AGENT RETRIEVAL BENCH: EVALUATING REPOSITORY CONTEXT RETRIEVAL FOR CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

While coding agents are typically evaluated on end-to-end patch generation, their success heavily depends on an upstream context-acquisition stage: identifying relevant repository files. We introduce Agent Retrieval Bench, a file-level code retrieval benchmark designed for this workflow-driven context discovery problem. Unlike traditional semantic code search, relevance in Agent Retrieval Bench reflects what an agent needs next across four core tasks: code2test, comment2context, trace2code, and edit2ripple, alongside a selective retrieval suite with natural no-gold and counterfactual controls. The benchmark comprises 427 curated samples across 25 repositories over frozen base commits. Evaluating lexical, dense embedding, and RepoMap baselines reveals sharp task-dependent trade-offs and limited selective success on natural no-gold labels. Logged agent trajectories miss all gold files on 27–35% of samples. In a 45-sample, single-run pilot, retrieval-derived seeds achieve higher final File F1 with fewer post-seed calls and read tokens than random non-gold seeds.

1 INTRODUCTION

Coding agents are often judged at the end of the workflow: did the generated patch pass tests, resolve the issue, or satisfy a benchmark oracle? This end-to-end measurement is useful, but it hides the context-acquisition layer that precedes editing. Before an agent can reason about a change, it must identify the files that matter. Modern agents may do this through a mixture of codebase indexes, repository maps, grep-like tools, file reads, and iterative search (Aider, n.d.; Yang et al., 2024), but the underlying question is the same: which repository context should enter the model before it edits?

This makes agentic repository retrieval different from generic semantic code search. In traditional code search, relevance is usually defined by query-document semantic similarity, such as a natural-language description matching a function. In coding-agent workflows, relevance is defined by the next useful step in the software task. A pull request may describe an implementation change while the useful context is a regression test. A review comment may point at one file while the missing constraint lives in another module. A failure trace may mention a test file or stack frame while the root cause is an implementation file that does not look semantically similar to the error text. Agentic relevance is therefore often indirect, structural, or workflow-dependent.

Interactive exploration does not remove the need to evaluate this layer; it changes the cost model. An agent can compensate for a weak initial context by spending more tool calls, tokens, and latency on search. In our logged trajectory track over the 287 `code2test`, `comment2context`, and `trace2code` samples, an OpenAI strict-context agent reads 3.2 files per sample on average, while Codex CLI recovered contexts use about 6 file/path events per sample. Yet these trajectories still never touch any gold file on 35.2% of OpenAI samples and 27.2–29.3% of Codex samples. Conditional on samples where gold is eventually reached, the median first hit is step 2 for OpenAI and step 3 for Codex. These results show why static top-k scores should not be treated as full agent performance, while establishing context acquisition as a measurable upstream failure surface and cost center.

Agent Retrieval Bench isolates this upstream problem. Agent Retrieval Bench is a file-level benchmark: given a workflow-derived query and a repository corpus at the base commit, a retriever or context engine must rank files that an agent would need to read. The benchmark is deliberately not patch-level. It does not ask whether an agent can synthesize the final edit. Instead, it measures

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

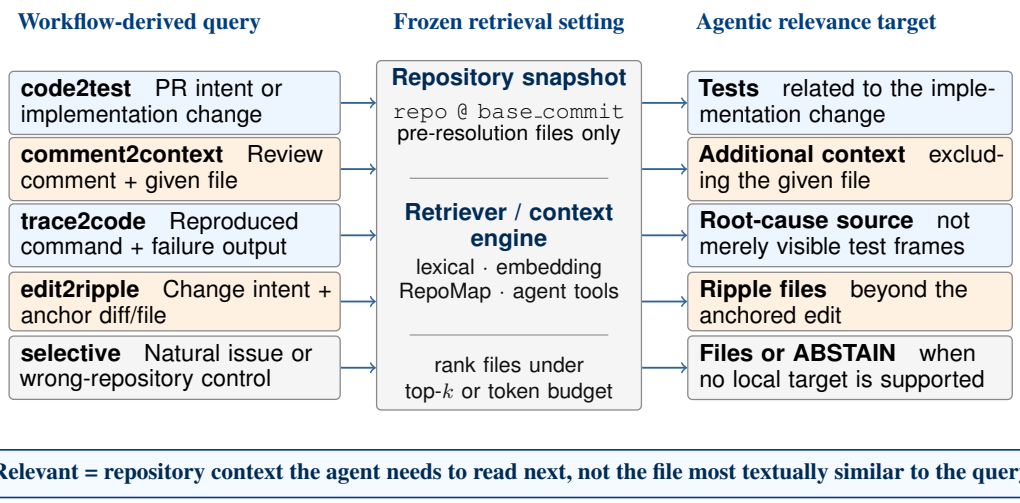

Figure 1: Agent Retrieval Bench protocol and task-specific agentic relevance. Every method receives a workflow-derived query and the repository frozen at its base commit, then ranks files or abstains. Positive targets represent the context needed for the next useful workflow step; selective cases test abstention under the resolution-target convention in Section 4.

whether the context-acquisition layer can find useful files before editing begins, and whether it can do so under realistic rank and context-budget constraints.

At the component level, we compare context acquisition and exploration cost across initial-context selections under a fixed agent scaffold. We do not claim that retrieval is the dominant cause of patch failure, that File F1 is a substitute for test-passing repair, or that the present results estimate the effect of retrieval on patch success. Establishing that link requires an executable repair benchmark with aligned outcome labels, which the current workflow-derived samples do not provide.

The benchmark contains four workflow-grounded positive-retrieval tasks. `code2test` asks for related tests from PR or implementation-change signals. `comment2context` asks for additional context files beyond the reviewed file. `trace2code` asks for root-cause source files from reproduced failure output, while treating visible tests as auxiliary context rather than main gold. `edit2ripple` provides an anchored change and asks for additional files affected by that change. A fifth selective subset combines 50 evidence-backed natural no-gold cases with 32 counterfactual wrong-repository controls. Each sample is evaluated against a base-commit corpus, preventing fixed-code leakage. Figure 1 summarizes the shared protocol and the task-specific meaning of relevance.

This paper makes four contributions:

1. We formalize *agentic relevance* as the next repository context needed by a coding workflow, and operationalize it in four positive tasks plus natural and counterfactual no-gold retrieval.
2. We construct and validate 427 samples against frozen base-commit corpora, with gold evidence, schema/corpus checks, and explicit controls for path, patch, commit, and generated-content leakage.
3. We define an agent-oriented evaluation suite that combines ranked-file retrieval, token-budget BCY curves, selective abstention, trajectory cost, and a controlled-input context-seed pilot.
4. We show empirically that no retrieval family dominates across workflow signals or context budgets: structural and semantic methods are complementary, Qwen3-4B and Qwen3-8B reverse order across `edit2ripple` budgets, and retrieval-derived seeds are more context-efficient than random context in the single-run intervention pilot.

The goal is not to claim that bug localization, test traceability, or code review assistance are new in isolation. The contribution is a unified evaluation semantics and hygiene standard for the context-acquisition layer used by coding agents.

Table 1: Positioning relative to nearby repository localization and agentic retrieval settings.

Work	Query signal	Target relevance	Distinguishing evaluation focus
SWE-bench / Agentless	Issue statements	Edit locations or files	Patch-pipeline localization
LocAgent	Issue statements	Code entities / locations	Graph-guided multi-hop localization
CORE-Bench	User and issue requests	Code, edit sites, context	Scale and broad retrieval coverage
SWE-Explore	Issue statements	Line regions from successful trajectories	Line-budget exploration and downstream repair association
Agent Retrieval Bench	PR, review, trace, and anchored-edit signals	Next-context files or abstention	Workflow semantics, base-commit hygiene, BCY/PES, selective retrieval

2 Related Work

Code search and repository context. CodeSearchNet and the code-search tasks in CodeXGLUE established benchmarks for natural-language–code matching, typically at the function or snippet level (Husain et al., 2019; Lu et al., 2021), while RepoBench evaluates cross-file retrieval, code completion, and their combined pipeline (Liu et al., 2024). Agent Retrieval Bench instead evaluates file-level next-context retrieval from multiple coding-workflow signals, rather than snippet retrieval for next-line prediction. Relevance may therefore follow semantic, structural, workflow-conventional, or causal relations rather than direct query–code similarity. Our task suite also intersects with software traceability and fault localization (Cleland-Huang et al., 2012; Just et al., 2014; Akbar & Kak, 2020), but casts them as instances of workflow-conditioned next-context retrieval.

