DID THE MODEL READ THE PAPER? ATTRIBUTING PAPER-TO-CODE SCORES TO THEIR SOURCES

Anonymous authors
Paper under double-blind review

ABSTRACT

Paper-to-code benchmarks are meant to measure whether models can turn research papers into working code, but their scores may also reflect information already available in the provided repository. We study this source-attribution problem in ResearchCodeBench, where only the target code span is masked while the other annotated code in the same file remains visible. We introduce paired interventions that separately remove the paper and this neighboring code while keeping the task and tests fixed. Across three models, removing the paper reduces pass rate by 1.6–8.1 percentage points (nominally significant for one model), whereas masking the neighboring code reduces it by 16.8–25.4 points and eliminates roughly two thirds of successful cases; most of this effect remains when the masked neighbors are replaced by a list of the names they bind. The asymmetry is concentrated in short targets nested inside another annotated snippet. A pre-specified split of the mask shows that most of it comes from the rest of that enclosing snippet, which the benchmark leaves visible, while the method's other components matter no more than the paper. The asymmetry also shrinks under the benchmark's line-weighted metric. We further find that naively masking multiple implementations makes the original prompt ambiguous, inflating the measured effect of code context. We fix this evaluation artifact by explicitly naming the target to restore, which recovers 6.5–8.6 points, while the larger effect of neighboring code remains. Our results show that repository-assisted paper-to-code benchmarks should control alternative implementation sources, starting with the code that encloses the target, before interpreting test-passing code as evidence that a model implemented the paper.

1 INTRODUCTION

Recent benchmarks evaluate whether language models can turn scientific papers into working research code, ranging from restoring missing implementation fragments to reproducing larger research repositories (Hua et al., 2025; Xiang et al., 2025; Starace et al., 2025; Seo et al., 2026). Their scores are naturally interpreted as evidence that a model can read a scientific description and translate it into an executable implementation. This interpretation, however, depends on where the information needed for a successful completion actually comes from. If a model receives not only the paper but also code related to the missing implementation, a passing completion shows that the task was solved, but not by itself that the paper supplied the information needed to solve it.

We focus on repository-assisted implementation, where the model receives a scientific description together with an existing research codebase and restores part of it (Hua et al., 2025; Xiang et al., 2025), rather than broader research replication (Starace et al., 2025; Seo et al., 2026). In this setting, code is intentionally part of the model's context, so a successful implementation may be supported by the paper, by information already present in the repository, or by both. This also makes the attribution question experimentally tractable: the paper and implementation context can be varied independently while the target, executable tests, and grading criterion remain fixed.

The issue is not simply whether additional code context helps. Code completion is already known to benefit from in-file and cross-file context (Ding et al., 2023; Zhang et al., 2023; Liu et al., 2024b).

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.

Instead, we ask a construct validity question: can a score read as evidence of paper use distinguish information from the paper from information elsewhere in the prompt?

In this paper, we study this attribution problem on ResearchCodeBench (RCB) (Hua et al., 2025). RCB constructs 212 code restoration tasks from 20 recent machine learning papers and their authors' repositories. For each task, the model receives the paper together with the source file containing the target implementation, with one expert-annotated span replaced by a placeholder. The generated completion is inserted back into the original file and evaluated with executable tests written against the authors' implementation (Hua et al., 2025). RCB suits this analysis: grading is execution-based, most repositories postdate the training cutoffs of the models it evaluated, and its pipeline is released. Its construction, however, creates an attribution ambiguity. Only the requested span is masked, so reference implementations of other annotated components in the same file remain visible. A successful restoration can therefore draw on the paper, the neighboring implementations, or both.

We separate these sources with paired interventions that keep the task, model, tests, grader, and target fixed while changing only the information available in the prompt. In one condition, we remove the paper. In the other, we mask the reference implementations of the other annotated spans in the same file as the target. Prior benchmarks have separately examined paper or repository access (Hua et al., 2025; Xiang et al., 2025), but do not isolate paper removal and same-file implementation removal under the same tasks and grader. This direct comparison produces a clear asymmetry. Across our three main models, removing the paper changes the pass rate by -1.6, -3.2, and -8.1 percentage points (pp), nominally significant on one model; replacing it with a length-matched wrong paper, which keeps prompt length and system prompt fixed, gives -6.5 to -10.2 pp. Masking the neighboring code instead reduces the pass rate by 16.8 to 25.4 pp and removes roughly two thirds of the successful cases at risk (67, 68, and 72%); across all six endpoints with both interventions, this share is 42–72%. The effect is not explained by literal duplication, and most of it remains when each masked neighbor is replaced by a list of the names it binds. It is concentrated in short targets, most of which are nested inside another annotated snippet, so that masking also removes part of the target's own enclosing code; on outermost targets the asymmetry is not established.

Establishing this attribution also revealed a second measurement problem. RCB's official instruction refers to the missing code in the "TODO blocks." This is unambiguous when the official prompt contains a single placeholder, but masking neighboring implementations introduces several placeholders while leaving the instruction and answer extraction procedure unchanged. A naive ablation therefore mixes the effect of removing information with an ambiguity introduced by the intervention itself. Explicitly naming the requested target recovers 6.5–8.6 pp, showing that 23–31% of the neighbor effect measured under the original instruction comes from this evaluation artifact. Thus, an intervention intended to measure a benchmark can itself change the measurement instrument.

The contributions can be summarized as follows.

- We formulate source attribution as a construct validity problem for repository-assisted paper-to-code evaluation and show on RCB that masking the other annotated code in the target's file disrupts substantially more successful restorations than removing the paper. With a pre-specified decomposition, we trace 76–84% of this effect on nested targets to the rest of the snippet that encloses the target, which RCB's one-span-at-a-time masking leaves visible; the method's other components matter about as much as the paper.
- We show that naive neighbor masking introduces a prompt and extraction artifact that materially inflates the measured effect, and correct it by naming the target.
- We release a fully masked variant of RCB derived from this diagnosis, with documented limitations; it is too small to resolve paper or specification effects (Appendix D.2).

2 Related Work

Research implementation benchmarks. Recent work evaluates language models on implementing research at different levels of scope. For instance, ResearchCodeBench (RCB) restores annotated code spans from recent machine learning repositories using the corresponding paper and source context, with generated code evaluated by executable tests (Hua et al., 2025). Moreover, SciReplicate-Bench evaluates function-level reproduction from NLP papers and repositories using separate agents for understanding the literature and retrieving implementation dependencies (Xiang

et al., 2025). LMR-BENCH similarly provides a research paper together with a repository containing masked functions (Yan et al., 2025), while RECODE-H studies research code development under iterative human feedback (Miao et al., 2026). At a broader scope, PaperBench evaluates agents on reproducing complete research projects (Starace et al., 2025), and Paper2Code studies automated generation of operational repositories from scientific papers (Seo et al., 2026). RExBench sidesteps both paper and pretraining leakage by asking agents to implement unpublished research extensions (Edwards et al., 2026). Our work differs in the question being asked. These benchmarks primarily evaluate whether a model can successfully implement research, whereas we ask which supplied source carries that success. RCB is the closest prior work because it already compares performance with and without the paper (Hua et al., 2025), finding a positive average effect that is larger for stronger models. Our results refine rather than contradict this finding: even under RCB's line-weighted metric, our paper effect remains small (Appendices C.1 and B.2). SciReplicate-Bench also varies access to literature and repository resources (Xiang et al., 2025), but its algorithm description remains available when literature access is removed, and repository access is considered as a whole. Our contribution is therefore not a paper ablation in isolation, but its direct comparison with same-file implementations under the same tasks and grader.

Repository context and source attribution. Repository context is known to improve code generation. CrossCodeEval evaluates completions that require information from other files (Ding et al., 2023); RepoBench and RepoCoder study retrieval and generation using in-file and cross-file repository context (Liu et al., 2024b; Zhang et al., 2023); and CoderEval evaluates generation in realistic repository environments (Yu et al., 2024). RepoExec varies the amount of dependency context and shows that it changes what code models produce (Le Hai et al., 2025). Closer to our concern, SWE-bench+ finds solutions or direct pointers to them in the issue text of many successful SWE-bench patches (Aleithan et al., 2024), and SpecAgent identifies *future-context leakage* in completion benchmarks: code that uses a removed target still reveals its interface (Ma et al., 2026). Much of our neighbor effect is of this kind: on nested targets it comes mostly from the code enclosing the target (Section 4.1), so that neighboring code helps is not itself surprising. What these benchmarks lack is a scientific paper as a competing source under the same grader; the paired comparison between the two, and the instruction artifact it exposes, are what we add. Context attribution methods such as ContextCite score which context parts support a fixed generation (Cohen-Wang et al., 2024); we instead intervene on the available information and measure the change in executable test outcomes.

Benchmark validity and evaluation sensitivity. Benchmark performance can be supported by annotation artifacts, lexical cues, or other signals that do not reflect the capability a score is intended to represent (Gururangan et al., 2018; Niven & Kao, 2019; Geirhos et al., 2020), motivating attention to construct validity in benchmark design and interpretation (Raji et al., 2021; Bowman & Dahl, 2021; Blodgett et al., 2021; Jacobs & Wallach, 2021; Bean et al., 2025). Our setting differs from conventional shortcut analyses because repository context is an intentional part of repository-assisted research coding. The validity question is whether a score obtained from several informative sources can be interpreted as evidence that the model used one particular source, namely the paper. This issue is also distinct from pretraining contamination and memorization (Sainz et al., 2023; Golchin & Surdeanu, 2024; Jain et al., 2025; Zhou et al., 2025; Liang et al., 2025), because the competing information is explicitly present at inference time; memorized methods remain a further non-paper source that our design does not isolate (Section 5). Finally, prompt formulation and answer extraction can materially affect language model evaluation (Sclar et al., 2024; Mizrahi et al., 2024; Alzahrani et al., 2024; Biderman et al., 2024). Our instruction artifact is unusual in that it is absent from the original benchmark and appears only after an ablation changes the number of visible placeholders while the instruction and extractor remain fixed. Thus, the intervention used to estimate an information source can itself alter the measurement procedure.

3 Evaluation Design

3.1 Task and estimands

We use RCB as a controlled testbed for source attribution. The prompt contains the paper's full LaTeX source and the target file, with the target snippet replaced by a `TODO` placeholder; an RCB task thus restores missing code in an existing repository instead of writing code from a paper alone.

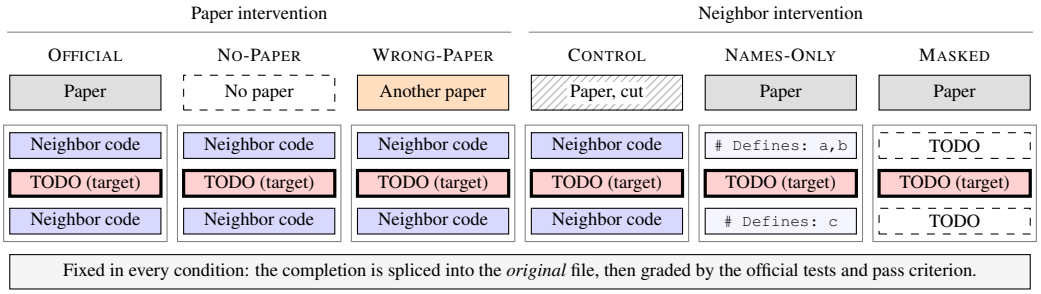


Figure 1: **What each condition shows the model.** Each column is one condition: the top box shows the paper, and the frame below shows the target file, where the target snippet is always masked. The paper intervention (left) removes the paper or replaces it with a length-matched wrong paper. The neighbor intervention (right) masks the other annotated snippets (*neighbors*) or replaces each with a list of the names it binds; CONTROL instead cuts as many characters from the paper as a length control. The instruction is either RCB's original one or NAMED, which names the target.

Table 1: **The comparisons behind our claims.** Each is paired on the same tasks, model and grader; only the prompt differs. Throughout, $A \to B$ denotes the paired change from A to B.

Comparison	What changes in the prompt	What it estimates
OFFICIAL → NO-PAPER	Paper deleted	RCB's own paper ablation
OFFICIAL → WRONG-PAPER	Wrong paper, same length	The paper's content
OFFICIAL → MASKED	Neighbors masked	The neighbors' contribution
CONTROL → MASKED	The same, at matched length	Length control
CONTROL → NAMES-ONLY → MASKED	Code removed, then names	Implementations vs. names

The other annotated snippets in the target file, which the official construction leaves visible, are the target's *neighbors*. To estimate how much the paper and the neighbors each contribute to task success, we compare paired conditions that keep the task, model, tests, and grader fixed while changing only the information available in the prompt. Figure 1 and Table 1 respectively summarize the prompt conditions and list the paired comparisons used in our analysis.

3.2 PROMPT INTERVENTIONS AND CONTROLS

Our goal is to determine which information source supports a successful restoration: the paper or the neighboring implementations in the target file. Because removing either source also changes prompt length or structure, we pair each intervention with controls for these side effects.

Isolating the contribution of the paper. OFFICIAL is the original benchmark setting, with both the paper and neighboring implementations visible. The direct paper intervention, NO-PAPER, removes the paper using the released RCB pipeline, but also shortens the prompt and changes the system prompt. We therefore introduce WRONG-PAPER, which replaces the correct paper with another benchmark paper truncated to the same length, preserving prompt length and system prompt while removing information from the correct paper (Appendix B.1). Together, NO-PAPER and WRONG-PAPER distinguish the effect of removing the paper from that of removing its content.

Isolating the contribution of neighboring implementations. MASKED masks the reference implementations of all other annotated spans in the target file, testing how much restoration depends on these neighboring implementations. Because masking also shortens the prompt, CONTROL removes the same number of characters from the paper while leaving the neighbors visible, providing a length-matched comparison. To determine *what* information the neighbors provide, NAMES-ONLY replaces each masked neighbor with a machine-extracted comment listing the names it binds (`# Defines: a, b` in Figure 1). Function signatures outside annotated spans stay visible in every condition, including MASKED; only 12 of the 186 runnable targets lose any `def` or `class` line to masking. The sequence CONTROL → NAMES-ONLY → MASKED therefore separates the contribution of full implementations from that of the names they bind.

Controlling target ambiguity. Neighbor masking also changes the structure of the task. RCB instructs the model to *"Please implement the missing code in the TODO blocks"*, which is unambiguous when the original prompt contains a single placeholder. After masking neighbors, however, only 7 of 212 tasks still contain one placeholder. A performance drop can therefore mix information loss with ambiguity about which block to complete. We address this confound with NAMED, which appends *"Implement ONLY the block named* `"{name}"` *; the other TODO blocks are not your task"*. This addition accounts for only 0.11% of the median prompt. Our main masked comparisons use NAMED whenever available, while comparing the original and NAMED instructions under identical masking directly measures the instruction artifact (Section 4.2). In the single-placeholder control, adding NAMED changes pass rate by only -2.2 pp ($p = 0.344$; Appendix A.2).

Preserving the target under masking. The masking operation must also leave the target itself unchanged. Annotated snippets are nested in 135 of the 212 tasks, so naively masking a neighboring span can overwrite a nested target. Our masker instead reproduces the official single-target masking on all tasks up to blank lines (Appendix A.3) and splits enclosing neighbors around the target, so the target placeholder occurs exactly once in every masked prompt. For a nested target, masking the neighbors therefore also masks the rest of the enclosing snippet.

3.3 EVALUATION PROTOCOL

We reproduce RCB locally because the benchmark releases its inference and grading pipeline but not the model outputs underlying its leaderboard. We generate prompts with the official inference code, insert completions into the original files, and run the official tests without modification. For the one leaderboard model we can host directly, Qwen2.5-Coder-32B-Instruct (Hui et al., 2024) (hereafter Qwen2.5-Coder-32B), our reproduction matches the published task-level outcomes on 91.5% of tasks (Cohen's $\kappa = 0.787$ (Cohen, 1960); Appendix A.1).

