

Hidden Dependencies in LLM-Enabled Code Generation: An Empirical Study of Regressions and Escaped Failures

Gyumin Nam 

Department of Computer Engineering, Hankyong National University, Anseong 17579, Republic of Korea

Geunseok Yang¹ 

Department of Computer Engineering, Hankyong National University, Anseong 17579, Republic of Korea

Abstract

Background. Regression testing has traditionally focused on source-code changes and well-specified software dependencies such as compiler versions, library versions, and operating-system platforms. However, LLM-enabled code-generation systems also depend on non-code artifacts (the model, the prompt template, and the retrieval corpus) that can change independently of the application code. Updates to any of these artifacts can silently regress previously passing tasks, and the lightweight partial-test oracles commonly used in continuous evaluation may not detect such failures. Despite the practical relevance, paired regression rates, measurement validity, and escaped-failure rates under controlled non-code dependency changes have received little empirical attention.

Aims. This work empirically analyzes how changes to models, prompts, and retrieval data affect the behavior of LLM-enabled code-generation systems, framing the problem as dependency-aware regression testing rather than a model or retrieval comparison. The questions addressed are failure-rate sensitivity, paired-regression frequency under the answer-bearing canonical retrieval condition, the measurement-validity decomposition of canonical regression signals (answer-bearing baseline rescue vs. non-canonical retrieval sensitivity), and how often generations pass a lightweight oracle yet fail a stronger oracle, reported over all generations and, conditionally, among stronger-oracle failures.

Method. We designed a controlled empirical study that treats models, prompts, and retrieval data as software dependencies of an LLM-enabled code-generation pipeline. Starting from a fixed baseline, we varied one dependency dimension at a time across three open-weight 7B-class code LLMs, three prompt templates, and four retrieval-data conditions (one being a synthetic stress test), evaluated on 120 CodeRAG-Bench HumanEval tasks under three random seeds (2,880 generations across eight conditions). For each (task, condition, seed) triple, we recorded syntactic validity and per-assertion execution outcomes to derive failure rate, paired regression rate, escaped-failure rate, and across-seed flakiness, and compared each changed condition to the baseline using paired McNemar tests.

Results. In the answer-bearing canonical benchmark condition, removing the retrieved document regressed 11.67% of previously passing tasks (D_1 , $p = 1.22 \times 10^{-4}$, stable under task-level Bonferroni correction). Replacing it with a different-task mismatched document regressed 9.17% (D_2 , $p = 9.77 \times 10^{-4}$, significant before correction but not after; reported as an auxiliary stressor). A copy-template audit showed that the M_0 baseline output was a normalized byte-level copy of the canonical retrieved document in 331 of 360 generations, and 12 of 14 D_1 -regressed tasks exhibited this property. A title-based self-excluded BM25 proxy and an answer-stripped D_0 check showed that most of this canonical signal was due to answer-bearing baseline rescue (BM25 D_1 : 3 regressions / 3 recoveries; D_2 : 3 / 4); thus, harmful non-canonical retrieval degradation was not established in this sample. The intentionally permissive O_1 probe let 36/360 D_1 and 26/360 D_2 generations pass despite failing O_2 ; an order-dependent first-three-substantive variant reduced these counts to 8/360 and 3/360. Model and prompt perturbations are reported as framework coverage checks rather than causal claims.

¹ Corresponding Author



© Gyumin Nam and Geunseok Yang;
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Empirical Software Engineering and Measurement (ESEM 2026).

Editors: Robert Feldt, Maria Paasivaara, Daniel Mendez, Stefan Wagner, and Marvin Muñoz Barón;
Article No. 51; pp. 51:1–51:20



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Conclusions. The strongest apparent regression signal arose from the answer-bearing canonical retrieval condition, but validity analyses showed that this signal primarily reflected a benchmark-induced copy-template dependency rather than an established deployment retrieval risk. These findings support dependency-aware paired regression testing and layered oracles, while cautioning that canonical answer-bearing retrieval regression rates should be interpreted as benchmark-specific stress signals rather than deployment-risk estimates.

2012 ACM Subject Classification Software and its engineering → Software testing and debugging; Software and its engineering → Software verification and validation; Computing methodologies → Artificial intelligence

Keywords and phrases LLM-enabled code generation, regression testing, retrieval-augmented generation, test oracles, empirical software engineering

Digital Object Identifier 10.4230/LIPIcs.ESEM.2026.51

Category Technical Track Paper

Supplementary Material

Software (Replication Package): <https://doi.org/10.5281/zenodo.21988303>

Funding This work was supported by the National University Development Project of Hankyong National University in 2026.

1 Introduction

Regression testing asks whether a change after a known baseline breaks behavior that previously passed [39]. In conventional software, the changed artifact is usually application source code or a pinned dependency such as a compiler, library, runtime, or operating-system version. LLM-enabled code-generation systems add less visible dependencies: model checkpoints, prompt templates, retrieval indexes, retrieval corpora, and oracle definitions. Any of these artifacts can change while the application code that invokes the generator remains unchanged. Prior work on hidden technical debt and software engineering for machine learning showed that ML behavior depends on data, configuration, and pipeline choices that are not always visible in source code [34, 2]. In code generation, these dependencies also appear at inference time and can become sources of silent behavioral regression.

This paper studies that problem for retrieval-augmented code generation (RACG). Existing code-generation evaluations usually report aggregate pass rates such as pass@k on fixed benchmark configurations, including HumanEval and extended-oracle variants such as EvalPlus [9, 26]. RACG benchmarks and systems further evaluate whether retrieved context improves generated code quality [24, 35]. These evaluations are valuable, but fixed-configuration scores do not show which previously passing tasks fail after one dependency changes, whether an observed retrieval effect is a deployment-relevant dependency or a benchmark artifact, or how often a weak oracle accepts a generation that a stronger oracle rejects. We therefore frame the study as dependency-aware regression testing rather than as a model leaderboard or a generic retrieval comparison.

The key measurement-validity issue is that the benchmark retrieval context can contain the answer. In the CodeRAG-Bench HumanEval setting used here, the canonical D_0 document contains the target entry-point definition and canonical solution body for all 120 selected tasks. Thus, D_0 is not a deployment-realistic retrieval condition; it is an answer-bearing canonical benchmark condition. If a model copies that document, removing or replacing it can create an apparent paired regression by removing the answer template rather than by

revealing harmful non-canonical retrieval-corpus sensitivity. For this reason, we interpret canonical retrieval numbers as benchmark-specific stress signals and decompose them with answer-stripped and self-excluded retrieval baselines.

We represent an LLM-enabled code-generation pipeline as a configuration $C = (M, P, D)$ and change one component at a time. The main experiment varies three open-weight 7B-class code models, three prompt templates, and four retrieval-data conditions over 120 CodeRAG-Bench HumanEval tasks and three seeds. Each generation is evaluated under a permissive partial visible oracle (O_1), the full HumanEval visible oracle (O_2), and an EvalPlus-derived extended oracle (O_3). We report generation-level failure rates, task-level paired regressions and recoveries, escaped failures between oracle tiers, and across-seed flakiness, using paired McNemar tests as the main inferential analysis.