Coding-agent evaluation and localization. SWE-bench evaluates issue resolution through executable tests (Jimenez et al., 2024), while SWE-agent studies how agent-computer interfaces affect performance on such tasks (Yang et al., 2024). Agentless and LocAgent explicitly study localization within repair workflows (Xia et al., 2024; Chen et al., 2025). These works include localization analyses, but their task setting centers on issue resolution. Agent Retrieval Bench evaluates file-level context acquisition across multiple workflow signals, reporting retrieval quality and exploration cost separately from patch success.

Agentic retrieval and exploration. Closest to our setting are CORE-Bench and SWE-Explore. CORE-Bench evaluates large-scale retrieval from issue and user-request inputs, including edit sites and broader context (Zhang et al., 2026a). SWE-Explore evaluates adaptive line-level repository exploration under a fixed line budget, derives relevance evidence from successful repair trajectories, and relates exploration metrics to downstream repair behavior (Zhang et al., 2026b). Agent Retrieval Bench instead focuses on file-level retrieval from PR, review-comment, failure-trace, and anchored-edit signals. Its relevance targets depend on the workflow and context already supplied: in `comment2context`, returning the reviewed file earns no credit because the target is additional context; in `trace2code`, visible test frames provide evidence but do not automatically qualify as root-cause source targets; and in `edit2ripple`, the supplied edit anchor is excluded from gold. These distinctions make the missing context, rather than the most conspicuous file in the query, the object of evaluation. Agent Retrieval Bench combines these task-specific targets with frozen pre-resolution corpora and explicit shortcut diagnostics. Its selective track adds evidence-backed natural no-gold cases and wrong-repository controls, testing whether to return files when no local resolution target is supported. This convention does not imply that all local files lack diagnostic value. These choices complement broader issue-centered retrieval and line-level exploration; file-level granularity or base-commit freezing alone is not the claimed novelty. Table 1 summarizes the differences in query signals, relevance targets, and evaluation focus. Our contribution is therefore a unified workflow semantics and evaluation protocol for the context-acquisition layer, rather than a new formulation of each underlying localization task.

3 PROBLEM DEFINITION

Task setting. A sample is $x_i = (q_i, \mathcal{C}_i, H_i, G_i)$: a workflow-derived query, a candidate corpus frozen at a repository base commit, optional given-context files, and labeled gold files. Positive samples use the repository state before the resolving change. A retriever receives q_i, including any given-context information, and ranks files in $\mathcal{C}_i$. The targets satisfy $G_i \subseteq \mathcal{C}_i$ and $G_i \cap H_i = \varnothing$: returning the already-reviewed file or edit anchor earns no gold credit. Selective retrieval additionally allows abstention; $G_i = \varnothing$ follows the resolution-target convention in Section 4. The next section specifies task-level inputs and targets; Appendix B provides the schema and real examples.

Agentic relevance. Reading an *agentically relevant* file would materially help the agent take the next correct workflow step, even without direct query–file semantic similarity. We analyze four overlapping relations: semantic-direct, structural-indirect, workflow-conventional, and causal-indirect. Appendix G.3 defines their operational predicates and projection into analysis groups. Gold labels are supported by task-specific evidence and may not exhaust all useful files.

Evaluation. For positive samples, we report Recall@k, $k \in \{5, 10, 20\}$, MRR, and Budgeted Context Yield (BCY). Recall measures the fraction of gold files in the top k unique files; MRR uses the reciprocal rank of the first gold file. For BCY, we pack available corpus file text in rank order after path deduplication, count headers toward the token budget B, and prefix-truncate the boundary file. Let $E_i(B)$ contain the files with at least one non-header content token packed. Over the N_+ positive samples $\mathcal{I}_+ = \{i : |G_i| > 0\}$,

$$\text{BCY@}B = \frac{1}{N_+} \sum_{i \in \mathcal{I}_+} \frac{|G_i \cap E_i(B)|}{|G_i|}.$$

BCY measures file exposure, not within-file localization or comprehension. All methods share a deterministic regex tokenizer; Appendix C specifies rendering, token accounting, and minimum-content sensitivity. We report 4k/8k/16k/32k curves and BCY@8k in compact tables. Selective evaluation and trajectory diagnostics are specified in Section 8.5 and Appendix H, respectively.

4 BENCHMARK TASKS

`code2test` simulates a coding agent receiving a PR or implementation-change signal and needing to identify relevant tests. The query may mention implementation files or behavior, but construction excludes exact test paths and final-patch evidence. The gold files are tests linked by released PR/change evidence.

`comment2context` simulates a code review setting. The agent sees a review comment and the reviewed file. The benchmark asks for additional files needed to understand or satisfy the comment. The reviewed file is treated as given context, not as gold.

`trace2code` simulates a reproduced failure. The query contains the command and failure excerpt. The gold files are root-cause source files. Tests and failure frames can be supporting context, but they are not counted as main gold unless they are supported as the root cause.

`edit2ripple` simulates impact analysis after a localized edit is known. The query contains an anchor file, an anchor diff, and the change intent. The anchor file is given context rather than gold; the retriever must find additional source or test files affected by the change. Gold files are backed by changed-file evidence and recorded ripple relations such as shared symbols, component proximity, or required test updates.

`Selective No-Gold Retrieval` The no-gold subset uses a resolution-target convention. Fifty *natural* cases have maintainer evidence attributing resolution to an upstream dependency, external service, or user error, with no supported local fix target after screening for local-fix signals and reviewing the evidence. We assign $G_i = \varnothing$ to reflect that resolution evidence; local files may still be useful for diagnosis. Thirty-two *counterfactual wrong-repository* controls pair queries with repositories unrelated to their resolution. The strata are evaluated separately: natural cases test abstention under externally resolved labels, while counterfactuals test gross query–repository mismatch. Neither stratum provides exhaustive judgments of every file's diagnostic utility.

Table 2: Dataset composition. Positive tasks use file-level retrieval metrics; no-gold samples are evaluated by selective abstention.

Subset	Evaluation role	Samples
`code2test`	positive retrieval	106
`comment2context`	positive retrieval	80
`trace2code`	positive retrieval	101
`edit2ripple`	positive retrieval	58
`abstention-natural`	no-gold decision	50
`abstention-counterfactual`	wrong-repository control	32
Positive subtotal		345
Total		427

5 Dataset Construction

Source collection. We mine public GitHub records from 25 purposively selected repositories with reconstructible Git history and usable PR, review, issue, or CI evidence. Crawls are bounded per repository and ordered by recent artifact updates, without a common date window. Available source URLs, identifiers, and timestamps are retained.

Query and gold construction. Queries are extracted from workflow records. `code2test` selects merged PRs changing both implementation and existing tests, using sanitized PR/change text as queries and evidence-linked tests as gold. `comment2context` pairs inline reviews with additional base-existing files changed by the first valid post-comment response commit, requiring relation evidence and excluding the reviewed file. `trace2code` applies test-side changes at the base commit to reproduce failures, uses the command and output as the query, and links source targets to resolving-change evidence. `edit2ripple` supplies an anchor edit and labels other changed files supported by ripple relations. Natural no-gold cases require maintainer evidence of external resolution; counterfactual controls pair queries with deliberately wrong repositories. A separate model-assisted audit checks label evidence without generating queries and is independent of the evaluated retrievers. Sanitization removes target-revealing fix content while retaining the permitted anchor diff; Section 6 details the schema, corpus, and leakage checks. These checks do not establish exhaustive gold coverage.

Corpus and composition. Corpora are frozen at sample base commits, preceding the resolving change for positives; every positive target must exist in its corpus. The 427 samples in Table 2 span six primary languages and use 271 repo/base pairs. Appendix A lists repositories, languages, and per-task counts. The combined manifest contains 308 snapshots across 29 repositories, 391,932 files, and 7,922,369 chunks; its remaining 37 snapshots are unused by these samples. Evaluation searches the full file corpus without task-specific candidate filtering. Implementation cues, reviewed files, and stack frames remain visible as potential distractors, with hard-negative annotations retained when available.