Our three main models are Gemma-4-12B-it and Qwen2.5-Coder-32B, self-hosted with vLLM (Kwon et al., 2023), and Mistral-Medium-3.5, accessed through a shared third-party router. Three more router endpoints (Mistral-Large and two undisclosed endpoints) serve only as robustness checks under the original instruction (Appendix F.3); these six span 17.0–36.3% on the official leaderboard metric. We also evaluate three commercial endpoints, Claude-Haiku-4.5, Claude-Sonnet-5, and GPT-6-Astra (Appendix F.5); "six endpoints" hereafter means these and the three main models. Self-hosted endpoints use one generation per prompt; router endpoints that vary under a determinism probe use three generations with majority vote (Appendix A); Mistral-Medium-3.5 passed a single-prompt probe and was run once, though our logs show it is not deterministic.

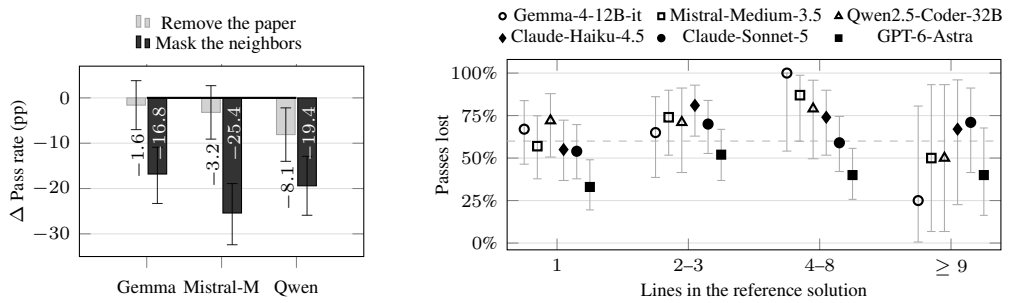
Of the 212 benchmark tasks, 26 cannot be graded in a fresh environment because dependencies no longer install or even the reference solution fails. We therefore evaluate the remaining *runnable* 186 tasks. Each paired comparison uses only tasks with a usable generation in all its conditions.

All statistical comparisons are paired at the task level. We use McNemar's exact test (McNemar, 1947), report discordant counts and the p-value, and provide a task-level bootstrap interval from 10,000 resamples for each paired pass-rate difference. These analyses treat the runnable tasks as the population, the appropriate target for claims about RCB as released. Because the tasks span only 17 repositories (3 to 37 tasks each), generalizing to paper-to-code tasks requires treating repositories as the sampling unit. We therefore also report repository-cluster intervals and p-values (Appendix E.2). Primary tables report unadjusted exact p-values. As a familywise sensitivity analysis, we apply Holm–Bonferroni correction across the 39 paired tests defined in Appendix E.3: 14 survive with task-level p-values, and 4 or 3 with repository-level ones (cluster-robust t or exact permutation test).

4 RESULTS

4.1 SOURCE ATTRIBUTION: PAPER VERSUS NEIGHBORING IMPLEMENTATIONS

Figure 2(a) gives our main source-attribution comparison. Under the same corrected instruction and on matched tasks, removing the paper causes a small decrease in pass rate on all three main models (-1.6, -3.2, and -8.1 pp). It is nominally significant on one model, survives multiplicity correction



(a) In percentage points

(b) As a share of the passes at risk

Figure 2: **Paper and neighboring-code ablations on the same tasks under the same instruction.** All conditions use the NAMED instruction, which names the target to restore, and differ only in the prompt; changes are paired, from baseline pass rates of 29.3–38.9% over $n = 184$–186 runnable tasks. The three models abbreviated in **(a)** are Gemma-4-12B-it, Mistral-Medium-3.5, and Qwen2.5-Coder-32B. **(a)** Bar labels are the paired changes in pp. Error bars are 95% task-level paired bootstrap percentile intervals (10,000 resamples); exact tests and repository-cluster intervals are in Table 10 and Appendix E.2. **(b)** The neighbor effect as a share of passes at risk: the percentage of an endpoint's successful cases removed by masking, grouped by reference-solution length, with Clopper–Pearson 95% intervals. Pooled, masking removes 333 of 556 passes (60%, dashed line). Hollow markers are self-hosted models and filled markers are the three commercial endpoints; no values are subtracted across backends (Appendices F.2 and F.5).

on none, and on Qwen2.5-Coder-32B depends on the serving engine (Appendix F.3); pooled, it is -4.1 pp and borderline once shared tasks and repositories are modeled (Table 11).

In contrast, masking neighboring implementations reduces pass rate by -16.8 to -25.4 pp. Our primary comparison contrasts the two interventions on the same tasks: removing neighboring implementations costs 15.3, 22.2, and 11.3 pp more than removing the paper. The difference is significant on all three models, its repository-cluster intervals exclude zero on all three, and it keeps its sign when any single repository is left out; which models survive multiplicity correction depends on whether tasks or repositories are the unit of inference (Table 11). **Successful restorations are consistently more sensitive to removing neighboring implementations than to removing the paper.** This conclusion is unchanged when missing generations are scored as failures (Appendix A.5).

Figure 3 asks a complementary question: *which intervention breaks individual successful cases?* Among tasks passed with both sources visible, only 1.3–12.3% survive neighbor masking but fail after paper removal (*Paper only*), while 39.6–51.9% survive paper removal but fail after neighbor masking (*Neighbors only*). *Neighbors only* also exceeds the loss from re-running an identical prompt on every endpoint with repeated generations, whereas *Paper only* does not on Claude-Haiku-4.5 or Claude-Sonnet-5 (Appendix F.6). These categories measure dependence under our interventions, not the model's internal use of either source.

Finally, Figure 2(b) asks how much of the benchmark's success is affected by this dependence. Masking neighbors removes roughly two thirds of successful baseline cases at risk on the three main models, with similar losses on the remaining endpoints except the strongest (42%). Because this dependence is concentrated in short targets, RCB's line-weighted leaderboard metric, which puts most of its weight on long targets, shrinks it: under that metric the neighbor effect on Gemma-4-12B-it is not significant, so our primary contrast depends on the metric (Appendix C.1).

Where the asymmetry lives. 126 of the 186 runnable targets are nested inside another annotated snippet, and for them MASKED also masks the rest of that snippet. The upper block of Table 2 shows that the primary contrast is concentrated there: on nested targets the neighbors cost -21.6 pp more than the paper, while on the 60 outermost targets the difference is not established. Nesting is nearly collinear with target length, so the two cannot be separated (Appendix C.4). Within nested targets, an analysis specified before any generation splits the MASKED mask into the rest of the enclosing snippet (MASK-ENCLOSING) and all other neighbors (MASK-OTHERS), on the 86 targets that have both (lower block). The enclosing snippet carries 76–84% of the effect on every model, while masking the other neighbors costs about as much as removing the paper. As we had predicted, the

Table 2: **Where the asymmetry lives.** Paired pass-rate changes (pp), pooled over models, with 95% intervals. Upper: the three main models by nesting (joint task bootstrap). Lower: the pre-specified split of the mask on nested targets, Gemma-4-12B-it, Qwen2.5-Coder-32B, Claude-Haiku-4.5 (repository-cluster bootstrap; each mask vs. its own length control). Per model: Tables 7 and 8.

Targets	n	Remove the paper	Mask all neighbors	Neighbors minus paper
Outermost	58–60	-6.1 $[-12.8, +0.3]$	-11.2 $[-20.6, -2.8]$	-5.1 $[-15.1, +4.5]$
Nested	125–126	-3.5 $[-8.3, +1.1]$	$\mathbf{-25.0}$ $[-31.3, -19.0]$	$\mathbf{-21.6}$ $[-28.2, -15.1]$
		Remove the paper	Mask enclosing snippet	Mask other neighbors
Nested, split	84	-7.5 $[-10.8, -2.8]$	$\mathbf{-23.5}$ $[-38.4, -12.0]$	-6.3 $[-11.1, +0.4]$

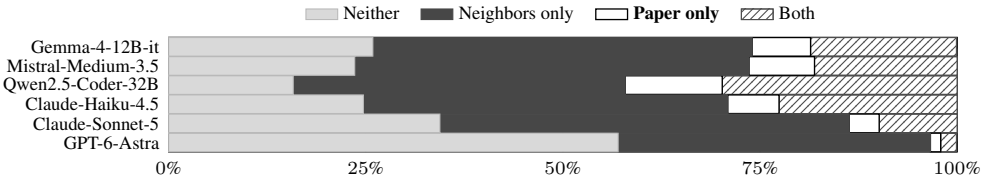


Figure 3: **Source dependence of successful baseline cases.** Every task an endpoint passes with everything visible, classified by whether the pass survives deleting the paper (NO-PAPER) and masking the neighbors (MASKED). *Paper only*: the pass survives masking but not deleting the paper; *Neighbors only*: the reverse. *Neither* mixes passes that need neither source (e.g., parametric knowledge or unannotated code) with passes the two sources can each supply alone; *Both* mixes passes that need each source with passes fragile to any prompt change. Counts: Table 19.

Table 3: **Both interventions change more than their intended source.** Paired changes in pass rate (pp) on the runnable tasks. Upper block: against the prompt with the paper, all under NAMED. Lower block: against CONTROL (neighbors visible, paper cut to the masked prompt length), which shows one placeholder under the original instruction. The pooled column combines the three models by inverse-variance weighting, with intervals that treat the models as independent (repository-cluster versions in Appendix E.2). Discordant pairs and p-values: Tables 4 and 9.

	Gemma	Mistral-M	Qwen	Pooled [95% CI]
Paper side: what replaces the paper				
NO-PAPER, as released	$\mathbf{-1.6}$	$\mathbf{-3.2}$	$\mathbf{-8.1}$	$\mathbf{-4.1}$ $[-7.4, -0.7]$
NO-PAPER, original system prompt	-4.3	-3.2	-8.6	-5.3 $[-8.4, -2.1]$
WRONG-PAPER, length-matched	$\mathbf{-6.5}$	$\mathbf{-7.1}$	$\mathbf{-10.2}$	$\mathbf{-7.8}$ $[-11.1, -4.5]$
Neighbor side: neighbors masked, under which instruction				
MASKED, original instruction	-26.1	-38.2	-26.3	-29.3 $[-33.9, -24.8]$
MASKED+NAMED	$\mathbf{-17.9}$	$\mathbf{-29.6}$	$\mathbf{-19.9}$	$\mathbf{-21.9}$ $[-26.2, -17.7]$
Instruction artifact (difference)	$+8.2$	$+8.6$	$+6.5$	$+7.7$ $[+5.1, +10.2]$

pre-specified test that the other components matter more than the paper fails (-1.2 pp, 95% interval $[-4.0, +2.9]$; Appendix C.4). **On nested targets the neighbor effect is thus mostly the target's own enclosing code, left visible because RCB masks one span at a time.**

4.2 CONTROLS FOR THE PAPER AND NEIGHBOR INTERVENTIONS

Each intervention changes more than its intended source. Table 3 puts the controls side by side: the upper block for paper removal, the lower block for neighbor masking.

Paper removal also changes prompt length and system prompt. The released NO-PAPER condition removes the paper, but also substantially shortens the prompt and changes the system prompt. Restoring the original system prompt changes the estimate only modestly. More importantly, our length-matched WRONG-PAPER condition replaces the paper with unrelated benchmark content while preserving prompt length and system prompt. This condition performs worse than remov-

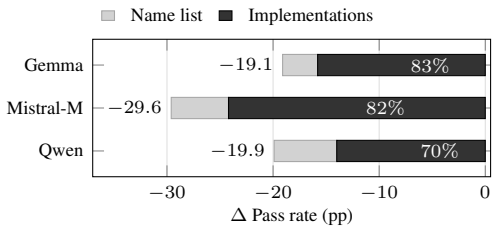


Figure 4: **What the neighbor effect consists of.** Each model's total neighbor effect (CONTROL → MASKED + NAMED, pp, at the bar's end), split into removing the implementations but keeping the names they bind (dark, with its share of the total; CONTROL → NAMES-ONLY) and then removing the names (light). Intervals: Appendix C.3.

ing the paper on all three models (-4.9, -3.8, and -2.2 pp), yielding a larger estimated paper effect (pooled -7.8 pp against zero, $p = 3 \times 10^{-6}$, and against -4.1 pp for the released condition, $p = 0.0083$; Table 3). An untruncated wrong paper gives a similar result (Appendix B.1).

Neither row isolates a pure paper contribution: removing the paper also removes the generic cost of a long document (Liu et al., 2024a), which understates the paper's benefit, while a wrong paper may mislead, which overstates it (a pooled distraction cost of -3.7 pp against no paper). The two bold rows answer different questions, so we report both per model rather than as one range. At the repository level the as-released estimate is marginal ($p = 0.0617$) while the length-matched one clears 0.05 ($p = 0.00445$; Appendix E.2).

Neighbor masking also introduces target ambiguity. Masking creates multiple placeholders under an instruction designed for a single target (Section 3.2). The lower block of Table 3 isolates this artifact by changing only the instruction: explicitly naming the requested target recovers 6.5–8.6 pp of pass rate. Multi-block responses and over-long extractions also largely disappear once the target is named, with a small residual on Gemma-4-12B-it (Appendix C.5). The pooled artifact is significant at the repository level, although no single model's artifact survives the repository-level multiplicity correction (Table 11). **Thus, naive masking overstates the neighbor effect by conflating loss of implementation context with ambiguity about which block to complete.**

We therefore use the corrected instruction rather than subtracting a fixed offset. The controls change the size of both effects, but not the main conclusion.

4.3 DECOMPOSING THE CONTRIBUTION OF NEIGHBORING CODE

We next ask *what information in neighboring code drives the effect*: its implementation or the names it exposes. Literal duplication is too rare to explain the overall effect: only 5.4% of tasks contain at least half of the reference solution verbatim elsewhere in the file (Appendix C.2). NAMES-ONLY replaces each masked neighbor with a comment listing the names it binds, decomposing the neighbor effect into the loss from removing full implementations (CONTROL → NAMES-ONLY) and the additional loss from removing the name list (NAMES-ONLY → MASKED), shown per model in Figure 4. Across all three models, most of the effect comes from full implementations (83%, 82%, and 70%). This component survives multiplicity correction for all three models using task-level p-values, and for Mistral-Medium-3.5 alone using repository-level ones. The name-list term is smaller, although its contribution cannot be ruled out. This decomposition remains after conservatively accounting for the instruction change in CONTROL → NAMES-ONLY (Appendix A).

Worked implementations also carry interface information: on 145 of the 186 runnable tasks, masking removes an identifier the reference solution uses (Ma et al., 2026), and the neighbor effect is larger there on every endpoint, though not significantly (Appendix C.3). **The neighbor effect is therefore driven primarily by full implementations, which carry interface information as well as method content, rather than by the list of names they bind.**

4.4 DOES THE PAPER MATTER MORE ONCE THE NEIGHBORS ARE MASKED?

One explanation for the small paper effect is *substitution*: neighboring implementations may already provide information that the paper would otherwise supply. If so, masking them should raise the paper effect. **We find no evidence of this substitution effect.** On the additive scale, where we predicted the interaction in advance, masking neighbors raises the paper effect on two models and lowers it on the third; the pooled interaction is $+1.0$ pp ($p = 0.674$; Appendix D.1).

A variant that masks every annotated snippet and targets the outermost one is released as a resource (Appendix D.2); with 65 tasks it cannot resolve a 5 pp paper effect, so we draw no conclusion.

