WHERE DO CONTEXT-ENGINE GAINS COME FROM? A COMPONENT-LEVEL DECOMPOSITION OF REPOSITORY RETRIEVAL UNDER MATCHED BASELINES AND AUDITED EXPOSURE

Aayam Bansal & Ishaan Gangwani
Synthetic Sciences
{aayam,ishaan}@syntheticsciences.ai

ABSTRACT

Context engines that index a repository and return files to a coding agent report large gains over lexical baselines, but the gains are rarely decomposed: how much is a learned reranker that would help any candidate pool, how much is the engine's own candidate generation, and how much was selected on the evaluation data? We answer this for one open-source engine, Delphi, with a protocol that records for every evaluation case whether development could have seen it, builds a ladder of conventional retrievers from the same embedding model with Delphi's own rerankers and query expansion attached, varies candidate generation and learned reranking factorially, and ablates Delphi's candidate branches one at a time on a fresh, never-scored draw. On three SWE-bench Verified draws that development never saw (62, 98, and 60 instances; sizes fixed before scoring), Delphi's advantage over the full conventional stack lies in its candidate pool: before any reranking its candidates lead by $+0.19$ MRR and $+0.07$ Recall@20; the shared rerankers add $+0.34$ MRR to conventional candidates but $+0.19$ to Delphi's, so after reranking the MRR gap closes to a tie while the recall gap ($+0.09$) persists; pooled over the three draws, Delphi's yield-metric leads ($+0.06$–0.09) exclude zero at a three-look-adjusted level and its MRR lead ($+0.065$) at 95% only. A branch-level ablation on the fresh draw attributes the candidate-pool advantage mostly to Delphi's chunking and index ($+0.15$ MRR for the same two-branch fusion) and its vector-heavy weighting ($+0.06$), with the four structural branches adding $+0.06$ jointly and no single branch more than 0.03. On a repository-disjoint commit-to-files set the pattern inverts: Delphi's pool has lower recall than two-branch fusion and the rerankers add nothing to either pool, so the reading "the reranker does the work" holds only for issue queries. Matched output contracts turn an apparent hosted-engine lead on documentation into a four-way tie; re-indexing a corpus from scratch reproduced Delphi's ordered top-20 on 1 of 18 cases; and a preregistered executable pilot (62 instances, two trajectories each) resolves no seed effect and audits its own seed. We release code, per-case artifacts, agent trajectories, and the exposure ledger.

1 INTRODUCTION

Coding agents increasingly rely on a context engine: a service that indexes a repository and, for each agent query, returns the files the agent should read. Such engines combine dense retrieval, lexical matching, structural indices, learned rerankers, and query expansion, and report large gains over lexical baselines. Those gains are hard to interpret. A margin over BM25 does not say whether the engine's design or a learned reranker that would help any candidate pool produced it. The evaluation cases were often available, at least in aggregate, while components were selected, so a margin partly measures selection. Comparisons with hosted engines mix output contracts and measure different

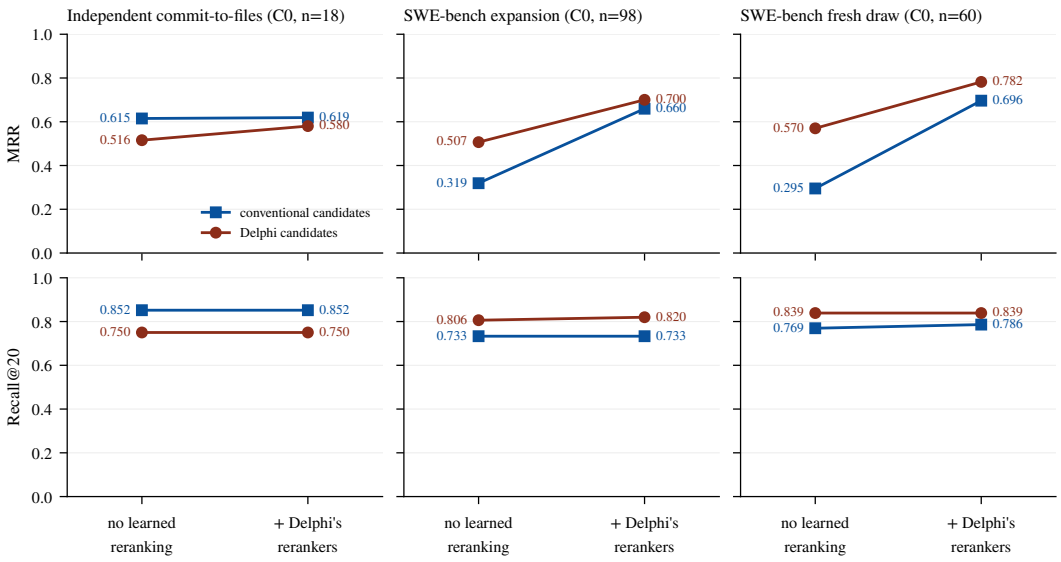

Figure 1: Candidate generation × learned reranking, expansion held constant, on sets no development decision could see. Blue: conventional candidates (dense+BM25 reciprocal-rank fusion over the same embedding model); red: Delphi's candidates (six branches, tuned fusion, quoted-path demotion). On SWE-bench issue queries (lower rows) Delphi's pool leads before any reranking; the shared rerankers add more to the weaker conventional pool, so the MRR lines converge while the Recall@20 gap, which reranking cannot close, persists. On the commit-to-files set (top row) Delphi's pool has lower recall and reranking changes little for either. Cells and contrasts: Appendix E.

quantities under one name, determinism among them. And retrieval quality is rarely connected to what an agent repairs.