6 Quality and Leakage Controls

Gold evidence and corpus validity. Table 3 summarizes evidence coverage and automated checks. All positive samples link gold files to workflow and base-commit evidence: `code2test`, `comment2context`, and `trace2code` retain span/block annotations; `edit2ripple` retains changed-file ripple evidence. Natural no-gold cases retain maintainer-resolution evidence, with counterfactual controls identified separately. Schema and corpus checks find no invalid rows. Evidence traceability and corpus membership do not independently establish semantic correctness or exhaustive gold coverage.

Shortcut diagnostics. We detect zero cases in the four fatal categories listed in Table 3. Patch/diff checks target answer-revealing fix content; the permitted `edit2ripple` anchor diff and visible

Table 3: Dataset validity diagnostics. Fatal leakage categories are exact shortcuts that would directly reveal a positive sample's answer.

Diagnostic	Result
Total samples	427
Evidence-backed positive gold	345 / 345
Natural evidence-backed no-gold decisions	50 / 50
Counterfactual wrong-repository controls	32 / 32
Schema or corpus invalid rows	0
Exact gold-path leakage	0
Raw patch / diff leakage	0
Fix-commit hash leakage	0
Autogenerated suggestion/review-artifact leakage	0
Positive task families with non-trivial method spread	4 / 4

failure frames remain valid inputs. Generated-artifact screening sanitizes source records and does not imply model-generated queries. The zero counts apply to these specified shortcut checks, not to every possible cue or source of leakage.

7 Baselines

We compare lexical, structural, and embedding retrieval. BM25 uses Robertson/Sparck Jones IDF and length normalization (Robertson & Zaragoza, 2009); `lexical` is a separate TF-IDF-style heuristic with path, basename, and symbol bonuses. RepoMap ranks files using repository structure, symbols, paths, and references. The five embedding models are Qwen3-4B and Qwen3-8B (Zhang et al., 2025), jina-code-embeddings-0.5b (Kryvosheieva et al., 2025), pplx-embed-v1-4b (Eslami et al., 2026), and nomic-embed-code (Suresh et al., 2025). Logged agent context selection is reported separately.

Embeddings use a common prompt-free interface: no added query or passage prefix, L2 normalization, cosine chunk scores, and maximum-score aggregation to unique files. These are zero-shot baselines without model-specific prompt tuning. Appendix D provides full checkpoint names, revisions, input rendering, pooling, sequence limits, and implementation caveats.

All positive-task runs search the full file corpus. The 345-sample leaderboard reports sample-weighted and unweighted task-macro averages; repository-macro scores average within each repository, then equally across 25 repositories. Taxonomy, RRF, and trajectory joins use the 287 `code2test`, `comment2context`, and `trace2code` samples with aligned annotations and artifacts. The 82 no-gold samples enter selective retrieval only and are excluded from positive-only Recall and MRR.

8 Main Results

8.1 Overall and Task-Specific Retrieval

On all 345 positive samples, the sample-weighted leaders differ by metric: Qwen3-4B leads MRR (0.2379), Qwen3-8B Recall@20 (0.7029), and RepoMap BCY@8k (0.3788; Table 4). Task-macro averaging retains the two Qwen rank-metric leaders but changes the BCY leader to Qwen3-8B. Repository weighting also matters: the four largest repositories contribute 58.8% of samples, and weighting all 25 repositories equally makes Qwen3-4B the Recall@20 leader (0.6344 versus 0.6193 for Qwen3-8B). Appendix G.1 gives the full sensitivity analysis.

Task-level winners likewise vary (Table 5). Qwen3-4B leads `code2test`, Jina leads `comment2context` MRR, and RepoMap leads `trace2code`. The latter provides the sharpest reversal: RepoMap reaches MRR 0.2742 and Recall@20 0.8366, whereas the aggregate MRR leader Qwen3-4B reaches only 0.0827 MRR. Lexical MRR (0.2075) also exceeds the embedding baselines on this task. Detailed `trace2code` rankings are in Appendix Table 15. The pplx `edit2ripple` MRR lead remains provisional pending the tokenizer correction noted in Appendix D.

Table 4: Complete positive leaderboard over 345 samples. Weighted metrics are sample-weighted over all four tasks; M-R@20, M-MRR, and M-BCY are unweighted task-macro averages. BCY uses canonical token-based file packing. The pplx row is provisional because of the tokenizer caveat in Appendix D.

Model	R@5	R@10	R@20	MRR	BCY@8k	M-R@20	M-MRR	M-BCY
Qwen3-4B	0.3182	0.4434	0.6306	**0.2379**	0.3409	0.6369	**0.2441**	0.3575
Qwen3-8B	**0.3565**	**0.5428**	**0.7029**	0.2336	0.3732	**0.6988**	0.2396	**0.3892**
pplx-4B†	0.3066	0.4450	0.6072	0.2267	0.3549	0.6136	0.2347	0.3712
RepoMap	0.3134	0.4805	0.6333	0.2158	**0.3788**	0.6230	0.2150	0.3717
Nomic	0.2777	0.3741	0.5244	0.1986	0.2781	0.5266	0.2113	0.2954
Jina-0.5B	0.2449	0.3355	0.4823	0.1914	0.2783	0.5042	0.2024	0.2953
Lexical	0.2292	0.3548	0.4940	0.1574	0.2650	0.5072	0.1674	0.2840
BM25	0.1899	0.3034	0.4452	0.1520	0.2051	0.4569	0.1551	0.2126

Table 5: Positive-task winners by MRR and Recall@20. The aggregate row uses all 345 positive samples.

Task	Best MRR Model	MRR	Best R@20 Model
Overall positive	Qwen3-4B	0.2379	Qwen3-8B
`code2test`	Qwen3-4B	0.3225	Qwen3-4B
`comment2context`	Jina-0.5B	0.3043	Qwen3-8B
`trace2code`	RepoMap	0.2742	RepoMap
`edit2ripple`	pplx-4B†	0.2877	Qwen3-4B

8.2 Retrieval under Context Budgets

The 58-sample `edit2ripple` task separates ranking depth from budgeted file exposure (Figure 2). Qwen3-4B has the highest Recall@20 (0.7112), but the two Qwen models reverse BCY order between fully packed budgets: at 4k tokens, Qwen3-4B scores 0.3707 versus 0.3333 for Qwen3-8B; at 8k, Qwen3-8B scores 0.5129 versus 0.5043. The provisional pplx run is higher at 4k. Values at 16k and 32k can underfill the budget with the stored top-20 prefix and are therefore lower bounds; they do not establish definitive larger-budget winners. Appendix Table 12 gives all rank and budget values.

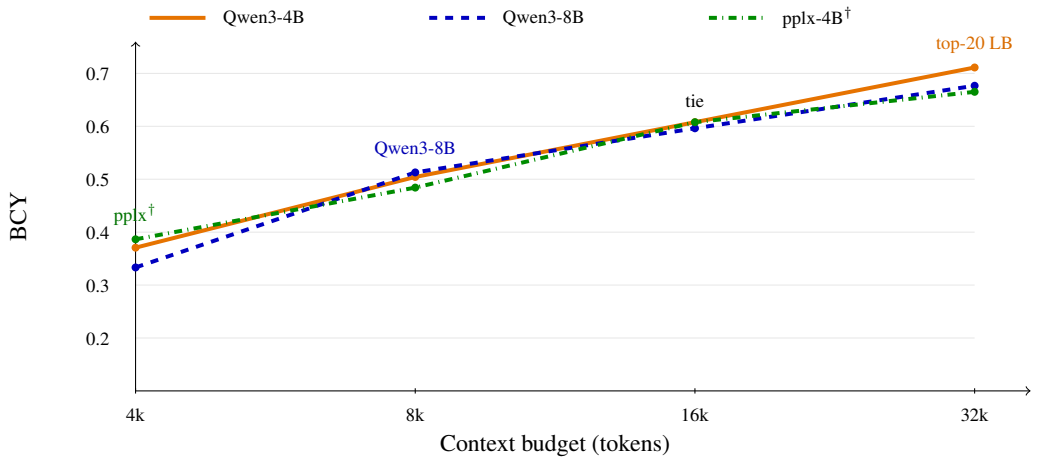

Figure 2: Canonical BCY curves for selected embedding models on `edit2ripple`. The verified Qwen ordering reverses between fully packed 4k and 8k budgets. † marks the provisional pplx run; 16k/32k points can be lower bounds from stored top-20 rankings.