5 DISCUSSION

What the small paper effect does and does not mean. A small paper effect does not imply that the paper contains no useful implementation information: masking one annotated span at a time leaves other routes to a successful restoration that the score alone cannot distinguish. Passes that survive both interventions are not attributable either: GPT-6-Astra passes 42.5% of runnable tasks with neither source (the three main models 6.5–7.7%), perhaps from a method it already knows.

Nested annotations leak the enclosing code. The largest source we found is how the benchmark was cut, not the task. When annotated spans nest, masking an inner span leaves the rest of its enclosing function visible, so the task becomes infilling an otherwise written function. Such benchmarks should mask at the outermost annotated unit, or report nested and outermost targets separately.

The instruction confound affects ablations, not the leaderboard. Official RCB scores are unaffected (one placeholder per prompt). Ablations that change the placeholder count should check that the instruction and extraction rule still identify the intended target (Biderman et al., 2024).

Limitations. Our claims concern RCB-style, single-file, repository-assisted restoration; on SciReplicate-Bench the neighbor contrast keeps its sign but is much smaller and fails our prespecified replication criterion (Appendix F.1). The three commercial endpoints give paper effects of -11.3, -4.3, and -1.6 pp, only the first significant (Appendix F.5). With 17 repositories, repository-level inference is imprecise (Appendix E.2). Mistral-Medium-3.5 is not deterministic and no longer served, Qwen2.5-Coder-32B's paper effect depends on the serving engine (Appendix F.3), and CONTROL used a process-dependent seed (Appendix A.3). The split of the neighbor mask cannot speak to outermost targets, which have no enclosing snippet.

6 CONCLUSION

On ResearchCodeBench, successful restorations are substantially more sensitive to masking other annotated code in the target file than to removing the paper. Much of this gap comes from nested targets, where the neighbor intervention also removes the implementation enclosing the target; the remaining annotated components contribute on a similar scale to the paper. When a benchmark supplies multiple routes to a correct implementation, source attribution is necessary before its score can be read as evidence that a model implemented research from the paper itself.

REPRODUCIBILITY STATEMENT

Section 3 states the task definition, the content and instruction conditions (the pre-specified split of the mask is defined in Appendix C.4), the repetition protocol per hosting mode, the runnable-subset denominator and the exact statistical tests; Appendix A gives the masker's self-check, the determinism probe and the flakiness measurement in full. Every experimental number in the paper is a LaTeX macro that one of two supplementary scripts generates from the raw per-task verdict files, and a check mode reports any macro that is stale. The benchmark's construction counts (tasks, papers, nested snippets, visible placeholders per ablation) are recomputed the same way from the released task set. There are two exceptions: the published leaderboard scores, which we did not measure, and a small number of appendix tables that quote a script's output directly and are marked as such in the source. The three sources of irreducible variation (prefix-cache non-determinism on self-hosted models, batching non-determinism on the shared router, and unseeded randomness in 43 of the benchmark's own tests) are each measured rather than assumed (Section 3 and Appendix A). One condition cannot be regenerated exactly: the paper cut of CONTROL was drawn from a process-dependent seed and its prompts were not stored (Appendix A.3). The construction of the released variant, including its departures from full masking, is documented in Appendix D.2. The benchmark itself, its papers and its tests are public (Hua et al., 2025); we release our masking, generation, splicing and analysis code, our raw generations and our per-task verdicts under every condition.

ETHICS STATEMENT

This work evaluates publicly released language models on a publicly released benchmark built from published papers and their authors' open-source repositories. No human subjects were involved and no new data about people was collected. The study is a critique of a measurement instrument, not of the people who built it: RCB is unusually careful and reproducible, and the confound we identify is findable only *because* the pipeline was released in full. We have taken care to state what our results do not show, in particular that they do not license a claim that paper-to-code evaluation is unsound or that any published leaderboard is wrong. Two of the endpoints we evaluate are served by a third-party router under names that hide the backing model; we report them separately, never in the main tables, and describe in Appendix F.3 our check rather than speculating about their identity.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic data (the natural-language specifications of the fully masked variant, Appendix D.2); to implement methods (the masking, generation, splicing and analysis code); to clean and reformat the per-task result files; to propose options for, and give feedback on, the experimental design; and to assist in interpreting results. We have not used generative AI tools to propose or refine hypotheses, to develop conceptual frameworks, or for translation, and formulating or proving mathematical claims is not applicable to this work. Additionally, we used generative AI tools to write and edit code, create figures, assemble the supplementary material, draft and copy-edit parts of the paper, search for and summarize related benchmarks, and check references. Language models also appear in this work as *objects of study*: they are the systems under evaluation, and their generations are the raw data.

The research question and every final design decision, including the decision rules written down before the runs, were made by the authors, who chose among options that AI tools proposed. We have reviewed all AI-assisted text. AI-written code was verified through its outputs and automated checks rather than read line by line: the masker aborts unless it reproduces the official prompt on every task up to whitespace; every experimental number in the paper is a macro generated from the raw per-task verdict files, and a check mode reports any stale macro; the macro-generation step imports its statistics from the analysis scripts rather than recomputing them; and the supplementary package recomputes all macros in a fresh environment. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

REFERENCES

Reem Aleithan, Haoran Xue, Mohammad Mahdi Mohajer, Elijah Nnorom, Gias Uddin, and Song Wang. SWE-Bench+: Enhanced coding benchmark for LLMs. *arXiv preprint arXiv:2410.06992*,

2024.

Norah Alzahrani, Hisham Abdullah Alyahya, Yazeed Alnumay, Sultan AlRashed, Shaykhah Alsubaie, Yousef Almushayqih, Faisal Mirza, Nouf Alotaibi, Nora Al-Twairesh, Areeb Alowisheq, M Saiful Bari, and Haidar Khan. When benchmarks are targets: Revealing the sensitivity of large language model leaderboards. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Andrew M. Bean, Ryan Othniel Kearns, Angelika Romanou, Franziska Sofia Hafner, Harry Mayne, et al. Measuring what matters: Construct validity in large language model benchmarks. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, et al. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*, 2024.

Su Lin Blodgett, Gilsinia Lopez, Alexandra Olteanu, Robert Sim, and Hanna Wallach. Stereotyping Norwegian salmon: An inventory of pitfalls in fairness benchmark datasets. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

Samuel R. Bowman and George E. Dahl. What will it take to fix benchmarking in natural language understanding? In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021.

C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. ContextCite: Attributing model generation to context. In *Advances in Neural Information Processing Systems*, 2024.

Yangruibo Ding, Zijian Wang, Wasi Uddin Ahmad, Hantian Ding, Ming Tan, Nihal Jain, Murali Krishna Ramanathan, Ramesh Nallapati, Parminder Bhatia, Dan Roth, and Bing Xiang. CrossCodeEval: A diverse and multilingual benchmark for cross-file code completion. In *Advances in Neural Information Processing Systems 36: Datasets and Benchmarks Track*, 2023.

Nicholas Edwards, Yukyung Lee, Yujun Audrey Mao, Yulu Qin, Sebastian Schuster, and Najoung Kim. RExBench: Can coding agents autonomously implement AI research extensions? In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

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.

Shahriar Golchin and Mihai Surdeanu. Time travel in LLMs: Tracing data contamination in large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

Tianyu Hua, Harper Hua, Violet Xiang, Benjamin Klieger, Sang T. Truong, Weixin Liang, Fan-Yun Sun, and Nick Haber. ResearchCodeBench: Benchmarking LLMs on implementing novel machine learning research code. In *Advances in Neural Information Processing Systems 38: Datasets and Benchmarks Track*, 2025. arXiv:2506.02314.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Abigail Z. Jacobs and Hanna Wallach. Measurement and fairness. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2021.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations (ICLR)*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Nam Le Hai, Dung Manh Nguyen, and Nghi D. Q. Bui. On the impacts of contexts on repository-level code generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025.

Shanchao Liang, Spandan Garg, and Roshanak Zilouchian Moghaddam. The SWE-Bench illusion: When state-of-the-art LLMs remember instead of reason. *arXiv preprint arXiv:2506.12286*, 2025.

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.

Tianyang Liu, Canwen Xu, and Julian McAuley. RepoBench: Benchmarking repository-level code auto-completion systems. In *International Conference on Learning Representations (ICLR)*, 2024b.

George Ma, Anurag Koul, Qi Chen, Yawen Wu, Sachit Kuhar, Yu Yu, Aritra Sengupta, Varun Kumar, and Murali Krishna Ramanathan. SpecAgent: A speculative retrieval and forecasting agent for code completion. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

Chunyu Miao, Henry Peng Zou, Yangning Li, Yankai Chen, Yibo Wang, Fangxin Wang, Yifan Li, Wooseong Yang, Bowei He, Xinni Zhang, Dianzhi Yu, Hanchen Yang, Hoang H Nguyen, Yue Zhou, Jie Yang, Jizhou Guo, Wenzhe Fan, Chin-Yuan Yeh, Panpan Meng, Liancheng Fang, Jinhu Qi, Wei-Chieh Huang, Zhengyao Gu, Yuwei Han, Langzhou He, Yuyao Yang, Yinghui Li, Hai-Tao Zheng, Xue Liu, Irwin King, and Philip S. Yu. RECODE-H: A benchmark for research code development with interactive human feedback. In *International Conference on Learning Representations (ICLR)*, 2026.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art? a call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949, 2024.

Timothy Niven and Hung-Yu Kao. Probing neural network comprehension of natural language arguments. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

Inioluwa Deborah Raji, Emily M. Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. AI and the everything in the whole wide world benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design. In *International Conference on Learning Representations (ICLR)*, 2024.

Minju Seo, Jinheon Baek, Seongyun Lee, and Sung Ju Hwang. Paper2code: Automating code generation from scientific papers in machine learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research. *arXiv preprint arXiv:2504.01848*, 2025.

Yanzheng Xiang, Hanqi Yan, Shuyin Ouyang, Lin Gui, and Yulan He. SciReplicate-Bench: Benchmarking LLMs in agent-driven algorithmic reproduction from research papers. In *Conference on Language Modeling (COLM)*, 2025.

Shuo Yan, Ruochen Li, Ziming Luo, Zimu Wang, Daoyang Li, Liqiang Jing, Kaiyu He, Peilin Wu, George Michalopoulos, Yue Zhang, Ziyang Zhang, Mian Zhang, Zhiyu Chen, and Xinya Du. LMR-BENCH: Evaluating LLM agent's ability on reproducing language modeling research. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6164–6186, 2025.

Hao Yu, Bo Shen, Dezhi Ran, Jiaxin Zhang, Qi Zhang, Yuchi Ma, Guangtai Liang, Ying Li, Qianxiang Wang, and Tao Xie. CoderEval: A benchmark of pragmatic code generation with generative pre-trained models. In *IEEE/ACM International Conference on Software Engineering (ICSE)*, 2024.

Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. RepoCoder: Repository-level code completion through iterative retrieval and generation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Xin Zhou, Martin Weyssow, Ratnadira Widyasari, Ting Zhang, Junda He, Yunbo Lyu, Jianming Chang, Beiqi Zhang, Dan Huang, and David Lo. LessLeak-Bench: A first investigation of data leakage in LLMs across 83 software engineering benchmarks. *arXiv preprint arXiv:2502.06215*, 2025.

A Evaluation protocol and validity checks

This appendix documents the pipeline behind every number in the paper: that it reproduces the leaderboard, that the masker and the length control do what they claim, how much noise the endpoints and the benchmark's own tests add, and which tasks enter each denominator.

A.1 Reproducing the leaderboard

RCB releases the benchmark, the papers, the tests and the generation pipeline, but not the model outputs behind the leaderboard, so we reproduce the evaluation locally. All prompts come from the official inference code path and are byte-identical to the official ones under Official; completions are spliced back and graded by the official tests with the official pass criterion verbatim. We patch, reorder and re-seed nothing, with one exception outside the main protocol: in the variant of Appendix D.2, 5 tasks are graded with a recalibrated test.

Every result in this paper is a difference between two conditions measured on our pipeline, so that pipeline must first reproduce the official one. Exactly one leaderboard model is small enough for us to host, Qwen2.5-Coder-32B. Under the official condition it passes 61 of 212 tasks versus 55 on the released leaderboard, with 91.5% task-level agreement (Cohen's $\kappa = 0.787$). The disagreements ($6/12$ leaderboard-only / ours-only) do not significantly favor either side (exact binomial $p = 0.238$). On the runnable subset, they favor us: $2/12$ in the same order ($\kappa = 0.822$, $p = 0.0129$), a net 10 tasks in our favor, observed after a change of serving engine (Appendix F.3). This offset concerns the level of pass rates relative to the leaderboard; every claim in the paper is a paired difference between conditions run on the same pipeline and engine.

For scale, the six main endpoints score 17.0–36.3% on the leaderboard's own denominator (all 212 tasks, unweighted pass@1), against 50.0% for GPT-4.1 and 64.2% for the strongest entry listed. The variant of Section 4.4 also uses two router endpoints, MiniMax-M3 and Muse-Spark-1.2.

A.2 THE INSTRUCTION AXIS

The official instruction, *"Please implement the missing code in the TODO blocks"*, is inert under OFFICIAL, where all 212 tasks show exactly one `TODO`, and ambiguous the moment an ablation masks the neighbors: of the 212 tasks only 7 (3.3%) still show a single `TODO` under MASKED, while 205 show two or more, up to 10. The NAMED sentence changes nothing else in the prompt: a median 97 characters or 0.11% of it. We run NAMED variants of MASKED, NAMES-ONLY, OFFICIAL, NO-PAPER and the wrong-paper conditions. CONTROL keeps the neighbors visible and so shows a single placeholder, for which the original instruction is already unambiguous; we run it under that instruction only. Comparisons that start from CONTROL (the neighbor length control and the decomposition of Section 4.3) therefore also change the instruction; on the one endpoint run under both, the instruction moves a single-placeholder condition by -2.2 pp ($p = 0.344$). All other comparisons across the two instructions measure the confound.

A.3 THE MULTI-SNIPPET MASKER AND THE LENGTH CONTROL

The official pipeline masks exactly one snippet, so it never has to decide what to do when two annotated snippets overlap. Our ablations mask many, and overlap is the common case rather than the exception: 135 of the 212 annotated snippets (63.7%) are nested inside another snippet.

A masker that simply overwrites each neighbor in turn produces a prompt in which an enclosing neighbor has swallowed the target, taking the target's own `TODO` placeholder with it. The prompt then asks the model to implement the *enclosing* block, whose name is the only one present, while the splicing step still pastes the model's output into the target's much smaller slot. Such a task fails by construction. Our masker instead splits an enclosing neighbor around the target and leaves the target's placeholder intact. Fixing the error changes pass rates less than one might expect, because nested targets are rarely solved once their neighbors are masked: on the two models we ran with both masker versions, the fix raised MASKED and CONTROL by about a point each (Appendix F.4).

One check is asserted on every run: restricted to a single snippet, the masker must match the official `mask_given_snippet()` on all 212 tasks after whitespace is removed; a mismatch aborts the run. The match is not byte for byte: the official masker leaves a blank line where each marker was and ours drops it, so the two differ by 1 to 3 blank lines on every task and by nothing else (212 of 212 tasks have identical non-blank lines). The self-check calls only the single-snippet path, so it does not exercise the nested split; we verified post hoc that the target placeholder occurs exactly once in the MASKED and NAMES-ONLY prompts of all 212 tasks, but this is not asserted at runtime.