Removing or replacing the answer-bearing canonical document regresses 14/120 and 11/120 tasks, respectively, but the baseline copies that document in 331/360 generations. A title-based self-excluded BM25 proxy yields small, bidirectional changes (3/3 and 3/4 regressions/recoveries), while removing only the canonical solution body reproduces 11/14 removal regressions. Thus, the strongest signal is a benchmark-induced copy-template dependency; harmful non-canonical retrieval degradation is not established in this sample.

This study makes three contributions.

- **Measurement-validity decomposition of answer-bearing retrieval.** We show how an answer-bearing canonical retrieval context can inflate apparent retrieval-regression rates, and we decompose the effect using a copy-template audit, BM25 self-excluded retrieval, and an answer-stripped D_0 mechanism check.
- **Dependency-aware paired regression-testing framework.** We formulate model checkpoints, prompt templates, retrieval contexts, retrieval corpora, and oracle tiers as versioned behavioral dependencies, and we evaluate one-factor-at-a-time changes with task-level paired regressions and recoveries.
- **Oracle-tier visibility analysis.** We quantify when dependency-induced failures pass a permissive partial oracle but fail a full visible oracle, and when visible-test-passing generations fail an EvalPlus-derived extended oracle.

The remainder of the paper reviews background (Section 2), formalizes the framework and analysis plan (Section 3), reports the experiments and validity decomposition (Section 4), discusses implications and threats to validity (Section 5), positions the work relative to related work (Section 6), and concludes (Section 7).

2 Background

2.1 LLM-Enabled Code-Generation Pipelines

A typical LLM-enabled code-generation system can be decomposed into three artifacts that jointly determine its behavior.

1. **Model M :** an instruction-tuned code LLM, for example Qwen2.5-Coder-7B [15], CodeLlama-7B [33], or DeepSeek-Coder-6.7B [13].
2. **Prompt template P :** a system-plus-user prompt format that typically includes a retrieval-context block, a task description, and instructions such as “return only executable Python code.”
3. **Retrieval context D :** usually the top- k documents returned by a retriever over a corpus. In CodeRAG-Bench HumanEval, $k = 1$ and the document is a code snippet semantically related to the task.

Fixing a task, seed, sampling configuration, and the versions of M , P , and D improves repeatability but does not make generation deterministic: runtime versions, batching, kernels, and stochastic sampling can still produce cross-run drift. Unlike a pinned compiler or library, these dependencies are probabilistic and can interact: a change in M can alter how a P or D change manifests, so their effects are not guaranteed to be separable, and we observe one such large model $\times$ data interaction empirically. Any of M , P , or D can change between deployments without modifying the application code that invokes the generator.

2.2 Traditional Regression-Testing Assumptions

Regression testing in classical software engineering [39] checks that code changes do not break previously passing behavior. The unit of change is typically a commit, the artifact under test is the application code, and dependencies (libraries, runtimes, and build configurations) are pinned to specific versions specified in a lockfile. LLM-enabled systems violate the first two assumptions: application-code changes are often absent, and the unit of behavioral perturbation becomes a model or retrieval-corpus update, with documented practical consequences such as the security regressions in GitHub Copilot output reported by Pearce et al. [30].

2.3 Retrieval-Augmented Code Generation and Retrieval Quality

Retrieval-augmented generation [24] augments LLMs with documents retrieved by a sparse retriever such as BM25 or by a dense retriever such as ColBERT [21] or DPR [20]. CodeRAG-Bench [35] provides task-specific retrieval documents (`docs[0]...docs[k]`) drawn from multiple corpora. The HumanEval split of this benchmark uses the `programming-solutions` corpus, in which each document is a candidate solution. The top-1 retrieved document for a HumanEval task is therefore a *canonical tier* document, which our audit confirms for the selected subset (Section 4.1). This raises a methodological concern: retrieval-perturbation experiments based on the canonical-tier `docs[0]` can overestimate the magnitude of retrieval-data dependency. This property motivates the main validity analysis in Section 4.4, which uses a BM25 self-excluded retrieval baseline to decompose the canonical RQ2 retrieval-perturbation effect.

We refer to D_0 consistently as the *answer-bearing canonical retrieval context*; it is a measurement setting that preserves a known property of CodeRAG-Bench HumanEval rather than a simulation of realistic deployment retrieval. The paper does not refer to D_0 as a “realistic retrieval baseline” or as “deployment-realistic retrieval.”

Our D_0 canonical condition departs from CodeRAG-Bench’s published RACG protocol, which excludes the canonical solution document from the retrieval corpus [35]; we deliberately retain the raw programming-solution document as an answer-bearing stress-test context. We therefore do not report the $D_0 \rightarrow D_1/D_2$ contrast as a standard leaderboard number or as evidence that the published protocol is flawed. Our contribution is to quantify, in this raw canonical setup, how much of the apparent retrieval regression signal is due to answer-bearing baseline rescue and how much remains as residual non-canonical sensitivity under a BM25 self-excluded retrieval proxy, via task-level paired regressions and recoveries.

2.4 A Three-Tier Oracle Scheme

The oracle problem in software testing [5] is particularly acute for generated code because multiple correct implementations can exist and the visible test suite can be incomplete; complementary approaches such as metamorphic testing [10] attempt to alleviate this gap via input-output relations. For HumanEval and HumanEval+, we can clearly define three oracles of increasing strictness.

■ **Table 1** Oracle-tier definitions.

Tier	Definition	Source
O1 (permissive partial visible oracle)	Passes syntactically and passes at least one assertion executed inside the HumanEval <code>check(candidate)</code> routine.	This work
O2 (benchmark-provided full visible assertion oracle)	Passes syntactically and the full HumanEval <code>check(candidate)</code> completes without assertion failures, exceptions, or timeouts. The shorthand “full visible test” refers to the visible test set provided by the benchmark and does not claim absolute oracle completeness; the O_3 /EvalPlus extended oracle tier addresses that gap.	Equivalent to standard pass@1
O3 (EvalPlus extended oracle)	Passes syntactically and passes all visible, base, and plus inputs from HumanEval+ that are included in the provided denominator.	EvalPlus [26]

The O_1 tier is intentionally permissive and is used as a low-strength oracle-sensitivity probe. This is not a claim that production smoke tests run every assertion and accept the program if any one of them passes. The $O_1 \rightarrow O_2$ gap captures *permissive partial-oracle escaped failures*, while the $O_2 \rightarrow O_3$ gap captures *visible-test escaped failures*. Section 4.3 reports both, and we treat the $O_2 \rightarrow O_3$ gap as the more direct evidence of oracle insufficiency.

3 Framework and Methodology

3.1 Research Questions and Analysis Plan

The empirical analysis is organized around four research questions.

1. **RQ1:** How do dependency-change conditions affect the failure rate?
2. **RQ2:** Under the answer-bearing canonical retrieval condition, how often do dependency changes cause tasks that previously passed under the fixed baseline to regress in a paired sense?
3. **RQ3:** How much of the canonical retrieval regression signal is attributable to answer-bearing baseline rescue, and how much remains as non-canonical retrieval sensitivity?
4. **RQ4:** How often do generations pass a permissive oracle but fail a stronger full-assertion oracle, measured over all generations and conditional on stronger-oracle failure?