BCY conclusions are insensitive to the minimum exposure threshold: increasing it from one to 128 content tokens preserves method ordering on both the three-task set and `edit2ripple`, with

Table 6: Logged context-acquisition cost on the 287 `code2test`, `comment2context`, and `trace2code` samples. Any-gold is the fraction of samples where any gold file appears in the logged context events. Median first hit is computed over successful samples. Codex rows use recovered file/path events, so they are cost proxies rather than wall-clock measurements.

Method	Events/sample	Any-gold	Median first hit
OpenAI GPT-5.4-mini strict	3.2	0.648	2
Codex CLI GPT-5.4	6.2	0.728	3
Codex CLI GPT-5.5	6.5	0.707	3

maximum absolute decreases of 0.0081 and 0.0058, respectively (Appendix G.5). File exposure still leaves substantial within-file search: across 345 positives the median gold file has 515 lines, while the median evidence-to-file ratio on the 287 span-annotated samples is 4.7%. A separate line-overlap diagnostic changes method orderings, so file-level success should not be interpreted as accurate span localization. Its legacy character-budget protocol and full results are reported in the same appendix.

8.3 Relevance Relations and Hybrid Retrieval

The taxonomy describes semantic-direct, structural-indirect, workflow-conventional, and causal-indirect query–gold relations on 287 samples. Relations overlap in 94.8% of samples; the disjoint primary projection is determined by task-conditioned rules, not exclusive membership. Direct and workflow-conventional slices favor embedding-heavy or fused rankers, while RepoMap leads causal-indirect MRR. The structural slice has only 23 samples. These are descriptive associations confounded with task identity; Appendix G.3 gives the predicates, precedence rules, and slice results.

To test whether the retrieval families can be combined without training, reciprocal-rank fusion (RRF) assigns $s_{\mathrm{RRF}}(f) = \sum_{m \in \mathcal{M}} (60 + r_m(f))^{-1}$, where $r_m(f)$ is a one-based file rank and an absent file contributes zero (Cormack et al., 2009). Fusion uses each input's stored top-20 list. On the same 287 samples, combining Qwen3-8B and RepoMap improves MRR from the best single-run value of 0.2296 to 0.2713 and Recall@20 from 0.7070 to 0.7331. It still trails Qwen3-4B on `code2test` MRR. Appendix G.4 provides per-task results; RRF-3 denotes the three-way Qwen3-4B, Qwen3-8B, and RepoMap fusion used in the taxonomy analysis.

Across the 287 `code2test`, `comment2context`, and `trace2code` samples, only 22 are missed by every baseline included in the top-20 analysis (Appendix G.4). This across-method union of hits further supports complementarity; it is not a single-system score.

8.4 Trajectory Cost and Seed Intervention

Across 287 samples, logged agents never touch any gold file on 35.2% of OpenAI strict-context runs and 27.2–29.3% of Codex runs (Table 6). Median first hits of two or three events are conditional on eventual gold contact. Codex events are recovered file/path observations, so these costs are proxies. On cases where Codex reaches gold only after more than three events, Qwen3-8B already places gold in its top 20 in 81.8–86.3% of samples. The full retriever–agent complementarity matrix includes both retriever-only and agent-only hits (Appendix H); neither method subsumes the other.

We intervene on initial context in a 45-sample pilot, with 15 samples from each of `code2test`, `comment2context`, and `trace2code`. Docker-isolated Codex GPT-5.5 uses a fixed closed-tool policy with at most 16 calls, an 8k total read-token budget including seed context, and a final three-file set. The arms are no seed, random non-gold, lexical, Qwen3-8B, RRF, and oracle gold. Each sample–arm pair has one trajectory; sampling was not repeated. Appendix H.2 specifies the remaining limits and paired results.

Random context already raises File F1 from 0.3222 to 0.3437. Lexical and RRF seeds reach 0.3981 and 0.3967 with fewer calls and post-seed tokens than random, while oracle gold reaches 0.6337. Seed lengths differ and consume different shares of the same read budget (Table 7); the results describe the tested context inputs and remaining exploration budgets rather than an isolated effect of retrieval quality. Some seeded runs worsen File F1, and small arm differences are not statistically resolved. The policy cannot edit or test code, so the pilot measures context acquisition and does not

Table 7: Closed-tool seed pilot: 45 samples across `code2test`, `comment2context`, and `trace2code`, one trajectory per sample–arm pair, and the same Codex GPT-5.5 policy. Any-gold includes seed and read hits. Med. hit is the median first-gold tool-call index among hit runs; seed hits count as 0 and misses are excluded. Post-seed tokens exclude seed context. Other score and cost columns are sample means; see Appendix H.2.

Arm	n	Final F1	ΔF1	Any-gold	Med. hit	Calls	Post toks	Seed toks
No seed	45	0.3222	0.0000	0.5111	3.0	3.71	2137.3	0.0
Random	45	0.3437	0.0215	0.7333	3.0	8.49	3681.7	2427.8
Lexical	45	0.3981	0.0759	0.8000	2.0	6.24	2243.9	3061.1
Qwen8B	45	0.3726	0.0504	0.7556	1.0	5.51	2047.7	3204.2
RRF	45	0.3967	0.0744	0.8000	1.0	5.42	1856.7	3298.8
Oracle	45	0.6337	0.3115	1.0000	0.0	4.18	1736.4	1781.3

Table 8: Repo-grouped selective retrieval with mixed and natural evidence-only no-gold protocols. Selective success@20 counts a no-gold abstention as success and an accepted positive as success only when any gold file appears in the top 20. No-abs. success is the corresponding score when every query returns files.

Protocol	Ranker	Pos. pass	No-gold abs.	Bal. acc	Succ.@20	No-abs. succ.	Coverage
Mixed	lexical	0.814	0.598	**0.706**	**0.496**	0.461	0.735
Mixed	Jina-0.5B	0.446	**0.793**	0.620	0.389	0.452	0.401
Mixed	BM25	0.672	0.451	0.562	0.377	0.429	0.649
Natural only	lexical	0.423	0.580	0.502	0.294	**0.499**	0.423
Natural only	Jina-0.5B	0.377	0.940	**0.658**	**0.334**	0.489	0.337
Natural only	BM25	0.188	**0.980**	0.584	0.220	0.463	0.167

establish patch-success gains. Auxiliary PES calibration and observational BCY correlations are reported separately in Appendix H.3.

8.5 Selective Retrieval with Abstention

Lexical, BM25, and Jina abstain when their top retrieval score falls below a threshold. We use repository-grouped five-fold cross-validation: four folds select the threshold maximizing balanced accuracy, which is applied once to held-out repositories. The mixed protocol uses 345 positives, 50 natural no-gold cases, and 32 counterfactual controls. The natural-only protocol removes the counterfactuals from both calibration and evaluation and rebuilds the folds over 395 samples.

Mixed lexical selective success@20 improves from 0.461 to 0.496, but lexical abstains on every counterfactual control and only 34% of natural no-gold cases under those decisions. After natural-only recalibration, all three rankers perform worse than always returning a ranking (Table 8). High no-gold abstention alone is insufficient: Jina correctly abstains on 94.0% of natural negatives but passes only 37.7% of positives. Under the resolution-target labels of Section 4, top-score thresholds optimized for balanced accuracy do not improve selective success. This evaluates agreement with those labels, not the absence of useful diagnostic context; pooling easy wrong-repository controls obscures the distinction.

9 Conclusion

Agent Retrieval Bench evaluates the repository context coding agents need from workflow signals, using frozen base-commit corpora, file-level gold, and no-gold cases. Results vary across tasks, context budgets, and repository weighting. Semantic and structural retrieval are complementary: simple rank fusion improves aggregate performance on the three-task subset. Natural no-gold cases expose weaknesses in score-threshold abstention. A 45-sample, single-run seed pilot also finds better context selection with less post-seed exploration than random context. Together, these findings motivate evaluating retrieval coverage, budgeted file exposure, and abstention as distinct aspects of context acquisition.