Length matching for CONTROL. CONTROL deletes randomly chosen spans of the paper amounting to the same number of characters that MASKED removes. The span choice was seeded with Python's per-process salted `hash()`, so it is not reproducible across runs, and the CONTROL prompts were not stored: each endpoint received a different random cut, and we cannot recover which spans any cell deleted. Rebuilding the paper from its spans also dropped inter-span whitespace, so CONTROL removes somewhat more than the intended length, while the logs record the intended length. Two independent draws on the near-deterministic Gemma-4-12B-it give 28.6 and 31.4% (9 of 185 verdicts differ). The headline contrasts (OFFICIAL $\rightarrow$ MASKED and NO-PAPER versus MASKED) do not use CONTROL. On Mistral-Medium-3.5, CONTROL scores $+4.3$ pp above OFFICIAL ($4/12$, $p = 0.0768$; bootstrap $[+0.5, +8.6]$; the two also differ in instruction and generation date), against $+1.6$ and $+0.5$ pp on the other two models, so CONTROL-based quantities on Mistral-Medium-3.5 (Table 3, lower block, and Figure 4) run about that much larger than its OFFICIAL-based neighbor effect. The number of characters is computed *against the masked prompt of the same task*, with a consequence: whenever the masking changes, CONTROL must be regenerated with it, or the pairing silently decays. Per task, the median absolute prompt-length difference between CONTROL and MASKED is 0.23%, and 0.61% against NAMES-ONLY.

A.4 NON-DETERMINISM AND GRADING NOISE

Determinism probes. Before using any endpoint we issue five identical requests at temperature 0 and compare the completions verbatim. This distinguishes two sources of non-determinism that a single run cannot: a self-hosted engine whose divergence comes from prefix-cache state, and a shared router whose divergence we attribute to batching with other users' requests. On the self-hosted models, repeated identical prompts diverge 22.5% of the time, and in 100% of those cases the first generation differs while the later ones agree, so a majority vote would only reproduce this fixed $2:1$ pattern rather than sample the model. We therefore use one generation per prompt there, and report that recomputing the entire run under the opposite reading flips 6 of 95 divergent groups and moves every reported figure by at most 1.1 pp. On the router, one probe returned five byte-identical completions and two returned distinct answers in no fixed pattern; the latter use three generations and a majority vote. The probe is a smoke test on one prompt and did not establish determinism for the endpoint that passed it: Mistral-Medium-3.5 was run once per prompt, but repeated identical prompts in our logs show that it is not deterministic. Of 65 prompts regenerated a day apart, 30 produced the same code and 59 of 63 verdicts agree; of 23 prompts regenerated about four weeks apart, 0 did (a gap that also admits a change of backing model; 18 of 23 for self-hosted Gemma-4-12B-it). The router no longer serves this model, so it cannot be re-run with three generations. Symmetric generation noise does not bias McNemar's test, but it lowers power and makes the per-task categories of Figure 3 noisier for this endpoint.

The benchmark's own grading noise. 43 tasks across 5 repositories are graded by tests that draw random inputs without setting a seed. Re-running one of them (`TabDiff`, the categorical-noise snippet) thirty times on a byte-identical file yielded 23 passes and 7 failures; the split itself is subject to the same randomness. Repairing this would mean changing the official pass criterion, so we bounded it instead. Re-running 86 task–condition pairs 7 times each changed exactly 1 verdict, 1.2% of pairs. That rate understates the risk: the pairs come from one model's original-instruction generations, and 61 of the 86 pairs never pass and so cannot flip; conditional on passing at least once, the flip rate is 4.0%. The neighbor effects of Section 4 are an order of magnitude larger than this noise, but the paper effect is of the same order. On the tasks whose tests are seeded, the paper effect is -5.7, -3.5, and -10.5 pp (Gemma-4-12B-it: $11/3$, $p = 0.0574$), and on the unseeded ones $+11.6$, -2.3, and $+0.0$ pp; a framing-only control is not elevated on the unseeded tasks, so we do not attribute the difference to test noise, and because the split is post hoc we do not use it elsewhere.

A.5 DENOMINATORS, DROPPED GENERATIONS, AND RECORDED PREDICTIONS

Of the 212 official tasks, 21 belong to repositories whose dependencies we could not install without pinning an obsolete toolchain, and a further 5 fail the official tests when the *authors' own reference solution* is spliced in, which we established by running the reference solutions through the identical pipeline. The runnable subset is the remaining 186. Because all 26 excluded tasks fail under every condition, excluding them changes no McNemar p-value at all; it changes only the denominator, and hence every effect expressed in percentage points, by about 12% relative. Our script computes both; the subset is a command-line flag. Each comparison further keeps only tasks with a usable generation under every condition, so denominators are 186 in most tables and 183–185 in some.

Dropped generations, and the worst case. A generation that reaches the token limit without emitting a code block yields nothing to splice, and the protocol drops the cell rather than scoring it: 15, 5 and 0 task–condition cells across the three endpoints. The drops are not neutral. Such a generation would almost certainly have failed, and most fall in the ablated conditions, so dropping them biases the paper intervention toward zero, which favors our claim. We therefore also score every dropped cell as a failure over all 186 tasks: Gemma-4-12B-it's paper intervention moves to -2.2 ($p = 0.557$) and its neighbor intervention to -16.7, and no Section 4.1 conclusion changes.

Recorded predictions. Several predictions and decision rules were written down before the corresponding later runs finished, but were not externally registered and should not be interpreted as a formal pre-registration. We report them for transparency together with the analyses they informed. The prediction concerning the instruction artifact is discussed in Appendix C.5, and the predicted interaction between neighbor masking and paper contribution in Appendix D.1.

B PAPER-SIDE CONTROLS

Section 4.2 and Table 3 give the controls for the paper deletion. This appendix gives their construction, per-model counts, and relation to the benchmark's own paper ablation.

B.1 CONTROLS FOR THE PAPER DELETION

NO-PAPER is the one condition we cannot length-match by cutting, since restoring the characters would mean inserting filler; its prompt is 93.1% shorter at the median, and it also swaps the official system framing from *"an expert in reproducing research code from a paper"* to *"an expert in completing research code"*, because that is what the released pipeline does. Both threats point toward the null rather than away from it: a 93.1% cut removes distracting as well as informative context, and long irrelevant context is known to cost accuracy on its own (Liu et al., 2024a), while the framing swap describes the task as code completion. We therefore run additional conditions that separate these from the deletion itself: one that removes the paper while keeping the original framing, and two that replace the paper with a *different* paper from the same benchmark, so that register is preserved and only the content is wrong (Section 4.2). Of the latter two, the first substitutes the other paper untruncated, leaving a median 23.0% prompt-length difference; the second truncates it to the target paper's exact character count (0.00% by construction) and supplies our estimates. Their pooled estimates agree (-7.6 untruncated, -7.8 pp length-matched), and the untruncated wrong paper is also worse than no paper, by -3.2 pp pooled (95% CI $[-5.9, -0.5]$).

Table 4 collects every paper intervention we ran, with discordant pairs and p-values.

Table 4: **Effect of removing or replacing the paper, under six conditions** (Sections 4.2 and 4.4). Paired changes against the same prompt with the paper present, under NAMED; parentheses give discordant pairs and McNemar's exact p. Row 1 is the ablation the benchmark constructs; rows 2–4 each address one of its confounds, and row 4, the length-matched wrong paper, keeps prompt length and framing fixed; row 5 repeats row 1 for Qwen2.5-Coder-32B on the earlier FP8 serving engine (Appendix F.3); row 6 repeats the ablation with the neighbors masked.

Neighbors	Paper intervention	Gemma-4-12B-it	Mistral-Medium-3.5	Qwen2.5-Coder-32B
Visible	NO-PAPER, as built	-1.6 (14/11, $p = 0.69$)	-3.2 (19/13, $p = 0.377$)	-8.1 (24/9, $p = 0.0135$)
	System prompt held fixed	-4.3 (17/9, $p = 0.169$)	-3.2 (16/10, $p = 0.327$)	-8.6 (23/7, $p = 0.00522$)
	Wrong paper, untruncated	-4.9 (16/7, $p = 0.0931$)	-11.4 (24/3, $p = 4.92e-05$)	-7.0 (23/10, $p = 0.0351$)
	Wrong paper, length-matched	$\mathbf{-6.5}$ (19/7, $p = 0.029$)	$\mathbf{-7.1}$ (20/7, $p = 0.0192$)	$\mathbf{-10.2}$ (26/7, $p = 0.00132$)
	NO-PAPER, FP8 engine	–	–	-5.4 ($p = 0.121$)
Masked	NO-PAPER, as built	-4.9 (13/4, $p = 0.049$)	-7.0 (14/1, $p = 0.000977$)	-4.3 (11/3, $p = 0.0574$)

Under the original instruction. The spread between models in Section 4.1 is not created by the corrected instruction. Under the original one (run on Qwen2.5-Coder-32B and Gemma-4-12B-it), removing the paper was already nominally significant on Qwen2.5-Coder-32B (-8.1 pp, $23/8$, $p = 0.0107$) and not on Gemma-4-12B-it (-2.7 pp, $p = 0.458$), as under NAMED.

B.2 WHY OUR PAPER ABLATION DIFFERS FROM THE ONE IN THE ORIGINAL BENCHMARK

RCB reports its own with-versus-without-paper comparison and finds a positive average effect that grows with model strength (Hua et al., 2025). Three differences separate it from ours. *Metric*: their headline paper-context figure is a difference in *line-weighted* scaled pass@1, which up-weights long snippets, while we report unweighted pass@1; Appendix C.1 recomputes both our interventions under their weighting. *Range*: their effect is concentrated at the top of their leaderboard. Most of our endpoints sit below its midpoint; the two frontier endpoints reach its upper range, and there the paper intervention gives -11.3 and -4.3 pp: one significant, one not, and no trend between them (Appendix F.5). *Instrument*: their comparison is unambiguous in *both* interventions, since removing the paper adds no placeholders, so our confound does not touch their number, though it does touch any later ablation that changes how many placeholders are visible. We read our result as bounding the paper's marginal contribution in the range we can measure, not as overturning theirs.

B.3 What separates removing the paper from replacing it

The two bold rows of Table 3, pooled at -4.1 and -7.8 pp, differ by the contrast between No-Paper and Wrong-Paper; the difference is not a confidence-interval width, and each estimate carries its own sampling uncertainty. The first is Official $\rightarrow$ No-Paper and the second Official $\rightarrow$ Wrong-Paper, so for each model the difference between them is No-Paper $\rightarrow$ Wrong-Paper up to which tasks each pair covers, the amount by which a wrong paper is worse than no paper at all; because each estimate is pooled with its own inverse-variance weights, the two pooled estimates differ only approximately by the pooled value of that quantity, -3.7 pp. A heterogeneity test does not reject a common value of that quantity across the three models ($Q = 0.72$, $p = 0.698$).

That -3.7 pp is not one mechanism. Relative to No-Paper, the wrong-paper prompt differs in three ways at once: it is longer by the size of a paper, it keeps the original system framing, and its content describes a method. The framing term we measure separately and it is small: -2.7, $+0.0$ and -0.5 pp (discordant $6/1$, $8/8$, $3/2$; $p = 0.125$, 1, 1). The length term we measure only at two smaller scales, and at both it is close to zero: the two wrong-paper constructions differ by 23.0% in median length yet the condition built on them differs by $+0.3$ pp pooled, and deleting a matched number of characters from the paper while leaving the file intact (the Control condition of Section 3.2) moves the 5 endpoints carrying both conditions by at most 1.6 pp, smallest p 0.549. Neither measurement reaches the scale at issue here, a whole paper against none, so we do not claim the length term is zero, only that we have not detected it at the scales we could test.

Separating the three terms would take one further condition, a document of the wrong paper's length and framing that describes no implementable method; we did not run it. We looked for the misleading term directly instead. Identifiers that appear in the wrong paper's equations and pseudocode, but nowhere in the task's own paper, do not measurably appear in the code generated under Wrong-Paper (no model's sign test approaches significance; smallest $p = 0.388$), while the same count applied to each task's own paper detects uptake under Official on all three models (largest $p = 0.0153$). The count catches borrowed names rather than a borrowed approach, so it bounds the misleading term rather than excluding it; this is why Section 4.2 reads the gap as a distraction cost.

C Neighbor-side controls and decomposition

This appendix checks what the neighbor effect is made of: how it depends on target length and line weighting, whether it is literal copying, how it splits between implementations and names, where nesting concentrates it, and how much of the original-instruction estimate is an instruction artifact.

C.1 Target length and the benchmark's line-weighted metric

Figure 2(b) plots the cells of Table 5. Each entry is the number of tasks an endpoint passes with everything visible that it fails once the neighbors are masked, over the number it passes, with the percentage and its Clopper–Pearson 95% interval. The last two columns test the four strata against each other *within* one endpoint, where the model, the serving backend and the generation run are all fixed: a permutation test on Pearson's statistic for heterogeneity, and a Cochran–Armitage permutation test for a monotone trend in length, both with $50{,}000$ relabelings. Neither rejects on any endpoint. These are the tests the stratum sizes allow, and against a gentle trend they have little power. The 24 intervals do not share a common value: GPT-6-Astra loses a smaller share (42% over all strata) than the other endpoints. **No quantity here is taken across two endpoints (Appendix F.2).**

The line-weighted metric. RCB's leaderboard reports *scaled* pass@1, defined in its released code as $\sum_i \ell_i \cdot \mathbf{1}[\text{pass}_i] / \sum_i \ell_i$ with ℓ_i the reference snippet's line count; we report unweighted pass@1 throughout. The reference lengths are heavily skewed (median 4 lines, maximum 58), so the longest third of the tasks carries 76% of the weight (Table 6).

Intervals are task-level paired bootstraps ($10{,}000$ resamples) on the weighted difference. Under the weighting, the neighbor effect shrinks to less than half its unweighted size and, on Gemma-4-12B-it, is no longer significant; the paper intervention is significant on Qwen2.5-Coder-32B alone, under both metrics. The cause is visible in the length strata: the neighbor effect is $-32.8/-35.4/-30.8$ pp on the 1–2-line tasks ($n = 64$–65) against $-5.3/-15.5/-5.1$ on the $\geq$ 6-line ones ($n = 57$–59), so a

Table 5: **Passes destroyed by masking the neighbors, by reference-solution length.** Each cell gives lost / passed and the percentage with its exact interval; n per stratum is 44, 49, 58 and 35 tasks, the same for every endpoint except one or two generations with no code block (Appendix A.5).

	Lines in the reference solution				Permutation p	
Endpoint	1	2–3	4–8	≥ 9	Heterog.	Trend
Gemma-4-12B-it	18 / 27 67% [46, 83]	11 / 17 65% [38, 86]	6 / 6 100% [54, 100]	1 / 4 25% [1, 81]	0.092	0.418
Mistral-Medium-3.5	17 / 30 57% [37, 75]	17 / 23 74% [52, 90]	13 / 15 87% [60, 98]	2 / 4 50% [7, 93]	0.161	0.488
Qwen2.5-Coder-32B	18 / 25 72% [51, 88]	10 / 14 71% [42, 92]	11 / 14 79% [49, 95]	2 / 4 50% [7, 93]	0.798	0.651
Claude-Haiku-4.5	18 / 33 55% [36, 72]	25 / 31 81% [63, 93]	17 / 23 74% [52, 90]	4 / 6 67% [22, 96]	0.133	0.469
Claude-Sonnet-5	22 / 41 54% [37, 69]	26 / 37 70% [53, 84]	23 / 39 59% [42, 74]	10 / 14 71% [42, 92]	0.404	0.469
GPT-6-Astra	14 / 43 33% [19, 49]	24 / 46 52% [37, 67]	18 / 45 40% [26, 56]	6 / 15 40% [16, 68]	0.312	0.96

Table 6: **Both interventions under the benchmark's own line-weighted metric.** Paired differences in the weighted score, with 95% task-level paired bootstraps (10,000 resamples) on the weighted difference. Under the weighting the neighbor effect falls to less than half its unweighted size; Figure 2(b) shows where that comes from.