We study these questions for one open-source engine, Delphi (six candidate branches with tuned reciprocal-rank fusion, query expansion, a cross-encoder, a listwise reranker), whose own history exhibits the selection problem: its largest evaluation partition (220 Agent Retrieval Bench cases) was re-drawn from a pool every case of which had been scored in an earlier development round whose aggregates gated which components shipped. The central question is a decomposition: where, component by component, does a context engine's advantage come from, on cases no development decision could see?

1. **RQ1 (decomposition).** Against lexical baselines, a component-matched conventional hybrid, and a factorial over candidate generation and reranking, which components account for Delphi's difference from conventional retrieval, and does the answer depend on query style?
2. **RQ2 (determinism).** Where does run-to-run variance come from, and when the same quantity is measured for every engine, which engines are stable?
3. **RQ3 (utility).** Holding the agent, its tools, and its budget fixed, does a retrieval seed change what the agent submits, and does it change verified repair?

Contributions.

- **A component-level decomposition.** A ladder of conventional retrievers adds Delphi's embedding model, BM25 fusion, its rerankers, and its expansion one at a time with Delphi's own prompts and parsers; a candidate-generator × reranker factorial separates the two factors the ladder confounds; and a branch-level ablation on a fresh draw asks which candidate branch carries the difference. On issue queries Delphi's contribution is its candidate pool (+0.19 MRR, +0.07 Recall@20 before reranking), most of whose MRR advantage the shared reranker recovers for conventional candidates; the branch ablation attributes it mostly to chunking and index and to weighting, not to any single structural branch; on commit-to-files queries the pool has lower recall and reranking adds nothing (§4, Figure 1).

- **An exposure protocol with a pooling rule.** A case-level ledger assigns every evaluation set one of three classes by what development could have seen; applied to Delphi, it reclassifies the 220-case ARB partition from held-out to validation evidence. Confirmatory results rest on 238 cases scored once and never before, drawn in three SWE-bench draws whose sizes were fixed in advance plus one repository-disjoint set, and pooled estimates are reported with intervals adjusted for the number of looks (§3).
- **Matched contracts and like-for-like repeatability for hosted engines.** Routing every engine's retrieved evidence—including a hosted synthesis engine's own recorded sources—through one frozen synthesis stage turns an apparent hosted lead into a four-way tie; repeatability is measured at each engine's observable unit, and a re-indexing experiment quantifies what "deterministic" means for a fresh installation (§5–6).
- **A preregistered executable pilot with an audited seed.** On the ranking evaluation's own 62 instances, a fixed agent with a retrieval seed resolves no more verified repairs than with a random-file seed; two trajectories per cell disagree by more than the effects under study; the seed's file heads rarely contained the edited region; and the pilot's variance yields a precision target for a decisive study (§7).

We claim no state of the art; several findings are null or negative for the engine we built, and we report them with the same prominence as the positive ones.

2 Related work

SWE-bench established executable repository-level issue resolution (Jimenez et al., 2024), while SWE-agent and Agentless expose repository navigation and localization as explicit stages (Yang et al., 2024; Xia et al., 2024). LocAgent studies a graph-guided localization agent and reports that better localization can improve multi-attempt repair (Chen et al., 2025). Repository-level code completion shows that cross-file retrieval matters once the edit location is known (Zhang et al., 2023; 2025). Dense retrieval, late interaction, hypothetical-document expansion, and BM25 motivate the components of practical context engines (Karpukhin et al., 2020; Khattab & Zaharia, 2020; Gao et al., 2023; Robertson & Zaragoza, 2009).

The closest work measures context acquisition directly. ContextBench provides human-verified file, block, and line contexts for issue-resolution tasks and scores agent trajectories (Li et al., 2026). CORE-Bench scales requirement-driven retrieval across code understanding, edit localization, and broader context (Zhang et al., 2026a). SWE-Explore derives audited line-level targets from successful trajectories (Zhang et al., 2026b). Agent Retrieval Bench (ARB) contributes workflow-derived next-context targets, base-commit hygiene, budget-aware file metrics, open-embedding and RepoMap baselines, and a closed-tool seed intervention with no-seed, random, retrieval-derived, and oracle arms (Qin & Xie, 2026). Our closed-tool experiment (§7) is a replication of that intervention with an additional seed, not a new design. What we add is narrower: a component-level decomposition under an exposure ledger, matched output contracts and like-for-like determinism for hosted engines, and a preregistered executable pilot on the ranking evaluation's own tasks.