RQ3 is the main empirical contribution and is answered in Section 4.4. Model- and prompt-replacement contrasts and the cross-model/GEE/scaling analyses are framework-robustness checks, not evidence for deployment-wide causal claims.

3.2 A Dependency-Aware Regression-Testing Framework

Let the *generation configuration* be $C = (M, P, D)$ and the *measurement configuration* be $E = (C, O_k)$. We fix a baseline $C_0 = (M_0, P_0, D_0)$ and a task set T . For each $t \in T$ and seed $s \in S$, the system produces $G(t, C, s)$, which oracle O_k maps to $\{\text{pass}, \text{fail}\}$ for $k \in \{1, 2, 3\}$.

A *changed configuration* C' perturbs exactly one component of C_0 ; that is, we change one factor at a time. Comparing C_0 and C' over (t, s) yields the following behavioral metrics, which are re-defined per oracle tier where applicable.

For each (C_0 vs. C_x) comparison, we compute the failure rate (per generation), the multi-seed paired regression rate (tasks passing all baseline seeds and failing the changed condition in at least one seed), the paired recovery rate, the signed paired change ($c - b$), the escaped-failure rate (O_1 -pass-but- O_2 -fail, analogously for $O_2 \rightarrow O_3$), and across-seed flakiness. We treat M , P , and D as testable dependencies for versioning, provenance, and paired comparison; O_k is the measurement layer rather than a perturbed generation dependency. The paired regression metric captures whether a change induces at least one failed generation in the evaluated seed set and is not the probability that a single deployment run will fail.

3.3 One-Factor-at-a-Time Condition Design

We define eight conditions on a single baseline (M_0, P_0, D_0) .

■ **Table 2** One-factor-at-a-time condition design.

ID	Model	Prompt	Data	Perturbed dimension
C_0 baseline	M_0	P_0	D_0 answer-bearing canonical	reference
C_1 model	M_1	P_0	D_0	model
C_2 model	M_2	P_0	D_0	model
C_3 prompt	M_0	P_1 (strict grounding)	D_0	prompt
C_4 prompt	M_0	P_2 (test-aware)	D_0	prompt
C_5 retrieval	M_0	P_0	D_1 (top document removed)	retrieval data
C_6 retrieval	M_0	P_0	D_2 (top document replaced with the offset-7 different-task mismatched document)	retrieval data
C_7 retrieval (stress)	M_0	P_0	D_3 (synthetic outdated-API memo prepended)	retrieval data

We designate C_7 as a stress-test condition and exclude it from claims about non-canonical retrieval-data changes.

We additionally audit the 120 HumanEval replacement contexts used for D_2 . The replacement source task is never the same as the target task, the source entry point is never the same as the target entry point, and the target entry point appears 0 out of 120 times in the D_2 context. The median token Jaccard overlap between D_0 and D_2 is 0.187 (IQR 0.147–0.219). Therefore D_2 is a deterministic different-task replacement stressor and is not a semantic-irrelevance control: the replacement document can still share general Python patterns with the target, and offset 7 is a deterministic but arbitrary construction choice. D_2 is therefore used only as a directional auxiliary stressor that supports D_1 and the Section 4.4 decomposition, not as the basis for any main retrieval claim.

3.4 Statistical Analysis

We report Wilson 95% confidence intervals [36] for each rate. The primary inferential unit is the task, and the McNemar test treats each task as one paired observation. At the task level, b counts a regression when the baseline passes every seed but the changed condition does not; c counts a recovery when the baseline does not pass every seed but the changed condition passes every seed. The primary estimands have distinct denominators: task-level paired regression (tasks passing the baseline), generation-level failure and $O_1 \rightarrow O_2$ escape rates (all rows), and conditional $O_2 \rightarrow O_3$ escaped failures (O_2 -passing rows).

The analysis comprises four tiers: (1) the main analysis (M_0 HumanEval D_1 paired regression under O_2 , with D_2 as an auxiliary stressor); (2) the central validity analysis (C_0 copy audit, BM25 self-excluded retrieval, answer-stripped D_0); (3) oracle-sensitivity ($O_1 \rightarrow O_2$ and $O_2 \rightarrow O_3$ escaped failures); and (4) robustness/exploratory checks (MBPP, cross-model, GEE, bootstrap).

For each (C_0 vs. C_x , oracle k) comparison, we use a paired McNemar test [29]; with fewer than 25 discordant pairs we use an exact binomial test, otherwise a continuity-corrected χ^2 . We define the multiple-comparison family as all 66 McNemar-style contrasts reported in the paper and companion package (14 main + 8 model $\times$ data interactions + 16 scaling + 10 MBPP + 10 BM25 self-excluded + 8 family-expansion), and we report both the uncorrected test and a conservative Bonferroni-equivalent threshold $\alpha/66 = 7.6 \times 10^{-4}$ (with $\alpha = 0.05$). Alternative step-down or FDR procedures [14, 6] could be less conservative, but we adopt Bonferroni as a strict upper bound. Only contrasts that pass this corrected threshold are interpreted as primary inferential results; the model-interaction, scaling, MBPP, and family-expansion contrasts are exploratory. As a sensitivity analysis, we also fit a binomial GEE model [25] with a logit link and an exchangeable working correlation structure clustered on `task_id` to confirm the direction of the retrieval-data effect under a population-average model.

3.5 Evaluation Pipeline

Each generation is evaluated under the three oracle tiers by an isolated subprocess runner with per-task wall-clock timeouts (30s for HumanEval visible tests, 20s for each side of the EvalPlus base/plus split). O_3 pre-computes expected outputs once by running the canonical solution on the HumanEval+ inputs; the provided denominator excludes 78 very-large-integer plus-output cells, leaving 1,570 base and 122,605 plus outputs, against which generations are compared by parallel worker processes (the released reproduction command uses 16; the worker count affects runtime only, not the recorded outcomes). O_2 runs the full `check(candidate)` routine inside an isolated subprocess; O_1 instruments every `assert` inside `check(candidate)` and counts whether at least one passes. Two early evaluator

errors (removal of generated import/helper definitions; consideration of only top-level visible assertions in O_2) were corrected: the final evaluator preserves top-level imports, helper functions/classes, and the prompt preamble, and runs the full visible-test check. Section 4.5 summarizes the sensitivity analysis. All quantitative tables in this paper were recomputed after the evaluator fixes.

4 Experiments

Sections 4.2 (RQ2) and 4.3 (RQ4) report the main analyses of failure rates, paired regressions, and oracle escaped failures. These should be read together with the Section 4.4 decomposition (RQ3), which determines how the RQ2 retrieval-perturbation rate should be interpreted. Section 4.5 presents robustness and exploratory analyses in compact form.