	Remove the paper		Mask the neighbors	
Model	Unweighted	Line-weighted	Unweighted	Line-weighted
Gemma-4-12B-it	-1.6	$+0.2$ $[-4.6, +5.5]$	-16.8	-3.7 $[-10.0, +3.4]$
Mistral-Medium-3.5	-3.2	-3.4 $[-8.0, +0.5]$	-25.4	-12.0 $[-17.7, -7.7]$
Qwen2.5-Coder-32B	-8.1	-4.8 $[-10.3, -0.2]$	-19.4	-7.5 $[-13.9, -1.9]$

weighting that puts three quarters of its mass on the long tasks is weighting down exactly where the effect is. The effect shrinks monotonically with length on all three models, as expected if models follow a neighbor's implementation. Our primary contrast therefore depends on the metric.

We report unweighted pass@1 because our question is what a *pass* is made of, and a weighting by restored lines answers a different one. The choice matters, and a reader comparing our numbers with the leaderboard's should use this table. It is also one of the three differences between our paper ablation and RCB's, and on its own it does not reconcile them (Appendix B.2).

C.2 The neighbors supply worked implementations, not copyable text

Section 4.3 argues that the neighbors supply *worked implementations*, not *copyable* text. The distinction is measurable without running a model at all: for each task we ask what share of the lines of the reference answer appear verbatim somewhere else in the same file *as the official prompt shows it*. The file content is taken from the official masker's output, never reconstructed from reference solutions; Appendix A.3 explains why that distinction matters when snippets nest.

Over the 185 tasks for which this is defined, the median share is 0% and the mean 6.5%. Only 10 tasks (5.4%) have half their answer available verbatim elsewhere in the file, and 5 (2.7%) have all of it. Line-level copying is therefore rare, and an ablation that removed duplicated text would not remove the effect we measure. What the neighbors supply is a completed implementation of other annotated code in the file, for nested targets including the rest of the enclosing snippet (tensor shapes, the order of operations, the names the target must bind, the file's conventions), which a model can follow without reproducing a single line.

One might expect verbatim overlap to make a task easy, so that the few overlapping tasks account for a disproportionate share of passes. The data show the opposite. On the 35 tasks with any verbatim overlap the unablated pass rate is *lower* on every endpoint we ran: 14.3 against 32.7%, 28.6 against 40.9%, 14.3 against 34.0%, 42.9 against 51.3%, 60.0 against 72.7%, and 65.7 against 83.3%. Those tasks are longer and harder than the rest. For this reason the fix we recommend is

masking the neighbor spans rather than de-duplicating the file: the channel is not text that can be copied, and removing duplicate text would leave it open.

C.3 IMPLEMENTATIONS VERSUS NAMES

NAMES-ONLY is MASKED with one machine-extracted comment per masked neighbor, `# Defines:`, listing the names it binds; the names are found by a regular expression over assignments, loops, `with` targets, and `def` and `class` lines. It adds a median of 272 characters and is identical to MASKED on 5 runnable tasks, because 14 of the 212 annotated snippets bind no name. Only 4 snippets contain a `def` or `class` line, so function signatures outside the annotated spans are visible in every condition; MASKED removes such a line on only 12 runnable tasks. The NAMES-ONLY $\rightarrow$ MASKED term therefore measures the removal of a list of names, not of signatures. Its task bootstrap intervals are $[-6.6, +0.0]$, $[-9.1, -1.6]$, and $[-10.2, -2.2]$ pp (Figure 4).

Totals in Figure 4. Each total is computed on the tasks where all three conditions were scored. Gemma-4-12B-it's total differs from Table 3 (n=183 versus 184) because a few generations hit the token limit without a code block (Appendix A.5). Mistral-Medium-3.5's total runs above its OFFICIAL-based effect because its CONTROL scores above OFFICIAL (Appendix A.3).

Bootstrapped shares for the decomposition. A task-level paired bootstrap (10^4 resamples) puts the leaked implementations' share of the total neighbor effect of Section 4.3 at $[65, 100]\%$ for Gemma-4-12B-it, $[69, 94]\%$ for Mistral-Medium-3.5 and $[52, 88]\%$ for Qwen2.5-Coder-32B. Gemma-4-12B-it's interval reaches the upper boundary, so it allows a zero name-list term.

The instruction change inside the leaked term. CONTROL $\rightarrow$ NAMES-ONLY changes the instruction as well as the content, because CONTROL was run only under the original instruction. The size of that change on a condition showing a single placeholder was measured on Qwen2.5-Coder-32B alone: -2.2 pp ($7/3$, $p = 0.344$, $n = 186$). For a worst case, we assume the same shift on every model, charge all of it to the instruction, and remove it from both the leaked term and the total. The leaked share then becomes 81% for Gemma-4-12B-it, 80% for Mistral-Medium-3.5 and 67% for Qwen2.5-Coder-32B, against 83, 82 and 70% as measured. The leaked implementations remain the larger term on all three models.

Interface identifiers. Masking also removes interface information. For each runnable task we collect the identifiers in the target's reference solution and check whether they are visible in the official prompt and in the masked prompts. On 145 tasks at least one disappears under MASKED, and on 107 of them it is still missing under NAMES-ONLY. The neighbor effect is larger on these tasks on 6 of the six endpoints and the paper effect larger on the remaining tasks on 6, but the endpoints share tasks: averaged over endpoints with a joint task bootstrap, the difference is -10.8 pp $[-21.6, +0.6]$ for the neighbor effect and $+5.6$ pp $[-0.4, +11.6]$ for the paper effect, and the split overlaps heavily with nesting. We read it as consistent with, not as evidence for, neighbors supplying the target's interface.

C.4 NESTED AND OUTERMOST TARGETS

Table 7 splits the three contrasts of Section 4.1 by whether the target is the outermost annotated snippet or nested inside another, for all six endpoints. For a nested target, MASKED also masks the rest of the enclosing snippet: across the 126 nested runnable targets (67.7%), 38% of masked lines belong to that snippet (median 57% per task), and on 40 targets every masked line does. On outermost targets every masked neighbor is a separate component. There the neighbor effect remains (six-endpoint average -12.8 pp) but so does a paper effect (-5.8 pp), and the paired contrast between them is not significant on any endpoint, including Claude-Sonnet-5 ($p = 0.0768$), where the two interventions taken separately reach different significance levels. Nesting is nearly collinear with reference length (medians 3 and 6 lines). In a per-task linear probability model of the neighbor effect, stacked over the three main models with model fixed effects and repository-clustered errors, the nesting coefficient is -13.8 pp alone and -5.1 pp ($p = 0.424$) after adding log reference length, so the data do not separate nesting from length.

Table 7: **The three contrasts of Section 4.1 by nesting.** Paired changes in pp (McNemar p), on tasks with all three conditions. The last column is the change from NO-PAPER to MASKED (negative: the neighbors cost more than the paper). Pooled rows average the endpoints, with joint task bootstrap 95% intervals; their n counts distinct tasks.

Endpoint	Target	n	OFFICIAL → NO-PAPER	OFFICIAL → MASKED	NO-PAPER → MASKED
Gemma-4-12B-it	Outer	58	$+3.4$ (0.727)	-8.6 (0.267)	-12.1 (0.118)
Gemma-4-12B-it	Nested	125	-4.0 (0.332)	-20.8 (2×10^{-7})	-16.8 (0.000104)
Mistral-Medium-3.5	Outer	60	-8.3 (0.18)	-15.0 (0.0117)	-6.7 (0.424)
Mistral-Medium-3.5	Nested	125	-0.8 (1)	-30.4 (7×10^{-11})	-29.6 (1×10^{-10})
Qwen2.5-Coder-32B	Outer	60	-13.3 (0.0386)	-10.0 (0.146)	$+3.3$ (0.774)
Qwen2.5-Coder-32B	Nested	126	-5.6 (0.189)	-23.8 (7×10^{-8})	-18.3 (7×10^{-5})
Claude-Haiku-4.5	Outer	60	-8.3 (0.0625)	-8.3 (0.227)	$+0.0$ (1)
Claude-Haiku-4.5	Nested	126	-12.7 (0.00372)	-43.7 (8×10^{-16})	-31.0 (9×10^{-10})
Claude-Sonnet-5	Outer	60	-6.7 (0.424)	-20.0 (0.000488)	-13.3 (0.0768)
Claude-Sonnet-5	Nested	126	-3.2 (0.424)	-54.8 (3×10^{-21})	-51.6 (9×10^{-19})
GPT-6-Astra	Outer	60	-1.7 (1)	-15.0 (0.0117)	-13.3 (0.0574)
GPT-6-Astra	Nested	126	-1.6 (0.625)	-41.3 (4×10^{-16})	-39.7 (2×10^{-15})
three main models	outer	60	-6.1 [-12.8, $+0.3$]	-11.2 [-20.6, -2.8]	-5.1 [-15.1, $+4.5$]
three main models	nested	126	-3.5 [-8.3, $+1.1$]	-25.0 [-31.3, -19.0]	-21.6 [-28.2, -15.1]
six endpoints	outer	60	-5.8 [-10.0, -1.6]	-12.8 [-20.6, -5.8]	-7.0 [-15.1, $+0.6$]
six endpoints	nested	126	-4.6 [-8.1, -1.3]	-35.8 [-41.1, -30.5]	-31.1 [-36.8, -25.7]

Splitting the neighbor mask on nested targets. This experiment asks which part of the neighbor effect on nested targets comes from the target's own enclosing snippet. It uses the nested runnable targets whose file also contains annotated snippets outside the enclosing one (86 targets from 9 repositories, 30 of them from one repository); on the other nested targets the two masks below coincide with existing conditions. We split the lines that MASKED removes into two disjoint parts whose union is the MASKED mask, and verified before generation that the removed characters add up on every task. MASK-ENCLOSING masks the rest of the outermost snippet that encloses the target, including any siblings inside it; MASK-OTHERS masks every other annotated neighbor. Each has its own CONTROL-style length control, which removes the same number of characters from the paper using a process-stable seed. All conditions use the NAMED instruction.

We specified the analysis before running any generation. The primary test is the paired difference between NO-PAPER and MASK-OTHERS, averaged over three models per task, with a repository-cluster bootstrap interval (10,000 resamples, seed 0); the claim that the paper matters less than the method's other components requires its lower bound to exceed zero. We predicted that the interval would include zero. The models are Gemma-4-12B-it and Qwen2.5-Coder-32B (self-hosted, one generation per prompt) and Claude-Haiku-4.5 (three generations, majority vote), replacing Mistral-Medium-3.5, which is no longer served and was not deterministic. All seven Gemma-4-12B-it conditions, OFFICIAL, MASKED and NO-PAPER included, come from one run; the other two models reuse their baseline runs from the same backends. Three Gemma-4-12B-it generations hit the output limit and returned no code, one in each of MASK-ENCLOSING, MASK-OTHERS and MASKED; the paired analysis drops them, and counting them as failures leaves the primary test unchanged, because Gemma-4-12B-it fails all three tasks with and without the paper.

Table 8 gives the result. The primary test fails: masking the other neighbors costs -1.2 pp more than removing the paper ($[-4.0, +2.9]$), and -1.2 pp ($[-4.8, +4.4]$) without the largest repository. The enclosing snippet carries most of the neighbor effect on every model (76–84% of the sum of the two parts), and the two parts roughly add up to MASKED (-28.6 pp pooled). Pooled, the length controls alone change pass rate by less than a point, so the length-adjusted and unadjusted estimates agree. With 9 clusters the bootstrap intervals are likely too narrow, and this split says nothing about outermost targets, which have no enclosing snippet. Splitting the targets at the median reference length (3 lines; not part of the pre-specified analysis) leaves the picture unchanged: the enclosing snippet carries 79% of the effect on short targets and 77% on long ones, and the primary difference is near zero in both (last two rows of Table 8). Because every target here is nested, this split cannot separate nesting from length.

C.5 THE INSTRUCTION ARTIFACT

Table 9 gives the lower block of Table 3 per model, with its discordant pairs.

Table 8: **Splitting the neighbor mask on nested targets.** Columns 3–4: how much more a mask costs than removing the paper, in pp (positive: the mask costs more). Columns 5–6: change from the matched length control to the mask (negative: the mask removes passes). Brackets are repository-cluster bootstrap 95% intervals. Pooled rows average the three models per task; the indented rows restrict the pool.

	n	MASK-OTHERS vs NO-PAPER	MASK-ENCLOSING vs NO-PAPER	Length control → MASK-OTHERS	Length control → MASK-ENCLOSING
Gemma-4-12B-it	84	$+3.6$ $[-4.8, +9.0]$	$+17.6$ $[+8.1, +29.0]$	-4.8 $[-11.1, +1.8]$	-17.6 $[-34.3, -5.7]$
Qwen2.5-Coder-32B	86	-4.7 $[-12.4, +2.3]$	$+10.5$ $[+0.0, +26.5]$	-2.3 $[-7.0, +2.3]$	-20.9 $[-39.2, -4.8]$
Claude-Haiku-4.5	86	-2.3 $[-9.5, +9.3]$	$+19.8$ $[+6.8, +40.6]$	-11.6 $[-21.5, +3.1]$	-31.4 $[-43.3, -21.1]$
pooled	84	-1.2 $[-4.0, +2.9]$	$+16.1$ $[+7.1, +31.7]$	-6.3 $[-11.1, +0.4]$	-23.5 $[-38.4, -12.0]$
without the largest repository	56	-1.2 $[-4.8, +4.4]$	$+20.2$ $[+6.8, +36.2]$	-3.0 $[-8.7, +1.7]$	-25.0 $[-42.6, -9.3]$
short targets ($\leq$ 3 lines)	50	-0.7 $[-4.6, +6.7]$	$+22.0$ $[+10.1, +41.7]$	-6.0 $[-11.0, +1.0]$	-29.3 $[-50.0, -16.0]$
long targets	34	-2.0 $[-7.2, +1.3]$	$+7.6$ $[+2.3, +14.1]$	-6.9 $[-11.7, +1.8]$	-15.2 $[-20.9, -5.6]$

Table 9: **The instruction artifact in the neighbor effect.** Rows one and two are drops from CONTROL; row three is their difference, measured directly as MASKED → MASKED+NAMED. Parentheses give discordant pairs and McNemar's exact p.

	Gemma-4-12B-it (n=184)	Mistral-Medium-3.5 (n=186)	Qwen2.5-Coder-32B (n=186)
Measured under original instruction	-26.1	-38.2	-26.3
True effect (NAMED)	$\mathbf{-17.9}$ (38/5, 2×10^{-7})	$\mathbf{-29.6}$ (56/1, 8×10^{-16})	$\mathbf{-19.9}$ (42/5, 2×10^{-8})
Template artifact	$+8.2$ (1/16, 0.00027)	$+8.6$ (2/18, 0.0004)	$+6.5$ (2/14, 0.0042)
as % of original	31%	23%	24%

The artifact leaves a trace in the responses that needs no pass rates to read. Where more than one `TODO` is visible, MASKED responses contain more than one fenced block 60–86% of the time and the extracted code runs a median of 3.12–7.50 times longer than the reference; under NAMED all three models collapse to a median length ratio of exactly 1.00 and a multi-block rate of 1–7%. The models are answering the question the instruction asks; the extractor is keeping the wrong answer. NAMED removes most but not all of the ambiguity: Gemma-4-12B-it's residual multi-block rate is the highest of the three, which is why Section 4.2 says "largely".

A reader may ask why the tasks that are *not* ambiguous do not serve as the control layer: the 5 runnable tasks that still show a single `TODO` under MASKED should carry no artifact at all, and comparing them against the rest would price it directly. They would, but $n = 5$ cannot distinguish an artifact of 0 from one of 20 pp, which is why we measure the artifact by changing the instruction rather than by subsetting the tasks.