3 Evaluation protocol

Exposure classes. Every evaluation set is assigned a class in a case-level ledger (Appendix C, Figure 3). **C0**: never scored by any system before its single confirmatory run, drawn from a pool no development decision could see. **C1**: development. **C2**: re-partitioned from a pool whose cases were all scored in an earlier round, where that round's aggregates influenced the shipped system. The 220-case ARB partition is C2: all 427 ARB v2 cases were scored in a first development round; that round's held-out aggregates were the acceptance test for query expansion, listwise reranking, the cross-encoder blend, reciprocal-rank fusion, and the path-affinity branch (Appendix H), and its per-workflow rates motivated the quoted-path demotion fix. A second round then drew a new $25/75$ split from the same pool, excluded 50 legacy-exposed IDs, and queried the 220-case partition once. Re-drawing does not un-expose a case, and the first round's per-case outputs are not retained, so we treat the partition as validation evidence and make no confirmatory claim on it. The C0 sets are an 18-case commit-to-files final over a repository-disjoint corpus locked before any scoring, and three SWE-bench Verified localization draws described next. A 40-case DS-1000 development subset

drives the documentation track and is C1. The ledger controls author exposure; it cannot control whether the foundation models inside any engine saw these public repositories during training, a caveat shared by every arm.

Three SWE-bench draws and the pooling rule. The SWE-bench Verified instances were drawn in sequence: 62 stratified instances in round 3 (12 repositories; leakage audit $0/62$); 98 instances in round 4, drawn under a fresh salt from the 438 no round had used (100 drawn, 2 excluded by two engine-agnostic rules fixed before scoring: gold files must exist at the base commit, and ARB's leakage check); and 60 instances drawn under a third salt from the remaining 338 for the branch-level ablation, with the same rules (0 excluded). Each draw's size and rules were fixed before any of its cases was scored, every system was scored once per draw, and no draw was stopped or extended on interim results. Because later draws were taken after earlier results were known, we treat the sequence explicitly: each draw is reported with its own intervals; the 98-instance draw is the prespecified replication of the 62-instance comparison; the 60-instance draw is a third replication and the only set with branch ablations; and pooled estimates carry both 95% intervals and intervals adjusted for three looks ($1 - 0.05/3$), the conservative reading of a pooled result after sequential draws (Appendix G).

Queries, corpus, metrics. Queries are the official ARB gold-free structured objects. File text comes from bare git clones at each case's base commit (binary and non-UTF-8 blobs skipped). Metrics are ARB's official per-case metrics (MRR, Recall@k) plus BCY@8k, the budget-constrained yield from packing ranked files into an 8,000-token context with ARB's packing routine. Documentation cases score *identifier hit*: whether the returned text contains the case's gold API identifier.

Freeze and attestation. All confirmatory Delphi numbers come from one frozen build with exact vector scan, API top-k=20, and response-level configuration attestation: every response carries the retrieval configuration actually served, and a run aborts on the first mismatch with its declared configuration (Appendix A). Paired deltas carry cluster bootstrap 95% intervals (20,000 resamples, seed 1042; repository clusters for repository tracks, case clusters for documentation), plus exact McNemar tests on any-gold@20 movement.

Matched ladder, factorial, and branch ablation. The lexical comparators (whole-file BM25, ARB's official lexical ranker, and their development-tuned fusion) cannot say how much of a margin is Delphi's engineering and how much is a learned reranker on top of any candidate pool. We therefore add four conventional systems, each adding one Delphi component to the previous one, over the official ARB chunks of the identical corpus: **dense**, `text-embedding-3-small`—the embedding model inside Delphi—max-pooled to files; **hybrid**, reciprocal-rank fusion (k=60, equal weights, untuned) of the dense ranking and BM25; **+ rerankers**, the hybrid's top 50 narrowed to 30 through Delphi's cross-encoder (`ms-marco-MiniLM-L-6-v2`, blend 0.4) and then Delphi's listwise `gpt-4o` reranker over the top 20 (seed 1042, identical prompt); **+ expansion**, the same with Delphi's hypothetical-document expansion (`gpt-4o-mini`, identical prompt and gate) feeding the dense query. We call the last rung the *full conventional stack*; prompts and parsers are imported from Delphi's source (Appendix B). What remains between it and Delphi is Delphi's own contribution: exact-symbol, exact-path, path-affinity, and trigram branches; its chunker and index; tuned fusion weights; quoted-path demotion. A ladder measures *conditional* increments in one construction order and does not allocate the final score uniquely among interacting components, so we also score a 2×2 factorial with expansion held constant—{conventional, Delphi} candidate generation $\times$ {none, Delphi's} learned reranking—where the Delphi-without-reranking cell is the frozen build with both rerankers disabled and everything else unchanged. Finally, on the fresh 60-instance draw only, we ablate Delphi's candidate generator branch by branch with rerankers off: each of the six branches removed in turn (fusion weight set to zero), and a two-branch configuration (vector and BM25 at equal weights) that reproduces the conventional fusion inside Delphi's own chunker and index (Appendix F). The factorial cells on the earlier draws were scored after their confirmatory runs and are explanatory; the branch ablation's set was drawn for that purpose and every configuration was scored once.