4.1 Experimental Setup

Experimental setup summary. The primary dataset is 120 CodeRAG-Bench HumanEval tasks (an evenly spaced subsample of 164), and the auxiliary dataset is 50 MBPP tasks. The primary models are Qwen2.5-Coder-7B (M_0) [15], DeepSeek-Coder-6.7B (M_1) [13], and CodeLlama-7B (M_2) [33]; the scaling analysis additionally includes Phi-3.5-mini (3.8B) [1], CodeLlama-13B, and SWE-agent-LM-32B [38]. The configured sampling parameters are temperature 0.2, top-p 0.95, and max_new_tokens 1024. Contemporaneous logs record vLLM 0.15.0 for the 2,880-row primary family. The disjoint five-seed extension used vLLM 0.20.0 with a public model identifier, making the combined eight-seed analysis a compound runtime-and-seed sensitivity; direct engine-version logs were not retained for the BM25 and stripped- D_0 families. The EvalPlus O_3 coverage spans 14,760 HumanEval generations and 2,160 BM25-proxy rows.

Model weights were loaded from the local paths or public identifiers recorded per run family (see the run-family provenance record in the replication package), and no model was fine-tuned in this study. The cited model and HumanEval/MBPP benchmark references ([15, 13, 33, 1, 9, 4]) are technical reports without a peer-reviewed venue; the citations are used only as a primary-source pointer for model and benchmark identification.

The HumanEval sample size was fixed prior to analysis by an evenly spaced deterministic rule to keep the full $O_1/O_2/O_3$ multi-seed pipeline tractable. The selected 120 and omitted 44 tasks differ by more than $|\text{SMD}| = 0.20$ on 2 of 17 locked static features (entry-point parts and AST loop count); this characterizes the subset but does not establish equivalent generation outcomes for omitted tasks. All selected tasks and seeds are retained in the reported analysis. The MBPP 50-task sample is a deterministic auxiliary replication rather than a cross-benchmark estimate, and it is interpreted under the same canonical-tier retrieval caveat as HumanEval.

On the selected 120 HumanEval tasks, D_0 contains the target entry-point definition and the canonical solution body for 120 of 120 tasks. Table 3 should therefore not be read as an estimate of deployment retrieval failure rates; its purpose is to quantify the benchmark-induced dependency signal that Section 4.4 decomposes against a BM25 self-excluded baseline.

4.2 Main Experiment: One Factor at a Time

We ran each of the eight conditions defined in Section 3.3 over 120 HumanEval tasks and three seeds, yielding 2,880 generations. Per-condition failure rates and paired-regression results under the O_2 full visible-test oracle are summarized in Table 3. These are results for the canonical answer-bearing stress-test setup rather than estimates of deployment retrieval risk.

■ **Table 3** Canonical answer-bearing stress-test results. Not an estimate of deployment retrieval risk and not a standard CodeRAG-Bench RACG leaderboard. Per-condition full visible-test oracle (O_2) results from the main experiment. Paired-regression rates, paired-regression confidence intervals, and McNemar p -values are computed at the task level; seed-level tests are descriptive. Boldface indicates a result that is stable under task-level Bonferroni correction. The dagger ($\dagger$) indicates significance before correction but not stability under the 66-comparison threshold. How much of the D_1/D_2 effect is answer-bearing baseline rescue versus residual non-canonical retrieval sensitivity is decomposed in Section 4.4 (Tables 5–7).

Condition	n	strong_fail	95% CI	Paired regression (task 95% CI, %)	McNemar p	Direction
C_0 baseline	360	0.000	[0.000, 0.011]	reference	reference	reference
C_1 model M_1 (DeepSeek)	360	0.011	[0.004, 0.028]	1.67% (2/120; [0.46, 5.87])	0.500	not significant
C_2 model M_2 (CodeL- lama)	360	0.081	[0.057, 0.113]	12.50% (15/120; [7.72, 19.60])	6.10×10^{-5}	regression
C_3 prompt P_1 (strict)	360	0.000	[0.000, 0.011]	0% (0/120; [0.00, 3.10])	1.000	no effect
C_4 prompt P_2 (test- aware)	360	0.025	[0.013, 0.047]	2.50% (3/120; [0.85, 7.09])	0.250	not significant
C_5 data D_1 (no document)	360	0.100	[0.073, 0.135]	11.67% (14/120; [7.08, 18.63])	1.22×10^{-4}	regression
C_6 data D_2 (different- task mis- matched docu- ment) $\dagger$	360	0.075	[0.052, 0.107]	9.17% (11/120; [5.20, 15.67])	9.77×10^{-4}	regression $\dagger$
C_7 data D_3 (stress)	360	0.000	[0.000, 0.011]	0% (0/120; [0.00, 3.10])	1.000	no effect (consistent with stress-test framing)

Because canonical C_0 passes all three seeds for all 120 tasks, $c = 0$ is structural in its contrasts and does not imply that recoveries are generally absent. Task-level McNemar gives $b = 14$, $p = 1.22 \times 10^{-4}$ for D_1 (Bonferroni-stable) and $b = 11$, $p = 9.77 \times 10^{-4}$ for D_2 (an auxiliary stressor that narrowly misses $\alpha/66 = 7.6 \times 10^{-4}$). Table 3 must be read with the Section 4.4 decomposition, not as independent evidence of deployment degradation.

In seed-aggregation sensitivity checks, the D_1 effect persists across the arbitrary-seed (14/120, 11.67%), majority-seed (12/120, 10.00%), and all-seed (10/120, 8.33%) aggregations. D_2 is directionally consistent but smaller (11, 9, 7/120).

4.3 EvalPlus-Derived Extended Oracle Re-evaluation and Oracle Sensitivity

We re-evaluated 14,760 HumanEval generations under the EvalPlus-derived HumanEval+ extended oracle (this is not an official leaderboard score). O_3 is computed after excluding 78 very-large-integer plus-output cells from HumanEval/83 and HumanEval/139. Expected outputs are pre-computed by running the canonical solution and then cached (1,570 base entries; 122,605 plus entries). Verification: the pre-computed cache matches the official EvalPlus ground truth, and the custom runner passes the canonical solution on every evaluated input. Section 4.4 separately reports a 2,160-row BM25 self-excluded O_3 check.

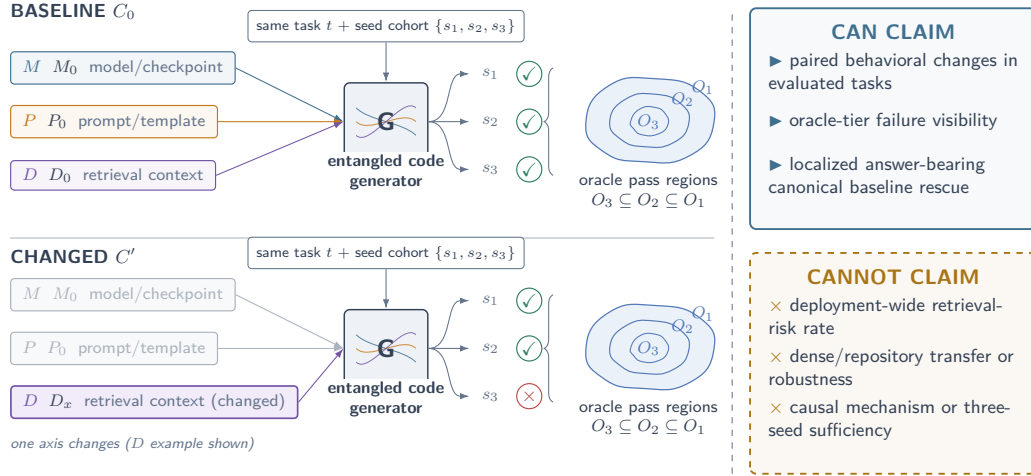
■ **Table 4** M_0 oracle results for the main experiment. O_3 is the EvalPlus-derived extended oracle (`strong_pass AND evalplus_all_passed`) over the non-excluded HumanEval+ denominator and is not an official leaderboard score. The $O_1 \rightarrow O_2$ column reports all-generation escape rates; $O_2 \rightarrow O_3$ is conditional on O_2 passing. O_1 is an intentionally permissive sensitivity probe, not a production smoke-test model. The C_7 synthetic stress condition equals C_0 and is omitted.