REFERENCES

Aider. Aider repository map documentation, n.d. URL https://aider.chat/docs/repomap.html. Accessed 2026-09-11.

Shayan A. Akbar and Avinash C. Kak. A large-scale comparative evaluation of IR-based tools for bug localization. In *Proceedings of the 17th International Conference on Mining Software Repositories*, pp. 21–31, 2020. doi: 10.1145/3379597.3387474.

Zhaoling Chen, Xiangru Tang, Gangda Deng, Fang Wu, Jialong Wu, Zhiwei Jiang, Viktor Prasanna, Arman Cohan, and Xingyao Wang. LocAgent: Graph-guided LLM agents for code localization. *arXiv preprint arXiv:2503.09089*, 2025. URL https://arxiv.org/abs/2503.09089.

Jane Cleland-Huang, Orlena Gotel, and Andrea Zisman (eds.). *Software and Systems Traceability*. Springer London, 2012. doi: 10.1007/978-1-4471-2239-5.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 758–759, 2009. doi: 10.1145/1571941.1572114.

Sedigheh Eslami, Maksim Gaiduk, Markus Krimmel, Louis Milliken, Bo Wang, and Denis Bykov. Diffusion-pretrained dense and contextual embeddings. *arXiv preprint arXiv:2602.11151*, 2026. URL https://arxiv.org/abs/2602.11151.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. CodeSearchNet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*, 2019. URL https://arxiv.org/abs/1909.09436.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002. doi: 10.1145/582415.582418.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

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*, pp. 437–440, 2014. doi: 10.1145/2610384.2628055.

Daria Kryvosheieva, Saba Sturua, Michael Günther, Scott Martens, and Han Xiao. Efficient code embeddings from code generation models. *arXiv preprint arXiv:2508.21290v1*, 2025. URL https://arxiv.org/abs/2508.21290v1.

Tianyang Liu, Canwen Xu, and Julian McAuley. RepoBench: Benchmarking repository-level code auto-completion systems. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=pPjZIOuQuF.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie Liu. CodeXGLUE: A machine learning benchmark dataset for code understanding and generation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c16a5320fa475530d9583c34fd356ef5-Abstract-round1.html.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Tarun Suresh, Revanth Gangi Reddy, Yifei Xu, Zach Nussbaum, Andriy Mulyar, Brandon Duderstadt, and Heng Ji. CoRNStack: High-quality contrastive data for better code retrieval and reranking. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=iyJOUELYir.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying LLM-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024. URL https://arxiv.org/abs/2407.01489.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Fuwei Zhang, Yanzhao Zhang, Mingxin Li, Dingkun Long, Lexiang Hu, Pengjun Xie, Zhao Zhang, and Fuzhen Zhuang. CORE-Bench: A comprehensive benchmark for code retrieval in the era of agentic coding. *arXiv preprint arXiv:2606.11864*, 2026a. URL https://arxiv.org/abs/2606.11864.

Shaoqiu Zhang, Yuhang Wang, Jialiang Liang, Yuling Shi, Wenhao Zeng, Maoquan Wang, Shilin He, Ningyuan Xu, Siyu Ye, Kai Cai, and Xiaodong Gu. SWE-Explore: Benchmarking how coding agents explore repositories. *arXiv preprint arXiv:2606.07297*, 2026b. URL https://arxiv.org/abs/2606.07297.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. URL https://arxiv.org/abs/2506.05176.

AI USE STATEMENT

We used generative AI tools, including OpenAI Codex and Claude, to assist with developing the evaluation framework and metrics, refining experimental designs, implementing data processing and evaluation code, auditing candidate samples, producing diagnostic annotations, and interpreting results. These tools also assisted with literature review, manuscript drafting, editing, and formatting. The roles of model-assisted auditing and diagnostic annotation are described in the paper. The authors take responsibility for the final manuscript, experimental claims, and associated artifacts.

A REPOSITORY AND LANGUAGE COVERAGE

Table 9 lists the 25 sample-bearing repositories, assigning each its primary implementation language. Counterfactuals are counted under the searched repository. The set covers 13 Python repositories, three each for Go, Rust, and TypeScript, two Java repositories, and one JavaScript repository. Go accounts for 142 of 345 positives, including 88 from Gin.

Table 9: Repository, primary language, and per-task sample counts. C2T=`code2test`, C2C=`comment2context`, T2C=`trace2code`, E2R=`edit2ripple`, Nat.=natural no-gold, and Ctr.=counterfactual wrong-repository.

Repository	Language	C2T	C2C	T2C	E2R	Nat.	Ctr.	All
`HypothesisWorks/hypothesis`	Python	0	1	0	0	0	5	6
`astral-sh/ruff`	Rust	0	6	0	0	1	3	10
`caddyserver/caddy`	Go	0	0	7	6	3	0	16
`clap-rs/clap`	Rust	4	0	1	5	0	1	11
`eslint/eslint`	JavaScript	0	1	0	0	1	4	6
`etcd-io/etcd`	Go	15	16	4	6	0	0	41
`fastapi/fastapi`	Python	3	0	0	0	0	0	3
`gin-gonic/gin`	Go	15	0	56	17	0	0	88
`huggingface/diffusers`	Python	6	9	0	1	0	2	18
`huggingface/transformers`	Python	15	13	0	8	0	0	36
`ipython/ipython`	Python	0	2	0	0	0	2	4
`microsoft/playwright`	TypeScript	0	4	0	0	3	1	8
`mockito/mockito`	Java	1	0	0	0	0	0	1
`numpy/numpy`	Python	0	1	0	0	5	1	7
`pallets/click`	Python	0	0	26	0	0	0	26
`pydantic/pydantic`	Python	0	1	0	0	0	1	2
`pypa/pip`	Python	0	5	0	0	4	2	11
`pytest-dev/pytest`	Python	3	2	1	1	14	5	26
`python/mypy`	Python	0	1	0	0	0	0	1
`scrapy/scrapy`	Python	0	3	0	0	3	0	6
`spring-projects/spring-boot`	Java	5	0	0	3	0	0	8
`tokio-rs/tokio`	Rust	15	11	6	6	2	0	40
`tox-dev/tox`	Python	0	2	0	0	0	3	5
`vitejs/vite`	TypeScript	19	1	0	5	14	2	41
`vuejs/core`	TypeScript	5	1	0	0	0	0	6
Total	–	106	80	101	58	50	32	427

B SAMPLE FIELDS AND EXAMPLES

Each JSONL record identifies a sample, task, repository, base commit, query, gold, and supporting evidence. Table 10 lists task-specific fields; optional `gold_spans` and `gold_blocks` locate supporting regions. Negative diagnostics use `hard_negative_files`; the unscored construction field `gold.negative_distractors` can overlap gold and is not an evaluated negative set. The examples below are excerpts, not complete records. Original query spelling is retained; source links identify their provenance.

`code2test`. clap-rs/clap PR #6319 changes `clap_mangen/src/render.rs`. **Query excerpt:** *fix(man)!: Put env support behind the env feature*. **Gold:** `clap_mangen/tests/testsuite/common.rs` and `clap_mangen/tests/testsuite/roff.rs`. **Evidence:** the same PR changes implementation and existing tests, with behavior and source–test path-overlap witnesses. The test paths are not in the query.

Table 10: Task-specific target and evidence fields in the released JSONL. Optional span fields provide finer-grained supporting regions but do not change the file-level gold set.