Hosted engines. Two commercial engines are compared on the documentation track: a *documentation engine* returning raw documentation context for a library query, and a *synthesis engine* return-

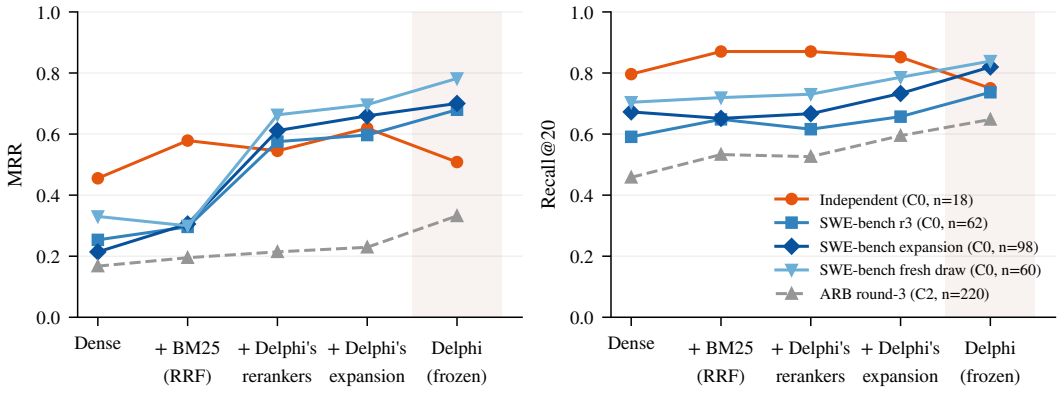

Figure 2: Conditional incremental gains along the matched ladder: each rung adds one Delphi component to the previous one over the identical corpus; the last point is the frozen Delphi stack. On issue queries the shared reranking head adds $+0.28$–0.31 MRR, on the commit-to-files set -0.03 with Delphi finishing below the full conventional stack, and on the ARB partition $+0.02$ against $+0.10$ for Delphi's own components. Dashed: the re-partitioned ARB set on which those components were selected (C2), for context only.

ing a grounded answer with cited sources; §6 matches their output contracts before comparing. The synthesis engine's repository surface could not be indexed at the required commits (Appendix L), so no repository number is reported for it in either direction.

Claim rule. "State of the art" would require leading every directly comparable engine on a C0 set with the complete comparable engine set runnable. No result in this paper meets it.

4 Decomposing retrieval quality (RQ1)

Against lexical baselines. Delphi leads whole-file BM25, the official lexical ranker, and their tuned fusion on every set (Figure 4, Tables 5–9): on the independent final its Recall@20 leads over BM25 and the lexical ranker exclude zero; on the three SWE-bench draws its MRR lead over the strongest lexical system is $+0.27$, $+0.21$, and $+0.35$, with Recall@20 tied on the first two and $+0.11$ $[+0.05, +0.19]$ on the third. None of this isolates Delphi's contribution.

The ladder: conditional increments that differ by query style. On the independent final (Table 5) the dense rung alone, using nothing but the embedding model Delphi already calls, ties Delphi on any-gold and exceeds it on Recall@20 (0.796 vs. 0.750); untuned fusion with BM25 reaches MRR 0.579 and Recall@20 0.870 against Delphi's 0.508 and 0.750; adding Delphi's rerankers *lowers* MRR to 0.545 and expansion raises it to 0.619, so Delphi's deficit against the full conventional stack (-0.111 $[-0.200, -0.034]$) excludes zero (six clusters; the direction, not the width, is the point). The SWE-bench draws have a different shape (Tables 6–8): the dense rung is weak (MRR 0.25–0.33), untuned fusion helps little, attaching Delphi's rerankers moves MRR by $+0.28$, $+0.31$, and $+0.40$, and Delphi then leads the full conventional stack by $+0.083$ MRR $[-0.107, +0.203]$, $+0.041$ $[-0.028, +0.112]$, and $+0.086$ $[-0.017, +0.323]$ on the three draws, with the yield metrics resolving where MRR does not. On the C2 ARB partition (Table 9) the rerankers add only $+0.019$ while Delphi's own components add $+0.103$ MRR $[-0.008, +0.189]$; that is the set on which those components were selected, so the margin is the expected direction of selection, not evidence. The reranking head is therefore the largest conditional gain on issue-style queries and nothing on the commit-to-files set (Figure 2); a "$+0.3$ from reranking" is a SWE-bench finding, not a property of the components.