Condition (M_0)	n	O_2 pass	O_3 pass	$O_2 \rightarrow O_3$ escaped	O_1 permissive $\rightarrow O_2$ escaped
C_0 baseline	360	100.0%	88.3%	11.67% (42/360)	0.00% (0/360)
C_5 data	360	90.0%	87.2%	3.09% (10/324)	10.00% (36/360)
D_1					
C_6 data	360	92.5%	87.5%	5.41% (18/333)	7.22% (26/360)
D_2					

RQ4 is an oracle-tier *visibility* analysis rather than a one-sided claim about extended-oracle degradation. The $O_2 \rightarrow O_3$ escaped-failure rate (11.67% at baseline) reflects baseline-level visible-test insufficiency; the conditional rates of 3–5% under D_1/D_2 are not evidence that retrieval perturbations reduce extended-oracle escaped failures but are a denominator effect.

O_1 *variants*. The intentionally permissive arbitrary-assertion probe produces all-generation escape counts of 36/360 under D_1 and 26/360 under D_2 ; conditional on O_1 passing, the escape rates are 36/360 and 26/359 (one D_2 generation fails O_1); and among O_2 failures, the corresponding failure-conditioned miss counts are 36/36 and 26/27. A less-permissive but order-dependent first-three-substantive variant yields 8/360 (2.22%) and 3/360 (0.83%), respectively; here a *substantive* assertion site is an executable assertion outside comments and docstrings, the first three are taken in source order, and generations with fewer than three such sites follow the released oracle script’s documented pass rule. These variants are sensitivity probes rather than models of a production smoke test.

Paired O_3 transitions are mixed: in the canonical setup, degradations and recoveries are of similar magnitude (22↓/18↑, 20↓/17↑), and in the BM25 self-excluded setup, recoveries dominate (8↓/20↑, 7↓/19↑). From the pooled O_3 robustness perspective, the absolute O_3 pass rate drops from 87.9% at baseline to 78.0% under D_1 and 79.4% under D_2 , but this does not replace the primary paired- M_0 transition analysis.



■ **Figure 1** Dependency axes, paired comparison, and claim boundary. D illustrates the one-factor change; M/P use the same pairing as framework-coverage checks. Check/cross marks depict primary O_2 outcomes. At task level, b is baseline all-seed-pass $\rightarrow$ changed not-all-seed-pass; c reverses this transition. For jointly eligible generations, $\text{pass}(O_3) \subseteq \text{pass}(O_2) \subseteq \text{pass}(O_1)$.

4.4 Main Empirical Contribution (RQ3): Measurement-Validity Decomposition of Answer-Bearing Canonical Retrieval Regressions

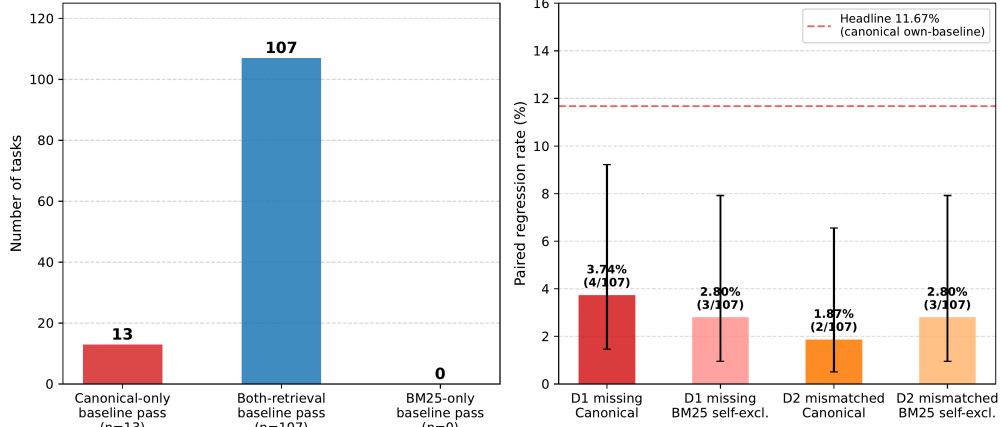
RQ3 decomposes how much of the Section 4.2 effect is answer-bearing baseline rescue rather than non-canonical retrieval sensitivity; Figure 1 summarizes the dependency axes and the resulting claim boundary. CodeRAG-Bench HumanEval docs[0] is a canonical tier; in our subset, D_0 contains the target entry-point definition and solution body for all 120 tasks. The central result is therefore the decomposition, not the 11.67% canonical stress rate alone.

Copy-template baseline audit. After NFKC normalization and whitespace removal, the M_0 baseline is byte-identical to canonical D_0 in 331/360 (91.9%) generations and all three seeds for 108/120 tasks. This holds for all three baseline seeds in 12/14 canonical D_1 regression tasks and at least two seeds in 13/14. Thus, removing D_0 usually removes a copied answer template; the headline contrast is a benchmark-induced dependency stress test, motivating the BM25 proxy and answer-stripped mechanism check below.

To address the canonical answer-bearing setup, we construct a title-based self-excluded BM25 proxy; title exclusion is not equivalent to answer exclusion.

1. Index the entire **programming-solutions** corpus (1,128 documents) with BM25Okapi [32], using regex word tokenization ($[A-Za-z0-9_]+$, discarding single-character tokens) and lowercasing.
2. For each of the 120 HumanEval tasks, query with the task prompt and select the top-1 retrieval document whose title is not the same as the task's own **entry_point**. For 120 of 120 tasks, the self-inclusive top-1 is the canonical solution itself (with an average BM25 score 124.5 higher than the self-excluded top-1), consistent with the answer-bearing D_0 audit.
3. Re-run M_0 with the same $D_1/D_2/D_3$ perturbation rules over this BM25 self-excluded D_0 , yielding 2,160 generations, and re-evaluate these rows under EvalPlus O_3 as an additional non-canonical oracle check.

No retrieved proxy context contains an exact or NFKC/whitespace-normalized canonical body, but three contain a target-named function and four tasks trigger the predefined overlap review. Excluding all four leaves the task-level D_1 and D_2 discordances unchanged (3/3 and 3/4), so flagged overlap does not drive the aggregate result; nevertheless, title exclusion does not guarantee answer exclusion.



■ **Figure 2** Task-level decomposition on the common intersection and D_1/D_2 regression rates.