A size band of $[+5, +10]$ pp, fixed after measuring the first two models, failed on the run it was fixed for (Qwen2.5-Coder-32B on the FP8 engine, $+4.8$ pp) and holds on the BF16 regeneration reported here ($+6.5$ pp); the band is about two standard errors of a single estimate wide, so we draw nothing from either outcome.

The artifact is larger on outermost targets (10.0–10.2 pp against 4.8–7.9 on nested ones) while the true effect runs the other way: those are the placeholders a model plausibly answers out of order.

D PAPER × NEIGHBOR INTERACTION AND FULLY MASKED EVALUATION

Both parts of this appendix ask one question: once the neighbors are gone, does the paper matter more? The first answers it on the tasks of the main protocol, the second on a variant in which every annotated snippet is masked.

D.1 THE FOURTH CELL OF THE 2×2, IN FULL

Section 4.4 reports that masking the neighbors does not make the paper load-bearing. The three estimates behind that sentence are $+3.3$, $+3.8$ and -3.8 pp: the paper's contribution grows once the neighbors are gone on two models and *shrinks* on the third. Had we stopped after the first two models, the result would have read as a confirmation, which is our reason for running three rather than two wherever the cost allowed.

What holds on all three is weaker and was observed after the fact: the deletion costs something under masking on every model, and its direction there is one-sided. Only Mistral-Medium-3.5 survives the multiplicity correction of Section 3 (Appendix D.1). We report the prediction and its failure together because a reader deciding whether to adopt the fix should know that its measured benefit is the removal of a substitute channel, not a restored paper effect.

Direction of the paper effect. With the neighbors masked, removing the paper survives the multiplicity correction on Mistral-Medium-3.5 alone (adjusted $p = 0.027$ against 0.785 and 0.803). Under MASKED it moves $38/8$ discordant pairs one way against the other ($p = 9 \times 10^{-6}$); with the neighbors visible the same deletion is less one-sided ($57/33$, $p = 0.0149$). These pooled binomial tests treat model $\times$ task pairs as independent; summing the three models' differences per task and permuting signs by repository gives $p = 0.0654$ and 0.0605, and the Mistral-only result does not survive the repository-level multiplicity correction (Table 12). We looked at direction only after seeing the pooled magnitudes, on 46 discordant pairs, so this is an observation and not a test; we record it because it is the one respect in which masking changes the paper intervention consistently across all three endpoints.

The share-of-passes scale (post hoc). On the scale of Figure 2(b), which we examined only after the fact, the share of passes that fail without the paper rises from 31% with the neighbors visible to 56% with them masked (difference $+24.7$ pp, task bootstrap $[+8.3, +41.0]$, repository-cluster $[+0.2, +40.1]$), in the same direction on all three models but on only 21–25 remaining passes per model. Most of this rise follows from the lower base alone: had removing the paper cost as many passes with the neighbors masked as with them visible, its net share would rise from 13% to 35%, against 44% observed. The two scales agree that the paper's absolute contribution does not grow; its relative weight among the few remaining passes does.

D.2 THE FULLY MASKED VARIANT: A RELEASED RESOURCE WITH KNOWN LIMITATIONS

The task unit is the *outermost* annotated snippet in each file rather than the official task, because 54 of the official snippets (25.5%) have a single line of code as their answer, and a natural-language specification of one line is longer than the line. Taking outermost snippets and dropping the projects whose reference solutions do not pass the official tests leaves 65 parent tasks over 18 projects; the three main models resolve 60, 65 and 65 of them under all four conditions, and every rung is scored on that same set. Two further router endpoints were also run. MiniMax-M3 resolves 64 and is discussed below. Muse-Spark-1.2 shows a neighbor effect of $+15.0$ pp ($p = 0.0703$) from the highest ORACLE rate we measured, 77.5%; we report it here and not in the main text because it resolves only 40 parent tasks, and those are the ones its provider returned a non-empty completion for four times running, which is not a random subset of the 65. We release this set as a resource derived from the diagnosis. Its construction departs from full masking in four ways, which we list because they change what the rungs measure. (i) In STUB, SPEC and SPEC-NO-PAPER each masked neighbor carries a `# Defines:` list of the names it binds, as in NAMES-ONLY (59 of 65 tasks); with the lists removed, many correct reference solutions would fail because the names they use are defined only in masked snippets. (ii) On 44 tasks the target slot also carries a machine-generated `# Must define:` line listing target-bound names that other snippets use, which is information about the answer's interface. (iii) No rung prepends the benchmark's context files, which affects 19 tasks, so ORACLE equals the official prompt only on the others and differs from STUB in more than the visible neighbors. (iv) 5 tasks from a repository whose reference solution fails the official tests are graded with a recalibrated test, an inclusion rule that differs from the runnable set of the main protocol. In addition, the specifications were generated by Gemma-4-12B-it from the full paper and the masked file without the `# Defines:` and `# Must define:` lines, then held fixed across solving endpoints, and no STUB-without-paper condition was run, so the paper rung

(SPEC-NO-PAPER → SPEC) measures the paper's value given a specification already derived from it. Denominators here are parent snippets and are not comparable to the runnable-186 figures used everywhere else in this paper.

Both rungs disagree in sign across the three main models: the paper rung runs -3.3, $+1.5$ and -1.5 pp ($p \geq 0.625$), and the specification rung runs -3.3, $+3.1$ and -1.5 pp. The intervals (Figure 5) permit specification gains of up to 7.9–12.9 pp and paper gains of up to 4.1–6.9 pp (conditional Clopper–Pearson; the task bootstrap gives 6.7–13.8 and 3.3–7.7 pp). The positive control does not behave as one. STUB → ORACLE is $+0.0$ pp on Gemma-4-12B-it ($3/3$), $+13.8$ pp on Mistral-Medium-3.5 ($p = 0.0352$), and $+10.8$ pp on Qwen2.5-Coder-32B ($p = 0.0654$); without the 19 context-file tasks it is $+2.4$, $+6.5$ ($p = 0.508$), and $+8.7$ pp, and the specification rung on Mistral-Medium-3.5 changes sign (-2.2 pp); the paper rung stays between -4.8 and $+2.2$ pp. On Gemma-4-12B-it, removing the `# Must define:` line changes STUB by -3.2 pp ($p = 0.625$) and leaves the positive control at $+3.4$ pp ($p = 0.727$), so the hint does not explain the missing dynamic range there. Resolving a paper effect of 5 pp with McNemar's test at the observed discordance rates would need about 207–240 tasks, and a specification effect of that size 433–578; RCB has 65 runnable outermost parents. We therefore report no finding from this set. On MiniMax-M3, the specification rung (SPEC minus STUB) is $+10.9$ pp ($p = 0.0923$, $n = 64$) and the paper rung $+7.8$ pp ($p = 0.125$). The former is the largest we measured and recovers most of the $+14.1$ pp that endpoint's neighbors supplied; it does not reach significance and its interval ($[-1.5, 18.3]$) contains both zero and a full recovery, so it neither establishes that a written specification replaces the neighbors nor lets us rule it out. We report it because it is a result likely to move with more tasks, and because leaving it out would hide that.

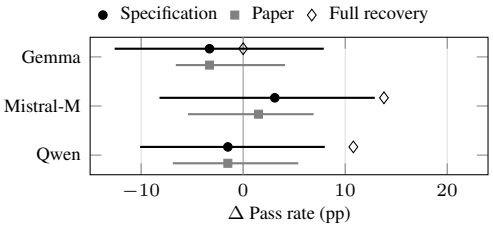


Figure 5: **Specification and paper rungs on the fully masked variant.** Per model, the paired change from replacing the placeholder with a generated specification (STUB → SPEC, circles) and from adding the paper to that specification (SPEC-NO-PAPER → SPEC, squares), with 95% conditional Clopper–Pearson intervals. Diamonds mark STUB → ORACLE, the positive control. Targets are redefined (outermost annotated snippet as the target), so the values are not comparable with Section 4; see the construction caveats in the text.

E STATISTICAL DETAILS

E.1 EXACT CONDITIONAL INTERVALS, AND HOW THEY COMPARE TO THE BOOTSTRAP

Figure 2(a) reports task-level bootstrap intervals. Table 10 gives the exact conditional alternative: a Clopper–Pearson interval (Clopper & Pearson, 1934) on the proportion of discordant pairs falling one way, rescaled by the number of discordant pairs over the number of tasks. It conditions on the observed number of discordant pairs, so its coverage is conditional, and in our data it is systematically narrower where discordance is large: on the paper intervention the two constructions agree to within 1.0 pp at every endpoint ($[-7.1, +3.8]$, $[-9.2, +2.7]$, $[-14.0, -2.2]$ under the bootstrap), but on the neighbor intervention the conditional interval is up to 6.4 pp narrower ($[-23.2, -10.8]$, $[-32.4, -18.9]$, $[-25.8, -12.9]$ under the bootstrap). The two interventions' intervals are disjoint on Gemma-4-12B-it and Mistral-Medium-3.5 under either construction; on Qwen2.5-Coder-32B they are disjoint under the conditional construction and overlap by about a point under the bootstrap, while the paired comparison of the two interventions separates them (11.3 pp, 95% CI $[3.9, 17.1]$, $p = 0.00246$).

Why we compare intervals rather than a ratio. The ratio of the two interventions' point estimates ranges from 2.4 to 10.5× across the three models of Section 4.1. The spread comes from

Table 10: **Every bar of Figure 2(a), with its discordant counts.** Discordant pairs are given as baseline-only / ablation-only, p is McNemar's exact test, and the interval is the conditional Clopper–Pearson construction described above. The bootstrap intervals plotted in the figure are quoted in the paragraph above this table.

		NO-PAPER (remove paper)		MASKED (remove neighbors)	
Model	n	Disc. (p)	95% CI	Disc. (p)	95% CI
Gemma-4-12B-it	184	14/11 (0.69)	$[-7.0, +4.1]$	36/5 (8×10^{-7})	$[-20.4, -10.5]$
Mistral-Medium-3.5	185	19/13 (0.377)	$[-9.1, +3.2]$	49/2 (1×10^{-12})	$[-27.3, -20.1]$
Qwen2.5-Coder-32B	186	24/9 (0.0135)	$[-13.0, -1.6]$	41/5 (4×10^{-8})	$[-22.9, -13.1]$

the denominator: the paper intervention is only a few points in size and its interval includes zero on two of the three models, so the ratio is unstable in a way the interventions themselves are not. Section 4.1 therefore rests on the paired comparison of the two interventions, not on their intervals.

E.2 REPOSITORY-LEVEL INFERENCE

The runnable tasks come from 17 repositories of 3 to 37 tasks each, which share code, tests and a paper. McNemar's test and the task bootstrap are exact or consistent for the tasks of RCB as released; for paper-to-code tasks in general the repository is the sampling unit. We therefore recompute every contrast in three ways: a cluster-robust t test (CR0 with a $\sqrt{J/(J-1)}$ correction and t with 16 degrees of freedom), an exact repository-level sign-flip permutation test that flips the sign of each repository's summed paired difference over all 2^J patterns, and a repository-cluster bootstrap (10^4 resamples of repositories). Table 11 gives the headline contrasts.

Table 11: **Headline contrasts under repository-level inference.** Repository-cluster bootstrap 95% interval in pp, cluster-robust p, and the range of the estimate when any one repository is left out. The paper-versus-neighbor row gives how many more points the neighbor intervention costs, as in Section 4.1.

Contrast	Model	Cluster bootstrap 95%	Cluster-robust p	Leave one repository out
OFFICIAL → NO-PAPER	Gemma-4-12B-it	$[-7.1, +4.0]$	0.557	-2.9 to $+0.0$
	Mistral-Medium-3.5	$[-9.5, +1.0]$	0.221	-4.1 to -1.6
	Qwen2.5-Coder-32B	$[-14.5, -1.8]$	0.0212	-9.4 to -6.7
OFFICIAL → MASKED	Gemma-4-12B-it	$[-25.5, -8.0]$	0.0019	-17.8 to -13.5
	Mistral-Medium-3.5	$[-34.2, -16.5]$	3×10^{-5}	-27.0 to -22.4
	Qwen2.5-Coder-32B	$[-28.4, -10.2]$	0.000697	-21.4 to -16.4
paper vs. neighbor	Gemma-4-12B-it	$[4.9, 25.1]$	0.00778	12.5 to 17.6
	Mistral-Medium-3.5	$[11.9, 30.9]$	0.000269	19.4 to 23.6
	Qwen2.5-Coder-32B	$[3.1, 19.4]$	0.0132	9.4 to 12.7

Pooled over the three models with the paper's fixed inverse-variance weights, the repository-level results are: paper removal -4.1 pp $[-8.7, -0.4]$ ($p = 0.0617$); length-matched wrong paper -7.8 pp $[-13.1, -3.5]$ ($p = 0.00445$); neighbor masking -20.2 pp $[-28.3, -12.3]$ ($p = 0.000154$); the paper-versus-neighbor contrast 15.9 pp $[7.1, 24.1]$ ($p = 0.00178$); and the instruction artifact $+7.7$ pp $[+3.2, +12.7]$ ($p = 0.00579$). The per-task differences are correlated across models because the models share tasks; with the joint bootstrap covariance of the three estimates, the generalized heterogeneity statistic is $Q = 3.5$ ($p = 0.172$) for the paper effect, 6.9 ($p = 0.0316$) for the neighbor effect, and 9.0 ($p = 0.0109$) for their contrast, so the neighbor effect is heterogeneous across models even though Cochran's test with independent-model variances does not detect it (for the paper effect that test gives $Q = 2.55$, $p = 0.28$).

Across the 39-test family (Table 12), 23 tests have unadjusted $p < 0.05$ under the cluster-robust test and 19 under the permutation test, but only 4 and 3 survive Holm correction. The two methods err in opposite directions: with 17 very unequal clusters the cluster-robust test tends to be liberal, while the permutation p-values are discrete, with a smallest attainable value of $2^{1-J'}$ when J' repositories have a non-zero summed difference, so some tests cannot survive a 39-test correction however strong the data. In a family restricted to the three paper-versus-neighbor contrasts, which we define only after seeing the data, the adjusted p are 0.0156, 0.000808, and 0.0156 under the cluster-robust test and 0.0586, 0.0205, and 0.0586 under the permutation test.

E.3 THE MULTIPLICITY FAMILY IN FULL

Section 3 reports a Holm–Bonferroni correction over a family of 39 paired tests, and Table 12 lists every one of them so that the correction can be checked rather than taken on trust. The family crosses every paired contrast run in Sections 4.1–4.4 with the models that carry all of them; each row is one contrast × model pair, tested on the same runnable task set used everywhere else in the paper. Rows are ordered by task-level adjusted p. We correct across the whole family, including the control and interaction tests that we expect to be null, rather than across the headline contrasts alone. The last two columns repeat the correction with repository-level p-values (Appendix E.2).

Table 12: **Every paired test in the multiplicity family.** One row per contrast × model pair. *Raw* p is McNemar's exact test for the paired contrasts and a task-level sign-flip permutation test for the interaction rows; *adjusted* p applies Holm–Bonferroni step-down across all 39 rows of this table. A tick marks the 14 rows with adjusted $p < 0.05$ under task-level inference. *CR* and *Perm.* are the same Holm correction applied to repository-cluster p-values: a cluster-robust t test and an exact repository-level sign-flip permutation test; an asterisk marks adjusted $p < 0.05$ (4 and 3 rows). The fully masked set of Appendix D.2 is not part of this family because it uses a different task unit. Conditions carry the NAMED instruction except in the MASKED → MASKED + NAMED row, which is the instruction contrast itself, and in the CONTROL condition, which keeps the neighbors and so has a single placeholder.