The factorial: candidates versus reranking. The ladder cannot say whether Delphi's margin on issue queries comes from a better candidate pool or from an interaction with the shared reranker, because it only ever adds the reranker to conventional candidates. The factorial (Figure 1, Appendix E)

Table 1: Branch-level ablation of Delphi's candidate generator on the fresh 60-instance SWE-bench draw (C0), rerankers off, expansion on. Each row removes one fusion branch (weight set to zero) or keeps only vector and BM25; Δ is ablation minus the full six-branch generator with repository-cluster bootstrap 95% intervals; * excludes zero (Recall@5 in Appendix F).

Configuration (rerankers off)	ΔMRR	ΔR@20	ΔBCY@8k
All six branches, tuned weights (absolute)	0.570	0.839	0.469
− exact symbol	-0.025 [-0.072, $+0.015$]	-0.033 [-0.214, $+0.024$]	-0.017 [-0.080, $+0.000$]
− exact path	-0.001 [-0.003, $+0.000$]	$+0.000$ [$+0.000$, $+0.000$]	$+0.000$ [$+0.000$, $+0.000$]
− path affinity	-0.001 [-0.007, $+0.000$]	-0.008 [-0.037, $+0.000$]	$+0.000$ [$+0.000$, $+0.000$]
− trigram	-0.009^{*} [-0.014, -0.000]	$+0.000$ [$+0.000$, $+0.000$]	-0.017 [-0.029, $+0.000$]
− BM25	-0.002 [-0.048, $+0.077$]	$+0.021$ [$+0.000$, $+0.036$]	$+0.044$ [$+0.000$, $+0.146$]
− vector	-0.188^{*} [-0.265, -0.090]	-0.126^{*} [-0.176, -0.029]	-0.206^{*} [-0.322, -0.071]
vector + BM25 only, equal weights	-0.121^{*} [-0.160, -0.070]	-0.064 [-0.253, $+0.000$]	-0.133^{*} [-0.208, -0.062]
vector + BM25 only, Delphi's weights (0.50/0.25)	-0.060^{*} [-0.096, -0.005]	-0.064 [-0.287, $+0.011$]	-0.067 [-0.107, $+0.000$]
vector only	-0.120^{*} [-0.170, -0.027]	-0.047 [-0.253, $+0.020$]	-0.081 [-0.159, $+0.136$]

answers this. On the 98-instance draw, before any learned reranking, Delphi's candidate generator beats the conventional dense+BM25 pool by $+0.188$ MRR [$+0.036$, $+0.317$], $+0.073$ Recall@20 [$+0.005$, $+0.159$], and $+0.160$ BCY [$+0.072$, $+0.310$]. The shared reranking head is worth $+0.341$ MRR [$+0.256$, $+0.468$] on conventional candidates but $+0.193$ [$+0.161$, $+0.285$] on Delphi's, so the two factors are partial substitutes: the reranker recovers most of the MRR the weaker pool had lost, and what remains is the recall it cannot manufacture ($+0.087$ Recall@20 [$+0.042$, $+0.167$]) plus a tied MRR ($+0.041$ [-0.028, $+0.112$]); it leaves conventional Recall@20 essentially unchanged by construction, since it reorders a 30-candidate window. The fresh 60-instance draw replicates the pattern with larger effects: candidates $+0.275$ MRR [$+0.195$, $+0.402$] and $+0.069$ Recall@20 [-0.018, $+0.253$] before reranking; rerankers $+0.401$ [$+0.297$, $+0.508$] on conventional candidates against $+0.212$ [$+0.119$, $+0.419$] on Delphi's; $+0.086$ MRR and $+0.053$ Recall@20 after. On the commit-to-files set the pattern inverts: Delphi's candidate pool trails the conventional hybrid before any reranking (-0.099 MRR [-0.190, -0.028]; -0.102 Recall@20 [-0.213, $+0.000$]), and the rerankers add nothing to either pool ($+0.004$ MRR [-0.093, $+0.119$] on conventional candidates, $+0.065$ [-0.044, $+0.176$] on Delphi's, same index). The heterogeneity therefore sits in candidate generation: Delphi's pool has more recall than two-way fusion on issue text and less on short commit messages, where the dense branch alone already reaches 0.796 Recall@20.