Figure 2 decomposes the canonical-tier effect into baseline-rescue and marginal-retrieval components. A naive comparison of paired regression rates between the canonical and BM25 self-excluded setups would use *different* baseline-pass sets and would ignore paired recoveries. We therefore report task-level McNemar b/c counts, where b counts regressions (the baseline passes under every seed and the changed condition fails under at least one seed) and c counts recoveries (the baseline does not pass under every seed but the changed condition passes under every seed).

■ **Table 5** Self-baseline paired regression, paired recovery, and signed task-level change. b/c ($c-b$) compresses regression, recovery, and net change into a single column.

Setup	Condition	Baseline-pass n	b / c ($c-b$)	McNemar p	Interpretation
Canonical D_0	D_1 none	120	14 / 0 (-14)	1.22×10^{-4}	Bonferroni-stable canonical stress
Canonical D_0	D_2 different-task	120	11 / 0 (-11)	9.77×10^{-4}	directional auxiliary stress
BM25 self-excluded D_0	D_1 none	107	3 / 3 (0)	1.000	inconclusive / bidirectional
BM25 self-excluded D_0	D_2 different-task	107	3 / 4 (+1)	1.000	inconclusive / bidirectional

Cross-run drift audit for D_1 none. Because D_1 none is a no-context anchor with identical rendered prompts between the canonical main run and the BM25 self-excluded re-run, we audit cross-run drift directly: 360/360 prompt hashes match, 340/360 generations are

byte-identical under NFKC, and 358/360 O_2 outcomes agree (only HumanEval/142 changes all-seed-pass status). Re-pairing moves the BM25 paired counts from $b = 3, c = 3$ only to $b = 4, c = 3$; harmful non-canonical retrieval degradation remains unestablished and bidirectional at this scale.

BM25 self-excluded prompt sensitivity. Because the M_0 baseline copies the answer-bearing D_0 on most tasks, the small canonical prompt effects in Table 3 could in principle be attenuated by the copy-template behavior. Under the BM25 self-excluded D_0 , the strict-grounding prompt yields $b = 4, c = 4$ and the test-aware prompt yields $b = 2, c = 2$ at the task level. Neither produces a directional degradation under either retrieval-source setup, which protects the prompt-effect interpretation from this alternative explanation.

Influence versus non-necessity. Across 360 same-task, same-seed BM25 C_0/D_1 pairs, 122 generations are code-invariant, 203 differ while both pass, 11 pass only with context, 8 pass only without it, and 16 fail in both conditions; task-level b/c remains 3/3. Thus, the null net contrast mixes invariance, functional non-necessity, help, and harm – it is not general retrieval robustness. HumanEval/39 is an illustrative counterexample: all three contextual outputs contain the retrieved context yet fail, while all no-context outputs pass; this demonstrates influence in one task, not a population-level causal effect.

■ **Table 6** Baseline-rescue decomposition of the canonical regression.

Task set	n	D_1 canonical regression	D_2 canonical regression	BM25 regression on the same set
Canonical-only baseline-rescue tasks	13	10/13	9/13	not defined
Common baseline-pass intersection	107	4/107	2/107	3/107 for both D_1 and D_2

The 11.67% canonical D_1 effect is concentrated on the 13 baseline-rescue tasks: 10/13 regress under canonical D_1 and 9/13 under D_2 . In the common 107-task baseline-pass intersection, canonical regressions are only 4/107 and 2/107, while BM25-proxy changes are small and bidirectional (3/3 for D_1 , 3/4 for D_2). Of the 14 canonical D_1 regressions, 10 are baseline-rescue cases and 4 lie in the common intersection; all 14 are visible-assertion failures rather than syntax errors, exceptions, or timeouts.

Answer-stripped D_0 direct validity test. As a more direct check than the BM25 proxy, we add an M_0 condition that preserves the canonical-tier document structure but removes only the canonical solution body: D_0^{stripped} retains the task’s function header, imports, and docstring, with no implementation body. This fixes the same answer-bearing-tier *document slot* and changes only the canonical answer body within the rendered document. We re-ran M_0 over 120 tasks $\times$ 3 seeds = 360 generations under the same O_2 oracle.

■ **Table 7** Answer-stripped D_0 paired contrast (M_0 , O_2 full visible test). b/c is the count of regression and recovery tasks.

Baseline	Changed condition	n	b / c	McNemar p	Interpretation
D_0 canonical (body present)	D_0^{stripped} (no body)	120	11 / 0	9.77×10^{-4}	direct rescue contribution of the solution body
D_1 none	D_0^{stripped}	106	0 / 3	0.250	header/docstring provides only a small additional rescue
title-excluded BM25 proxy	D_0^{stripped}	107	2 / 4	0.688	task-level agreement with the retriever- independent control

The answer-stripped contrast is a mechanism check, not a separate correction-stable claim: $p = 9.77 \times 10^{-4}$ exceeds $\alpha/66 = 7.6 \times 10^{-4}$. Removing only the solution body reproduces 11/14 (78.6%) canonical D_1 regressions; removing the entire document adds three recoveries relative to stripped D_0 ($b = 0, c = 3$), indicating little additional rescue from the header/docstring. The retriever-independent stripped control and BM25 proxy are similar at task level ($b = 2, c = 4, p = 0.69$), but the latter is not guaranteed answer-excluded. Together, these contrasts are consistent with the solution body being the operative element of the canonical-tier dependency; because direct engine-version logs were not retained for the stripped- D_0 family, we read this as strong supporting evidence rather than a fully isolated causal mechanism.

Accordingly, 11.67% is a canonical-tier stress rate, removing the solution body reproduces most of it (11 of 14 regressions), and the small bidirectional BM25-proxy changes do not establish harmful non-canonical degradation. This negative decomposition prevents a deployment-wide interpretation of the headline rate.

BM25 O_3 check. Re-evaluating the 2,160 BM25 self-excluded rows under EvalPlus reveals no additional O_3 degradation under D_1/D_2 . The strict O_3 pass rate is 83.89% for BM25 C_0 and 87.22% for both BM25 D_1 and D_2 . The conditional $O_2 \rightarrow O_3$ escaped-failure rate is 7.65% at C_0 , 3.09% under D_1 , and 5.14% under D_2 . This result should not be over-interpreted as a retrieval benefit; instead, it strengthens the main validity argument that removing the self document does not reproduce the large canonical D_1/D_2 effect at a stronger extended-oracle layer.

4.5 Robustness and Artifact Checks

The remaining analyses are auxiliary checks rather than separate claims. In the disjoint five-seed rerun, the D_1 and D_2 regression task sets recur exactly (14/14 and 11/11); 8 and 6 tasks, respectively, have no passing seed in the combined record. Because the extension changed vLLM version and model-loading path, this is compound rerun persistence, not evidence that three seeds are sufficient. Cross-model, MBPP, GEE [25], bootstrap/permutation [12], scaling, and pooled EvalPlus analyses remain exploratory because benchmark exposure and answer-bearing retrieval are major confounders.

5 Discussion

5.1 Dependency Sensitivity Depends on the Retrieval Source and Model Recipe

The Section 4.4 decomposition is the central empirical finding. The large canonical model $\times$ data interaction also attenuates under the BM25 self-excluded setup (M_1/M_2 paired regression rates fall from 24.58–57.14% to 6.17–19.51%). The cross-model results are exploratory heterogeneity evidence; the main contribution is separating canonical-retrieval inflation from inconclusive non-canonical retrieval change.

5.2 Implications of Canonical vs. BM25 Self-Excluded Retrieval

The residual non-canonical retrieval sensitivity (common 107-task intersection: canonical D_1 3.74% vs. BM25 2.80%) is small and comparable across the two retrieval sources. Future benchmark designs should use retrieval corpora whose top-1 documents are not the canonical solution (for example, `online-tutorials` or `stackoverflow-posts`) or use an explicit answer-stripping variant. Reviewers and practitioners should read effect sizes from canonical-tier RACG benchmarks as benchmark-specific stress-test numbers rather than as deployment retrieval-risk estimates.

5.3 Extended and Permissive Oracles Reveal Different Blind Spots

The $O_2 \rightarrow O_3$ escaped-failure rate (11.67% at baseline) reflects visible-test insufficiency, and paired O_3 transitions are mixed. The all-generation permissive $O_1 \rightarrow O_2$ gap grows from 0/360 at baseline to 36/360 under D_1 and 26/360 under D_2 ; the order-dependent first-three-substantive variant yields 8/360 and 3/360, so the measured escape magnitude is itself a function of the visibility rule. Dependency updates therefore should not be evaluated with permissive probes alone but must be tested with layered oracles.

5.4 Practical Implications for LLM-Enabled Software Testing

The practical implications concern benchmark design and evaluation governance rather than quantifying deployment retrieval risk. RACG benchmarks should pair answer-bearing results with answer-stripped or self-excluded controls. A minimal CI record should version model/checkpoint, prompt, retriever/corpus, runtime, sampling configuration, and seeds; report task-level b/c by oracle tier; and escalate correction-stable or security-relevant escaped failures for review.

5.5 Threats to Validity

This section follows the four-type validity-threat classification of Wohlin et al. [37]: construct, internal, external, and conclusion validity, framed within the broader measurement-validation guidance of Kitchenham et al. [22].

Construct validity. Model replacement is entangled with capability differences and is not a causal estimate of deployment-relevant model-update risk. The paired-regression design partially controls for capability on the easier subset but cannot achieve full separation. O_1 is an intentionally permissive sensitivity probe and is not a model of any production smoke test. D_1/D_2 are controlled stress conditions rather than direct simulations of every real corpus update.

Internal validity. The primary analysis uses three seeds. Although the same D_1/D_2 regression task sets recur in the five-seed extension, its different vLLM version and model-loading path prevent interpreting this as a pure seed-count study or as proof that three seeds are sufficient. Two evaluator errors (the removal of import/helper definitions and consideration of only top-level visible assertions in O_2) were discovered and corrected; Section 4.5 summarizes the sensitivity analysis. The CodeRAG-Bench HumanEval top-1 canonical tier is a known benchmark property rather than a newly discovered defect [35].

External validity. The primary results use 120 HumanEval Python function-level tasks under answer-bearing canonical retrieval, with auxiliary checks on 50 MBPP tasks and the sparse BM25 proxy. Static comparison of the selected 120 and omitted 44 tasks does not establish outcome equivalence, so absolute rates should not be generalized to all 164 tasks. Generalization to closed-weight or multilingual models, dense retrieval, repository-level or multi-turn agents, and non-functional outcomes such as security, performance, and maintainability remains future work. Benchmark contamination and memorization particularly limit cross-model interpretation [28, 8].

Conclusion validity. The Bonferroni-equivalent threshold for 66 comparisons ($\alpha/66 = 7.6 \times 10^{-4}$) is a strict upper bound. D_1 ($p = 1.22 \times 10^{-4}$) passes this threshold, whereas D_2 ($p = 9.77 \times 10^{-4}$) does not. The GEE robustness check supports the direction but is not a condition-specific proof and should be interpreted together with the task-level McNemar results.

6 Related Work

This work sits at the intersection of the four areas reviewed below: on top of a task-level *paired regression* design, we treat model, prompt, retrieval, and oracle changes as *controlled non-code dependency perturbations* and combine an *answer-bearing retrieval measurement-validity decomposition* with an *oracle-tier visibility* analysis. To our knowledge, prior work has not combined these four axes.

6.1 Evaluation of LLM-Based Code Generation

HumanEval [9] and MBPP [4] are foundational benchmarks for Python function-level code generation, and EvalPlus [26] extends them with automatically generated additional inputs. BigCodeBench [42], LiveCodeBench [16], and SWE-bench [18] extend evaluation toward realistic library use, contamination avoidance, and repository-level tasks, respectively. Beyond pass@ k leaderboards, LLM-augmented test generation such as CodaMosa [23] uses pre-trained models to escape coverage plateaus on Java fault databases like Defects4J [19]. All of these benchmarks report aggregate pass rates under a fixed configuration; none measure paired regression under controlled non-code dependency changes, which is the methodological focus of our work.

6.2 Defects and Quality in Retrieval-Augmented Generation

Retrieval-augmented generation [24] and dense passage retrieval [20] form the basis of the broader RAG literature, which has since scaled to trillion-token corpora through retrieval-time pretraining such as RETRO [7]. Repository-level extensions such as RepoCoder [40] shifted from QA-style to code-specific retrieval, and Self-RAG [3] adds self-reflective adaptive retrieval.

CodeRAG-Bench [35] characterizes failure modes (irrelevant retrieval, contradictions, retrieval-induced hallucinations) and provides the retrieval-context infrastructure we use. Our work is complementary: we treat the retrieval corpus as a *software dependency*, quantify paired regression under controlled perturbations, and audit how answer-bearing canonical-tier retrieval can inflate the apparent dependency effect, which links retrieval evaluation to benchmark-contamination concerns when the context closely encodes the answer.

6.3 Regression Testing for ML and LLM Systems

Sculley et al. [34] argue that ML systems acquire dependencies not visible in source code; Amershi et al. [2] and Zhang et al. [41] document how ML practices differ from classical SE assumptions; DeepXplore [31] exposes DNN defects through differential whitebox testing; and D’Amour et al. [11] show that small perturbations change behavior in ways that traditional evaluation hides. Our work extends this lineage to LLM-enabled code generation by treating prompt templates, retrieval corpora, retriever-exclusion policies, and oracle tiers as versioned dependencies and by showing that answer-bearing retrieval can create a hidden dependency. Flakiness studies in classical testing [27] also support multi-seed evaluation.

6.4 Test Oracles and Silent Failures

Test-oracle insufficiency has long been studied in mutation testing [17] and surveyed under the *oracle problem* [5]. The extended test set of EvalPlus [26] targets this for LLM-generated code via automatically generated edge-case inputs. We use EvalPlus as the O_3 tier inside a dependency-regression design rather than as an aggregate leaderboard, and our two-stage escaped-failure metrics ($O_1 \rightarrow O_2$ and $O_2 \rightarrow O_3$) use *non-code dependency changes* as the source of behavioral perturbation instead of mutations.