Contrast	Sec.	Model	Raw p	Adj. p	Surv.	CR	Perm.
OFFICIAL → MASKED	§4.1	Mistral-Medium-3.5	1×10^{-12}	5×10^{-11}	✓	0.0013^{*}	0.0381^{*}
CONTROL → NAMES-ONLY	§4.3	Mistral-Medium-3.5	4×10^{-12}	2×10^{-10}	✓	0.002^{*}	0.0452^{*}
NO-PAPER → MASKED (paper vs. neighbor)	§4.1	Mistral-Medium-3.5	6×10^{-9}	2×10^{-7}	✓	0.01^{*}	0.232
OFFICIAL → MASKED	§4.1	Qwen2.5-Coder-32B	4×10^{-8}	2×10^{-6}	✓	0.0251^{*}	0.0879
OFFICIAL → MASKED	§4.1	Gemma-4-12B-it	8×10^{-7}	3×10^{-5}	✓	0.0647	0.258
CONTROL → NAMES-ONLY	§4.3	Gemma-4-12B-it	2×10^{-6}	8×10^{-5}	✓	0.171	0.264
CONTROL → NAMES-ONLY	§4.3	Qwen2.5-Coder-32B	1×10^{-5}	4×10^{-4}	✓	0.0526	0.205
NO-PAPER → MASKED (paper vs. neighbor)	§4.1	Gemma-4-12B-it	3×10^{-5}	8×10^{-4}	✓	0.197	0.762
OFFICIAL → WRONG-PAPER (untruncated)	§4.2	Mistral-Medium-3.5	5×10^{-5}	0.0015	✓	0.171	0.258
MASKED → MASKED + NAMED	§4.2	Gemma-4-12B-it	3×10^{-4}	0.0082	✓	0.197	0.258
MASKED → MASKED + NAMED	§4.2	Mistral-Medium-3.5	4×10^{-4}	0.0117	✓	0.155	0.34
MASKED → MASKED + NO-PAPER	§4.4	Mistral-Medium-3.5	1×10^{-3}	0.0273	✓	1	1
OFFICIAL → WRONG-PAPER (length-matched)	§4.2	Qwen2.5-Coder-32B	0.0013	0.0356	✓	0.0759	0.0381^{*}
NO-PAPER → WRONG-PAPER (untruncated)	§4.2	Mistral-Medium-3.5	0.0015	0.0387	✓	0.103	0.633
NO-PAPER → MASKED (paper vs. neighbor)	§4.1	Qwen2.5-Coder-32B	0.0025	0.0615		0.305	1
MASKED → MASKED + NAMED	§4.2	Qwen2.5-Coder-32B	0.0042	0.1		0.631	1
OFFICIAL → NO-PAPER (system prompt held fixed)	§4.2	Qwen2.5-Coder-32B	0.0052	0.12		0.171	0.355
NAMES-ONLY → MASKED	§4.3	Qwen2.5-Coder-32B	0.0074	0.162		0.241	1
NAMES-ONLY → MASKED	§4.3	Mistral-Medium-3.5	0.0129	0.272		0.613	1
OFFICIAL → NO-PAPER	§4.1	Qwen2.5-Coder-32B	0.0135	0.272		0.467	1
OFFICIAL → WRONG-PAPER (length-matched)	§4.2	Mistral-Medium-3.5	0.0192	0.364		0.88	1
OFFICIAL → WRONG-PAPER (length-matched)	§4.2	Gemma-4-12B-it	0.029	0.521		0.641	1
OFFICIAL → WRONG-PAPER (untruncated)	§4.2	Qwen2.5-Coder-32B	0.0351	0.596		0.64	1
MASKED → MASKED + NO-PAPER	§4.4	Gemma-4-12B-it	0.049	0.785		1	1
NO-PAPER → WRONG-PAPER (length-matched)	§4.2	Gemma-4-12B-it	0.049	0.785		0.186	1
MASKED → MASKED + NO-PAPER	§4.4	Qwen2.5-Coder-32B	0.0574	0.803		1	1
OFFICIAL → WRONG-PAPER (untruncated)	§4.2	Gemma-4-12B-it	0.0931	1		1	1
NAMES-ONLY → MASKED	§4.3	Gemma-4-12B-it	0.109	1		0.601	1
OFFICIAL → NO-PAPER (system prompt held fixed)	§4.2	Gemma-4-12B-it	0.169	1		1	1
NO-PAPER → WRONG-PAPER (length-matched)	§4.2	Mistral-Medium-3.5	0.21	1		1	1
NO-PAPER → WRONG-PAPER (untruncated)	§4.2	Gemma-4-12B-it	0.238	1		1	1
paper × neighbor interaction	§4.4	Qwen2.5-Coder-32B	0.32	1		1	1
paper × neighbor interaction	§4.4	Mistral-Medium-3.5	0.325	1		1	1
OFFICIAL → NO-PAPER (system prompt held fixed)	§4.2	Mistral-Medium-3.5	0.327	1		1	1
OFFICIAL → NO-PAPER	§4.1	Mistral-Medium-3.5	0.377	1		1	1
paper × neighbor interaction	§4.4	Gemma-4-12B-it	0.394	1		1	1
NO-PAPER → WRONG-PAPER (length-matched)	§4.2	Qwen2.5-Coder-32B	0.503	1		1	1
OFFICIAL → NO-PAPER	§4.1	Gemma-4-12B-it	0.69	1		1	1
NO-PAPER → WRONG-PAPER (untruncated)	§4.2	Qwen2.5-Coder-32B	0.815	1		1	1

F GENERALIZATION AND ENDPOINT ROBUSTNESS

This appendix asks whether the findings depend on the benchmark or on the endpoints: the same conditions on a second benchmark, which runs can be compared with which, and the endpoints themselves, including the three commercial ones.

Table 13: **The four conditions on SciReplicate-Bench.** Paired changes with McNemar's exact test, the same presentation as the main tables. Pass rate is given as before → after; discordant pairs are (only-before passes / only-after passes).

Contrast	Model	n	Pass rate	Δ (pp)	95% CI	Disc.	p
CONTROL → MASKED Same-file neighbors	Gemma-4-12B-it	56	23.2 → 17.9	-5.4	$[-8.8, +3.9]$	4/1	0.38
	Qwen2.5-Coder-32B	56	21.4 → 16.1	-5.4	$[-8.8, +3.9]$	4/1	0.38
	Claude-Haiku-4.5	58	32.8 → 25.9	-6.9	$[-6.9, +1.4]$	4/0	0.12
CONTROL → MASKED All tasks	Gemma-4-12B-it	78	17.9 → 14.1	-3.8	$[-6.3, +2.8]$	4/1	0.38
	Qwen2.5-Coder-32B	78	16.7 → 12.8	-3.8	$[-6.3, +2.8]$	4/1	0.38
	Claude-Haiku-4.5	80	27.5 → 20.0	-7.5	$[-7.5, -0.6]$	6/0	0.031
OFFICIAL → NO-PAPER All tasks	Gemma-4-12B-it	78	15.4 → 16.7	$+1.3$	$[-6.6, +8.4]$	4/5	1
	Qwen2.5-Coder-32B	78	17.9 → 14.1	-3.8	$[-9.8, +4.6]$	6/3	0.51
	Claude-Haiku-4.5	80	21.2 → 26.2	$+5.0$	$[-3.8, +10.8]$	3/7	0.34

F.1 THE SAME FOUR CONDITIONS ON A SECOND BENCHMARK

We re-ran the four prompt conditions on SciReplicate-Bench (Xiang et al., 2025), whose tasks are built from a different paper set and graded by a different harness. Of its 100 published tasks, the authors' reference solution passes in our environment on 82, and those are the tasks we generate for; a further 2 of them fail the reference check at final grading and are dropped from all four conditions. The neighbor contrast (CONTROL → MASKED) was specified before grading, on the 59 tasks that have at least one other benchmark task in the same file, because for the remaining tasks the two prompts are identical by construction; the whole-set version is reported alongside it. The paper contrast (OFFICIAL → NO-PAPER) uses all tasks. Per-model denominators differ and are printed in the table; holding the task set fixed across models instead changes no direction (Table 14). Intervals are the exact conditional intervals of Appendix E.1, which is why a direction with no discordant pairs against it puts one bound at the point estimate.

What replicates, and what does not. The neighbor contrast points the same way for every model. On the pre-specified same-file subset, masking the neighbors costs -5.4 pp for Gemma-4-12B-it, -5.4 pp for Qwen2.5-Coder-32B and -6.9 pp for Claude-Haiku-4.5, and no model gains. None of the three reaches significance ($p = 0.38$, 0.38 and 0.12). We fixed the replication criterion in advance as *both* the direction holding and at least two models reaching $p < 0.05$. That criterion fails here. The direction agrees on every model: consistent estimates, underpowered evidence.

The discordant column says why. McNemar's exact test only sees tasks that flip between the two conditions, and here that is 4/1, 4/1 and 4/0 pairs. With a handful of flips the exact test cannot return $p < 0.05$ at all unless at least six of them land on the same side, however large the underlying effect is. This is a property of the benchmark's size, not evidence that the effect is absent, and it is the reason we treat ResearchCodeBench rather than SciReplicate-Bench as the setting that carries the argument.

The whole-task version of the same contrast is larger for Claude-Haiku-4.5 (-7.5 pp, $p = 0.031$) and does cross the threshold. We do not count it. The same-file subset is the analysis we specified, and moving to whichever task set turned out significant is the specific failure that fixing the contrast in advance was meant to rule out; we print it because it belongs in the table, not as support for a claim.

The paper contrast. Removing the paper moves the pass rate by $+1.3$ pp for Gemma-4-12B-it, -3.8 pp for Qwen2.5-Coder-32B and $+5.0$ pp for Claude-Haiku-4.5, with intervals $[-6.6, +8.4]$, $[-9.8, +4.6]$ and $[-3.8, +10.8]$. The three do not agree on a sign and none is significant. We specified this contrast for estimates and intervals only and draw no conclusion from it; we report it so that the four conditions are shown on both benchmarks, not only where they agree.

F.2 WHICH RUNS ARE COMPARABLE WITH WHICH

Every effect in this paper is a paired difference between two conditions of the same block of Table 15. Inside a block the endpoint, the serving backend, the decoding settings, the repetition protocol and

Table 14: **Sensitivity: only tasks that all three models have a pair for.** A few tasks exceed the context window of the two open-weight models under the three conditions that include the paper, so the denominators in Table 13 differ by model; this table holds the task set fixed instead. Cells are Δ (pp) with (discordant pairs, p).

Contrast	n	Gemma-4-12B-it	Qwen2.5-Coder-32B	Claude-Haiku-4.5
CONTROL → MASKED, same-file	56	-5.4 (4/1, $p = 0.38$)	-5.4 (4/1, $p = 0.38$)	-7.1 (4/0, $p = 0.12$)
CONTROL → MASKED, all	78	-3.8 (4/1, $p = 0.38$)	-3.8 (4/1, $p = 0.38$)	-7.7 (6/0, $p = 0.031$)
OFFICIAL → NO-PAPER	78	$+1.3$ (4/5, $p = 1$)	-3.8 (6/3, $p = 0.51$)	$+5.1$ (3/7, $p = 0.34$)

the masker version are all fixed, so they cancel in the difference. Across blocks these factors do not cancel, and we never subtract across them: a serving change alone moves a pass rate by a few points (Appendix A). The endpoint, backend and masker columns are recorded by hand, because the serving configuration is not written into the generation records, which is itself the reason this table exists. The remaining columns are measured from those records by the script that emits the table. Where a cell reads *not recorded*, no file in our repository states it, and we say so rather than reconstruct it.

Table 15: **Every generation run behind the paper.** One row per block and instruction generation. *Masker* is the neighbor masker in the tree when the run was made; the corrected one splits an enclosing neighbor around the target (Appendix A). Task counts are in parentheses.

Block	Endpoint	Backend	Instr.	Masker	n_{rep}	Conditions (tasks)
G0	Gemma-4-12B-it	self-hosted (vLLM), bf16 KV cache, T=0, seed 0	original	earlier	3	with_paper (212), wo_paper (212)
G1	Gemma-4-12B-it	self-hosted (vLLM), bf16 KV cache, T=0, seed 0	original	earlier	1	scaffold_ctrl (212), scaffold_masked (212)
G2	Gemma-4-12B-it	self-hosted (vLLM), bf16 KV cache, T=0, seed 0	names-only	corrected	1	scaffold_api_named (212), scaffold_masked_named (212), scaffold_masked_wo_paper_named (212), with_paper_named (212), wo_paper_keepframe_named (212), wo_paper_named (212), wrong_paper_lenmatch_named (212), wrong_paper_named (212)
			original	corrected	1	scaffold_api (212), scaffold_ctrl (212), scaffold_masked (212)
G3	Gemma-4-12B-it	self-hosted (vLLM), bf16 KV cache, decoding not recorded	original	leak-free set	1	oracle_paper (65), spec_nopaper (65), spec_paper (65), stub_ambig (65), stub_nomd (65), stub_paper (65)
G4	Gemma-4-12B-it	self-hosted (vLLM), bf16 KV cache, T=0, seed 0	names-only	corrected	1	mask_enclosing_named (91), mask_others_named (91), scaffold_ctrl_enc_named (91), scaffold_ctrl_oth_named (91), scaffold_masked_named (91), with_paper_named (91), wo_paper_named (91)
M1	Mistral-Medium-3.5	shared router, T=0, seed 0	original	earlier	1	scaffold_ctrl (212), scaffold_masked (212), with_paper (212)

Table 15, continued

Block	Endpoint	Backend	Instr.	Masker	n_{rep}	Conditions (tasks)
M2	Mistral-Medium-3.5	shared router, T=0, seed 0	names-only	corrected	1	scaffold_api_named (212), scaffold_masked_named (212), scaffold_masked_wo_paper_named (212), with_paper_named (212), wo_paper_keepframe_named (212), wo_paper_named (212), wrong_paper_lenmatch_named (212), wrong_paper_named (212)
			original	corrected	1	scaffold_api (212), scaffold_ctrl (212), scaffold_masked (212)
M3	Mistral-Medium-3.5	shared router, decoding not recorded	original	leak-free set	1	oracle_paper (65), spec_nopaper (65), spec_paper (65), stub_paper (65)
Q1	Qwen2.5-Coder-32B	self-hosted (vLLM), fp8 KV cache, T=0, seed 0	original	earlier	1	scaffold_ctrl (212), scaffold_masked (212), with_paper (212), wo_paper (212)
Q2	Qwen2.5-Coder-32B	self-hosted (vLLM), fp8 KV cache, T=0, seed 0	names-only	corrected	1	scaffold_masked_named (212), scaffold_masked_wo_paper_named (212), with_paper_named (212), wo_paper_keepframe_named (212), wo_paper_named (212), wrong_paper_lenmatch_named (212), wrong_paper_named (212)
Q3	Qwen2.5-Coder-32B	self-hosted (vLLM), bf16 KV cache, T=0, seed 0	names-only	corrected	1	mask_enclosing_named (91), mask_others_named (91), scaffold_api_named (212), scaffold_ctrl_enc_named (91), scaffold_ctrl_oth_named (91), scaffold_masked_named (212), scaffold_masked_wo_paper_named (212), with_paper_named (212), wo_paper_keepframe_named (212), wo_paper_named (212), wrong_paper_lenmatch_named (212), wrong_paper_named (212)
			original	corrected	1	scaffold_ctrl (212), scaffold_masked (212), with_paper (212), wo_paper (212)
Q4	Qwen2.5-Coder-32B	self-hosted (vLLM), bf16 KV cache, decoding not recorded	original	leak-free set	1	oracle_paper (65), spec_nopaper (65), spec_paper (65), stub_paper (65)
H	Claude-Haiku-4.5	commercial API, not sent	original	corrected	3	scaffold_ctrl (212)
			names-only	corrected	3	mask_enclosing_named (91), mask_others_named (91), scaffold_ctrl_enc_named (91), scaffold_ctrl_oth_named (91), scaffold_masked_named (212), with_paper_named (212), wo_paper_named (212)
S	Claude-Sonnet-5	commercial API, not sent	original	corrected	3	scaffold_ctrl (212)
			names-only	corrected	3	scaffold_masked_named (212), with_paper_named (212), wo_paper_named (212)
A	GPT-6-Astra	commercial API, reasoning effort low, not sent	names-only	corrected	3	scaffold_masked_named (212), scaffold_masked_wo_paper_named (212), with_paper_named (212), wo_paper_named (212)
R1	Agnes-2.5-Flash	shared router, T=0, seed 0	original	earlier	3	scaffold_ctrl (212), scaffold_masked (212), with_paper (212)

Table 15, continued

Block	Endpoint	Backend	Instr.	Masker	n_{rep}	Conditions (tasks)
R2	Agnes-2.0-Flash	shared router, T=0, seed 0	original	earlier	3	scaffold_ctrl (212), scaffold_masked (212), with_paper (212)
R3	Mistral-Large	shared router, T=0, seed 0	original	earlier	1	scaffold_ctrl (212), scaffold_masked (212), with_paper (212)
R4	MiniMax-M3	shared router, decoding not recorded	original	leak-free set	1	oracle_paper (65), spec_nopaper (65), spec_paper (65), stub_paper (65)
R5	Muse-Spark-1.2	shared router, decoding not recorded	original	leak-free set	1	oracle_paper (65), spec_nopaper (65), spec_paper (65), stub_paper (65)

F.3 MODEL ENDPOINTS

Table 16 maps the model names used in the text to the identifiers we queried. Two of the six main endpoints, Agnes-2.5-Flash and Agnes-2.0-Flash, are exposed by the router under vendor-obscured names; the provider does not disclose which models back them.