Which branch carries it. The fresh draw was taken to answer this (Table 1, Appendix F). With rerankers off, removing any one structural branch costs little (exact symbol -0.025 MRR [-0.072, $+0.015$]; trigram -0.009; exact path and path affinity -0.001), removing BM25 costs nothing (-0.002), and removing the vector branch costs -0.188 [-0.265, -0.090]. The branches are redundant one at a time but not jointly: reducing the generator to vector and BM25 at Delphi's relative weights costs -0.060 [-0.096, -0.005], and at equal weights -0.121 [-0.160, -0.070]. That last configuration is the conventional two-branch fusion computed inside Delphi's own index, and it scores 0.449 MRR against 0.295 for the same fusion over ARB chunks. The candidate-pool advantage therefore decomposes into Delphi's chunking, indexing, and fusion implementation ($+0.15$ MRR at identical fusion design), its vector-heavy weighting ($+0.06$), and its four structural branches taken together ($+0.06$ MRR, $+0.06$ Recall@20), none individually necessary. The reading that exact-path and symbol branches "match the paths in issue text" is not supported; what is supported is that a vector-dominant fusion over a finer index with several weak, redundant lexical branches yields a better pool than equal-weight two-branch fusion over file-and-symbol chunks.

Where Delphi stands, draw by draw. Against the full conventional stack, the 98-instance draw—the prespecified replication of the 62-instance tie—gives $+0.041$ MRR [-0.028, $+0.112$], $+0.061$ Recall@5 [$+0.023$, $+0.148$], $+0.087$ Recall@20 [$+0.042$, $+0.167$], and $+0.082$ BCY [$+0.030$, $+0.152$], the three yield metrics excluding zero at the three-look-adjusted level as well (Table 17); the fresh 60-instance draw gives $+0.086$ MRR [-0.017, $+0.323$], $+0.053$ Recall@20 [-0.020, $+0.226$], and $+0.119$ BCY [$+0.026$, $+0.295$]. Pooled over all three draws (220 instances): $+0.065$ MRR [$+0.007$, $+0.132$], $+0.059$ Recall@5, $+0.076$ Recall@20, and $+0.087$ BCY, the yield metrics excluding zero at the adjusted level ([$+0.008$, $+0.182$], [$+0.025$, $+0.160$], [$+0.037$, $+0.162$]) and MRR at 95% only; point estimates are positive in every draw for every metric (the two-draw pool is Table 10). The answer to RQ1 is therefore conditional on query style: on issue queries

Table 2: Like-for-like repeatability on ten documentation cases × ten runs (450 run pairs per engine). Retrieved-set columns use each engine's *observable* retrieval unit—retrieved chunks for Delphi and the documentation engine, the set of cited URLs for the synthesis engine—and are therefore not directly comparable across engines: a stable URL set does not imply that the same passages were selected from those pages. Byte-exact is the delivered context text; hit agreement is whether the identifier-hit outcome agreed across the pair.

Engine	Set exact	Order exact	Set Jaccard	Byte exact	Hit agreement
Delphi (raw context, k=5)	0.980	0.980	0.993	0.980	1.000
Documentation engine (quality-guided IDs)	0.542	0.362	0.835	0.356	0.918
Documentation engine (oracle IDs)	0.496	0.258	0.806	0.253	0.909
Synthesis engine (cited-URL set)	1.000	1.000	1.000	0.000	0.929

Delphi's own additions are a better candidate pool, worth $+0.06$ to $+0.09$ on yield metrics after the shared reranker has done its work; on commit-to-files queries they are worth less than nothing, at n=18 with six clusters. Latencies (Table 13) and development-time ablations (Appendix H) are in the appendix.

5 Determinism, like for like (RQ2)

The pipeline was the variance. Repeating eight ARB queries ten times against the pre-freeze stack gave a mean exact top-10 rate of 30.6%. Paired traces isolated uncached, unseeded LLM stages; insertion-order ties in fusion SQL; and approximate nearest-neighbour search under concurrency. Total SQL ordering, single-flight sharing of concurrent identical hosted calls, an explicit seed, and a persistent stage cache take exact repeats to 100% in-process; a cold restart still moved $7/8$ lists until the cache was warm, after which $8/8$ survive two restarts. We claim restart-durable repeatability from the first persisted answer, not that two clean installations agree on their first answer, and the factorial quantified the difference: re-indexing the independent corpus from scratch and re-running the frozen configuration with an empty stage cache reproduced the ordered top-20 on 1 of 18 cases and the top-20 set on 7; Recall@20 and BCY were identical for every case, and MRR moved by $+0.072$ $[+0.005, +0.159]$ (Appendix E), which is as large as several differences reported on that set. Embedding APIs were not the problem (Appendix J).

Same quantity, every engine. A naive comparison—98.0% byte-identical context for Delphi against 35.6% and 0.0% for the hosted engines—compares retrieval stability for two engines with generated-text stability for the third. Table 2 separates the layers and measures each engine at its observable retrieval unit: the synthesis engine's cited-source URL sets are identical across every pair (1.000); Delphi's retrieved chunks across 98.0%; the documentation engine's retrieved chunks across 54% (quality-guided) or 50% (oracle IDs). The units differ in granularity, so these numbers do not rank the engines' underlying retrieval stability: the synthesis engine could cite the same pages while selecting different passages, which its API does not expose. What is observable is that its answer text never repeats byte-for-byte and that this flips the identifier-hit outcome in 7.1% of pairs, against 0% for Delphi. The defensible statements are narrow: Delphi's caching and total ordering make its retrieved text, and hence downstream decisions, repeatable; the documentation engine's retrieval drifts at the chunk level; the synthesis engine's cited sources do not drift, its prose does.