Task	Target fields	Relevance or abstention witnesses
code2test	gold.related_tests	metadata.evidence.source, signals, implementation_files, and related_tests
comment2context	gold.given_files; gold.must_context_files[].path	must_context_files[].evidence; metadata.evidence.gold_definition, review-comment and response-commit identifiers, and post_comment_commits
trace2code	gold.root_cause_files	metadata.evidence.signals, run_status, test_files, and combined_log
edit2ripple	gold.given_files; gold.files	metadata.gold_evidence[].path, signals, is_test, additions, and deletions
Natural no-gold	gold.files=[]; gold.no_gold=true; gold.reason	metadata.organic=true, evidence_summary, and evidence_urls
Counterfactual no-gold	gold.files=[]; gold.no_gold=true; gold.reason	metadata.organic=false, source_repo, source_sample_id, pairing_profile, and evidence_urls

comment2context. tokio-rs/tokio PR #7686 provides `tokio/tests/sync_mpsc.rs` as given context. **Query excerpt:** Please also test the `drop(tx)` case. **Gold:** `tokio/src/sync/mpsc/chan.rs`, excluding the given test. **Evidence:** `modified_after_review_comment`, `implementation_context_for_reviewed_change`, and `explicit_behavior_or_api_dependency` connect the response change to the review. The active hard-negative set does not contain `chan.rs`.

trace2code. gin-gonic/gin PR #4404 is reproduced with `go test ./..`. **Failure excerpt:** `./debug_test.go:118:2: undefined: runtimeVersion`. **Gold:** `debug.go`, not the visible `debug_test.go`. **Evidence:** `test_only_patch_applied`, `failure_observed`, and `compile_error` record the base-commit failure.

edit2ripple. clap-rs/clap PR #6129 provides `clap_builder/src/parser/parser.rs` and its anchor diff. **Intent excerpt:** feat: Add `default_values_if` and `default_values_ifs`. **Gold:** `clap_builder/src/builder/arg.rs` and `tests/builder/default_vals.rs`. **Evidence:** path overlap and shared changed symbols link the anchor to the API declaration and tests; the anchor itself earns no credit.

Natural no-gold. pytest-dev/pytest issue #13024 has the title *SystemError occured in Windows Python 3.13.0.* **Query excerpt:** *I wonder if this is just a bug with Python 3.13.0 and not pytest's.* **Gold:** `files=[]`, `no_gold=true`, `reason=upstream_dependency`. **Evidence:** maintainer-resolution evidence records no repository-local fix. This is a natural issue in the correct repository, not a wrong-repository control; the label does not rule out useful local diagnostic context.

C Context-Budget Evaluation Protocol

File packing and token accounting. We deduplicate ranked paths and render each file as `### {path}\n{content}\n`, using the available `kind=file` text from the released base-commit corpus. Long-file text may already be truncated by corpus construction. The tokenizer, `regex_code_tokenizer_v1`, counts matches of

```
[A-Za-z_][A-Za-z0-9_]*|[0-9]+|[^A-Za-z0-9_\s]
```

with Unicode whitespace handling. The counter includes path-header tokens and adds one separator budget unit for each nonempty file. Whitespace, including the newline ending a header, produces no regex token. We greedily pack files by rank. If a file exceeds the remaining budget, we include only the prefix that fits and stop; a header alone earns no coverage credit.

Minimum-content sensitivity. Let L_{ig} be the number of available content tokens in gold file g and $m_{ig}(B)$ the number packed under budget B, excluding headers and separators. For threshold τ,

$$\mathrm{BCY}_\tau@B = \frac{1}{N_+} \sum_{i \in \mathcal{I}_+} \frac{1}{|G_i|} \sum_{g \in G_i} \mathbf{1}[L_{ig} > 0 \ \wedge \ m_{ig}(B) \geq \min(\tau, L_{ig})] \,.$$

Canonical BCY uses $\tau = 1$; the min term gives complete short files credit at larger thresholds. Empty file text earns no credit. The sensitivity analysis uses $\tau \in \{1, 16, 32, 64, 128\}$.

Scope of reported curves. The details artifacts retain the top 20 unique files, so curves at 4k, 8k, 16k, and 32k tokens use that stored prefix. Larger-budget points are lower bounds whenever those files do not fill the budget. Non-gold context share remains a diagnostic rather than a primary metric because useful but unjudged files may otherwise be counted as waste.

D Embedding Baseline Implementation

Qwen3-4B and Qwen3-8B denote Qwen3-Embedding-4B and Qwen3-Embedding-8B throughout the main text and figures. Table 11 lists the full checkpoint names.

All checkpoints use SentenceTransformers `encode` with L2-normalized output, no query/passage prefix, and no `prompt_name`. Their `default_prompt_name` is unset. This common prompt-free interface is not each model's best task-specific configuration.

Table 11: Embedding baseline implementation. Revision gives the first 12 hexadecimal characters of the Hugging Face commit. All rows use no added prompt, L2-normalized output, right truncation, and max chunk-to-file aggregation.

Checkpoint	Revision	Pooling	Similarity	Dim.	Max tokens
Qwen3-Embedding-4B	`5cf2132abc99`	last token	cosine	2560	40,960
Qwen3-Embedding-8B	`1d8ad4ca9b3d`	last token	cosine	4096	40,960
jina-code-embeddings-0.5b	`4db235132daf`	last token	cosine	896	32,768
pplx-embed-v1-4b	`2cd0f789519b`	mean	cosine	2560	32,768
nomic-embed-code	`11114029805c`	last token	cosine	3584	32,768

Each corpus chunk is rendered as follows; the `symbol` line is omitted when no symbol is available.

```
path: {path}
kind: {kind}
symbol: {symbol}
content:
{text}
```

Queries use the benchmark's evaluation query text without an added instruction. All tokenizers use right padding and right truncation. Table 11 reports the effective SentenceTransformers sequence cap, i.e., the active limit after reconciling the checkpoint's model and tokenizer settings. Chunks are scored by cosine similarity. File-level rankings use max aggregation over chunks: for query q, file f, and its chunks C_f,

$$s(q, f) = \max_{c \in C_f} \left\langle \frac{e(q)}{\|e(q)\|_2}, \frac{e(c)}{\|e(c)\|_2} \right\rangle .$$

Ties are resolved deterministically by path and chunk identifier, after which the evaluator retains the first occurrence of each file.

The `edit2ripple` runs use an NVIDIA H20 with embedding and query batch sizes of 8. Runtime depends on hardware and cache state and is not compared as a model-quality result.

The pplx run is provisional: Transformers 4.57.6 recommended `fix_mistral_regex=true`, but the evaluator did not forward that keyword. The results use the tokenizer as loaded, not a corrected-tokenizer rerun. The other four checkpoints did not emit this warning.

E FULL EDIT2RIPPLE RESULTS

Table 12 gives the values behind Figure 2. Canonical BCY uses the stored top-20 rankings: 4k/8k are fully packed, whereas 16k/32k are lower bounds when that prefix underfills the budget.

Table 12: `edit2ripple` retrieval results on 58 evidence-backed samples. † marks the provisional pplx run; BCY@32k is a top-20-prefix lower bound (LB).

Model	Family	R@5	R@20	MRR	BCY@4k	BCY@8k	BCY@16k LB	BCY@32k LB
Qwen3-Embedding-4B	embedding	0.4813	**0.7112**	0.2791	0.3707	0.5043	**0.6078**	**0.7112**
Qwen3-Embedding-8B	embedding	**0.4914**	0.6825	0.2652	0.3333	**0.5129**	0.5963	0.6767
pplx-embed-v1-4b†	embedding	0.4670	0.6782	**0.2877**	**0.3865**	0.4842	**0.6078**	0.6652
jina-code-embeddings-0.5b	embedding	0.3678	0.6466	0.2414	0.2773	0.3851	0.4928	0.6408
nomic-embed-code	embedding	0.3807	0.5733	0.2860	0.2917	0.3980	0.5244	0.5690
lexical	lexical	0.4124	0.5876	0.2428	0.3218	0.4468	0.5187	0.5876
RepoMap	structure	0.3635	0.6164	0.2344	0.2040	0.3592	0.5187	0.5848
BM25	lexical	0.2069	0.4986	0.1536	0.1221	0.2428	0.3103	0.4641

F CLOSED-TOOL CONTEXT ACQUISITION DIAGNOSTIC

Codex runs in a Docker container without a corpus mount and returns JSON actions from `list_dir`, `grep`, `read_file`, and `submit`. The evaluator executes them outside the container and returns observations; shell or unsupported-tool events count as protocol violations. Table 13 reports the 287-sample three-task diagnostic. Scripted grep uses fixed query terms without adapting to observations; it is a tool-path lower bound, not an estimate of interactive lexical search.

Table 13: Closed-tool results. Overall rows use all 287 samples; task rows use the same Codex policy. Both policies have zero recorded protocol violations. Calls are reported for the overall runs.