Table 16: **Model endpoints.** Names used in the text and the identifiers we queried. "Router" is one shared third-party API router.

Name in text	Identifier	Access	Used in
Gemma-4-12B-it	`gemma-4-12B-it`	self-hosted (vLLM)	all sections
Qwen2.5-Coder-32B-Instruct	`Qwen2.5-Coder-32B-Instruct`	self-hosted (vLLM)	all sections
Mistral-Medium-3.5	`mistral-medium-3-5`	router	all sections
Mistral-Large	`mistral-large`	router	Table 17
Agnes-2.5-Flash	`agnes-2.5-flash`	router	Table 17
Agnes-2.0-Flash	`agnes-2.0-flash`	router	Table 17
MiniMax-M3	`minimax-m3-free`	router (no-cost tier)	variant only
Muse-Spark-1.2	`muse-spark-1.2`	router	variant only
Claude-Haiku-4.5	`claude-haiku-4-5`	commercial API	App. F.5
Claude-Sonnet-5	`claude-sonnet-5`	commercial API	App. F.5
GPT-6-Astra	`gpt-6-astra`	commercial API	App. F.5

We report them only as a robustness check, and separately, because their protocol differs from the rest ($n_{\text{repeat}} = 3$ with a majority vote, since the router is not deterministic).

A deployment fingerprint taken from a single prompt was identical for the two endpoints, which initially suggested that one backend was being served under two names. The full-corpus comparison did not support that conclusion. Across all 212 tasks the two endpoints produced verbatim-identical output on 11.3% of them. That is inside the 9.0–18.4% band we measure between endpoints known to be distinct (13.7% for Mistral-Large against Mistral-Medium-3.5, 9.0% and 18.4% against a self-hosted Qwen model), and in fact below the highest of them: not the level expected of one backend served twice. We report this because the cheap check gave the wrong answer: two different models agree by default on short, low-entropy completions, so on a shared router only a full-corpus verbatim comparison separates aliasing from coincidence. The check costs nothing beyond two runs that were being made anyway.

Serving configuration for Qwen2.5-Coder-32B. Partway through this work the vLLM instance serving Qwen2.5-Coder-32B changed its KV-cache precision from FP8 to unquantized BF16. The serving configuration is part of the experimental condition, so we regenerated every Qwen2.5-Coder-32B condition on the new engine; all Qwen2.5-Coder-32B figures in this paper come from it. Before switching we fixed an acceptance rule: each condition's pass rate may move by at most 3 pp between engines, and a move above 5 pp triggers an audit of the grading environment. 9 of the 10 conditions stayed within 2.7 pp. The validity-anchor condition of Appendix A.1 did not: it moved from 29.6 to 32.8% ($+3.2$ pp; discordant $1/7$, $p = 0.0703$), above the first threshold and below the second. We audited the environment regardless. A broken environment changes the verdict on code that did not change. The two engines generated byte-identical code in 232 of 372 runnable task–condition cells, and 1 of those changed verdict. We therefore attribute the shift to the engine, and switched although the condition exceeded the threshold.

The rule bounded each condition separately, while our claims are differences between conditions, and a difference can move by up to twice the per-condition bound. Qwen2.5-Coder-32B's paper intervention (Section 4.1) was -5.4 pp ($p = 0.121$) under the old engine and is -8.1 pp ($p = 0.0135$) under the new one; the validity anchor's disagreements with the leaderboard on the runnable subset were $3/7$ ($p = 0.34$) and are $2/12$. Both shifts are reported as measured. With the FP8 value for Qwen2.5-Coder-32B, the pooled paper effect of Section 4.1 would be -3.2 pp (independent-model $p = 0.0574$, repository-cluster $p = 0.0968$) instead of -4.1 pp ($p = 0.016$ and 0.0617): the only nominally significant per-model paper effect, and the nominal significance of the pooled one, both depend on the engine. We do not fold this into the intervals; we report the paper effect under both engines instead. The neighbor effect is unaffected at this resolution.

Because Qwen2.5-Coder-32B's paper intervention is two discordant pairs from the conventional threshold, we re-ran its three NAMED conditions twice more and took a majority vote, a sensitivity analysis whose reporting we fixed before seeing it; the main text keeps the single-generation protocol of Section 3. Of the 558 task–condition cells, 8 had non-identical verdicts across the three generations and the vote overturned the first generation in 4. With the vote, removing the paper gives -8.6 pp ($25/9$, $p = 0.00904$), removing the neighbors -20.4 pp ($42/4$, $p = 5.1e-9$), and the neighbors exceed the paper by 11.8 pp ($p = 0.00126$): the same conclusions as the single-generation figures.

F.4 PER-ENDPOINT NEIGHBOR EFFECT

Table 17: **The ablated condition is flat.** Six endpoints, all under the original instruction and all but Qwen2.5-Coder-32B under the earlier masker, so that they are as comparable as possible. Qwen2.5-Coder-32B's CONTROL and MASKED use the corrected masker, which moves MASKED by about a point where both were run on one engine (Appendix A.3). All three rate columns are measured; the last column is their difference, taken before rounding. Across a $2.1\times$ range of unablated performance the MASKED pass rate stays within 3.8–5.9%. † marks the two vendor-obscured endpoints.

Endpoint	OFFICIAL	CONTROL	MASKED	CONTROL → MASKED
Agnes-2.5-Flash†	19.4%	15.6%	4.8%	−10.8
Agnes-2.0-Flash†	27.4%	26.3%	4.8%	−21.5
Qwen2.5-Coder-32B	32.8%	31.2%	4.8%	−26.3
Gemma-4-12B-it	30.8%	28.6%	3.8%	−24.9
Mistral-Large	36.6%	39.8%	5.9%	−33.9
Mistral-Medium-3.5	41.4%	40.3%	3.8%	−36.6

All six numbers were measured under the *original* instruction, the only instruction for which every endpoint has data, and all but Qwen2.5-Coder-32B under the earlier masker; under the corrected protocol the three re-run models give -17.9, -29.6 and -19.9, with a MASKED+NAMED condition at 13.0, 13.4 and 11.3%. Where both maskers were run on one engine, for two of the six, the corrected one raises the MASKED condition by about a point and raises CONTROL with it (since CONTROL's length matching is computed against MASKED), moving those two drops to -26.1 and -38.2; it does not change the flatness of the MASKED column, which is what the table is for.

F.5 FRONTIER ENDPOINTS

The six endpoints above, which include the three of Section 4.1, sit in the middle-to-lower part of the RCB leaderboard, which leaves open what happens at the top. We therefore re-ran the headline contrast (NO-PAPER and MASKED, both under the corrected NAMED instruction, with the same splicer, grader and runnable denominator) on two commercial endpoints reached through one shared backend, with extended thinking disabled and $n_{\text{repeat}} = 3$ majority voting: the weaker endpoint returns verbatim-identical output on only 23.8% of task–condition cells and the stronger on 47.9%, so a single sample is not enough on either.

Read in isolation, the weaker endpoint suggests a trend with capability: its paper effect, -11.3 pp ($p = 0.00032$), is the largest we have measured and lies outside the range of Section 4.2. Leaving out its two largest contributors (`eomt`, then `schedule_free`) moves it to -8.1 and -7.4 pp ($p = 0.0169$, 0.0213; $n = 149$, 135) and it stays significant, so it is not one project's artifact. The stronger endpoint then reverses the pattern: -4.3 pp, not significant, bounded above at 9.4 pp.

Table 18: **The paper intervention stays below the neighbor intervention; the neighbor intervention is the largest we measured.** All three endpoints, $n = 186$ runnable tasks, paired McNemar against the same-endpoint NAMED baseline; each cell gives the drop with its discordant pair and p. The stronger endpoint is near the top of the leaderboard (over all 212 tasks its rate falls just below the strongest listed entry, 64.2%, since the excluded tasks fail under every condition in our environment) and yet shows the *smaller* paper effect, while its neighbor effect is the largest we have measured anywhere, with not one task in the → MASKED column passing after masking that failed before it. Claude-Haiku-4.5 and Claude-Sonnet-5 can be compared directly with each other, but not subtracted from endpoints run on other backends, because they share a backend with each other and with no other condition; GPT-6-Astra runs on a backend of its own and is read only within its row (Appendix F.2). We did not run CONTROL on GPT-6-Astra. The last column is the length-matched neighbor effect, the same CONTROL → MASKED quantity reported for the other endpoints in Table 17 (62/4 and 83/3, $p = 2\times10^{-14}$ and 3×10^{-21}).

Endpoint	Baseline	→ NO-PAPER	→ MASKED	CONTROL → MASKED
Claude-Haiku-4.5	50.0%	−11.3 (27/6, 0.00032)	−32.3 (64/4, 6×10^{-15})	−31.2
Claude-Sonnet-5	70.4%	−4.3 (18/10, 0.18)	−43.5 (81/0, 8×10^{-25})	−43.0
GPT-6-Astra	80.1%	−1.6 (5/2, 0.45)	−32.8 (62/1, 1×10^{-17})	–

These two endpoints do not establish a trend in either direction. The neighbor intervention does not reverse: the neighbor effect is larger on the stronger of these two, and on both endpoints the paired comparison of Section 4.1 separates the two interventions directly, by 21.0 and 39.2 pp ($p = 3\times10^{-7}$ and 6×10^{-18}). We draw no trend from that either, and none from the six of Table 17, where a drop that grows with the baseline is what a flat ablated condition produces by arithmetic. Because those six use the original instruction, we do not place them on a common axis with the two frontier endpoints.

The neighbor effect is not a prompt-length effect. Masking the neighbors shortens the prompt, so we also ran the length-matched CONTROL condition on both endpoints, deleting the same number of characters from the paper instead of from the file. Neither endpoint moves: CONTROL scores 48.9% and 69.9% against baselines of 50.0% and 70.4% (−1.1 and −0.5 pp, $p = 0.815$ and 1), so the length-matched drops in the last column of Table 18 are within a point of the raw ones. On the three endpoints of Section 4.1 CONTROL lands *above* baseline instead, so there the length-matched drops of Table 9 are no smaller than the raw ones of Figure 2. In both cases, removing this much paper text has no measurable effect, which is what the control is meant to establish.

A third commercial endpoint. We added GPT-6-Astra after the two endpoints above, through its vendor's API with reasoning effort set to low, the lowest setting the endpoint accepts. The endpoint ignores temperature and seed and returns verbatim-identical output on only 57.5% of task–condition cells, so we again use $n_{\text{repeat}} = 3$ majority voting. It has the highest baseline of any endpoint we measured, 80.1%, and the smallest paper effect: removing the paper lowers the pass rate by 1.6 percentage points ($p = 0.45$), while masking the neighbors lowers it by 32.8; compared directly, the two interventions differ by 31.2 percentage points ($p = 5\times10^{-15}$). Of its 149 baseline passes, 1.3% fall in the paper-only category of Figure 3. On this endpoint we also ran the cell that removes the paper after masking the neighbors. From 47.3% the pass rate drops by a further 4.8 percentage points (14/5, $p = 0.064$), which is 16% of the passes that survive masking. Even with its neighbors masked, this endpoint's successes mostly do not depend on the paper.

F.6 THE SOURCES OF A PASS, PER ENDPOINT

A same-prompt noise floor for Figure 3. A pass can be lost by re-running an identical prompt. We measure this floor as the share of OFFICIAL passes that fail on another generation of the same prompt: 3.9% for Gemma-4-12B-it (from repeated original-instruction generations, the larger of its two conditions), 2.9% for Qwen2.5-Coder-32B (three NAMED generations), 11.6%, 4.7%, and 1.1% for the three commercial endpoints; Mistral-Medium-3.5 has no repeated generations under this protocol. *Paper only* exceeds this floor on Gemma-4-12B-it (7.4%) and Qwen2.5-Coder-32B (12.3%), sits at it on GPT-6-Astra (1.3%), and falls below it on Claude-Haiku-4.5 (6.5%) and Claude-Sonnet-5 (3.8%); *Neighbors only* exceeds every floor by at least 35 pp. The raw ratio of the two categories, at least 3.4× on every endpoint, therefore overstates the case where *Paper only* is within noise. The

commercial floors come from single generations while Figure 3 uses a three-generation majority vote, so they are conservative there.

The sources of a pass, per endpoint. Table 19 gives the cross-classification behind Section 4.1's claim that the paper is rarely the sole source of a pass, for each endpoint separately. Each row is computed within one endpoint, so the rows are six separate replications rather than one pooled figure, and no two of them are subtracted. They share the task set, so they are not statistically independent of one another; what differs between rows is the model and its generations.

Table 19: **What each pass needed, in counts and percentages.** The cells behind Figure 3. Tasks passed with everything visible, classified by whether the pass survives deleting the paper and whether it survives masking the neighbors. Rows are percentages of that endpoint's passes. The first three rows are the main endpoints of Section 4.1, the last three the commercial endpoints; every row is computed inside one endpoint.

Endpoint	n	Passed	Neither	Neighbors only	**Paper only**	Both
Gemma-4-12B-it	183	54	25.9%	48.1%	**7.4%**	18.5%
Mistral-Medium-3.5	185	72	23.6%	50.0%	**8.3%**	18.1%
Qwen2.5-Coder-32B	186	57	15.8%	42.1%	**12.3%**	29.8%
Claude-Haiku-4.5	186	93	24.7%	46.2%	**6.5%**	22.6%
Claude-Sonnet-5	186	131	34.4%	51.9%	**3.8%**	9.9%
GPT-6-Astra	186	149	57.0%	39.6%	**1.3%**	2.0%

The "Both" column should not be over-read: it counts passes that neither deletion survives, which includes tasks where the two sources are redundant with each other and tasks where each supplies a different necessary piece. The column that carries the argument is the pair beside it, and the asymmetry there is the same asymmetry the headline reports, counted in tasks rather than in points.