6 Documentation retrieval under matched output contracts

Documentation retrieval is where hosted engines are strongest and where output contracts differ most. The synthesis engine returns a fluent answer with citations; the documentation engine and Delphi return raw context. Scored naively, the synthesis engine leads (identifier hit 0.575 against Delphi's 0.400 and the documentation engine's 0.375). That comparison conflates retrieval with the synthesis model's parametric knowledge.

Matching the contract. Routing only the two raw-context engines through a frozen synthesis stage (`gpt-5.4-mini`, 1,600 output tokens, one system prompt) and comparing them with the

synthesis engine's *native* pass aligns the output format but not the synthesis model or prompt. The synthesis engine's raw-retrieval mode returned no sources during the recorded round, but its synthesized responses carry the five retrieved source chunks it grounded on; we reconstructed that context from the recorded responses (mean 2,754 tokens), packed it under the same 8,000-token budget with the same routine as every other arm, and routed it through the same frozen stage, three repeats, no new hosted calls (Figure 10, Appendix I). Three findings follow. First, on raw retrieved context the synthesis engine is ahead of Delphi: its reconstructed five-source context hits the gold identifier on 0.475 of cases against Delphi's 0.400 (k=5) and 0.425 (k=20). Second, under the matched contract the four arms finish within 0.05 of one another and every engine-versus-engine interval includes zero (Table 19); the synthesis engine's matched arm (0.583) differs from its native pass by $+0.008$ $[-0.050, +0.075]$, so no difference between its own synthesis and the frozen stage is resolved on this metric. Third, the metric is near saturation: the model alone scores 0.492 and the retrieval-over-floor differences are small ($+0.13$, $+0.13$, $+0.09$); because retrieval may help some cases and hurt others, the floor bounds the room the metric leaves rather than decomposing scores into parametric and retrieved parts. We detect no statistically resolved difference between any two engines under this evaluation; "parity or better" is not supported either, since an interval of $[-0.09, +0.16]$ does not establish it, and non-inferiority would require a predeclared margin and a sample several times this size.

7 AGENT UTILITY (RQ3)

Two preliminary measurements. A replication of ARB's closed-tool seed intervention (40 paired development cases per arm) finds that seeds change which files a frozen agent submits, ordered by seed quality: file-F1 rises from 0.046 with no seed to 0.234 with a Delphi seed against an oracle ceiling of 0.858 (Appendix K). Executing generated code on 40 DS-1000 cases $\times$ 3 repeats leaves every retrieval arm within $[-0.14, +0.20]$ of the no-retrieval control (Appendix I).

Executable pilot (preregistered). To measure retrieval and executable outcome on the same tasks, we preregistered a pilot (Appendix K) on the 62 C0 SWE-bench instances whose rankings every system in Table 6 had already produced once. The agent is mini-SWE-agent with `gpt-5.4-mini`, ordinary shell tools in every condition, and a fixed step budget; a seed is a block appended to the issue naming the top five files a recorded ranking produced, with the first 40 lines of each file, capped at 3,000 tokens. Conditions: no seed; five random non-gold files in the same block (prompting control); Delphi's recorded top five; the full conventional stack's recorded top five. The outcome is verified repair under the official harness, paired by instance, with instance- and repository-cluster intervals. Every condition was run twice (two independent trajectories per instance); Figure 12 (Appendix K) reports both repeats and their pooled per-instance means.

The two repeats disagree by more than the effects under study, which is the first result. Resolved counts were 25/30/31/23 (none/random/Delphi/conventional) in the first repeat and 25/27/25/25 in the second; within a condition, 7–10 of 62 instances flip between repeats. Pooled, no seed effect is resolved. A Delphi seed changes the per-instance repair rate by $+0.048$ $[-0.008, +0.113]$ against no seed, and by -0.008 $[-0.073, +0.056]$ against a block of five random non-gold files, which itself sits at $+0.056$ $[-0.016, +0.137]$ over no seed. The conventional stack's seed is the only arm that trails a comparator with an interval excluding zero (-0.073 $[-0.145, -0.008]$ against the random control); we have no explanation and offer none. Budgets were not binding: agents used 7–8 of 50 steps and cost about two cents per instance.

What the seed contained. Two audits qualify what the pilot tested. The seed is a *file hint plus file head*, not the engine's task-relevant evidence: among the 50 gold files Delphi's seed named, the first 40 lines covered a region the gold patch edits in only 9 (median first edited line 200). The random control excludes gold files but is not thereby uninformative: its 310 files were 30% tests, 25% Python sources, 15% documentation, and 30% other, with 1.3% in the same directory as a gold file, so what it contributes is repository orientation, not the fix location. To test the interface we ran two further arms on the same 62 instances with one trajectory each (exploratory, not preregistered): Delphi's top five as paths only, and as paths plus each file's best-matching chunk for the issue text (BM25, capped at 60 lines, same budget; it covered an edited region in 13 of the 50 seeded gold files). Paths only resolved 26 and paths plus chunks 24, against 25 with no seed and 31 and 25 in

the two file-head repeats; neither interface differs from no seed ($+0.016$ $[-0.097, +0.129]$; -0.016 $[-0.097, +0.065]$), and the spread across interfaces is smaller than the spread between repeats of one interface (Table 23). Pooled per-instance paired differences in the pilot have standard deviations of 0.23–0.30; a study on 200 instances with two trajectories each would have an expected 95% half-width of about 0.04 before repository clustering, and we set that, not a favourable interval, as the confirmatory design (Appendix K). At about \$11 for 496 trajectories it is affordable; what it needs first is the seed interface fixed on development tasks.

8 Discussion and limitations

Supported. Delphi leads lexical baselines on every set; against the full conventional stack it leads by $+0.06$–0.09 on Recall@5, Recall@20, and BCY pooled over three SWE-bench draws (220 instances; intervals excluding zero at the three-look-adjusted level; MRR $+0.065$ at 95% only), with positive point estimates in every draw, and trails on the 18-case independent final. The shared reranking head is the largest conditional gain on issue queries ($+0.28$–0.40 MRR) and adds nothing on commit-to-files queries or the ARB partition given expansion. On issue queries Delphi's own contribution sits in its candidate pool ($+0.19$ MRR, $+0.07$ Recall@20 before reranking), the reranker recovers most of the MRR but not the recall for conventional candidates, and the branch ablation attributes the pool advantage to chunking and index ($+0.15$ MRR), vector-heavy weighting ($+0.06$), and the structural branches jointly ($+0.06$) rather than to any single branch; on commit-to-files queries Delphi's pool has lower recall than the conventional hybrid (-0.10 Recall@20) and the reranker adds nothing to either pool. Under one frozen synthesis stage no two documentation engines are statistically distinguishable at $n{=}40$. Delphi's retrieved text is repeatable in-process and across restarts with a warm cache; the hosted synthesis engine's cited-source sets repeat exactly. Retrieval seeds change a closed-tool agent's file selection, ordered by seed quality.

Not supported. State of the art, anywhere; any confirmatory result on the C2 ARB partition; a pooled yield-metric lead that survives adjustment for three looks; documentation parity or non-inferiority; downstream utility on DS-1000 at $n{=}40$ or in executable repair at $n{=}62$, in either direction; a ranking of the engines' underlying retrieval stability (the observable units differ); a decomposition of documentation scores into parametric and retrieved parts; that the pilot's seed delivered the engine's evidence rather than file names and heads; and MRR differences on the 18-case set smaller than the $+0.07$ re-indexing shift.

Limitations. One engine is studied in depth; the hosted engines only where their surfaces permitted a matched measurement, and one arm is missing. The C0 sets are small (18, 62, 98, 60) and their two query styles live in different repository collections, so query style and corpus are confounded in the heterogeneity we report. The factorial's Delphi-candidates-without-reranking cell exists only where an index could be provisioned, and the branch ablation is a single draw of 60 instances. Every arm shares one embedding model and one family of foundation models, so training-set exposure of the public repositories is uncontrolled. The pilot uses a small policy model, one agent scaffold, and a seed interface that rarely carried the edited region; two trajectories per cell are too few. The ledger records what the authors could see, not what the models had seen.

9 Conclusion

Decomposed against baselines built from its own parts, on cases its development could not see, a context engine's advantage over lexical retrieval on issue-style queries is mostly the part any careful practitioner would assemble; what is Delphi's own is a better candidate pool, worth $+0.19$–0.28 MRR and $+0.07$ Recall@20 before reranking and $+0.06$ to $+0.09$ on yield metrics after it, and that pool owes more to how the repository is chunked, indexed, and weighted than to any structural branch. On commit-to-files queries its candidate pool has lower recall than two-branch fusion and the shared reranker adds nothing to either. Margins measured on the re-partitioned ARB set do not survive the ledger that classifies how that set was used, and margins over hosted engines do not survive matching the quantity being measured. What survives is the protocol, and a pilot whose audited seed and measured variance specify the study that could connect retrieval to repair.

REFERENCES

Zhaoling Chen, Robert Tang, Gangda Deng, Fang Wu, Jialong Wu, Zhiwei Jiang, Viktor Prasanna, Arman Cohan, and Xingyao Wang. Locagent: Graph-guided llm agents for code localization. In *ACL*, pp. 8697–8727, 2025. doi: 10.18653/v1/2025.acl-long.426.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *ACL*, 2023.

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 *ICLR*, 2024.

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. In *EMNLP*, 2020.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *SIGIR*, 2020.

Han Li, Letian Zhu, Bohan Zhang, Rili Feng, Jiaming Wang, