Xinsen Zhang, Zhenkai Ding, Tianjun Pan, Run Yang, Chun Kang, Xue Xiong, and Jingnan Gu. OPSDL: On-policy self-distillation for long-context language models, 2026b. URL https://arxiv.org/abs/2604.17535.

Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, Tetsuya Sakai, Tian Feng, and Hayato Yamana. ToolBeHonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11388–11422, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.637. URL https://aclanthology.org/2024.emnlp-main.637/.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models, 2026. URL https://arxiv.org/abs/2601.18734.

Wenxuan Zhou, Sheng Zhang, Hoifung Poon, and Muhao Chen. Context-faithful prompting for large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14544–14556, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.968. URL https://aclanthology.org/2023.findings-emnlp.968/.

Yigeng Zhou, Wu Li, Yifan Lu, Yequan Wang, Xuebo Liu, Wenya Wang, Jun Yu, Min Zhang, and Jing Li. Mitigating context-memory conflicts in LLMs through dynamic cognitive reconciliation decoding, 2026. URL https://arxiv.org/abs/2605.12185.

This appendix answers four questions, in the order the questions arise rather than the order the appendices run. **The record.** Appendix A gives every preregistered test's status and the departures that make a difference. **What the models were shown.** Appendix C builds the layouts, services and made-up names; Appendix D gives every layout's effect. **What the diagnosis found beyond the main text.** Appendices H, I and J separate delivery, layout and model; Appendices E and F ask whether the name's surface or an expected default explains it; Appendices G, M and U vary the readout, the last in an agent loop, and Appendix N repeats the measurement on an API model. **How the repair was trained and scored.** Appendix B gives the recipe and the held-out split; Appendices K and L its results, with the chat and tool renders in Appendix K.1 and the preservation families in Appendix K.2; Appendix O the inference-time alternatives and Appendix Q scores every one of them on both requests at once, and Appendix R supervises the asked-for-default family; Appendix P repeats the import-alias measurement of Yang (2026) in the six models and trains the repair on it, and Appendix T asks whether two path-valued fields behave like the port, and Appendix S summarises the rest of the repair and tool-use literature. Appendix V gives the limitations.

A Preregistration, claim status and deviations

The diagnosis preregistration fixed the six models, the 48 services and their observed values, the 17 layouts, the prompt construction per delivery condition, the scoring rules, the analysis code and the decision rules, and was committed before any of the six models had produced output on these stimuli; the models had been run on the pilot's stimuli, whose results stayed unopened. The repair design — the training recipe, the evaluation battery and four primary tests (P1 to P4) — was committed at the same time, before any adapter was trained. Supplement S2 gives every claim's status and every departure from the two documents, row by row; the paragraphs below state each confirmatory outcome and every preregistered test that failed.

Departures that make a difference. We record twenty-one departures; these are the ones a reader should weigh. The repair trained for 300 steps at a fixed learning rate with one seed per arm and fold (two for NCSD), where the design called for 600 steps, a tuned rate and three seeds; it trained and evaluated on raw text, where the design put half the examples in the tool-result form; its efficacy test reuses the training action templates and value sampler, so only services and layouts are held out; its remembered default is the documented default throughout, where the design took the model's own answer for the 20 rule-selected services; and the anchor carries two KL terms, not the three designed. P2 compares one answer per service against a 3-point threshold instead of a non-inferiority bound, and P3 uses one kind of non-port number where the design pooled three. Tool delivery ran after the chat analysis and on 11 of the 17 layouts, the bridge condition in a later bfloat16 regeneration, and the sampling scale, the free-form condition and the instruction, context-aware-decoding and constrained-copying baselines were designed after the diagnosis was analysed. The fifth anchor family and the supervised asked-for-default arms of Section 4.3, trained on raw text or in all three deliveries, were designed later still, after the trade-off had been measured, so no preregistered test covers them and every statement about them is exploratory (Appendix R).

Every preregistered test has a status. The families are A, the headline contrast of `docker ps` against key=value; B, its scope and mechanism, that is, which other layouts and which single edits to a layout carry the effect; C, whether prior strength predicts where it happens; K, the same substitution in code; and P, the repair's four primary tests — the effect removed, the memory kept, no credulous

copying of planted numbers, and label-free training matching supervised training (NCSD within ± 5 points of SFT on Δ_{def}). A1, A2, B1–B9, the C family's two rank statistics and K1–K4 in the diagnosis and P1–P4 in the repair were all computed, as were the C1 headline gate on A1 and the C2 agreement check between the greedy and nats scales, and the paragraphs below give each outcome. The tests that fail are K4, the repair's P2, the C family, B2 on the same-form subset, B5, which is significant against its predicted sign, and the five B contrasts that do not survive Holm correction (B3, B4, B6, B8, B9). K2 was missed by the first analysis and computed from the stored score records for this version. The plans also list runs we did not make, none of them a preregistered test: the final raw-text wave with further base and instruction-tuned models and its scale slopes, the vLLM-against-Hugging-Face scoring check, the two report-form action templates, and the designed preference-optimisation, reinforcement-learning, docker-trained and Llama-3.2-1B arms of the repair and its position-masked mechanism check.

Table 1: The main claim families by status: C, a preregistered test; S, a preregistered secondary analysis; E, exploratory: designed after the diagnosis data were seen, or run outside the preregistered design. Supplement S2 gives every claim with its numbers.

Claim		Outcome	Where
A1: `docker ps` above key=value, chat	C	passes; 3 of 6 models meet the per-model criterion	Sec. 2
A2: the same against key=value with no variable	C	passes	App. A
B1–B9: other layouts and single edits	C	4 of 9 survive Holm, B5 against its predicted sign; B2 not on the same-form subset	App. A
C family: prior strength ranks services by risk	C	fails	App. A
K1–K3: a rebound import alias pulls code	C	pass	Sec. 3.5, App. A
K4: relabelling the alias beats anonymising it	C	fails	App. P
P1, P3, P4: the repair removes the effect, copies no planted number, NCSD matches SFT (1.5B)	C	pass, one seed per fold; P4 with both adapters at 0.00	Sec. 4.3, App. A
P2: no-observation answers unchanged	C	fails	App. K
Tool delivery raises the effect on command output	S	3.2 to 9.7 points; five of six models	Sec. 3.1
Delivery steps, retrieved document, name in the action line, no prefill, agent loop, API model	E		Secs. 3.2–3.4
Repair on Olmo-3-7B-Instruct; inference-time methods	E		Sec. 4.3
Fifth anchor family; supervised second-request arms, 1.5B and 7B	E		Sec. 4.3, App. R

Confirmatory results. In chat, A1 — `DOCKER1` minus `KV` — is $+3.68$, CI $[+2.26, +5.34]$, passing on all three preregistered metric variants. Three models clear the per-model criterion, an A1 of at least 3 points with a 95% interval above zero: Qwen3.5-9B $+10.42$ $[4.86, 16.67]$, Olmo-3-7B-Instruct $+5.79$ $[1.27, 11.46]$, Olmo-3.1-32B-Instruct $+5.56$ $[2.08, 9.72]$. Qwen3.5-27B $+0.35$ $[-3.12, 5.56]$ and both Gemma models 0.00 do not. The preregistered rule then confines the confirmatory claim to the passing families, which we read as the three passing models: both Olmo 3 models and Qwen3.5-9B. A2, the same quantity against `KV-NOVAR`, is $+4.09$ $[+2.60, +5.79]$. `KV-NOVAR` offers no shell variable to write in place of the number, and A2's interval is above zero, so the preregistered downgrade of C1 for an effect resting on that variable does not apply. A1 is also identical whether or not the outputs that write the variable are counted. Four of the nine B contrasts survive Holm correction in chat, two in each direction: `kubectl` and `ss` above key=value, and `host:port` below a bare port (-1.33 points, B5, whose preregistered prediction was positive) and a port printed twice below once (-1.77, B7). The preregistration also requires B1–B3 to hold on the same-form subset, the action whose name tokens match the observation's, before they clear the name's surface form. There `kubectl` is larger ($+8.8$ against $+5.8$ in chat, $+16.3$ against $+12.6$ under tools), and `ss` keeps its size ($+1.5$) but no longer survives Holm correction ($p = 0.052$), so we cannot separate it from the name's surface form. The C family is null in chat over the 20 rule-selected services: Kendall $\bar{\tau} = 0.02$ $[-0.03, 0.12]$, and -0.01 $[-0.05, 0.10]$ with the name's familiarity in a `docker ps` row partialled out, so prior strength is no risk score and all statements

about it are exploratory. In code K1, that a rebound alias pulls the continuation toward the library the alias conventionally names, and K3, that the error is higher under a rebound alias than under one relabelled to the bound library's own convention ($+27.08$ points, CI $[11.54, 50.83]$), both pass; K4, that relabelling the alias beats anonymising it, does not (Holm $p = 0.86$). K2 passes: the name effect in nats when the import binds the alias (`import pandas as np`) minus when a separate line assigns it (`import pandas`, then `np = pandas`), averaged over the two use sites, is $+3.92$, CI $[+1.11, +6.65]$ (Holm $p = 0.023$; $[+0.31, +9.33]$ when models are resampled as well), and is positive in all six models, so the pull toward the conventionally named library is stronger when the import itself binds the alias. In the repair, P1 passes in both folds, NCSD taking Δ_{def} from the frozen model's 4.63 and 4.82 points to 0.00, and P4 holds, with SFT also at 0.00. P3 passes, since neither adapter writes a planted number in either fold, and P2 fails: each changes its answer on one or two of the 21 services with nothing to read (Appendix K).

Preregistered secondary analyses. Under tools A1 is $+13.34$ $[+9.46, +17.48]$ and five of six models qualify. In both deliveries A1 is smallest at the default plus 13 and largest when the observed value shares no leading digit with it, and in chat it is not carried by the 28 pilot services alone (A1 $+3.9$ on them against $+3.3$ on the rule-selected ones). C2, the sign agreement of the greedy and nats scales, holds in 49 of the 84 preregistered chat contrasts and 38 of 60 under tools, and in 21 of 28 and 26 of 29 among contrasts whose greedy interval excludes zero.

B Training and evaluation detail

Prompts and labels. Every training prompt is raw text with no chat template: an observation where there is one, a newline, and the action line the model completes; evaluation uses the same raw text, and Appendix K.1 rerenders it through chat and tool templates. The remembered default is the documented default recorded in the frozen service list. A training binding prompt puts the target in one of nine training layouts, with zero to five shuffled rows for other services of the training fold, each at its own default or at a random port with equal probability, and completes one of five action templates. The observed value is a uniform random port in $[1024, 65535]$, a near miss 2 to 99 away from the default, another service's default, or, for a congruent prompt, the default itself; the diagnosis's default plus 13 never occurs. Supplement S3 gives the layouts, templates, anchor families and loss terms in full, with the sampler probabilities, the construction rules, the leakage check, the probe accounting and the teacher's own statistics.

Adapter and optimisation. Qwen2.5-1.5B-Instruct with a rank-16 LoRA, $\alpha = 32$ and no dropout on `q/k/v/o_proj` and `gate/up/down_proj`, giving 18.46M trainable parameters, the base weights in bfloat16 and the adapter in float32, gradient checkpointing on and the KV cache off. AdamW, $\beta = (0.9, 0.999)$, no weight decay, peak learning rate 10^{-4}, linear warmup over 20 steps and then cosine decay to zero, gradient norm clipped at 1.0, 300 steps (Appendix A). Each step holds 32 sequences, 16 binding prompts, 12 anchor prompts and 4 general-text chunks, in micro-batches of one or two sequences with gradients accumulated over the step. One example stream, seeded from the run seed alone, is generated before training, so NCSD and SFT at a seed see the same prompts in the same order and differ only on binding prompts; anchor continuations are resampled in each run.

Objective. The student p_θ carries the adapter, the frozen p_{θ_0} is the same network with it disabled, and every divergence is a forward KL toward the frozen model or a mixture built from it. For a binding prompt x, three made-up names ν_1, ν_2, ν_3 are drawn independently from a pool of 500 five-letter consonant-vowel names absent from a public code corpus, and $\tilde{x}_j$ is x with every surface form of the service name replaced by ν_j. The frozen model samples up to 8 tokens at temperature 1 from one $\tilde{x}_j$ chosen uniformly, the sample is cut to at most 16 characters, and an example whose span is then empty is skipped. The span y is appended to all three made-up name prompts and to x, and all four are scored teacher-forced on y, so every distribution conditions on the same history:

$$q_t = \frac{1}{3}\sum_{j=1}^{3} p_{\theta_0}(\cdot \mid \tilde{x}_j, y_{<t}), \qquad \ell_{\text{bind}}(x) = \frac{1}{|y|}\sum_{t=1}^{|y|} \mathrm{KL}(q_t \,\|\, p_\theta(\cdot \mid x, y_{<t})).$$

No ground-truth value enters this term. The SFT arm replaces it with $\ell_{\text{SFT}}(x) = -\frac{1}{|v|}\sum_t \log p_\theta(v_t \mid x, v_{<t})$, for v the observed value's digits, and changes nothing else. An anchor prompt takes the

same forward KL over an uncut 8-token frozen continuation, ℓ_A, and a 128-token general-text chunk the same over every position, ℓ_G. One step then costs $\frac{1}{16}\sum_{x\in B}\ell_{\mathrm{bind}}(x) + \lambda_A\frac{1}{12}\sum_{a\in A}\ell_A(a) + \lambda_G\frac{1}{4}\sum_{c\in G}\ell_G(c)$ over its binding prompts B, anchor prompts A and chunks G, with $\lambda_A = 1.0$ and $\lambda_G = 0.2$ and a skipped binding example contributing zero to the first sum.

Anchor prompts and general text. Anchor prompts come from four families, each rendered for a training-fold service with one of the five training action templates: the action alone with no observation (35%), an observation listing one to four other services but not the target (30%), a note naming the service's default without a number (15%), and a `svcstat` row planting a number that is not a port, a process id, an uptime, a memory size, a thread count or a connection count (20%). NCSD with the fifth anchor family and the supervised arm draw 15% of anchor prompts from a family asking for the documented default, taken from the no-observation family, which falls to 20%. General-text chunks are 600-character windows of Python source from a public code corpus, truncated to 128 tokens.

Folds. The 48 services are split into two halves of 24, stratified on two properties of the frozen 1.5B model. Its prior strength S is the log-probability of the documented default under the real name minus the mean under three made-up names, with no observation, averaged over three templates (a Python keyword argument, a command-line flag and an INI section). Its base error is whether it writes the default on any of its DOCKER1 conflict prompts in a 600-prompt raw-text probe, which gives each service 3 to 19 conflict prompts. Binned by base error (any or none) crossed with tertiles of S and sorted by S inside each of the six bins, services are assigned to the halves alternately; the halves have mean S of 6.90 and 7.04 nats and contain 10 and 11 services with any base error. Fold A is the adapter trained on half A, measured together with the frozen model on half B; fold B is the reverse.

What is and is not held out. Services and observation layouts are held out, action templates and anchor families are not: each adapter trains on its own half's 24 services in the nine training layouts, disjoint from the five evaluation layouts, and is scored with the same five templates. No held-out service reaches the binding or anchor draws under any spelling.

Evaluation. Each fold is scored on 400 binding prompts for its 24 held-out services, 316 with a conflict and 84 congruent, drawn with the training value sampler and rendered by the diagnosis code in five command layouts held out of training: DOCKER1, D-NOHASH, KUBECTL, SS and D-MAP. Each shows only the target's row, and D-MAP prints the observed value on both sides of its arrow (`0.0.0.0:6392->6392/tcp`), so no layout prints the default. Each prompt is paired with one carrying a made-up name from the teacher's pool, not length-matched as the diagnosis names were, so Δ_{def} here is a difference against one made-up name rather than the mean over three. Decoding is greedy with at most 8 new tokens, and an output is read as its leading integer of 2 to 5 digits. Δ_{def} is the real name's default rate minus the made-up names' on conflict prompts, per service and then averaged over services; the copy rate is the share of a service's conflict prompts answered with the observed value, averaged over services as Δ_{def} is. One fixed draw of 200 anchor-family prompts, the same for the frozen model and every adapter, supplies two probes. Its 70 prompts with no observation cover 21 of the 24 held-out services. The 70 prompts are unevenly spread over those services, so the nothing-to-read probe (E0, the preregistered test P2) scores one prompt per service — that service's last in the fixed draw — and counts the services where the adapter's answer differs from the frozen model's on it. One service is worth 4.76 points, so the 3-point threshold tolerates none. The test counts every disagreement as a failure, whatever its direction; for every anchored 1.5B adapter, each disagreement observed falls on a service the frozen model answered wrongly (Appendix K). Its 48 prompts that plant a non-port number on the target's row, over 21 services, form the non-port probe. Thirty of the 48 planted numbers have two to five digits, so only those can be written as if they were ports, and the same 30 are what the conditional rule and constrained copying of Appendix O can act on. Preservation with an observation present uses a separate 900-prompt draw from the same four-family mix, of which only its 133 prose and 281 row-absent prompts are scored, 414 per fold over all 24 held-out services, correct when the answer is the documented default and a misbinding when it is another service's listed port. Intervals on Δ_{def} are cluster bootstraps over services with 10,000 resamples. Preservation is compared with the frozen model on identical prompts, and those paired

differences carry the same service-cluster bootstrap. A preservation accuracy on its own carries a Wilson interval, which ignores clustering by service.

Compute. The diagnosis and its regeneration ran on rented A100 GPUs with vLLM. The 1.5B repair ran on one 8 GB laptop RTX 5070 with a peak allocation of at most 7.0 GB, and the 7B repair on one rented A100 with micro-batches of 8 and 39.98M trainable parameters. An anchored 1.5B NCSD run of 300 steps takes 1.6 to 2.9 hours and an anchored SFT run 0.8 to 1.1 hours, because NCSD samples a teacher span and scores three made-up-name prompts for every binding example.

C Stimuli

Every layout renders the same fact, that a service listens on a port that is not its documented default, and the layouts differ only in the surface it arrives on. Table 2 lists all 17 for `redis`, whose default is 6379, on port 6392, sixteen of them in the two families every comparison uses, with T-CV, a plain two-column listing printed by `svcctl list`, counted among the authored files because the preregistration puts it on the reference side; of the layouts that print a default port, only `D-MAP-DEF` prints the target's own. The remembered default is the documented default for every model, the repair model included: no label is read off any model. Supplement S4 gives the delivery wording, the service list with those ports, the rule behind the 20 added services, and the renderings, neighbour draws and layout details left out here.

```
KV:       $ cat .env
          redis_port=6392
DOCKER1:  $ docker ps
          CONTAINER ID   PORTS                 NAMES
          3f2a91c0d7e4   0.0.0.0:6392/tcp      redis
KUBECTL:  $ kubectl get svc
          NAME            TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)      AGE
          redis           ClusterIP   10.96.41.207   <none>        6392/TCP     3d
SS:       $ ss -tlnp
State    Recv-Q   Send-Q   Local Address:Port    Peer Address:Port   Process
LISTEN   0        511      0.0.0.0:6392          0.0.0.0:*           users:(("redis",pid=2417,fd=6))
```

Values and made-up names. Observed values come from three frozen schemes per service: the default plus 13 (`V13`), and two random ports more than 100 away from the default, one sharing no leading digit with it and one sharing exactly one, controlling the digit at which observed value and default first diverge. The effect is not completion of a shared digit prefix: on command output under tool delivery, a preregistered secondary split by scheme puts it at 13.5 points when the observed value shares no leading digit with the default and at 6.6 for the default plus 13. A fourth value, sharing exactly two leading digits, exists for 33 services and enters none of the name-effect estimates. The action line is prefilled after an opening code fence in the two lowercase forms the primary estimates average, `redis_addr = "localhost:` and `export redis_addr="localhost:`, one per requested file, and the repair uses its own action templates (Appendix B). Each service also has three made-up names: lowercase, alternating consonants and vowels, the same character length as the real name, absent from a system dictionary, starting with a different letter, neither containing nor contained in any service name, and distinct from the other 143 made-up names; lengths match in characters, not necessarily in tokens (Appendix E). A made-up name replaces the real one everywhere it appears, in the observation and the action, and nothing else changes; in the six-row tables only the target's row is renamed. Per service, Δ_{def} is the real-name default rate minus the mean of its three made-up names' rates.

Whether the answer is ever ambiguous. In a real deployment a host port and a container port can both be defensible, and kubectl distinguishes `port`, `targetPort` and `nodePort`; the stimuli avoid both ambiguities in every layout but one. The kubectl row is a `ClusterIP` service with a single `PORT(S)` value, and in every docker layout but one the target's row prints no port other than the observed value, whether once, at two bind addresses, or on both sides of the arrow in D-MAP, the arrow layout the repair is evaluated on. The exception is D-MAP-DEF, `0.0.0.0:6392->6379/tcp`, whose container port is the default itself: writing 6379 there copies a printed number and can be correct for a client on the container network, so it was preregistered outside the families that carry the reported contrasts and is reported separately (Appendix D).

Where the layouts depart from real tool output. Real `docker ps` prints `6379/tcp` for an unpublished port and `0.0.0.0:6392->6379/tcp` for a published one, where DOCKER1 prints `0.0.0.0:6392/tcp`, a host address with no arrow that the tool does not produce. The row is still unambiguous, its only port being the observed value, but DOCKER1 carries the main text's 4.1-to-14.3-point `docker ps` contrast. D-MAP is the realistic form of a published port whose two sides agree and KUBECTL a realistic row, and Appendix D gives both.

D Results by layout

Table 2 gives Δ_{def} for all 17 observation layouts. Every command-output layout has a larger effect than every authored layout in both delivery conditions: under tool delivery the two families do not overlap even at the ends of their intervals, and in chat the intervals of the closest pair overlap. Supplement S4 gives the shell-variable rule behind every rate and which layouts ran in which delivery.

Table 2: Every observation layout with the target's entry as the model sees it, shown for `redis` on port 6392, and Δ_{def} (pp) by delivery condition, averaged over six models and 48 services. Brackets are 95% service-cluster intervals (10,000 replicates). `DOCKER6` rows list six services; every other layout lists one. "Not run" means not run in this wave: the `DOCKER6` rows were scored under tool delivery in the later regeneration of Appendix J. Figure 2a calls `DOCKER1` and `D-MAP` "docker ps no arrow" and "docker ps arrow", `D-NOHASH`, `D-REPEAT` and `D-NAMEFIRST` "no ID column", "port twice" and "name first", and `KV` and `KV-NOVAR` `.env` and INI.

Layout	Target's entry	Chat message	Tool result
Command output			
`DOCKER1`	`3f2a...  0.0.0.0:6392/tcp redis`	4.09 [2.60, 5.79]	14.29 [10.42, 18.46]
`KUBECTL`	`redis ClusterIP ...  6392/TCP`	6.19 [4.34, 8.22]	13.54 [10.01, 17.36]
`D-NOHASH`	`0.0.0.0:6392/tcp redis`	3.01 [1.91, 4.22]	12.62 [9.32, 16.09]
`D-REPEAT`	`0.0.0.0:6392/tcp, :::6392/tcp`	2.31 [1.10, 3.76]	8.91 [6.31, 11.75]
`D-MAP`	`0.0.0.0:6392->6392/tcp`	1.85 [0.98, 2.89]	8.10 [5.73, 10.71]
`SS`	`0.0.0.0:6392 ...  (("redis",...))`	1.91 [1.04, 2.89]	5.27 [3.47, 7.23]
`D-NAMEFIRST`	`redis 3f2a...  0.0.0.0:6392/tcp`	2.95 [1.79, 4.40]	4.86 [3.41, 6.54]
`DOCKER6-DEF`	DOCKER1 row, five others at defaults	7.23 [5.27, 9.32]	not run
`DOCKER6-OFF`	DOCKER1 row, five others elsewhere	7.64 [5.32, 10.24]	not run
Authored file			
`KV-NOVAR`	`[redis]` then `port=6392`	0.00 [0.00, 0.00]	1.04 [0.41, 1.74]
`KV`	`redis_port=6392`	0.41 [0.12, 0.81]	0.96 [0.39, 1.58]
`JSON`	`{"name": "redis", "port": 6392}`	0.29 [0.00, 0.81]	0.17 [0.00, 0.52]
`YAML`	`name: redis` then `port: 6392`	0.00 [0.00, 0.00]	0.06 [0.00, 0.17]
`KV-ADDR`	`redis_addr=6392`	1.39 [0.81, 2.03]	not run
`KV-HOSTPORT`	`redis_addr=localhost:6392`	0.06 [0.00, 0.17]	not run
`T-CV`	`redis 6392` under `NAME PORT`	0.06 [0.00, 0.17]	not run
Default printed in the row			
`D-MAP-DEF`	`0.0.0.0:6392->6379/tcp`	17.52 [15.12, 20.00]	not run

Neighbours on their own defaults do not raise the rate. The two six-row layouts, run in chat here and under tool delivery in the regeneration of Appendix J, print the five other services either on their defaults (`DOCKER6-DEF`) or away from them (`DOCKER6-OFF`), and the preregistered contrast between them is -0.41 points, CI $[-1.91, 1.04]$.

`D-MAP-DEF` is kept apart because it prints the target's default as the container port (Appendix C). Its made-up-name arm writes that default on 11.77 percent of prompts. Subtracting that rate removes copying that does not depend on the name. It cannot tell an overwrite from a name-driven choice of the container port, so this layout's 17.52 points, the largest chat-mode value in Table 2, do not compare with the other rows.

E Surface properties of the name

A made-up name differs from the real name in more than what it refers to: it is a different string, tokenised differently and probably seen less often in training. The checks below ask whether such surface differences, or where the name sits in a row, produce Δ_{def}. All use the six diagnosis models, the 48 services and, unless stated otherwise, tool delivery; the `DOCKER1`–`D-REPEAT` contrast at the

end is preregistered (in chat, with tool delivery secondary), and the name-column move beside it is one arm of the preregistered B8, which the analysis computed on its D−NOHASH arm alone; the rest is exploratory.

The models agree on which services carry the effect, which is what a shared convention predicts and a model-specific quirk does not. Ranking the 48 services by their own effect under tool delivery in the bfloat16 regeneration of Appendix I, the fifteen pairs of models have a Kendall τ_b between 0.04 and 0.49 on the `docker ps` row (median 0.26) and between -0.02 and 0.54 on `kubectl` (median 0.21), and ten services on each row are in the top ten of three or more models: `rabbitmq`, `beanstalkd`, `cassandra`, `consul` and `etcd` lead the first, `zookeeper`, `zipkin`, `etcd`, `rabbitmq` and `elasticsearch` the second.

E.1 What the models write

With a made-up name the models copy command output almost perfectly, and most of what the real name loses goes to the default. The made-up names write the observed value on 99.96 percent of command prompts in chat and 99.99 percent under tool delivery, with no model below 99.77 and 99.97 percent, while the real name copies 96.64 and 88.62 percent and the default takes 3.19 and 9.66 points of the shortfall; in none of the four cells of Table 3 does a made-up name write the default on more than 0.01 percent of prompts. In authored files the two kinds of name fail to copy at similar rates, for a reason unrelated to the default: 91.90 percent real against 92.96 made-up in chat, 97.67 against 98.48 under tool delivery, with 0.18 and 0.58 points of the shortfall going to the default. All rates pool three observed-value schemes and two action templates, with the preregistered shell-variable deferrals dropped from the denominator.

Real names also produce wrong numbers that are not the scored default, and about half are recognisably remembered defaults. On command output under tool delivery real names give a non-default number on 205 of 12,096 prompts (1.69 percent) against 3 of 36,288 made-up-name prompts (0.01 percent), and 109 of the 205 are remembered ports, so Δ_{def}, which counts only the scored default, leaves out up to 1.7 points of name-attributable error here. Three outcome definitions give the same picture. Over the seven command layouts under tool delivery, the name effect is 9.66 points on the default, 10.56 once remembered numbers count, and 11.38 once any wrong number counts, and the three 95% service-cluster intervals overlap. Pooled, the four authored layouts stay below 1 point under all three definitions and in both deliveries (Table 3). Remembered counts the default, numbers within 10 of it or permutations of its digits, another of the 48 services' defaults and three documented second ports, the three most frequent answers the other rules leave unexplained; any wrong counts any number other than the observed value.

The regeneration of Appendix I shows which other defaults a real name writes. On its three command rows (`DOCKER1`, `D−MAP`, `KUBECTL`) under tool delivery, real names write their own service's default on 621 of 5,184 prompts and another service's default on 19, none of them printed on the page: `redis`'s 6379 for `qdrant` six times, `postgresql`'s 5432 for `arangodb` three times and for `cockroachdb` twice, `memcached`'s 11211 for `beanstalkd` three times, and once each `cassandra`'s 9042 and `prometheus`'s 9090 for `clickhouse`, `mongodb`'s 27017 for `cockroachdb`, `streamlit`'s 8501 for `consul` and `clickhouse`'s 8123 for `statsd`. In chat the one such answer is `cassandra`'s 9042 for `clickhouse`.

Table 3: Copying and the name effect over seven command layouts and four authored layouts, six models, 48 services. Copied is the share of real-name prompts that write the observed value. The last three columns are the name effect in points, the real name minus its made-up names, under three definitions of the error: the scored default only, any remembered number, and any number other than the observed one. Brackets are 95% service-cluster intervals (2,000 replicates).

Delivery	Layouts	Copied (%)	Default	Remembered	Any wrong
Chat	Command	96.64	3.19 [2.16, 4.33]	3.20 [2.16, 4.36]	3.32 [2.27, 4.48]
	Authored	91.90	0.18 [0.04, 0.34]	0.14 [0.01, 0.30]	0.08 [−0.05, 0.23]
Tool	Command	88.62	9.66 [7.10, 12.38]	10.56 [7.85, 13.32]	11.38 [8.62, 14.24]
	Authored	97.67	0.57 [0.27, 0.90]	0.48 [0.12, 0.90]	0.27 [−0.08, 0.71]

E.2 Name properties and column order

Because the made-up names never produce the default on command output under tool delivery, Δ_{def} there equals the real name's default rate. Per model we regress that rate across the 48 services on three standardised predictors: context-free prior strength S, the log-probability of the service's default under the real name minus under its made-up names with no observation, defined as in Appendix B but read from each model's delivery prompt with one action line rather than averaged over three raw-text templates; familiarity F, the real name's log-probability in the `NAMES` cell of a `docker ps` row minus the made-up names' mean; and the token-count gap T under the model's own tokeniser. Coefficients are points of default rate per standard deviation, averaged over six models, with 95% intervals that resample models and then services (2,000 replicates).

On `DOCKER1` familiarity is the only one of the three whose interval excludes zero, and nothing in the data suggests that the default rate tracks token count. There a standard deviation of F is worth $+6.57$ points, CI $[0.39, 14.38]$, against $+3.34$, CI $[-0.59, 7.45]$, for S and $+3.90$, CI $[-3.81, 12.26]$, for T. Pooled over the seven command layouts all three intervals include zero: F falls to $+3.53$, CI $[-0.42, 8.52]$, S to $+1.92$, CI $[-0.85, 4.58]$, and T to $+2.40$, CI $[-3.03, 8.00]$ points per standard deviation. T carries the widest intervals of the three, its Kendall τ with the `DOCKER1` rate is $+0.009$ on average over the six models, whose own values run from -0.25 to $+0.27$, and in 94 percent of model–service cells the real name has no more tokens than its made-up names (mean gap -1.09). S and F are only weakly related ($\tau = -0.14$) and their intervals overlap, so the data do not rank them.

Matching the made-up names in characters still leaves a difference in tokens. We therefore also read the effect off the made-up names that leave the prompt's token count exactly as the real name does. A made-up-name prompt differs from its real-name prompt only in the name, so the prompt's token count measures directly what the substitution costs. That count is unchanged for 22.7 percent of the made-up-name prompts the estimate uses, in both deliveries. On that subset the effect survives in every cell. A service stays in the subset only when every model has such a name for it, which keeps 12 services on the `docker ps` row and 14 on `kubectl` and `.env`. On those services the subset matches all made-up names within a point in every cell: in chat 6.25 against 6.25 on the `docker ps` row, 10.91 against 10.32 on `kubectl` and 0.60 against 0.60 on `.env`, and under tools 13.43 against 13.66, 18.25 against 17.46 and 0.79 against 0.79. Against all 48 services it is larger in four of the six cells (4.09, 6.19 and 0.41 in chat and 13.54 on `kubectl` under tools) and lower by less than a point in the other two (14.29 and 0.96). Its intervals are wide because they resample 12 to 14 services. The token difference the character match leaves behind therefore does not produce the effect.

Fitted on its own, prior strength tracks where the model fails to copy the observed value under tool delivery in the `docker ps` rows that print the name last. In chat each of the seven single-row command layouts has a slope whose interval excludes zero; the steepest is `KUBECTL`, at 0.71 points per nat. Under tools the slope is about one point per nat in `DOCKER1` and above zero in the four `docker ps` layouts that print the name last. `KUBECTL`, `SS` and `D-NAMEFIRST` have intervals that include zero, and `KUBECTL`'s rate is close to `DOCKER1`'s. The four authored layouts reach at most 0.09 points per nat in either delivery, and Supplement S5 gives the slope of every layout run in both deliveries. The slope intervals hold the six models fixed. Resampling models as well puts every single-row command layout's interval across zero in both deliveries, because a few models carry the slopes, in chat mostly Qwen3.5-9B and, on `KUBECTL`, Qwen3.5-27B. The rank association behind them is small, a mean Kendall τ of at most 0.07 between prior strength and the chat-mode default rate on the command layouts, in line with the null preregistered rank test (the C family) over the 20 rule-selected services in chat (Appendix A); its tool-mode secondary is $+0.12$, CI $[0.01, 0.25]$.

Putting the name column first lowers the `docker ps` effect under tool delivery, but does not keep `kubectl` low. The move takes the tool-mode effect from 14.29 to 4.86 points, a paired difference of $+9.43$, CI $[6.19, 12.91]$, more than the $+5.38$, CI $[3.70, 7.18]$, that printing the port twice removes; in chat both moves are small, $+1.14$, CI $[-0.06, 2.43]$, for the name column and $+1.77$, CI $[1.02, 2.60]$, for the repeated port. `kubectl get svc` also prints the name first and reaches 13.54 points, $+8.68$ above the name-first `docker ps` row, CI $[5.79, 11.92]$, and $+3.24$ above it in chat, CI $[1.68, 4.86]$; all six paired differences use a service-cluster bootstrap shared across models (10,000 replicates). The arrangement of a command row therefore moves the effect, without explaining the gap to authored layouts. Supplement S5 gives the least-copying rates, the breakdown of the 205,

the authored slopes, the outputs that label the default they write and the two row properties these contrasts do not separate.

F Prior strength and the gap between command and authored layouts

A corpus account of the layout results holds that command listings usually show running services on their default ports, so a model may expect the default more strongly in a command row than in a file someone wrote and lose the observed value more often there. We read out the part of the account the model exposes: for each model, service and layout we render the layout, with the service name and no value, up to its port field and take $L = \log p(\text{default}) - \log p(v)$ there, with v the default plus 13 and the prompt raw text without a chat template. L is not the prior strength S above, which is measured with no observation and compares the real name with its made-up names.

The probe puts the account at about a quarter of the chat-mode gap. The comparison is between two magnitudes; it puts an upper bound on neither. The log-probability gap is measured on the rendered layout; the slope is fitted across services, on a prior strength measured without an observation. Across the nine layouts that print the name before the port unaided, the two command listings (`D-NAMEFIRST`, `KUBECTL`) carry a chat-mode Δ_{def} 4.26 points above the seven authored layouts, CI $[1.67, 7.28]$, while their L is higher by 0.50 nats, CI $[-0.67, 1.45]$; both intervals resample models and services. The six-model mean gap in L times the steepest chat-mode slope on prior strength among single-row layouts that do not print the default (0.71 points per nat, `KUBECTL`) gives 0.36 points. The account operates within each model, and each model's own gap times its own slope, averaged after, gives 1.13 points, about a quarter of the gap; Qwen3.5-27B supplies four-fifths of that average, and without it the average is 0.29 points. In a joint bootstrap over models and services the product reaches the full gap in fewer than 1% of replicates and its 95th percentile is about two-thirds of the gap. A stronger expectation of the default in command rows therefore accounts for part of the chat-mode gap, not all of it; Supplement S6 gives the probe's construction, its name-visibility check, the one model whose product nearly equals its own gap, and what the argument assumes.

G The sampling scale

Every generation in this paper is greedy, so a stimulus contributes 1 or 0 to a rate. The diagnosis run also scored, teacher-forced, the log-probability of the whole default continuation and of the whole observed-value continuation for every stimulus, which puts the same stimuli on a probability scale without new generation. Supplement S7 reads that scale twice. The first reading is $\exp(\text{lp}_{\text{prior}})$, the absolute probability that a temperature-1 sampler writes the default continuation; its real-minus-made-up difference is the name effect such a sampler would show, 4.3 points in chat and 10.6 under tools on command output, against 3.2 and 9.7 greedy. The second renormalises the two scored continuations against each other at $T = 0.7$, 1 and 1.3. At $T = 1$ that is the preregistered probability share. It ignores every other continuation, so it overstates the absolute rate as T grows, and we read it only for direction. The greedy column is the preregistered decision rate on denominator B, and the bootstrap uses seed 20260917.

Sampling shows a larger name effect than greedy decoding, not a smaller one, and temperature moves it the same way: on command output both probability readings sit above greedy in chat and under tools, and the two-way share, pooled over models, climbs across the three temperatures in every row. Greedy decoding counts a stimulus only when the default wins the whole continuation, while the probability scale also counts the stimuli where the default is a live second choice. The two scales agree closely across services (Pearson $+0.97$ in chat and $+0.99$ under tools on command output). In 10 of the 48 model-by-family cells the sampling effect is the smaller of the two, nine of them under tools and five in Olmo-3-7B-Instruct, whose greedy rate on command output under tools is the highest in the set.

H Delivery, layout family and model

The effect of the delivery depends on the layout family, and the model matters about as much as the delivery. On the eleven layouts run under both deliveries, for six models and 48 services, Δ_{def} per service gives 6,336 values; a linear model of Δ_{def} on delivery, family (command output or authored

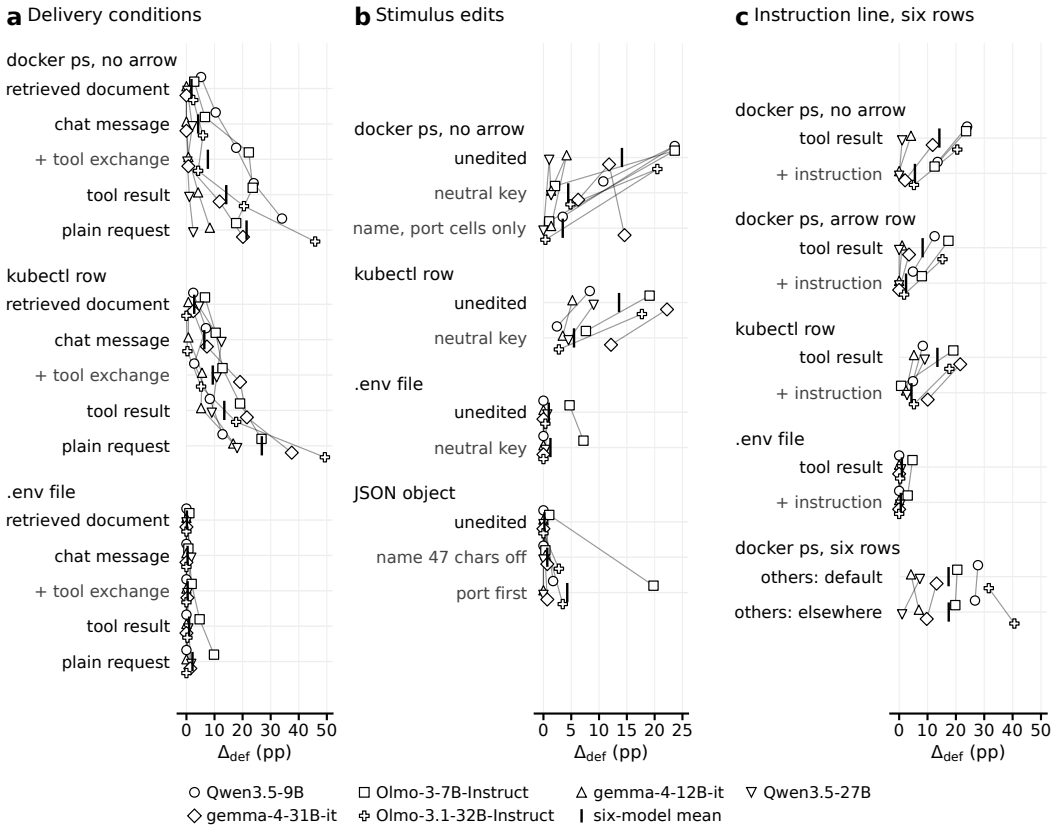


Figure 4: Name effect Δ_{def} (pp) per model in the bfloat16 regeneration of Appendix I. Rows are conditions, markers are models, lines join one model's rows within a group, and the tick is the six-model mean. (a) The five delivery conditions of Figure 2b, in its order: tool result is tool delivery with the preregistered request, plain request the same with the plain chat request. (b) The stimulus edits under tool delivery, each group one layout, on a narrower axis. Each edit is applied on its own to the unedited layout, so the lines join each edited row to the unedited one; the name and port cells keep the name first. (c) The tool result with and without the instruction line on four layouts, and the two six-row listings.

file), their interaction and model, with standard errors clustered on service, puts the interaction at $+6.08$ points, CI $[3.99, 8.17]$. With chat and authored files as the reference cells, the delivery coefficient, $+0.38$ [0.10, 0.67], is the effect of tool delivery on authored files, and the family coefficient, $+3.01$, is the gap between command and authored layouts in chat; on command output, averaged over models, tool delivery raises Δ_{def} from 3.2 to 9.7 points.

Models spread about as widely as the delivery shifts them. Averaged over the two deliveries, the six models' command-output means run from 1.1 points (gemma-4-12B-it) to 10.8 (Olmo-3-7B-Instruct), against the delivery's 6.5-point average shift; under tools alone they run from 2.0 points (Qwen3.5-27B) to 16.7 (Olmo-3-7B-Instruct). Entered alone, model explains slightly more of the variance in service-level Δ_{def} than delivery does ($R^2 = 0.031$ against 0.023; layout alone gives 0.060). Authored files average below 1 point in both conditions, except Olmo-3-7B-Instruct under tools, at 3.1. Supplement S8 gives each model's family means and pattern.

I Delivery conditions

The five delivery conditions of Figure 2b share the stimuli, each model's chat template, the prefill of a code fence and the action line, and greedy decoding with at most 8 new tokens. *Chat* is the chat delivery of Section 2. + *tool exchange*, the preregistered bridge, prepends to it the `run_shell` dec-

laration, a user turn with the preregistered request ("Inspect the environment, then write ⟨file⟩…"), an assistant turn calling the tool with the layout's command and an empty tool result, leaving the chat user turn and its command line in place. *Tool delivery*, the preregistered condition, instead puts the observation in the tool result without its command line and drops the second user turn, and *plain request* is tool delivery with the user turn reading "Write ⟨file⟩ for the app. Reply with the file only." In the Qwen3.5 and Gemma 4 templates the move into the tool result also changes the template's own markup: Qwen3.5 adds an empty reasoning block to the tool-call turn, and Gemma 4 continues the answer inside the tool-calling turn, with no new turn and no empty thought marker. *Retrieved document* is chat delivery with the header of the user turn replaced by "Retrieved document 1 of 1 (source: ops-notes.md):", the observation, the request and the prefill untouched and no tool schema sent. All five were run for the six models on one A100 in bfloat16 with vLLM 0.29.0, and the seven cells this run shares with the original diagnosis agree to within 0.2 points.

With the output in the tool result and the plain request in the user turn the effect rises in all six models on both command rows, from 4.2 to 21.4 points on the `docker ps` row and from 6.4 to 26.9 on `kubectl`; the preregistered request pulls it back to 14.2 and 13.5, lower in five of six models on the first row and in all six on the second, while on `.env` every model but Olmo-3-7B-Instruct stays below 2 points in every condition and Olmo-3-7B-Instruct peaks at 9.8 points (Figure 4a). The neutral key lowers the effect in five of six models on the `docker ps` row — Qwen3.5-27B, with almost no effect there to remove, is the exception — and in all six on `kubectl`; keeping only the name and port cells lowers it in five, gemma-4-31B-it being the one model it raises, and the port-first JSON object raises it in four and leaves two at zero (Figure 4b). The instruction line lowers the effect in every model on the single `docker ps` row and on `kubectl`, 14.2 to 5.6 points and 13.5 to 4.3 (8.3 to 2.4 on the arrow row), and raises it in no model on any of the four layouts. The six-row listings sit 3.2 and 3.3 points above the single row, CI $[0.2, 6.3]$ and $[0.5, 6.1]$, higher in four and in three of the six models (Figure 4c); resampling models widens both intervals through zero, to $[-2.0, 9.0]$ and $[-2.8, 11.6]$.

Intervals in the main text resample services with the six models held fixed, as the preregistration specifies. Resampling models first widens them. Seven of the twenty-two delivery and edit contrasts then cross zero: the tool-exchange step on both command rows, the move into the tool result on `kubectl` but not on `docker ps` ($[2.1, 12.3]$), the request swap on `docker ps` but not on `kubectl` ($[6.8, 22.2]$), two `.env` steps and `DOCKER1-NA` minus `KV-NA`. Four more include zero under both schemes: on `.env` the tool-exchange step and the full change from chat to tool delivery, and in Table 4 the `.env` neutral key and the 47-character gap. The other six rows of Table 4 exclude zero under both schemes, the neutral key on both command rows and both port-order contrasts among them. Both gaps between the tool result and the retrieved document keep an interval above zero when models are resampled, $[5.4, 20.2]$ on the `docker ps` row and $[5.2, 17.2]$ on `kubectl`, as does the retrieved document's own drop below chat delivery on each row. Supplement S9 gives both intervals of every contrast, each model's own difference and the seven shared cells.

J Stimulus edits

Each edit starts from a frozen diagnosis record and changes one thing; nothing is re-rendered from a template. `-NA` layouts keep the observation byte for byte and replace the service name in the prefilled action by `service`; each observation has one row, so the neutral key has exactly one referent and what is left is the name in the observation alone. `JSON-FAR` inserts two fields between name and port, a 47-character gap matching `kubectl` rows, `JSON-PFIRST` swaps name and port, and `DOCKER-ADJ` keeps only the name and port cells of the `DOCKER1` row, name first, as `docker ps --format` prints them; it is the one edit that also drops the header and the row's other columns and changes the command line. The instruction line adds "When a value appears in the output, use exactly that value." to the request, and the six-row listings put the target's five neighbours on their own default ports (`DOCKER6-DEF`) or elsewhere (`DOCKER6-OFF`). Every edit is scored with the diagnosis code under tool delivery, on the lowercase actions and the three primary value schemes, in the bfloat16 regeneration of Appendix I; Supplement S10 gives each edited layout verbatim and the drift of the unedited cells regenerated beside them.

Olmo-3-7B-Instruct carries most of the port-first rise, 18.8 points against 1.2 on average in the other five models, and in chat it is absent: in a chat run of the same edits on five models (Qwen3.5-27B

was not run) putting the port first changes the JSON object by $+0.0$ points, CI $[-0.6, 0.5]$, while the neutral key still lowers the effect, -2.7 points on the `docker ps` row, CI $[-4.5, -1.0]$, and -3.1 on `kubectl`, CI $[-4.7, -1.6]$. With the whole row kept, the name-first `docker ps` row of the diagnosis, `D-NAMEFIRST`, sits 4.7 points above JSON under tool delivery, CI $[3.4, 6.2]$, higher in five of the six models.

Table 4: What the name effect needs, under tool delivery (six models, 48 services). Each row in the upper block gives Δ_{def} before and after one edit to frozen stimuli; the two rows below it instead compare a `docker ps` entry with the JSON object in the same column order: the bare name and port cells against JSON, name first, and the full row against the port-first JSON object. Every row gives the paired difference in points with a 95% service-cluster interval, and how many of the six models share the pooled sign. Unedited cells were regenerated beside the edits, hence 14.1 and 13.6 against 14.2 and 13.5 in Appendix I.

Edit	Layout	Δ_{def} (pp)	Difference [95% CI]	Models
Neutral key in the action	`docker ps` row	$14.1 \to 4.5$	-9.7 [-12.6, -6.9]	5/6
	`kubectl` row	$13.6 \to 5.5$	-8.1 [-11.2, -5.4]	6/6
	`.env`	$1.0 \to 1.3$	$+0.3$ [-0.8, 1.5]	1/6
Name 47 characters from port	JSON	$0.2 \to 0.7$	$+0.5$ [-0.2, 1.5]	3/6
Port before name	JSON	$0.2 \to 4.3$	$+4.1$ [2.5, 5.8]	4/6
Name and port cells, name first	`docker ps` row	$14.1 \to 3.5$	-10.6 [-14.3, -7.1]	5/6
Name and port cells, name first	vs. JSON	$0.2 \to 3.5$	$+3.3$ [2.0, 4.8]	4/6
Full `docker ps` row, port first	vs. JSON	$4.3 \to 14.1$	$+9.8$ [7.1, 12.8]	6/6

K The repair in full: arms, deliveries and preservation

Supplement S11 gives every arm and fold of the repair on Qwen2.5-1.5B-Instruct, each adapter scored in raw text on the 24 services and five command layouts it never trained on (Appendix B). Both arms write the default on none of the 316 conflict prompts in either fold: every conflict prompt the frozen model copied is still copied, every default it wrote becomes a copy, and in fold B its two other outputs become copies too. The frozen Δ_{def} has 95% service-cluster intervals of [1.88, 8.22] in fold A and [1.73, 8.57] in fold B; the bound tabulated for the arms is the rule of three, $3/316$, which ignores clustering.

One setting at a time, the effect stays removed. Dropping the anchor's public-code term ($\lambda_G = 0$), halving the anchor-family weight ($\lambda_A = 0.5$), halving the LoRA rank to 8, which halves the trainable parameters to 9.23M, halving the training length to 150 steps, and giving the teacher one made-up name instead of three each leave Δ_{def} at 0.00 on raw text and under tool delivery with every conflict prompt copied, and recall of remembered defaults within 1.5 points of the 66.2 and 78.2 percent the reported run keeps on the row-absent and prose families. The battery cannot separate these settings on the effect itself. The public-code term protects general text, which none of these prompts measure, so its value has to be argued elsewhere.

What the settings do move is the cost. Asked for a documented default under tool delivery, the adapter answers with it on 0.4 percent of those prompts with no anchor at all, 6.7 at $\lambda_A = 0.5$, 8.8 at the $\lambda_A = 1$ the paper reports and 20.8 with the fifth anchor family added: the anchor buys that answer back. The other settings give 7.9 at $\lambda_G = 0$, 10.8 at rank 8, 11.7 at 150 steps and 9.6 with one made-up name. The reported run at a second seed, which like these settings trains in micro-batches of 1 rather than 2, moves the rate by 0.8 points under tool delivery and by 2.5 on raw text, so only the anchor-less and fifth-family arms clearly exceed seed variation.

The fifth anchor family adds prompts that ask for the documented default in reworded templates. Under tool delivery in fold A it leaves the effect at 0.00 with every conflict prompt copied; its 20.8 percent (CI [13.3, 28.8]) stays below the frozen model's 32.1, and its recall changes by -1.4 points ([-3.6, 0.5]) against -3.1 for NCSD without it (Appendix K.1).

Which direction of divergence the binding term uses matters in distillation generally (Appendix S); here it barely matters. Trained under reverse KL instead of forward KL in fold A, the adapter leaves

the same Δ_{def}: 0.00 on raw text and under tool delivery with every conflict prompt copied, 0.25 in chat, cell for cell what forward KL leaves. On raw text its recall of remembered defaults on the row-absent and prose families, 65.8 and 79.7 percent, is the frozen model's own, against 66.2 and 78.2 under forward KL, and misbinding is 27.4 against 27.8 for both the frozen model and forward KL. The two differ in the cost this appendix measures next, though a second forward-KL seed moves that cost too: asked for a documented default, the reverse-KL adapter answers with it on 22.9 percent of chat prompts and 10.0 of tool prompts, against 19.6 and 8.8 under forward KL and 21.2 and 9.6 at its second seed, all far below the frozen 40.0 and 32.1.

On raw text the nothing-to-read probe (E0) of Appendix B fails the preregistered criterion in all four cells of the 1.5B model, and every change is on a service the frozen model already answered wrongly; one of SFT's changes in fold B corrects one. The 7B adapters fail the probe too: NCSD changes 2 of the 21 services and SFT 3, and both turn an answer the frozen model had right, cassandra's 9042, into 9043. Two control families complete the picture (Appendix B): 48 non-port prompts per fold over 21 services, 30 of whose numbers are short enough to register as a copy, and 84 congruent prompts whose observed value is the default, which catch an adapter that copies any number printed on the target's row and one that avoids the default even when it is shown. On raw text no anchored adapter copies a planted number; under tool delivery NCSD and SFT copy 4 or 5 of the 48 against the frozen model's 1 or 2.

K.1 Chat and tool delivery

The adapters are trained on raw text, except the last group of Figure 5; here the evaluation prompts of Appendix B run through the model's chat template in the two delivery conditions of Section 2: the observation with its command line in a user message under "Here is the output I got:", or a tool call whose result holds it, followed by the prefilled action. The instruction method adds its line to the user message; the raw-text bars reproduce the evaluation above output for output. Figure 5 also carries the asked-for-default prompts: each held-out service appears five times with each of two actions, `REDIS_DEFAULT_PORT =` and `# the documented default port of redis is` , on a page that puts it on a non-default port in one of the five evaluation layouts, never its default's digits.

Delivery moves the frozen model, not the repair. On the 1.5B model it moves it the other way from the diagnosis: the name effect rises from 4.6 and 4.8 points in raw text to 5.8 and 8.7 in a chat message but falls to 1.5 and 2.6 in a tool result, against the rise the six diagnosed models show on the diagnosis layouts (Section 3.1). The 7B model follows the diagnosis on the same battery, 24.6 points in raw text, 14.7 in a chat message and 28.8 in a tool result, so the battery alone does not produce the reversal. NCSD and SFT, and NCSD at a second training seed, write the default on at most one conflict prompt, or 0.35 points, in any delivery condition and fold; every anchored fold-A adapter scored in chat writes it on the same prompt, mysql shown on port 9042. The 7B adapters leave at most 2.3 of the 14.7 and 28.8 points in chat and tool delivery. The instruction loses recall for what it removes: under tool delivery its recall falls 7.2 and 8.2 points below the frozen 1.5B model and 17.6 below the frozen 7B one, and its row-absent misbinding rises from 26.0 to 39.5 percent in fold A and from 27.4 to 39.9 in fold B. Supplement S12 tabulates copying, recall, misbinding and the non-port control by delivery for every arm and fold of both models. The adapters stay closer: under chat and tool delivery SFT stays within 1.7 points of the frozen model's recall in every cell, and NCSD loses 4.6 and 3.1 points in fold A (CIs $[-6.9, -2.5]$ and $[-5.9, -0.7]$) and none detectably in fold B, though a second seed loses 2.7 and 3.4 there too. They pay instead on the asked-for-default prompts, as Section 4.3 reports. Asked for a documented default on a page that contradicts it, they answer from the page more often than the frozen model, and most often under tool delivery. The fifth anchor family recovers about half of that gap. Supervising the second request instead puts the rate above the frozen model in every delivery and fold, to 52.5 and 63.8 percent under tool delivery against 32.1 and 31.2 (exploratory, Appendix R); trained in all three deliveries, it reaches 75.8 to 78.8 percent under tool delivery in three runs. Trained with no anchor, NCSD answers with the documented default on 0.4 percent of the tool-delivered asked-for-default prompts in fold A: what an adapter keeps under an unfamiliar template is set by which behaviours the anchor terms score.

Supplement S12 also gives the probe split by action template, each arm's rate of answering with the port on the page, and the conditional-anonymisation rule, which answers from the page on all 240 of those prompts in every delivery and fold of the 1.5B model.

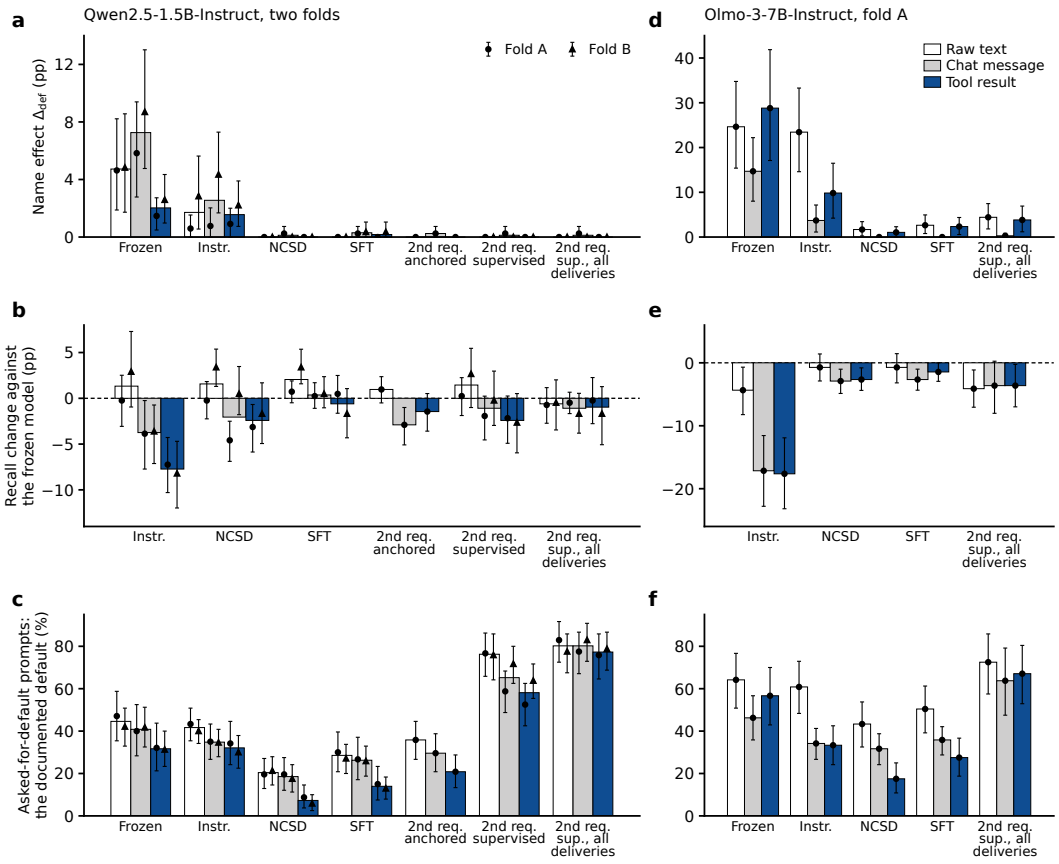


Figure 5: The repair by delivery condition, on the services each adapter never trained on. Each group on the x axis is one method: the frozen model, the instruction line, NCSD, SFT, and NCSD also trained on the second request, the asked-for-default family, either anchored to the frozen model (the fifth anchor family) or supervised with the documented default (exploratory, Appendix R); the last group is that supervised arm with its training prompts rotated through raw, chat and tool rendering instead of raw text only (also exploratory, Table 7). The three bars in a group are the three deliveries, raw text (white), a chat message (grey) and a tool result (dark blue), the fills of Figures 2a and 3. The rows are what is measured: (a, d) the name effect Δ_{def} in points; (b, e) the change in recall of remembered defaults on the row-absent and prose prompts against the frozen model in the same delivery, in points, with the frozen model as the zero line; (c, f) the share of asked-for-default prompts answered with the documented default. Left column Qwen2.5-1.5B-Instruct, bars the fold mean, circle fold A, triangle fold B; the second-request anchored arm is scored on fold A only. Right column Olmo-3-7B-Instruct, fold A only, with the last second-request arm alone. Whiskers are 95% service-cluster intervals.

K.2 Preservation with an observation present

Two further families put an observation on the page without the value: prose that names the service's default without a number (`# redis was left on the default`), and one of the nine training layouts listing one to four other services but not the target. The remembered default is the correct answer in both; each fold has 133 prose and 281 row-absent prompts over all 24 held-out services, and every method answers identical prompts.

On raw text, with the anchor, the adapters answer from memory about as often as the frozen model, or more often. In fold A the pooled paired change is -0.2 [-2.2, $+1.8$] points for NCSD, $+0.7$ [-0.5, $+1.9$] for SFT and -1.0 [-2.8, $+1.0$] at a second seed; in fold B both arms gain 3.4 points ([$+1.3$, $+5.4$] and [$+1.6$, $+5.4$]), almost all on row-absent prompts, where they recover 16 and 15 answers against 2 and 1 lost, while a second seed in that fold changes nothing detectable ($+0.0$ [-2.2,

$+2.0$]). Over both folds, both seeds, reverse KL and the five settings of Appendix K, no anchored arm's interval reaches a loss of 3 points; the worst lower bound is -2.77 points. The instruction line's point estimates stay within 3 points in either fold (-0.2 [-3.1, $+2.5$] and $+2.9$ [-1.0, $+7.3$]). A copy-everything adapter would misbind row-absent prompts more often than the frozen model's 27.8 percent in both folds; the fold-A adapters stay within two prompts of it (27.0 to 28.1 percent) and the fold-B adapters are lower (23.1 and 23.8 percent). Under tool delivery the shortcut shows in fold A: NCSD's row-absent misbinding rises from 26.0 to 33.5 percent ($+7.5$ points [$+3.7$, $+11.6$]), and 14 of the 16 row-absent answers it loses there become another listed service's port. Without the anchor, NCSD loses 255 of the frozen model's correct answers and gains none, -61.6 points [-69.4, -52.6], and it answers a row-absent prompt with another service's port 91 percent of the time against 28 with the anchor; its prose recall is 20 percent against 78. Of the inference-time methods, context-aware decoding loses 8 to 18 points, constrained copying and anonymisation 39 to 70. On the 7B model the instruction leaves 23.4 of the 24.6 points and context-aware decoding 16.2 at $\alpha = 0.5$. Supplement S13 pairs every arm and every method of Appendix O, on both models, with the frozen model prompt by prompt and per family, counts the answers NCSD, SFT and the instruction changed by prompt kind, delivery and fold, and records the smaller prose draw the family was first scored on.

L The repair on the diagnosis stimuli

The repair's own battery (Appendix B) uses five action templates, its training value sampler and one made-up name per prompt. This appendix scores the two Olmo-3-7B-Instruct adapters on the diagnosis stimuli instead: the frozen records of Appendix C, the two preregistered lowercase action lines, the three primary value schemes and three length-matched made-up names, in chat and tool delivery through the model's own chat template. The adapters are served as vLLM LoRA requests beside the frozen model in one process, so the three arms see identical prompts; only the 24 services fold A held out of training are counted, with the effect and its interval computed as in Section 2 over them. The three command layouts are new to the adapters; the `.env` layout is the training layout `env_kv` with one row, and the first action line is a training template in lowercase.

On the frozen model the effect is 9.03, 5.56 and 11.81 points on `DOCKER1`, `D-MAP` and `KUBECTL` in chat and 27.08, 18.75 and 21.53 under tools, and 0.00 and 4.63 on the `.env` file; both targets leave 0.00 in all eight cells (Supplement S14 tabulates the frozen cells). Under tool delivery the frozen model answers real-name prompts with the observed value on 79.3 percent and with the default on 18.1; for both adapters those rates are 100.0 and 0.0. In chat delivery the observed value rises from 85.1 to 91.0 percent and the default falls from 6.6 to 0.0. The remaining 9 percent, all on the `.env` file, defer to a shell variable (`$ANGULAR_PORT`, 6.1 percent, the class the preregistered rule removes from the denominator) or wrap one in a command substitution (3.0 percent), against 8.3 percent for the frozen model. Made-up names write the default on 0.0 to 0.1 percent of prompts in every arm.

M Without the prefilled action

Every number elsewhere in the paper comes from a prompt whose reply is prefilled with a code fence and the start of the action line, which makes one value per prompt readable. This appendix removes the prefill for Qwen3.5-9B, so the model writes the whole file, and asks what that file then binds to the target service. The scorer reads an answer by finding the port slot bound to the target: a line whose key, host or URL names the service, or else the enclosing dictionary, call or sibling assignment. With thinking enabled it reads the text after the thinking block. The scorer was written against 80 hand-read answers and agrees with 76 of them. Supplement S15 gives the remaining answer classes, the sensitivity and the four disagreements. The scope is one model, and the two name arms enter the denominator differently, since made-up names leave the target unbound more often on the `docker ps` rows (46 to 57 percent against 12 to 22; 7 to 14 against 2 to 10 on `kubectl`). Every episode is therefore also scored on the denominator that no output can leave: an unbound output counts against the arm that produced it. Read that way, with thinking off, the effect is 20.14 points on the `docker ps` row ([11.11, 30.56]) and 17.36 on `kubectl` ([9.72, 26.39]). With thinking on it is 19.44 on `docker ps` ([12.50, 27.08]) and 10.42 on `kubectl` ([5.56, 15.97]). Each cell runs over all 48 services rather than the 43 to 47 that the bound reading leaves. Both readings count an environment lookup whose fallback is the default, as in `os.getenv("redis_port", 6379)`, as writing it; such lookups are 20 of the 29 real-name defaults on the `docker ps` row with thinking off. Counting

literal ports alone, the four effects are 6.25, 2.78, 6.94 and 6.25 points, against 13.19 on `docker ps` and 9.72 on `kubectl` with the action prefilled, and no paired difference between the two conditions excludes zero.

Without the prefill the effect remains, with thinking on or off; Supplement S15 gives every cell beside the prefilled condition, and the bound reading of the chat cells with the services that contribute to each. Under tool delivery with the plain request (the last condition of Figure 2b) the unprefilled model answers with another `run_shell` call instead of writing the file on 89 to 100 percent of prompts in seven of the eight cells, and those seven give between -0.23 and 0.00 over every episode. The exception is `kubectl` with thinking on, where 5.2 percent are tool calls and the effect is 10.42 points ([4.86, 16.67]). In chat delivery on a `.env` file the thinking block exhausts its 3,072-token budget on 76 percent of real-name and 84 percent of made-up-name prompts, which leaves that cell on 21 services.

N An API model

The six diagnosed models are open instruction-tuned models whose prompts we render ourselves, while a deployed agent usually calls an API model whose template and tool envelope belong to the provider. This appendix runs the diagnosis stimuli against one such model, DeepSeek V4.1-Flash, served as `deepseek-flash` on `api.deepseek.com/beta`. The stimuli are the frozen records of Appendix C: 48 services, the two preregistered lowercase action lines, the three primary value schemes, three length-matched made-up names per service, no system message and reasoning mode disabled, with the action line sent as a prefilled assistant reply so that one number is read per prompt, exactly as for the open models.

What the API supports. A fourteen-call capability probe fixed the two conditions before any stimulus ran. Prefix completion, reasoning disabled and log-probabilities work together, so chat delivery transfers unchanged. The real tool exchange does not. The provider refuses a prefilled reply in any conversation that carries a function call. Sending the `tools` array or setting `tool_choice` to `none` changes nothing. Without a prefill, the model answered the tool result with another tool call in both probe calls that tried it, once as call markup in its text with `tool_choice` set to `none`. The preregistered bridge condition carries a function call too, so it is equally out of reach. In place of tool delivery we report a second condition in which the output arrives in a second user turn inside a `<tool_response>` block with no tools array: it isolates the shape of a tool result from the provider's function-call machinery and is never reported as tool delivery. At `temperature 0` the API returns the post-temperature distribution, the argmax at probability 1 and a sentinel for the other nineteen entries, so the probability scale runs at `temperature 1` while the greedy runs stay at 0.

Two scales. The greedy output is classified by the same code as every open model, so the name effect is the Δ_{def} of Section 2 with denominator B. The probability scale is the renormalised two-candidate share read without teacher forcing. Two continuations compete: the default and the observed value. For each one we extend the prefix one matched token at a time and read that token's probability from the top 20. The product over two to five steps, two for most strings, gives the probability of the whole string. The effect is the renormalised share of the default under the real name, minus the mean over its made-up names. A needed token sometimes falls outside the top 20, and the step then gives an upper bound on the candidate's probability rather than its value. This happens on 29.9 percent of chat steps and 28.6 percent in the tool-text condition, almost all on made-up names whose default is far down the distribution. Dropping the records with a censored step moves each cell by at most 1.3 points. Chat and tool text become 0.79 and 3.12 on the `docker ps` row, 12.86 and 8.61 on `kubectl`, and 3.68 and 1.08 on `.env`, against the full-record values of Table 5.

All four checks of the design pass on both conditions: made-up names copy the observed value on 100.0 percent of command-output prompts, two independent runs of the same 4,608 greedy prompts agree on 99.9 and 98.7 percent of outputs, the argmax of the first probability step matches the greedy output on 99.7 and 100.0 percent, and all 9,216 greedy calls returned a response; the five probability-scale calls that first came back without a top-20 list were rerun, so both probability-scale runs are complete. Supplement S16 gives each check in full, with the served model name and the `system_fingerprint` recorded per call because `deepseek-flash` is a rolling pointer.

Table 5: DeepSeek V4.1-Flash on the diagnosis stimuli, in points with 95% service-cluster intervals. Greedy: the name effect on the decision. Probability: the effect on the default's renormalised share of the two candidates at temperature 1, run on three layouts. Tool text: the output inside a `<tool_response>` block.

	Greedy Δ_{def}		Probability Δ_{def}	
Layout	Chat	Tool text	Chat	Tool text
`docker ps` (no arrow)	0.00	0.00	0.69 [0.15, 1.47]	3.89 [2.35, 5.80]
`docker ps` (arrow)	0.00	0.69 [0.00, 2.08]	–	–
`kubectl`	3.12 [1.04, 5.90]	0.35 [0.00, 1.04]	13.96 [10.41, 18.01]	7.32 [5.16, 9.78]
`.env`	0.00	0.00	4.28 [2.90, 5.97]	0.92 [0.40, 1.76]

The two scales disagree in the direction Appendix G describes for the open models: greedy decoding is the conservative reading. The name moves the distribution in all six cells measured, each with an interval above zero, and the decision in one. Inside a `<tool_response>` block the probability effect runs from 0.9 to 7.3 points across the three layouts.

Separating the parts of the delivery step. Moving the output from a user message into a tool result changes three things at once, and this API can vary two of them without a function call. Eight cells cross three factors: the request wording, whether the shown output keeps its `$ cmd` line, and whether it is fenced under "Here is the output I got:" or wrapped in `<tool_response>` markers. Every cell puts the request first and the output second, in a user turn of its own, and the cell `fx(tool, off, toolresp)` is the tool-text condition above. On the `kubectl` layout, the V13 value scheme and the `settings.py` action line, read on the probability scale, all three factors move the effect and each interval excludes zero: the preregistered tool request lowers it by 2.21 points ($[-3.87, -0.91]$) against the plain chat request, as it does on the open models; dropping the command line raises it by 1.84 ([0.95, 2.77]); and the tool-response markers lower it by 4.01 ($[-6.32, -2.04]$), leaving the cells between 3.58 and 11.99 points. The Qwen3.5 templates render the `tool` role as exactly such markers in a user turn, but after the model's own declared tool call, and isolating that call needs a provider accepting a prefilled reply beside a function call, so the step-by-step comparison of Section 3.2 is not repeated here.

Is the drop under retrieval the header string? The retrieved document of Section 3.2 differs from the chat message only in its header line, which drops the lead-in sentence and names a file, so six wrappers run in the factorial's cell (`kubectl`, V13, `settings.py`, probability scale) on the same chat request, the same kept command line and the same observed value, changing nothing but what surrounds it: the lead-in sentence with a fence, the fence alone, blockquote lines, `<tool_response>` markers, the retrieved-document header, and a metadata line giving the tool, the command and a capture time. The three wrappers that only dress the text sit together at 10.10, 8.71 and 9.52 points; paired by service against the lead-in, the fence alone lowers the effect by 1.39 points ($[-2.54, -0.18]$) and the blockquote by 0.58 ($[-2.52, 1.05]$). The three that say where the text came from all lower it: the markers by 4.36 ($[-7.67, -1.56]$) to 5.74 points, the retrieved header by 5.58 ($[-8.65, -2.95]$) to 4.52, and the provenance line by 6.55 ($[-9.99, -3.69]$) to 3.55. Most of the drop turns, then, on whether the wrapper names a source, not on one string. The runner, the capability probe's fourteen responses, the verification files and every record are released with the rest of the artefacts.

O Inference-time baselines

Six methods run on the frozen Qwen2.5-1.5B-Instruct over the prompts of Appendix B and on the frozen Olmo-3-7B-Instruct over its own repair's prompts, with the same scoring code. The instruction prepends the line `# When a value appears in the output below, use exactly that value.` to every prompt. Context-aware decoding (Shi et al., 2024) takes at each step the argmax of $(1 + \alpha)$ times the logits given the full prompt minus α times those given the action line alone, appending the token to both. Constrained copying scores every two- to five-digit

integer in the observation as a continuation, adds the log-probability that the next token is not a digit, and writes the most probable, or decodes greedily with no candidate. Anonymisation replaces every surface form of the target's name with one made-up name per prompt, by the repair's substitution rule, then decodes greedily. The two conditional methods act only where a rule finds the target on the page: a surface form of its name on a line that also holds a two- to five-digit whole number. It fires on all 400 binding prompts per fold, none of the preservation prompts and those 30 non-port prompts (Appendix B). Conditional anonymisation anonymises such a prompt; the conditional rewrite turns its service-and-port rows into `<name>_port=<port>` lines under `$ cat .env`, the authored layout of the preregistered contrast, where the effect stays below 1 point. Prompts the rule leaves alone keep the greedy output, which reproduces the frozen model's evaluation file output for output in both folds, as context-aware decoding at $\alpha = 0$ does on 64 test prompts.

Only conditional anonymisation reaches zero while leaving every preservation answer unchanged. The instruction is the cheapest unconditional method: no training, no extra forward pass. It writes the default on 2 of 316 conflict prompts in fold A and 8 in fold B, and on raw text recall stays within about 3 points of the frozen model, though it costs 3.6 to 8.2 points in chat and tool delivery (Figure 5). It also fails the no-change test on the prompts with nothing to read, on more services than the adapters: the E0 probe scores one prompt per service against a 3-point threshold and counts every changed answer as a failure, and the instruction changes 8 and 5 of the 21 services against the adapters' one or two. Those changes gain 4 correct answers and lose 1 in fold A, and gain 1 and lose 1 in fold B. Constrained copying also copies the non-port number, on the same 30. Anonymisation reaches zero by taking away the name the default is recalled from. NCSD trained without the anchor, fold A only, learns the same shortcut, which the anchor prevents. On the 7B model's larger effect the instruction removes 5 percent, against 87 and 41 on the 1.5B folds, and context-aware decoding 34 percent at $\alpha = 0.5$ and 56 at $\alpha = 1$, against 81 and 54 on the 1.5B at either weight. The conditional rewrite removes 65 percent, against 63 and 58. Constrained copying, anonymisation and conditional anonymisation remove all of it, the first two at a cost of 33 and 57 points of recall and conditional anonymisation at none.

Context-aware decoding needs two forward passes per generated token, constrained copying about 2.2 scoring passes per prompt, every other method one. Supplement S17 gives every method on both models.

P The code domain

The code measurements follow Yang (2026), who binds a conventional alias to another library, scores the pull at the use site on six library pairs in templated prompts and on twelve in forced-choice and naturalistic arms, and tests prompt-level fixes. Here all twelve pairs, over four bound libraries, run as greedy generation in the six models of the diagnosis, and this appendix adds a two-line binding and a trained repair. Each pair is probed at the two use sites of that work: a line that begins with the alias (`np.`) and an assignment (`x = np.`). All of it runs in the chat condition, with the import in a user message and the use site in the reply. Three arms bind a conventional alias to another library (`import pandas as np`), replace it with one of three length-matched made-up tokens (`import pandas as vo`), or change it to the bound library's own convention (`import pandas as pd`); each continuation is classified by checking the attribute it names against the installed modules.

A rebound alias pulls the continuation toward the library the alias conventionally names. Among continuations naming an attribute that belongs to only one of the two libraries, the share naming the conventional library is 49.48 points higher under a rebound than under a made-up alias, CI $[27.30, 69.93]$, averaged over the twelve pairs with intervals that resample bound libraries and then pairs within them (10,000 replicates).

Counted over all continuations, rebinding moves about 27 points from the bound library's attributes to the conventional library's: the bound library's share falls from 54.4 percent under a made-up alias to 27.1 under a rebound one, while the excluded classes barely move: attributes of both libraries are 2 to 4 percent of continuations in every arm, of neither library 19 to 20 and continuations naming no attribute 23 to 25 percent, so the conditional statistic is not an artefact of naming nothing.

On rebound imports, relabelling the alias to the bound library's own convention buys nothing over anonymising it, and K4 does not pass. With the error taken as the share of all continuations that do not name an attribute unique to the bound library, rebinding raises it by 27.08 points relative to the relabelled alias, CI $[11.54, 50.83]$, while the made-up minus relabelled difference is -0.23 points, CI $[-9.26, 8.33]$, Holm $p = 0.86$. By bound library the rebound-minus-relabelled difference is 29.2 points for numpy, 15.0 for pandas, 62.5 for sklearn and 8.3 for xgboost, which has one pair. On conventional imports anonymising has a cost that relabelling avoids: with an import such as `import pandas as pd` the models name an attribute of the imported library on 70.1 percent of continuations and under a made-up alias on 61.8. Scored with attributes of both libraries counted correct, since here the two libraries coincide, the preregistered anonymisation cost is 8.33 points, positive in all six models and above the 2-point threshold at which the preregistration reports it as a cost.

The repair on code. NCSD was trained on the alias task with the objective and anchor of Section 4.1, in two folds split by bound library, with the alias bound in two lines (`import pandas` then `np = pandas`); fold A trains on the pairs whose bound library is pandas or xgboost. The designed SFT arm, with bound-library attributes as targets, was not run. The one-line form used everywhere else in this appendix was the designed evaluation, and there the conventional library's share barely moves: by less than 4 points in fold A and not at all in fold B (51.3 against 51.4). Scoring the two-line form it trained on was added after that result, so what follows is exploratory. There the share falls by about two thirds in fold A, from 3 of 5 unambiguous continuations to 1 of 5 on held-out bound libraries (numpy and sklearn) and from 3 of 6 to 1 of 6 on training ones, and not by naming fewer attributes: on the held-out libraries the continuations naming an attribute only the bound library has go from 2 to 4 of 12. In fold B it falls from 3 of 6 to 2 of 7 (50.0 to 28.6 percent). Each two-line cell is 12 distinct conflict prompts decoded greedily with one seed. Every interval, resampling pairs, includes zero, the frozen model's among them, and the two-line cell on training libraries is in-sample. Supplement S18 gives the class table, the adapter on both import forms and library groups, the conditional statistic's denominator and the effect in Qwen2.5-1.5B-Instruct.

Q Both requests at once

A repair or an inference-time method can stop overwriting the observed value by always answering from the page, even when the request asks for something the page does not show. The binding prompts alone cannot tell those two behaviours apart. Table 6 scores the repair arms and every inference-time method on both, with the same raw-text prompts and the same code: the conflict prompts, where the answer should be the port the page shows, and the asked-for-default prompts, where the page shows the same service on a non-default port and the question asks for its documented default. The supervised arm, designed after this comparison, is scored in Appendix R.

The conditional rewrite costs accuracy on the first request. It writes the observed value on 90.2 and 86.9 percent of conflict prompts, averaged over services, against the frozen model's 95.4 and 94.4. That is 5.17 points lower in fold A ($[-11.37, +0.80]$) and 7.56 in fold B ($[-13.29, -1.96]$), paired by service. The lost prompts do not become default answers, which fall to 1.7 and 2.0. They become key names: on 7.4 and 9.7 percent the model writes `redis_port`, the key the rewrite put on the page, instead of the number beside it. Every other method writes the observed value at least as often as the frozen model, and every adapter and constrained copying do so on every prompt.

The last column reverses the order. Constrained copying and the two anonymisations never answer with the documented default. NCSD answers with it on 19.6 and 21.2 percent and SFT on 30.0 and 27.1, against the frozen model's 47.1 and 42.1. Only the conditional rewrite is above the frozen model, at 62.5 and 63.8. No method here is better on both requests. The methods that answer from the page on every prompt give up between a quarter and all of the second request; the one that gains there gives up part of the first. They also differ in when they act: the rewrite and conditional anonymisation change a prompt only when a rule finds the target's row on it, and the others act on every prompt.

An adaptive rule buys back more of the second request without escaping the trade-off. Context-aware decoding uses one strength for every step. Setting the weight at each step to the Jensen-Shannon divergence between the two streams, the form Wang et al. (2025b) use, reimplemented inside the decoding loop of Appendix O rather than taken from their code, removes 56 and 32 percent of the

Table 6: The repair arms and every inference-time method on both requests, Qwen2.5-1.5B-Instruct, fold A / fold B (rows with one value: fold A only), raw-text prompts, on the services no adapter trained on. The first four columns classify what was written on the 316 conflict prompts: each service's shares over its own prompts, averaged over the 24 services. Observed value is the value printed on the page, the answer the request asks for. Default is the service's documented default, the error this paper measures; the two CAD weights write it on the same 2 and 7 prompts, so those cells coincide. Key name is an identifier written in place of the value: one ending in `_port`, such as the key `redis_port` the rewrite puts on the page, or the service's own name. Other is anything else, another number or nothing parseable. The intervals quoted in the text resample services. Asked-for-default is the share of the asked-for-default prompts answered with the documented default, the correct answer there; higher is better, up to what the model recalls with nothing on the page competing (79.7 and 81.2 percent for the frozen model).

	What the conflict prompts were answered with (%)				Asked-for-
Method	Observed value	Default	Key name	Other	default (%)
frozen model	95.4 / 94.4	4.6 / 4.8	0.0 / 0.0	0.0 / 0.8	47.1 / 42.1
NCSD	100.0 / 100.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	19.6 / 21.2
SFT	100.0 / 100.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	30.0 / 27.1
NCSD, fifth anchor family	100.0	0.0	0.0	0.0	35.8
NCSD, no anchor	100.0	0.0	0.0	0.0	3.8
instruction	99.4 / 96.4	0.6 / 2.8	0.0 / 0.0	0.0 / 0.8	43.3 / 40.0
CAD, $\alpha = 0.5$	98.8 / 97.8	0.9 / 2.2	0.0 / 0.0	0.3 / 0.0	26.2 / 22.1
CAD, $\alpha = 1$	95.4 / 96.7	0.9 / 2.2	2.3 / 0.8	1.4 / 0.3	18.3 / 11.7
constrained copying	100.0 / 100.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0
anonymisation	100.0 / 99.7	0.0 / 0.0	0.0 / 0.0	0.0 / 0.3	0.0 / 0.0
conditional anonymisation	100.0 / 99.7	0.0 / 0.0	0.0 / 0.0	0.0 / 0.3	0.0 / 0.0
conditional rewrite	90.2 / 86.9	1.7 / 2.0	7.4 / 9.7	0.7 / 1.4	62.5 / 63.8
adaptive CAD (JS weight)	97.7 / 95.2	2.0 / 3.3	0.0 / 0.0	0.3 / 1.5	38.8 / 35.0

effect, leaving 2.05 and 3.30 points against the frozen model's 4.63 and 4.82, and answers with the documented default on 38.75 and 35.00 percent of the asked-for-default prompts. Fixed strengths answer with it on 26.2 and 22.1 at $\alpha = 0.5$ and 18.3 and 11.7 at $\alpha = 1$, and the frozen model on 47.1 and 42.1. Deciding per token how much the page should count therefore lands between the fixed strengths and no contrast at all on both requests: it leaves more of the effect than either fixed strength, answers with the documented default more often than either, and stays below the frozen model on the second.

An instruction separates the two requests only after the model has stopped overwriting. Three arms run on the same prompts: no instruction line, the copy instruction of Appendix O, and an instruction naming both requests, the two lines `# When the line you are writing records the current setup, use exactly the value that appears in the output below.` and `# When the request asks which value the service normally uses, answer with its documented default instead.` On the frozen model the copy instruction raises the observed-value rate by 4.04 points in fold A ([1.41, 7.65]) and 1.99 in fold B ([0.68, 3.51]). It moves the documented-default rate by -3.75 ([-9.58, 2.08]) and -2.08 ([-9.58, 5.42]). The instruction naming both requests reverses that: it raises the documented-default rate by 5.42 ([1.67, 9.58]) and 7.92 ([1.67, 14.17]), and it no longer raises the observed-value rate, which changes by -0.41 ([-2.13, 1.23]) and -1.29 ([-3.27, 0.00]). On the adapters every conflict prompt is answered from the page under all three arms, so the observed-value change is 0.00 exactly, and the same instruction raises the documented-default rate by 6.67 ([2.92, 10.83]) and 3.75 ([0.00, 7.50]) for NCSD and by 9.58 ([5.00, 14.17]) for SFT in fold A. In fold A part of that is not specific to naming the second request, and in fold B none of it is: the copy instruction alone raises NCSD's documented-default rate by 2.92 ([0.83, 5.00]) in fold A and by the same 3.75 ([0.42, 7.08]) in fold B, and SFT's by 2.50 ([-2.08, 6.67]) in fold A. All differences are paired by service and resampled over services.

R SUPERVISING THE SECOND REQUEST

The four anchor families are divergences toward θ_0, so an anchor can hold the frozen model's answer on a family but cannot improve on it. On the asked-for-default prompts the frozen model answers with the documented default on 32.1 percent under tool delivery, and the fifth anchor family reaches 20.8: the anchor recovers part of the loss but cannot pass the frozen model (Qwen2.5-1.5B, fold A). The arm here replaces that anchor with supervision. Prompts of the family have a known documented default, since the generator records it when planting a different value, so the label needs no new annotation; the term is the negative log-likelihood of that default, at weight 1. As in the arm with the fifth anchor family, the family takes 15 percent of the anchor prompts from the no-observation family, whose share falls from 35 to 20 percent; the other three distillation anchors and the public-code term are unchanged. Its two action templates, `{N}_STOCK_PORT =` and `# by default {n} listens on port` , reword the evaluation's constant `{N}_DEFAULT_PORT =` and comment `# the documented default port of {n} is` (Appendix K.1), and it trains on the fold's training services in the nine training layouts, so the evaluation's services and layouts are held out and its templates reworded.

In this training mix, what removes the trade-off is supervising the second request, not only including it. In the NCSD and SFT arms every page appears with exactly one request, so "answer from the page" fits the training set exactly and nothing in the objective requires the model to read the request. The fifth anchor family puts the second request on training pages but holds the frozen model's answer there, and it stays below the frozen model (35.8 against 47.1 percent on raw text in fold A). Supervising those pages with the documented default gives the same kind of page two correct answers, one per request, which makes that rule unfit. This is the logic of counterfactually augmented data, which pairs minimally different inputs with different labels to remove a shortcut (Kaushik et al., 2020; Geirhos et al., 2020). Minder et al. (2025) build a training set on the same principle. Their two instructions ask one question of a false context; here the page is true and the two requests ask for different values. Only the second request carries a label, and the first keeps NCSD's target. DisentQA writes both answers in one output, and they separate only once empty and unrelated passages join training (Neeman et al., 2023). Irpan et al. (2025) point to supervised data on look-alike prompts as the standard remedy for the benign refusals that bias-augmented consistency training adds; the supervised arm tests that remedy on the second request. Both arms were designed after the trade-off had been measured, so both are exploratory and no preregistered test covers them (Appendix A).

Table 9 gives these arms on one battery. Supervising the family lifts the asked-for-default rate above the frozen model in all nine cells, and the three supervised runs, two folds and a second seed in fold A, agree to within 0.9 points on raw text. SFT with the same supervised term reaches 78.8, 60.0 and 60.4 percent in fold A, so the result does not rest on NCSD's target. On raw text Δ_{def} is 0.00 and every conflict prompt is answered from the page in all three. Row-absent recall stays within the range the other arms occupy, and prose recall is below the frozen model's by 3 of 133 prompts (2.3 points) in each fold. The non-port probe stays at zero on raw text and in chat; under tool delivery the arm copies the planted number on 3 or 4 of its 48 prompts, against 2 and 1 for the frozen model and 5 and 4 for NCSD in folds A and B. With no observation the arm changes the answer on one to three of 21 services against NCSD's two, so the preregistered criterion still fails, and every changed service is one the frozen model already answered wrongly.

Two things about the frozen model set what this arm can reach. The first is that most of its errors here are not failures of memory. In fold A, on raw prompts that ask for the documented default and show a page, Qwen2.5-1.5B writes the value on the page on 50.4 percent and something else on 2.5. Of those 121 page answers, 86 are on services it recalls on the prose family at least half the time. Most of what the training adds is therefore not knowledge but the habit of reading the request. The second is the prose family, which names the default with no number to compete with it: the frozen model recalls it on 79.7 percent of raw prompts and the supervised arm on 77.4, about the 76.7 the arm reaches on the asked-for-default prompts.

That match averages the two action templates: in fold A the arm answers with the documented default on 88.3 percent of the comment prompts and 65.0 of the constant prompts on raw text. On the comment it does so on 79.2 to 90.8 percent in every delivery and run, reaching the frozen model's prose recall everywhere: paired by service, the difference is $+5.2$ to $+11.4$ points on raw text, above zero in one of the three runs, and -7.4 to $+6.2$ in chat and tool, where every interval covers zero.

On the constant it reaches 65.0 to 69.2 percent on raw text and 25.0 to 52.5 in chat and tool, and 95 of the 96 raw-text answers that copy the page, over the three runs, are on the constant. The shortfall under chat and tool, 15 to 37 points below the arm's own prose recall, is mostly the constant's. The frozen model shows the same split, 71.7 against 22.5 percent on raw text in fold A.

Both evaluation templates contain the word "default", so four more request forms on the same fold-A pages test whether the arm reads the request or the word. Two ask for the documented default without the word, `{N}_STANDARD_PORT =` and `# out of the box {n} listens on port` ; two use the word but ask for the value on the page, `{N}_NON_DEFAULT_PORT =` and `# {n} is not on its default port here; it listens on port` . On the first two the supervised arm writes the documented default far more often than the frozen model, by 54.2 and 35.8 points on raw text and by 22.5 to 40.8 in chat and tool, every interval above zero when paired by service; NCSD does so on 0 to 30 percent. On the two that ask for the page it follows the word part of the time on raw text, writing the default on 19.2 and 17.5 percent, 15.8 and 15.0 points above the frozen model ([7.5, 25.8] and [7.5, 24.2]); under tool delivery it matches the frozen model, and in chat it matches it on the constant and is 7.5 points above on the comment. The arm learns the request, with a partial pull toward the word on raw text.

Every arm so far trains on raw text, so under chat and tool delivery the supervised arm meets the second request in deliveries it never saw in training. Rotating training prompts through raw, chat and tool rendering tests this (Table 7); the two partial arms run in fold A with one seed. Rendering only the supervised asked-for-default prompts lifts the rate to 90.4 percent in chat and 86.7 under tools, but in chat the name effect rises from the frozen model's 5.83 points to 21.50 (CI [13.37, 30.50]): in that delivery the arm has learned to write the default, and it does so on 80.8 and 73.3 percent of chat requests that name the default but ask for the page. Rendering the binding prompts as well takes the effect to 0.00 in every delivery, but the anchors, still raw, stop holding in chat and tool delivery: row-absent prompts are answered with another listed service's port on 38.8 and 43.1 percent against the frozen model's 19.9 and 26.0, and the planted non-port number is copied on 18.75 percent of tool prompts against 4.17. Each partial arm moves the trade-off to the pairing of request and delivery that training left out. Rendering the anchors too removes both costs, and the result holds with a second seed and in fold B. Across the three runs the arm answers 75.8 to 82.9 percent of asked-for-default prompts in every delivery, against 31.2 to 47.1 for the frozen model, and leaves an effect of at most 0.25 points, one prompt. Its memory holds: prose recall falls by at most 3.8 points, row-absent misbinding rises by at most 3.2, and the no-observation answer changes on one or two of 21 services, each one the frozen model answered wrongly. The planted non-port number is copied as often as by the frozen model, except in fold B under tools, on 2 of 48 prompts against 1. The remaining cost is the word itself: on the two request forms that name the default but ask for the page, the arm writes the default on 5.0 to 60.0 percent, against 0.0 to 20.8 for the frozen model.

Answering from the page is not confined to the smaller model. The 7B repair ran the same prompts on Olmo-3-7B-Instruct (Table 8), a different family, so this one pair does not separate scale from family. The frozen 7B reads the request better: it leaves 11.0 points between what it recalls and what it says on raw text, against the 1.5B's 32.6, while recalling slightly less, 75.2 percent against 79.7. NCSD takes 20.8 points from the 7B on raw text and 39.2 under tool delivery, against 27.5 and 23.3 from the 1.5B: the larger model gives up less on raw text and in chat and more under tools.

The rotated supervised arm, run once on the 7B model in fold A, reduces the trade-off there without removing it. It answers 72.5, 63.8 and 67.1 percent of asked-for-default prompts in raw, chat and tool delivery, against the frozen model's 64.2, 46.2 and 56.7, and narrows the gap in every delivery (Table 8). It leaves Δ_{def} at 4.41 on raw text (CI [1.81, 7.47]), 0.28 in chat and 3.82 under tools ([1.17, 6.92]), where NCSD leaves 1.68, 0.00 and 1.05. Prose recall falls by 5.3, 12.0 and 4.5 points, row-absent misbinding rises by at most 3.2, and the planted non-port number is copied on 5, 6 and 12 of 48 prompts against the frozen model's 3, 4 and 8. With no observation the arm changes the answer on 8 of 21 services, each one the frozen model answered wrongly. On the two request forms that name the default but ask for the page, it writes the default on 13.3 to 48.3 percent, against 0.0 to 23.3 for the frozen model.

Table 7: The supervised arm trained in the deliveries it is scored in, Qwen2.5-1.5B-Instruct, the 24 services held out of each fold's training, one seed per row, exploratory. Asked is the share of asked-for-default prompts answered with the documented default (%); Δ_{def} is the name effect on the conflict prompts (pp). The indented rows rotate training prompts through raw, chat and tool rendering: first the supervised asked-for-default prompts only, then the binding prompts too, then the anchor prompts too; the last two rows repeat that arm with a second seed and in fold B. Misbound is the share of row-absent prompts answered with another listed service's port under chat and tool delivery (%). The frozen model's prose recall is 79.7, 88.7 and 88.0 percent in the three deliveries in fold A and 81.2, 87.2 and 91.7 in fold B.

		raw		chat		tool		Misbound	
Arm	Fold	Asked	Δ_{def}	Asked	Δ_{def}	Asked	Δ_{def}	chat	tool
Frozen	A	47.1	4.63	40.0	5.83	32.1	1.47	19.9	26.0
Frozen	B	42.1	4.82	41.7	8.69	31.2	2.58	18.9	27.4
NCSD	A	19.6	0.00	19.6	0.25	8.8	0.00	23.5	33.5
SFT, second request supervised	A	78.8	0.00	60.0	0.25	60.4	0.00	21.4	31.0
NCSD, second request supervised	A	76.7	0.00	58.8	0.25	52.5	0.00	20.6	30.2
its asked prompts rendered	A	81.2	0.00	90.4	21.50	86.7	1.14	3.9	3.9
its binding prompts rendered too	A	82.5	0.00	75.0	0.00	76.7	0.00	38.8	43.1
its anchor prompts rendered too	A	82.9	0.00	77.5	0.25	75.8	0.00	21.0	29.2
the same, seed 1	A	79.2	0.00	80.4	0.25	75.8	0.25	18.1	28.5
the same	B	77.5	0.00	82.9	0.00	78.8	0.00	21.7	28.5

Table 8: What each model recalls and what it says when the request asks for the documented default, fold A, held-out services. Says is the share answered with the documented default; Gap is the prose family's recall minus that share, the part the model has and does not use. The frozen 7B leaves a smaller gap than the frozen 1.5B in every delivery, and the repair widens the gap on both. Sup., rotated is NCSD with the second request supervised and its training prompts rotated through the three deliveries (exploratory, one run on each model); it narrows the gap on both, and a negative gap means the model says the default more often than it recalls it in prose.

		raw		chat		tool	
Model	Arm	Says	Gap	Says	Gap	Says	Gap
Qwen2.5-1.5B	frozen	47.1	32.6	40.0	48.7	32.1	55.9
Qwen2.5-1.5B	NCSD	19.6	58.6	19.6	65.4	8.8	79.2
Qwen2.5-1.5B	sup., rotated	82.9	−5.5	77.5	11.2	75.8	15.2
Olmo-3-7B	frozen	64.2	11.0	46.2	30.4	56.7	25.3
Olmo-3-7B	NCSD	43.3	29.6	31.7	41.3	17.5	63.0
Olmo-3-7B	sup., rotated	72.5	−2.6	63.8	0.9	67.1	10.4

S The rest of the repair literature

Inference-time work has moved from a fixed contrast toward deciding when the context should win. A two-stage decoder predicts a conflict from the attention map and routes the step accordingly (Zhou et al., 2026), another generalises contrastive decoding by allocating authority between the prior and the context from conflict signals (Jiang et al., 2026), and activation steering reaches the same end with no weight update at all (Anand et al., 2026). The competition itself has also been studied as a conflict task borrowed from psychology, between a learned mapping and an in-context rule (Hu et al., 2026b).

Tool-use reliability is charted as taxonomies and benchmarks. ToolFailBench labels traces for skipping a tool, ignoring its result and fabricating output (Soni, 2026). ToolCritic detects eight tool-calling error types, one of them misinterpreting an output (Hamad et al., 2025). A survey collects such modes (Hu et al., 2026a), and recognising that a tool is needed diverges from calling it (Cheng et al., 2026). What a tool returns changes behaviour even when the call is right: a correct code-interpreter result substitutes for reasoning (Fatahi Bayat et al., 2026), the kind of tool shifts how well

Table 9: The frozen model, NCSD and the two arms that add the second request, on one battery, fold A or B of Qwen2.5-1.5B-Instruct, scored on the 24 services held out of its training. The first three columns are the share of asked-for-default prompts answered with the documented default, in the three deliveries. Δ_{def} and Copied are the name effect and the copy rate on the conflict prompts in raw text, each averaged per service as in Table 6. Recall is the remembered default on the row-absent and prose families in raw text, pooled over each family's prompts. E0 counts the held-out services whose no-observation answer in raw text differs from the frozen model's, which the preregistered criterion allows none of.

		Asked-for-default (%)			Δ_{def}	Copied	Recall (%)		
Arm	Fold	raw	chat	tool	(pp)	(%)	absent	prose	E0
Frozen	A	47.1	40.0	32.1	4.63	95.4	65.8	79.7	0 of 21
Frozen	B	42.1	41.7	31.2	4.82	94.4	56.9	81.2	0 of 21
NCSD	A	19.6	19.6	8.8	0.00	100.0	66.2	78.2	2 of 21
NCSD	B	21.2	17.5	5.8	0.00	100.0	61.9	81.2	2 of 21
NCSD, fifth anchor family	A	35.8	29.6	20.8	0.00	100.0	68.0	78.2	1 of 21
NCSD, second request supervised	A	76.7	58.8	52.5	0.00	100.0	67.3	77.4	2 of 21
the same, seed 1	A	76.2	60.8	54.6	0.00	100.0	64.8	77.4	1 of 21
NCSD, second request supervised	B	75.8	71.7	63.8	0.00	100.0	61.9	79.0	3 of 21
SFT, second request supervised	A	78.8	60.0	60.4	0.00	100.0	66.9	79.7	1 of 21

stated confidence tracks accuracy (Xuan et al., 2026), and the calling protocol itself costs accuracy in the presence of distractors (Zhang et al., 2026a).

Training-time repairs that supervise the answer include counterfactual targets (Longpre et al., 2021), identifier-robust code training (Henkel et al., 2022), preference pairs (Bi et al., 2025; Zhang et al., 2025), training toward memory when retrieval misleads (Lin et al., 2026) and training toward calibrated trust (Sun et al., 2026). Four methods train both sides of the choice between context and memory. Knowledge-aware fine-tuning labels a context that entails the answer with that answer, and an irrelevant or empty context with the pretrained model's own closed-book answer (Li et al., 2023). DisentQA writes a contextual and a parametric answer in one output and labels the parametric one with the original answer even where a counterfactual passage contradicts it (Neeman et al., 2023). CR-DPO prefers the model's own reasoning paths that trust the context when its closed-book answer is wrong and its own answer when the context is wrong (Huang et al., 2025). Minder et al. (2025) label one question on a false context under an instruction to use the context and under one to ignore it (Appendix R). Closest in structure to our two requests are knowledge-aware fine-tuning's hard irrelevant contexts: SQuAD 2.0 asks, of the same paragraphs, questions they answer and questions they do not, and the latter are trained toward the pretrained model's closed-book answer. Our second request differs in that the page shows a different true value of the very attribute it asks about, its label is the documented default rather than the model's own answer, and each request is made through the line being written rather than through a question or an instruction. Such training can reuse a context-versus-memory direction already present in the model (Gupta & Gupta, 2026). A direction learned on a fine-tuned model also switches the untuned instruct model between context and memory (Minder et al., 2025). Instruction tuning itself first raises and then lowers context reliance under knowledge conflicts, as examples whose context agrees with memory come to dominate the gradient (Goyal et al., 2025). Counterfactual augmentation against that drop helps mainly on conflicts like the augmented ones and lowers SQuAD accuracy on Alpaca from 76 to 62 percent, where the context agrees with memory (Goyal et al., 2025). Instruction tuning also reshapes which properties of a context a model leans on (Wangsajaya et al., 2026).

Consistency training is a close precedent for the binding term. Bias-augmented consistency training fine-tunes a model on prompts that carry a biasing cue toward its own responses sampled without the cue, with no gold labels (Chua et al., 2024). On tasks that require repeating a user's mistake, it scores 45.0 percent, against 52.4 for GPT-3.5-Turbo and 49.5 for the authors' control. Irpan et al. (2025) apply it to sycophancy and jailbreaks and add a variant that matches residual-stream activations. Han et al. (2026) make it on-policy and lose no more capability than supervised consistency training. Gautam et al. (2026) extend it to MLP and attention states; matching MLP states removes persona adoption but degrades alignment on benign persona prompts, and training against one threat can hurt

another that calls for the opposite policy. Africa & Mani (2026) find that on models already fine-tuned to misbehave it suppresses reward hacking and emergent misalignment but raises sycophancy, significantly for the activation variant, because the clean-prompt targets already contain it. Imran et al. (2026) match only the rate of a target behaviour with reinforcement learning, which reduces bias-following on held-out biases while largely preserving how often the model mentions the cue. Our clean prompt replaces the name with three made-up names drawn from a pool of five-letter names instead of deleting a cue. Our target averages the frozen model's next-token distributions over those names on a span of at most eight tokens it sampled; the methods above take their target from a single clean prompt, whether a sampled response (Africa & Mani, 2026), soft token-level labels (Gautam et al., 2026), the student's own samples or internal states. Our anchors hold the frozen model on four prompt families where the name should still recall the default, and on public code, rather than on generic instruction data or the training prompts. With the name gone the frozen model copies the observed value, so the target does not contain the error it removes (Section 4.1). In these papers the cue cut from the teacher's prompt is treated as irrelevant to the answer. Our name is also what the second request needs, which is where the trade-off of Section 4.3 appears.

Two neighbours of our teacher remove knowledge rather than keep it. Anonymising entities in the pretraining corpus grounds answers in the context and pays in closed-book recall (Cohen et al., 2026); our adapter keeps the default and moves only what the name does to a value on the page. In agents, tools hand a forgotten fact back (Chen et al., 2026).

The self-distillation literature differs in the direction of the divergence. Reverse KL keeps a student from covering a teacher's low-probability regions (Gu et al., 2024). Propagating a definition into the parameters minimises a forward KL to the teacher (Padmanabhan et al., 2023). On-policy training narrows what the direction costs (Agarwal et al., 2024), though across distillation tasks the family still matters, with symmetric divergences beating both one-sided ones (Wen et al., 2023). Our binding term uses forward KL, and reverse KL changes little here (Appendix K). Wang et al. (2026) add to on-policy context distillation a forward-KL term that pulls the student's output with the privileged context shown toward its own stop-gradient output without it. In 12 of 14 settings the term cuts how often a re-shown context flips answers the student already gets right. Reversed, in one ablation on Llama-3.1-8B-Instruct and MedMCQA, it degrades accuracy after about 30 training steps. Their target moves with the student; ours is the frozen model under made-up names. Knowledge-aware fine-tuning proposes, without running it, a cross-entropy of the same form for irrelevant contexts (Li et al., 2023). Jukić & Titov (2026) warn that KL matching can buy agreement with a teacher at high-entropy states.

T Two path-valued fields: data directory and configuration path

Every configuration measurement so far uses the port. The same battery also ran on the repair model with two path-valued fields, the data directory and the configuration file, each asked for with a pre-filled action (`redis_dir = "/`) and shown either in a command-output layout — a `docker ps` `MOUNTS` column or a `ps -eo user,pid,args` line — or in an authored `.env` line. The rest of the design is unchanged: the same 48 services, three length-matched made-up names, three value schemes, here path prefixes, and both deliveries, with one action template on the conflict prompts. The slash is prefilled because without it none of the 1,152 probe answers is an absolute path: two thirds are placeholders such as `path/to/your/data`, and most of the rest bare names or file names.

Neither field carries the effect, for two reasons. With nothing to read, Qwen2.5-1.5B-Instruct holds almost no path conventions. Two of its three action templates agree on an answer for 237 of 384 cues, but for 59 of the 64 real-name cues among them that answer is a placeholder built from the name (`/path/to/redis/data`), the form the made-up names get too. Only five services get a real path, all in the configuration field (consul, netdata, prometheus, rabbitmq and squid), and three of those take the form `/etc/squid/squid.conf`, which some made-up names also get. And when a path is on the page it is copied: 3,726 of 3,726 conflict prompts (1,422 in chat, over the 237 cues with a stable answer, and 2,304 under tools, over all 384), including all 186 configuration-field prompts on those five services, so the name effect is 0.00 in every cell with nothing to remove.

Read with the port results, this fits an overwrite that needs a remembered value to compete with the observed one as a completion of the line being written. A port has one; for paths the model mostly

has none, and on the 60 real-name configuration prompts of the five services where it has one it still copies the page. The same model has a name effect on the port of 5.8 and 8.7 points in chat and 1.5 and 2.6 under tool delivery, and writes a remembered path on none of the path prompts.

U Agent episodes without a prefill

Every other measurement reads one completion, and all but Appendix M prefill the action line. Without the prefill, the tool-delivered prompts of Appendix M mostly end in another `run_shell` call, which a single turn cannot score. Here the episode goes on. It starts from the tool delivery of Section 2 (the request, the assistant's `run_shell` call and the `docker ps` or `kubectl` row it returned) with nothing prefilled, after one system line: "You are a coding agent working on a developer's machine. The project is in /workspace. Use the run_shell tool to run shell commands when you need to, and complete the user's task." Each further call runs in real bash under bubblewrap, with no network, the host read-only and a fresh `/workspace` that holds `README.md`, `app/main.py` and `requirements.txt` but not the file to write. The service commands (`docker`, `kubectl`, `curl`, `nc`, `ss`) answer for one service, the observed one, on the observed port only: `docker port` and `kubectl describe` report that port, a connection to it succeeds, a connection to any other port is refused, the service's image has no shell, and nothing prints the documented default. An episode ends when the target file exists, when the model replies without a tool call (the reply is then its file), or after twelve assistant turns; the last three tool results note the turns left. The stimuli are the `docker ps` and `kubectl` rows under the three primary value schemes, both target files and all 48 services under the real name and its three made-up names, each with the preregistered request and the plain one: 4,608 episodes per model, at temperature 0. The retrieved-document condition and the code task are measured in single turns only.

Scoring. The written file's code, without comments and docstrings, is read for the two numbers: the episode sets the default when the documented default appears and the observed port does not. An environment lookup counts by its fallback, because the sandbox sets no variables and the fallback is the value the program runs with. A default that appears as boilerplate for another service, such as a `DB_PORT` of 5432 in a generic file, is what the made-up names measure and subtract. Every episode is in the denominator, so an unfinished episode or a file that sets neither number counts against its arm. Δ_{def} is the real name's share minus its made-up names' share, per service and then averaged over services, with the service-cluster bootstrap. Two readings check the scorer: one counts the default wherever it appears, and one only where the free-form scorer of Appendix M binds it to the service's name.

Models. Qwen3.5-9B runs with thinking on, its default in agent use. With thinking off, Qwen3.5-9B repeated one `docker exec` call and wrote no file in any of 16 trial episodes. DeepSeek V4.1-Flash runs through its API with native function calling and thinking disabled. Olmo-3-7B-Instruct wrote no file in any of 48 trial episodes under three tool set-ups, so it is not run. The service commands and the turn budget were fixed on 24 trial episodes of DeepSeek, before any full run. In an earlier sandbox, process 1 still held the host's environment; 39 DeepSeek episodes and no Qwen3.5-9B episode read it, and the 39 were rerun once bash was process 1 with no host environment.

Results. Table 10 gives both models. Qwen3.5-9B writes the file in 99.7 percent of real-name episodes and sets the documented default in 24.2 percent, against 0.7 percent under made-up names: Δ_{def} is 23.6 points (CI [18.1, 29.5]), 38.2 on the `docker ps` row and 8.9 on `kubectl`, 19.5 with the preregistered request and 27.6 with the plain one. The two scorer checks give 23.2 and 20.6. It reads or tests the port with a tool (`docker port` or `inspect`, `kubectl get -o` or `describe`, `ss`, `netstat`, `curl`, `nc`) in 1.9 percent of real-name episodes and never connects to it. In 2 of the 279 real-name files that set the default, a comment states the observed port: "`# Qdrant settings (since Qdrant is running on port 4167)`" stands above `QDRANT_PORT = 6333`. DeepSeek V4.1-Flash reads or tests the port in 74.5 percent of real-name episodes, connects to it in 17.6, and sets the default in 0.7 percent, against 0.5 under made-up names (Δ_{def} 0.2 points, CI $[-0.8, 1.0]$). It leaves more episodes unfinished at twelve turns, 29.9 percent under the real name and 47.2 under made-up names; every episode counts, so these lower both arms. Its small effect is not ignorance: asked for each service's documented default, it answers all 48 correctly. On the probability scale of Appendix N the real name moves probability toward the

default on every command row, but the default's share of the two candidates exceeds one half on at most 4.2 percent of real-name prompts (`kubectl`, chat), so the name seldom flips the choice. In the 294 real-name episodes that never read or test the port it sets the default in 2.4 percent, as made-up names do in theirs.

Table 10: Agent episodes, 4,608 per model, with the default counted on every episode. Wrote is the share of episodes that write the file and Default the share whose file sets the documented default and not the observed port, under the real name / made-up names (%). Δ_{def} is their difference per service in points, with its 95% service-cluster interval, overall and on each command row. Checked is the share of real-name episodes that read or test the port with a tool (%).

Model	Wrote	Default	Δ_{def} [CI]	`docker ps`	`kubectl`	Checked
Qwen3.5-9B	99.7 / 99.3	24.2 / 0.7	23.6 [18.1, 29.5]	38.2	8.9	1.9
DeepSeek V4.1-Flash	70.1 / 52.8	0.7 / 0.5	0.2 [−0.8, 1.0]	0.3	0.0	74.5

V Limitations

Preregistered and exploratory results. The preregistered confirmatory tests are A1 and A2 in chat, the nine B contrasts, the C family, K1 to K4 in code and P1 to P4 in the repair (Table 1). Tool delivery is a preregistered secondary analysis, as are the split by value scheme, the check on the 28 pilot services and C2. The `DOCKER6-DEF` against `DOCKER6-OFF` and `DOCKER1` against `D-REPEAT` contrasts are preregistered too, and the bridge condition was preregistered but ran only in the later bfloat16 regeneration. Tool delivery ran after the chat analysis, on 11 of the 17 layouts (Appendix A). Every other result is exploratory: it was designed after the diagnosis data were seen, or it ran outside the preregistered design. On the diagnosis side this covers the delivery steps other than the bridge, the retrieved document, the stimulus edits of Section 3.3, the delivery-by-family regression, the regressions on name properties, the prior-strength probe, the sampling scale, the unprefilled and agent runs, the API model and the path-valued fields. On the repair side it covers the run on Olmo-3-7B-Instruct, every inference-time method, the two-line scoring of the code repair, the fifth anchor family and every supervised second-request arm. K2 was missed by the first analysis and computed later from the stored score records. We record twenty-one departures from the two preregistered documents, and Supplement S2 lists each one (Appendix A).

Preregistered tests that fail. A1 passes in chat on all three preregistered metric variants, but only three of the six models meet the per-model criterion of at least 3 points with an interval above zero: Qwen3.5-9B, Olmo-3-7B-Instruct and Olmo-3.1-32B-Instruct (Appendix A). Qwen3.5-27B reaches $+0.35$ points (CI $[-3.12, 5.56]$) and both Gemma 4 models 0.00. The preregistered rule therefore confines the confirmatory claim to the passing families, which we narrow to the three passing models. On the `docker ps` row both Gemma 4 models score zero in chat, and their 4.2 and 12.5 points under tools come from the secondary analysis (Section 3.1). Five of the nine B contrasts do not survive Holm correction: B3, B4, B6, B8 and B9. B5 survives against its predicted sign, with a `host:port` line 1.33 points below a bare port. B2, `ss` above key=value, keeps its size on the same-form subset ($+1.5$) but not its Holm significance ($p = 0.052$), so it is not separated from the name's surface form. B8 was computed on its `D-NOHASH` arm alone (Appendix E). The C family fails: prior strength does not rank the 20 rule-selected services by risk in chat (Kendall $\bar{\tau} = 0.02$, CI $[-0.03, 0.12]$), so every statement about prior strength is exploratory. K4 fails: relabelling a rebound alias to the bound library's own convention does no better than anonymising it (Holm $p = 0.86$; Appendix P). P2 fails for NCSD and SFT in both folds of the 1.5B model, and the 7B adapters fail the same probe (Appendix K). P2 compares one answer per service against a 3-point threshold, where the design used a non-inferiority bound, and P3 uses one kind of non-port number where the design pooled three. P1, P3 and P4 pass on one training seed per arm and fold, and P4 passes with both targets at 0.00 (Table 1).

The six diagnosed models. The pooled diagnosis numbers average six instruction-tuned models and are not estimates for model families at large. Models spread about as widely as the delivery shifts them (Appendix H). Averaged over the two deliveries, the command-output effect runs from 1.1

points in gemma-4-12B-it to 10.8 in Olmo-3-7B-Instruct. Under tools alone it runs from 2.0 points in Qwen3.5-27B to 16.7 in Olmo-3-7B-Instruct. Entered alone, model explains slightly more of the variance in service-level Δ_{def} than delivery does ($R^2 = 0.031$ against 0.023). The threefold rise under tool delivery is pooled over the six models; five of them show a rise (Section 3.1). Olmo-3-7B-Instruct is the one model whose authored files average above 1 point, at 3.1 under tools (Appendix H). The effect is concentrated in a subset of services: under tools 41 of the 48 have a positive effect on command output, and the ten largest carry about half of it (Section 3.4). The final raw-text wave, with further base and instruction-tuned models and its scale slopes, was not run (Appendix A).

Intervals. Every configuration interval in the main text resamples services with the models held fixed, as the preregistration specifies; the code interval resamples bound libraries and then pairs. Resampling models as well widens seven of the twenty-two delivery and edit contrasts through zero (Appendix I). The seven are the tool-exchange step on both command rows, the move into the tool result on `kubectl`, the request swap on `docker ps`, two `.env` steps and `DOCKER1−NA` minus `KV−NA`, the last being the 3.2 points that the name in the observation alone leaves on the `docker ps` row (Section 3.3). Two delivery steps survive resampling models, each on one row: the move into the tool result on `docker ps` ($[2.1, 12.3]$) and the request swap on `kubectl` ($[6.8, 22.2]$). Four more contrasts include zero under both schemes, all on `.env` or JSON. The rise of the six-row listings over the single row also widens through zero, to $[-2.0, 9.0]$ and $[-2.8, 11.6]$. Resampling models puts the prior-strength slope of every single-row command layout across zero in both deliveries (Appendix E). In the repair, the bound tabulated for an arm that writes no default is the rule of three, $3/316$, which ignores clustering by service (Appendix K). A preservation accuracy on its own carries a Wilson interval, which also ignores clustering (Appendix B).

Services, field and layouts. The configuration results cover one field, a port, on 48 services chosen for well-known defaults (Section 2). Twenty of them were added under a rule that requires one documented default port shared with no other service in the set. Beyond configuration, the diagnosis covers only import aliases in code. The two path-valued fields, a data directory and a configuration path, ran only on Qwen2.5-1.5B-Instruct (Appendix T). That model holds almost no path conventions: with nothing to read, only five services get a real path, all in the configuration field. The null on those fields therefore does not test a model that holds path conventions, and no such model was run. `DOCKER1`, the row of the preregistered contrast, prints a host address without an arrow, a form real `docker ps` does not produce (Appendix C). That row carries the 4.1-to-14.3-point `docker ps` rise, while the realistic `D−MAP` and `KUBECTL` rows rise from 1.9 to 8.1 and from 6.2 to 13.5 (Section 3.1). `D−MAP−DEF` prints the default as the container port, and its 17.52 points stay out of every contrast because the made-up names cannot separate an overwrite there from a name-driven choice of that port (Appendix D). Under tool delivery the authored family has four layouts (Table 2). Every command layout exceeds every authored layout in both deliveries, but in chat the intervals of the closest pair overlap (Appendix D).

Greedy decoding and the prefilled action. Every scored output is decoded greedily (Appendix G). Outside Appendices M and U the model completes a prefilled action line with reasoning modes off, so those numbers describe one completed line, not a model that also chooses whether to call a tool and what else to write. Read as the probability that a temperature-1 sampler writes the default, the command-output effect is 4.3 points in chat and 10.6 under tools, against 3.2 and 9.7 greedy. In 10 of the 48 model-by-family cells the sampling effect is the smaller one, nine of them under tools and five in Olmo-3-7B-Instruct. The renormalised two-candidate share ignores every other continuation and overstates the absolute rate as temperature grows, so it is read only for direction. The greedy and nats scales agree in sign on 49 of the 84 preregistered chat contrasts and on 38 of 60 under tools (C2; Appendix A). The unprefilled measurement covers one model, Qwen3.5-9B (Appendix M). Its scorer agrees with 76 of 80 hand-read answers. Made-up names leave the target unbound more often than the real name, on 46 to 57 percent of `docker ps` prompts against 12 to 22. Both readings count an environment lookup whose fallback is the default as writing the default, and with thinking off such lookups are 20 of the 29 real-name defaults on the `docker ps` row. Counting literal ports alone, the four unprefilled effects are 6.25, 2.78, 6.94 and 6.25 points, against 13.19 and 9.72 with the action prefilled, and no paired difference between the two conditions excludes zero. Under tool delivery with the plain request, the unprefilled model answers with another `run_shell` call on 89 to 100 percent of prompts in seven of the eight cells, which a single turn cannot score. In chat on

a `.env` file the thinking block exhausts its 3,072-token budget on 76 percent of real-name and 84 percent of made-up-name prompts, which leaves that cell on 21 services.

What the made-up names control. A made-up name differs from the real one in tokenisation and familiarity as well as in identity, so Δ_{def} measures everything the real name carries, not its identity alone (Appendix E). The made-up names match the real ones in characters, not in tokens. On the `docker ps` row the default rate does not track the gap in token count between the real name and its made-up names (Kendall $\tau = 0.009$). The check restricted to made-up names that leave the prompt's token count unchanged uses 22.7 percent of the made-up-name prompts and keeps 12 to 14 services, so its intervals are wide. Familiarity predicts the `docker ps` rate only weakly. It is the one name property whose coefficient on the `DOCKER1` rate excludes zero, $+6.57$ points per standard deviation (CI $[0.39, 14.38]$), and pooled over the seven command layouts its interval includes zero, as do those of prior strength and token count. The data do not rank familiarity against prior strength. Made-up names copy command output almost without exception, but in authored files they copy the observed value on 92.96 percent of prompts in chat and 98.48 under tools, close to the real name's 91.90 and 97.67. That shortfall is unrelated to the default. Δ_{def} counts only the scored default and leaves out up to 1.7 points of name-attributable error on command output under tools. Δ_{def} is also a difference between two rates, so a model that ignored names would score zero on it while writing nothing right; every repair arm is therefore also scored on what it wrote (Section 4.3). The prior-strength probe attributes 1.13 points, about a quarter of the chat-mode gap between the name-first command listings and the authored layouts, to a stronger expectation of the default in command rows (Appendix F). Qwen3.5-27B supplies four-fifths of that average, and without it the average is 0.29 points. The probe compares two magnitudes and puts an upper bound on neither. No experiment examines the models' internals: the variants of Sections 3.2 and 3.3 locate the effect in the prompt, not in the model, and the designed position-masked mechanism check was not run.

Delivery variants. The five delivery conditions, the stimulus edits and the instruction line ran in a later bfloat16 regeneration on vLLM 0.29.0, whose seven cells shared with the original diagnosis agree within 0.2 points (Appendix I). Moving the output into the tool result changes at least three things at once: the message role, the command line, which it drops, and the repeated request, which it also drops (Section 3.2). In the Qwen3.5 and Gemma 4 templates the move also changes the template's own markup: Qwen3.5 adds an empty reasoning block to the tool-call turn, and Gemma 4 continues the answer inside the tool-calling turn (Appendix I). The rise under tool delivery therefore belongs to the whole delivery, not to the message role. On the API model's probability scale, dropping the command line alone raises the effect by 1.84 points on the `kubectl` layout (Appendix N). The retrieved-document result rests on one header string, on the `docker ps` and `kubectl` rows, in single turns (Section 3.2). Its comparison with other wrappers ran only on the API model, on one layout, one value scheme and one action line, on the probability scale (Appendix N). The stimulus edits of Section 3.3 ran under tool delivery, and Olmo-3-7B-Instruct carries most of the port-first rise there, 18.8 points against 1.2 on average in the other five models (Appendix J). In a chat run of the same edits on five models, without Qwen3.5-27B, putting the port first changes the JSON object by $+0.0$ points, and the neutral key lowers the effect by 2.7 points on the `docker ps` row and by 3.1 on `kubectl`.

The agent loop. The agent loop ran two models (Appendix U). Qwen3.5-9B shows the effect, 23.6 points (CI [18.1, 29.5]), and DeepSeek V4.1-Flash does not, 0.2 points (CI $[-0.8, 1.0]$). Qwen3.5-9B ran with thinking on only. With thinking off it repeated one `docker exec` call and wrote no file in any of 16 trial episodes. Olmo-3-7B-Instruct wrote no file in any of 48 trial episodes under three tool set-ups and was not run. Calls run in real bash, but the service commands are simulated: they answer for the observed service on the observed port only, other ports refuse connections, there is no network, and nothing prints the documented default. An episode ends after at most twelve assistant turns, and every episode is decoded at temperature 0. The service commands and the turn budget were fixed on 24 trial episodes of DeepSeek before any full run. DeepSeek leaves 29.9 percent of real-name and 47.2 percent of made-up-name episodes unfinished at twelve turns, and each unfinished episode counts against its arm. DeepSeek reads or tests the port in 74.5 percent of real-name episodes, Qwen3.5-9B in 1.9 percent. In the 294 real-name DeepSeek episodes that never read or test the port, it sets the default in 2.4 percent, as made-up names do in theirs. Qwen3.5-9B's effect is 38.2 points

on the `docker ps` row and 8.9 on `kubectl`. The retrieved-document condition and the code task were measured in single turns only.

The API model. The API check covers one model, DeepSeek V4.1-Flash, served as `deepseek-flash`, a rolling pointer whose served version is recorded per call (Appendix N). The provider refuses a prefilled reply in any conversation that carries a function call, so neither tool delivery nor the bridge condition ran on it. Its tool-text condition wraps the output in `<tool_response>` markers inside a user turn and is not tool delivery. Under greedy decoding the name moves the decision in one cell, `kubectl` in chat, by 3.12 points (Table 5). The probability scale is read from the top 20 log-probabilities, and a needed token falls outside them on 29.9 percent of chat steps and 28.6 percent of tool-text steps. Dropping the records with such a step moves each cell by at most 1.3 points.

The code domain. Yang (2026) reported the pull of a rebound import alias, and the code results repeat that measurement in the six models (Appendix P). Every code run is in the chat condition. The page holds no value to copy there, so the code task shows the name's pull without an overwrite (Section 3.5). The code repair trained NCSD only, and the designed SFT arm was not run. On the one-line import form, the designed evaluation, the repair moves the conventional library's share by less than 4 points in fold A and not at all in fold B (51.3 against 51.4). Scoring the two-line form it trained on was added after that result and is exploratory. Each two-line cell is 12 distinct conflict prompts decoded greedily with one seed. Every interval there includes zero, and the cell on training libraries is in-sample.

The repair: models, folds and seeds. The repair on Olmo-3-7B-Instruct is exploratory and ran in one fold with one seed (Section 4.2; Appendix A; Table 1). Its held-out half was chosen on the 1.5B model's prior strength and base error, not its own (Appendix B). On raw text both 7B adapters leave an interval above zero: 1.7 points for NCSD (CI [0.2, 3.4]) and 2.6 for SFT ([0.8, 4.9]; Section 4.3). Under tool delivery NCSD leaves 1.0 points (CI [0.0, 2.3]). The two folds, the second NCSD seed, the anchor and setting ablations, reverse KL and the two-fold comparison with inference-time methods ran on Qwen2.5-1.5B-Instruct, which is not one of the six diagnosed models. The frozen 1.5B model has a small effect to remove: on raw text it writes the default on 13 and 14 of the 316 conflict prompts per fold, a name effect of 4.63 and 4.82 points (Appendix A). Delivery moves that model opposite to the diagnosed models: its effect rises to 5.8 and 8.7 points in a chat message and falls to 1.5 and 2.6 in a tool result (Appendix K.1). The design called for 600 steps, a tuned learning rate, three seeds and half the training examples in tool-result form. The repair trained 300 steps at a fixed, untuned rate on raw text, with one seed per arm and fold and two for NCSD (Appendix A). The anchor carries two KL terms where the design had three, and its general-text term uses Python source only (Appendix B). The repair's remembered default is the documented default throughout, where the design took the model's own answer for the 20 rule-selected services (Appendix A). The designed preference-optimisation, reinforcement-learning, docker-trained and Llama-3.2-1B arms were not run. Every ablated setting leaves Δ_{def} at 0.00 on raw text and under tool delivery, so the battery cannot separate the settings on the effect (Appendix K). The public-code term protects general text, which no evaluation prompt measures. NCSD and SFT differ in scope as well as in source. NCSD matches a distribution over a sampled span of at most eight tokens, which usually runs past the digits, and SFT is a hard label on the digits alone. A difference between them can therefore come from the made-up-name teacher or from the wider target, and this paper does not separate the two (Section 4.1).

What the repair holds out. The repair holds out services and observation layouts, not action templates or anchor families (Appendix B). Its efficacy test reuses the training action templates and value sampler (Appendix A). Training and held-out services both come from the 48 diagnosis services, where the design used a disjoint pool of 68 lesser-known services (Supplement S2). Each evaluation prompt is paired with one made-up name from the teacher's pool, not length-matched, so the repair's Δ_{def} is a difference against one made-up name rather than the mean over three (Appendix B). The check on the diagnosis stimuli, which also changes the value schemes and one of the two action lines, ran on the 7B adapters and on the 24 services held out of fold A only (Appendix L). In that check the `.env` layout is a training layout with one row, and the first action line is a training template in lowercase.

Costs of the repair. With nothing on the page, the 7B adapters change their answer on 2 (NCSD) and 3 (SFT) of 21 services, and both turn an answer the frozen model had right, cassandra's 9042, into 9043 (Appendix K). On the 1.5B model every such change falls on a service the frozen model answered wrongly. Under tool delivery NCSD and SFT copy the planted non-port number on 4 or 5 of 48 prompts, against the frozen 1.5B model's 1 or 2. Under tool delivery in fold A, NCSD answers a row-absent prompt with another listed service's port on 33.5 percent, against the frozen model's 26.0 ($+7.5$ points, CI [3.7, 11.6]; Appendix K.2). In chat and tool delivery NCSD loses 4.6 and 3.1 points of recall in fold A and none detectably in fold B, where a second seed loses 2.7 and 3.4 (Appendix K.1). In those deliveries NCSD and SFT write the default on at most one conflict prompt per fold, and every anchored fold-A adapter scored in chat writes it on the same prompt, mysql shown on port 9042. Asked for the documented default on a page that shows another port, NCSD answers with it on 19.6 and 21.2 percent of raw prompts, against the frozen 1.5B model's 47.1 and 42.1 (Section 4.3). On the 7B model that rate falls from 57 to 18 percent of tool-delivered prompts. The prose family was re-drawn at 133 prompts per fold after a first draw of 22, on which fold-A NCSD sat nine points below the frozen model (Appendix K.2; Supplement S2).

Inference-time methods. The instruction, context-aware decoding and constrained copying were designed after the diagnosis was analysed (Appendix A). The adaptive variant of context-aware decoding is our reimplementation inside our decoding loop, not the authors' code (Appendix Q). Conditional anonymisation reaches zero at no recall cost, but it needs the target's surface forms at inference time (Section 4.3). It answers from the page on all 240 asked-for-default prompts in every delivery and fold of the 1.5B model (Appendix K.1). The conditional rewrite is the one method of Table 6 above the frozen model on the second request. It writes the observed value on 90.2 and 86.9 percent of conflict prompts, against the frozen model's 95.4 and 94.4, and the prompts it loses become key names such as `redis_port`, on 7.4 and 9.7 percent. An instruction naming both requests raises the frozen 1.5B model's documented-default rate by 5.42 and 7.92 points and changes its observed-value rate by -0.41 and -1.29, so it gains on the second request without improving the first (Appendix Q).

Supervising the second request. The fifth anchor family and every supervised arm were designed after the trade-off had been measured, so no preregistered test covers them (Appendix A). The fifth anchor family ran in fold A only. The supervised label is the documented default that the stimulus generator records (Appendix R). The raw-trained supervised arm ran three times, twice in fold A and once in fold B. SFT with the same supervised term ran once, in fold A. On raw text the raw-trained arm answers 75.8 to 76.7 percent of asked-for-default prompts, 3.0 to 5.4 points below the frozen model's prose recall (Section 4.3). Under chat and tool delivery it stays above the frozen model but falls 15 to 37 points below its own prose recall, mostly on the constant action `REDIS_DEFAULT_PORT =` , where it reaches 25.0 to 52.5 percent (Appendix R). On the comment action it reaches the frozen model's prose recall in every delivery: its paired difference runs from -7.4 to $+6.2$ points in chat and tool, and every interval covers zero. Its prose recall is 3 of 133 prompts (2.3 points) below the frozen model's in each fold. Under tool delivery it copies the planted non-port number on 3 or 4 of 48 prompts, against 2 and 1 for the frozen model. With nothing on the page it changes the answer on one to three of 21 services, so it fails the P2 criterion too. Both evaluation templates for the second request contain the word "default", and so does one of the two training templates, `# by default {n} listens on port .` The probe that separates the word from the request ran on fold-A pages only. On the two request forms that use the word but ask for the page, the raw-trained arm writes the default on 19.2 and 17.5 percent of raw prompts, 15.8 and 15.0 points above the frozen model (CIs [7.5, 25.8] and [7.5, 24.2]). In chat it is 7.5 points above the frozen model on the comment form.

Training in all three deliveries. The two partial rotations of Table 7 ran once each, in fold A, and each moves the trade-off to the pairing of request and delivery that its training left out (Appendix R). Rendering only the supervised asked-for-default prompts raises the chat name effect from the frozen model's 5.83 points to 21.50 (CI [13.37, 30.50]). That arm writes the default on 80.8 and 73.3 percent of chat requests that name the default but ask for the page. Rendering the binding prompts as well takes the effect to 0.00, while row-absent prompts are answered with another listed service's port on 38.8 and 43.1 percent in chat and tool, against 19.9 and 26.0 for the frozen model. That arm also copies the planted non-port number on 18.75 percent of tool prompts, against 4.17. The full rotation, with anchor prompts rendered too, ran three times, twice in fold A and once in fold B. It writes the

default on 5.0 to 60.0 percent of requests that name the default but ask for the page, against 0.0 to 20.8 for the frozen model. Its prose recall falls by at most 3.8 points, and its row-absent misbinding rises by at most 3.2. With nothing on the page it changes the answer on one or two of 21 services, so it fails the P2 criterion too. In fold B under tools it copies the planted non-port number on 2 of 48 prompts, against 1 for the frozen model. On the 7B model the full rotation ran once, in fold A, and reduces the trade-off without removing it (Table 8). It leaves Δ_{def} at 4.41 points on raw text (CI [1.81, 7.47]) and 3.82 under tools ([1.17, 6.92]), where NCSD leaves 1.68 and 1.05. Its prose recall falls by 5.3, 12.0 and 4.5 points in the three deliveries. It copies the planted non-port number on 5, 6 and 12 of 48 prompts, against the frozen model's 3, 4 and 8. With nothing on the page it changes the answer on 8 of 21 services, each one the frozen model answered wrongly. On requests that name the default but ask for the page it writes the default on 13.3 to 48.3 percent, against 0.0 to 23.3 for the frozen model. Olmo-3-7B-Instruct and Qwen2.5-1.5B-Instruct come from different families, so this pair does not separate scale from family.