7 Conclusion

Answer-bearing canonical-tier retrieval can create a hidden copy-template dependency in LLM code-generation benchmarks: under answer-stripped and BM25 self-excluded decompositions, the apparent canonical regression signal is largely localized to the solution body and baseline rescue. *Reporting paired regression rates without such decompositions measures a benchmark-induced stress signal rather than deployment-wide risk.* These results motivate dependency-aware regression testing, layered oracles, and replication on answer-excluded corpora.

Data Availability Statement

In accordance with the ESEM 2026 Open Science Policy, the replication package is archived at <https://doi.org/10.5281/zenodo.21988303>. It contains the configurations, selected tasks and conditions, raw generations, $O_1/O_2/O_3$ results, evaluator pre-fix snapshots, tables, figures, and analysis pipeline. The README maps claims to CPU-only reproduction commands that do not require rerunning inference. Model weights are not redistributed. Original primary rows record local model paths but not checkpoint revisions; public identifiers and cache-resolved commits are post-hoc provenance aids, and the run-family provenance matrix distinguishes these limitations from directly observed engine metadata. The artifact code is released under the MIT license and the derived data under CC BY 4.0, while HumanEval, MBPP, CodeRAG-Bench, and EvalPlus remain subject to their upstream licenses.

References

- 1 Marah Abdin, Jyoti Aneja, Hany Awadalla, et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024. arXiv preprint arXiv:2404.14219. doi:10.48550/arXiv.2404.14219.
- 2 Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. Software engineering for machine learning: A case study, 2019. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300.
- 3 Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*, 2024.
- 4 Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. arXiv preprint arXiv:2108.07732. arXiv:2108.07732.
- 5 Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. The oracle problem in software testing: A survey, 2015. *IEEE Transactions on Software Engineering*, 41(5):507–525.
- 6 Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing, 1995. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300.
- 7 Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pages 2206–2240, 2022. URL: <https://proceedings.mlr.press/v162/borgeaud22a.html>.
- 8 Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying memorization across neural language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- 9 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code, 2021. arXiv preprint arXiv:2107.03374. arXiv:2107.03374.
- 10 Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, T. H. Tse, and Zhi Quan Zhou. Metamorphic testing: A review of challenges and opportunities. *ACM Computing Surveys*, 51(1):1–27, 2018. doi:10.1145/3143561.
- 11 Alexander D’Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning, 2022. *Journal of Machine Learning Research*, 23(226):1–61.
- 12 Bradley Efron. Bootstrap methods: Another look at the jackknife, 1979. *The Annals of Statistics*, 7(1):1–26.
- 13 Daya Guo, Qihao Zhu, Dejian Yang, et al. DeepSeek-Coder: When the large language model meets programming – the rise of code intelligence, 2024. arXiv preprint arXiv:2401.14196. doi:10.48550/arXiv.2401.14196.
- 14 Sture Holm. A simple sequentially rejective multiple test procedure, 1979. *Scandinavian Journal of Statistics*, 6(2):65–70.
- 15 Binyuan Hui, Jian Yang, Zeyu Cui, et al. Qwen2.5-Coder technical report, 2024. arXiv preprint arXiv:2409.12186. doi:10.48550/arXiv.2409.12186.

- 16 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, 2025. In International Conference on Learning Representations (ICLR).
- 17 Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing, 2011. *IEEE Transactions on Software Engineering*, 37(5):649–678.
- 18 Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues?, 2024. In International Conference on Learning Representations (ICLR).
- 19 René Just, Darioush Jalali, and Michael D. Ernst. Defects4J: A database of existing faults to enable controlled testing studies for Java programs. In *Proceedings of the 2014 International Symposium on Software Testing and Analysis (ISSTA)*, pages 437–440, 2014. doi:10.1145/2610384.2628055.
- 20 Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering, 2020. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6769–6781.
- 21 Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 39–48, 2020. doi:10.1145/3397271.3401075.
- 22 Barbara Kitchenham, Shari Lawrence Pfleeger, and Norman Fenton. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering*, 21(12):929–943, 1995. doi:10.1109/32.489070.
- 23 Caroline Lemieux, Jeevana Priya Inala, Shuvendu K. Lahiri, and Siddhartha Sen. CodaMosa: Escaping coverage plateaus in test generation with pre-trained large language models. In *Proceedings of the 45th International Conference on Software Engineering (ICSE)*, pages 919–931, 2023. doi:10.1109/ICSE48619.2023.00085.
- 24 Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks, 2020. In Advances in Neural Information Processing Systems (NeurIPS).
- 25 Kung-Yee Liang and Scott L. Zeger. Longitudinal data analysis using generalized linear models, 1986. *Biometrika*, 73(1):13–22.
- 26 Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation, 2023. In Advances in Neural Information Processing Systems (NeurIPS).
- 27 Qingzhou Luo, Farah Hariri, Lamyaa Eloussi, and Darko Marinov. An empirical analysis of flaky tests, 2014. In ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), pages 643–653.
- 28 Inbal Magar and Roy Schwartz. Data contamination: From memorization to exploitation, 2022. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL, Short Papers), pages 157–165.
- 29 Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages, 1947. *Psychometrika*, 12(2):153–157.
- 30 Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. Asleep at the keyboard? assessing the security of GitHub Copilot’s code contributions. In *IEEE Symposium on Security and Privacy (S&P)*, pages 754–768, 2022. doi:10.1109/SP46214.2022.9833571.
- 31 Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. DeepXplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*, pages 1–18, 2017. doi:10.1145/3132747.3132785.

- 32 Stephen E. Robertson and Stephen Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval, 1994. In Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pages 232–241.
- 33 Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, et al. Code Llama: Open foundation models for code, 2023. arXiv preprint arXiv:2308.12950. doi:10.48550/arXiv.2308.12950.
- 34 D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems, 2015. In Advances in Neural Information Processing Systems (NeurIPS).
- 35 Zora Zhiruo Wang, Akari Asai, Xinyan Velocity Yu, Frank F. Xu, Yiqing Xie, Graham Neubig, and Daniel Fried. CodeRAG-Bench: Can retrieval augment code generation?, 2025. In Findings of the Association for Computational Linguistics: NAACL 2025.
- 36 Edwin B. Wilson. Probable inference, the law of succession, and statistical inference, 1927. Journal of the American Statistical Association, 22(158):209–212.
- 37 Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Springer, 2012. doi:10.1007/978-3-642-29044-2.
- 38 John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems 38 (NeurIPS), Datasets and Benchmarks Track*, 2025. OpenReview id: 63iVrXc8cC.
- 39 Shin Yoo and Mark Harman. Regression testing minimization, selection and prioritization: A survey, 2012. Software Testing, Verification and Reliability, 22(2):67–120.
- 40 Fengji Zhang, Bei Chen, Yue Zhang, et al. RepoCoder: Repository-level code completion through iterative retrieval and generation, 2023. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2471–2484.
- 41 Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 48(2):1–36, 2022. doi:10.1109/TSE.2019.2962027.
- 42 Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widayarsi, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions, 2025. In International Conference on Learning Representations (ICLR).