Policy / task	n	Final F1	Final R	Final P	Any-gold	Calls
Scripted grep	287	0.0719	0.1144	0.0557	0.2683	15.79
Codex closed-tool	287	0.3279	0.4692	0.2712	0.5470	3.82
`code2test`	106	0.2387	0.3601	0.1887	0.4151	–
`comment2context`	80	0.1325	0.1583	0.1292	0.2625	–
`trace2code`	101	0.5762	0.8300	0.4703	0.9109	–

A post-hoc prefix check scores files read within the first B calls: File F1 rises from 0.1863 at $B = 2$ to 0.3276 at $B = 4$, remaining unchanged at 6 and 8. This re-scores existing trajectories, not policies rerun with smaller budgets, and does not measure repair success.

G ADDITIONAL RETRIEVAL ANALYSES

This appendix reports repository-weighted scores, task-level rankings, relation-label rules, and granularity diagnostics supporting the main retrieval results.

G.1 REPOSITORY WEIGHTING

Gin contributes 88 of 345 positives; the four largest repositories contribute 203 (58.8%). Table 14 compares sample weighting with equal repository weighting. Qwen3-4B remains the MRR leader and becomes the Recall@20 leader. Repository-macro BCY is unavailable because the canonical BCY artifacts are task-aggregated rather than per-sample.

G.2 DETAILED TRACE2CODE RANKINGS

Table 15 shows the task-level reversal: RepoMap leads MRR and Recall@20, while Lexical MRR exceeds most embedding baselines.

Table 14: Repository-imbalance sensitivity on 345 positive samples. W- columns weight samples equally; R- columns average within repository and then weight all 25 repositories equally.

Model	W-R@20	R-R@20	W-MRR	R-MRR
Qwen3-4B	0.6306	**0.6344**	**0.2379**	**0.2501**
Qwen3-8B	**0.7029**	0.6193	0.2336	0.2238
pplx-4B†	0.6072	0.5714	0.2267	0.1871
RepoMap	0.6333	0.4619	0.2158	0.1298
Nomic	0.5244	0.4869	0.1986	0.1858
Jina-0.5B	0.4823	0.5424	0.1914	0.2031
Lexical	0.4940	0.3632	0.1574	0.1195
BM25	0.4452	0.4318	0.1520	0.1336

Table 15: `trace2code` task-level results.

Model	R@5	R@10	R@20	MRR
aider-style-repomap	0.4488	0.7195	0.8366	0.2742
lexical	0.3432	0.4818	0.6964	0.2075
Qwen3-8B	0.2442	0.5281	0.7970	0.1654
BM25	0.2228	0.3218	0.4934	0.1638
pplx-embed-v1-4b	0.1601	0.2855	0.5380	0.1372
nomic-embed-code	0.1584	0.1733	0.3581	0.0871
Qwen3-4B	0.0743	0.1815	0.5050	0.0827
jina-code-embeddings-0.5b	0.0693	0.1188	0.2607	0.0607

G.3 Relevance Relation Rules and Results

The relevance taxonomy describes overlapping query–gold relations: semantic-direct (D) uses target path or symbol cues; structural-indirect (S) uses repository links from an anchor; workflow-conventional (W) uses role transformations such as implementation-to-test; and causal-indirect (C) uses evidence connecting a failure symptom to a root-cause file. These mechanism labels are generated by deterministic Codex-authored rules, separately from the benchmark gold. Let Q_i be the set of non-path query tokens after identifier splitting and stop-token removal, and let V_i contain tokens from gold paths and released gold-symbol evidence. Directness is

$$d_i = \frac{|Q_i \cap V_i|}{\max(1, \min(|Q_i|, |V_i|))}.$$

Let s_i count recorded same-module, symbol/path-overlap, and source–test-path-overlap witnesses, plus same-module or stem links from a given/changed-file anchor to a gold path; let p_i indicate token overlap between an anchor path and a gold path. Let w_i count workflow witnesses such as same-PR changed tests or review-requested tests, a test-file gold, and the `code2test` workflow prior. The released annotations retain the exact witness-field names, summarized in Appendix Table 10 and instantiated by the examples that follow it.

Candidate relations may overlap. C is present for an evidence-backed `trace2code` symptom-to-root-cause pair; D and S remain possible secondary relations when their witnesses fire. For `code2test`, D is present when $d_i \geq 0.10$ or at least two query/gold cue tokens overlap, W is always present by task semantics, and S is present when $s_i > 0$ or $p_i = 1$. For `comment2context`, D is present when $d_i \geq 0.12$, at least two cue tokens overlap, or the released evidence records symbol/path overlap; W is present when $w_i > 0$; and S is present when $s_i > 0$, with S as the fallback when no other witness fires. The primary projection is deterministic: `trace2code` selects C; `code2test` selects D only when $d_i \geq 0.30$, otherwise W; and `comment2context` selects strong D when $d_i \geq 0.16$, otherwise W when $w_i > s_i$, otherwise S when available, followed by the remaining detected relation. Thus *primary* means dominant under a published rule, not exclusive membership or an ontological claim.

Of the 287 taxonomy-labeled samples, 272 (94.8%) have secondary relations: 146 receive two, 126 receive three, and 15 receive one. Table 16 partitions them by the primary rule; `edit2ripple` has not been processed by these rules.

Table 16: Primary projection of the multi-label agentic-relevance taxonomy. Secondary relations are retained in the released annotations but omitted here so rows partition the 287 samples.

Primary relation	Total	`code2test`	`comment2context`	`trace2code`
Semantic-direct	96	44	52	0
Structural-indirect	23	0	23	0
Workflow-conventional	67	62	5	0
Causal-indirect	101	0	0	101

Table 17 reports slice winners. RRF-3 combines Qwen3-4B, Qwen3-8B, and RepoMap; RRF-Q8+RM combines Qwen3-8B and RepoMap. The structural slice has only 23 samples. Because the projection is task-conditioned and relations overlap, these associations do not identify a task-independent causal effect.

Table 17: Best methods by primary agentic-relevance projection. R@20 and MRR are computed within each disjoint projected slice.

Primary relation	n	Best MRR	Best R@20
Semantic-direct	96	0.3939 RRF-3	0.7448 RRF-3
Structural-indirect	23	0.2213 Qwen3-4B	0.5725 RRF-3/Q8+RM
Workflow-conventional	67	0.2876 Qwen3-4B	0.6463 RRF-3
Causal-indirect	101	0.2742 RepoMap	0.8795 RRF-Q8+RM

G.4 Hybrid Ranking Metrics

RRF ($k = 60$) fuses stored top-20 file rankings from Qwen3-8B and RepoMap. Table 18 reports rank metrics; its top-20-limited BCY appears in Table 19. nDCG@20 uses binary file relevance and standard discounted cumulative-gain normalization (Järvelin & Kekäläinen, 2002).

Table 18: Post-hoc RRF fusion of Qwen3-8B and RepoMap on the 287 `code2test`, `comment2context`, and `trace2code` samples.

Task	R@20	MRR	Any@20	nDCG@20
overall	0.7331	0.2713	0.8153	0.3651
`code2test`	0.6972	0.2811	0.7642	0.3662
`comment2context`	0.5958	0.2635	0.7375	0.3136
`trace2code`	0.8795	0.2672	0.9307	0.4048

The fusion improves three-task MRR to 0.2713 and Recall@20 to 0.7331, but trails Qwen3-4B on `code2test` MRR. A separate top-20 union over Qwen3-4B, Qwen3-8B, pplx, RepoMap, Jina, Nomic, and Lexical misses only 22 of the 287 samples; this union is not a single-system score.

G.5 Context Selection and Granularity

Table 19 compares canonical BCY@8k and matched top-3/final-context File F1. Static rankings and multi-step agent trajectories remain different cost settings.

With $\tau \in \{1, 16, 32, 64, 128\}$ in Appendix C, all 12 three-task methods retain their BCY ordering; the largest decrease from $\tau = 1$ to 128 is 0.0081. All eight `edit2ripple` methods also retain their order, with a maximum decrease of 0.0058.

Table 19: Context-budget and context-selection results on `code2test`, `comment2context`, and `trace2code`. Static retrievers are measured by BCY@8k and matched top-3 file F1 where available; agent rows use logged final-context trajectories.

Method	Type	BCY@8k	Top-3 / final File F1
RRF(Qwen3-8B+RepoMap)	retriever	0.4884	–
RepoMap	retriever	0.3828	0.1102
Qwen3-4B	retriever	0.3079	0.1247
Lexical	retriever	0.2282	0.0721
Grep strict	retriever	0.2157	0.0841
OpenAI GPT-5.4-mini strict	agent	–	0.3113
Codex CLI GPT-5.4	agent	–	0.3342
Codex CLI GPT-5.5	agent	–	0.3255

G.5.1 File-versus-Span Sensitivity

Table 20 counts sample–gold-file occurrences. Across 345 positives, the median file has 515 lines; 51.8% exceed 500 lines, 24.1% exceed 1,000, and the maximum is 10,218. The 287 span-annotated samples have median evidence of 27 lines and median evidence/file ratio of 4.7%; 75.7% of span-file pairs occupy at most 10%. Thus file exposure leaves substantial within-file localization work.

Table 20: Gold-file size and annotated evidence density. Gold-file counts are sample–file occurrences in the base-commit corpus. Span statistics cover the 287 `code2test`, `comment2context`, and `trace2code` samples; `edit2ripple` does not yet have span annotations.

Task	Gold files	Med. file	P90 file	Span pairs	Med. evidence	Med. evidence/file	$\leq$10% file
`code2test`	148	518	1,506	139	14 lines	4.1%	79.1%
`comment2context`	140	491	1,910	128	32 lines	6.3%	70.3%
`trace2code`	127	725	3,020	124	27 lines	3.9%	77.4%
`edit2ripple`	99	414	1,500	–	–	–	–
Overall	514	515	2,011	391	27 lines	4.7%	75.7%

Table 21 measures overlap with annotated lines using stored chunk rankings and their legacy 8,000-character packing rule, not canonical token BCY. Line metrics use unions of packed intervals. Qwen3-8B leads file Recall@20 but ranks fourth by line F1; Qwen3-4B leads line F1 at only 0.0276. RepoMap has no line-resolved output. File-level performance therefore does not establish accurate span localization.

Table 21: File exposure versus line overlap on the 287-sample span-annotated core. "Exposure" and line metrics use the legacy 8,000-character chunk packing and are not canonical token BCY.

Method	File R@20	Exposure	Line R	Line P	Line F1
Qwen3-4B	0.6143	0.2516	0.0918	0.0216	**0.0276**
pplx-embed-v1-4b	0.5929	0.1359	0.0803	0.0186	0.0234
jina-code-embeddings-0.5b	0.4491	0.1568	0.0630	0.0136	0.0181
Qwen3-8B	**0.7070**	0.1533	0.0573	0.0137	0.0165
nomic-embed-code	0.5145	0.0832	0.0525	0.0110	0.0155
BM25	0.4344	0.0993	0.0400	0.0086	0.0121
Lexical	0.4750	0.0738	0.0223	0.0044	0.0062
RepoMap	0.6367	0.0592	–	–	–

H Trajectory Diagnostics and Seed Protocol

The analyses below define the trajectory measures and document the seed protocol and proxy-validity checks.

H.1 Gold Contact and Complementarity

Trajectories are joined to static rankings by sample ID. A gold contact means that any labeled gold file appears in the logged context events; Codex events are recovered file/path observations. First-hit medians exclude misses. Late-hit samples first reach gold after more than three events (Table 22).

Table 22: Retriever coverage on late-hit samples from the three trajectory tasks. Late-hit samples are those where the logged agent first reaches a gold file after more than three context events.

Agent	Retriever	Samples	Events/sample	Any@10	Any@20
Codex GPT-5.4	Qwen3-8B	77	7.6	0.675	0.818
Codex GPT-5.4	Qwen3-4B	77	7.6	0.623	0.779
Codex GPT-5.5	Qwen3-8B	73	8.1	0.726	0.863
Codex GPT-5.5	Qwen3-4B	73	8.1	0.658	0.836

Complementarity is defined for a *pair* of a retriever and an agent scaffold. Let A_i indicate whether the trajectory touches any gold file and $R_i@k$ whether the retriever finds any gold in its top k. Their 2-by-2 matrix gives both-hit, retriever-only, agent-only, and both-miss rates. Potential Exploration Savings is $\mathrm{PES}@k = \frac{1}{N}\sum_{i:A_i=1}\max(T_i-1,0)\mathbf{1}[R_i@k]$, where T_i is the first gold-touch event and N counts all joined samples. Agent-miss cases contribute zero because no first hit is observed. PES is an upper-bound proxy for avoidable gold-localization delay, not a causal estimate of saved calls. Table 23 reports these measures for Codex CLI GPT-5.4 at top 20.

Table 23: Trajectory-conditioned complementarity for Codex CLI GPT-5.4 at top 20. The four outcome proportions sum to one, up to rounding, over 287 samples; PES is reported separately.

Retriever	Both hit	Retriever-only	Agent-only	Both miss	PES@20
Qwen3-4B	0.509	0.178	0.220	0.094	1.213
Qwen3-8B	0.627	0.167	0.101	0.105	1.345
RepoMap	0.592	0.125	0.136	0.146	1.233
RRF(Qwen3-8B+RepoMap)	0.652	0.164	0.077	0.108	1.373

The four outcome rates in Table 23 show both directions of complementarity. Static top-20 reachability neither guarantees compact final context nor subsumes interactive exploration.

H.2 Seed Protocol and Paired Results

Scoring and aggregation. For selected files S_i and gold G_i, final File F1 is $2|S_i \cap G_i|/(|S_i|+|G_i|)$. Table 7 averages over 45 samples; ΔF1 is relative to no seed. Any-gold includes preloaded seeds and reads. First-hit indices count all tool calls; seed hits have index 0 and misses are excluded only from the median. Calls and tokens are sample means; post-seed tokens exclude seed tokens.

The pilot has 15 samples per task and one trajectory per sample–arm pair. Docker-isolated Codex GPT-5.5 uses `list_dir`, `grep`, `read_file`, and `submit`, with limits of 16 calls, 20 model turns, 8k total read tokens including seeds, and 1,200 tokens per file, plus a final three-file set. Seeds use at most three files and a 4k cap. No-seed, random non-gold, Lexical, Qwen3-8B, RRF, and oracle arms share these limits, but seed length changes the remaining read allowance. Saved CLI metadata specifies neither temperature nor random seed; sampling was not repeated.

Table 24 shows that some seeded runs worsen F1, including oracle-seeded runs. Aggregate gains are not universal or statistically resolved; unequal seed lengths and single-run policy sampling limit causal interpretation.

Evidence boundary. The policy cannot edit or test code. These outcomes measure context acquisition, not repair. Historical resolving commits are construction evidence, not evaluated agent outcomes, and the samples have no one-to-one mapping to executable SWE-bench instances. We therefore report neither patch success nor an ARB–SWE-bench success correlation.

Table 24: Paired seed-intervention deltas against the no-seed arm. Rescue/no-seed miss is the fraction of no-seed misses that become any-gold hits under the seed.

Arm	Improved	Worsened	Rescue/miss	Seed-hit ΔF1
Random non-gold	0.2667	0.3333	0.5455	0.0000
Lexical	0.3333	0.2889	0.6364	0.0625
Qwen3-8B	0.2889	0.3556	0.5909	0.2119
RRF(Qwen3-8B+RepoMap)	0.4222	0.3333	0.7273	0.1154
Oracle gold	0.6444	0.1778	1.0000	0.3115

H.3 Proxy-Validity Checks

Table 25 checks context-acquisition efficiency (CAE) proxies. A separate 40-sample Codex GPT-5.4 intervention calibrates PES against observed event savings; first-gold-step reduction is available for 16 pairs. The BCY rows are observational joins of RRF rankings with agent trajectories, not interventions or evidence of downstream repair effects.

Table 25: Proxy-validity checks for CAE metrics on `code2test`, `comment2context`, and `trace2code`. PES rows use paired control/seeded Codex GPT-5.4 trajectories. BCY rows are observational joins for RRF(Qwen3-8B+RepoMap); ρ is Spearman rank correlation.

Metric	Target quantity	n	ρ
PES@20	First-gold-step reduction	16	0.674
PES@20	Event savings	40	0.529
BCY@8k / OpenAI strict	Any-gold touched	287	0.230
BCY@8k / OpenAI strict	Final File F1	287	0.237
BCY@8k / Codex GPT-5.4	Any-gold touched	287	0.172
BCY@8k / Codex GPT-5.4	Final File F1	287	0.129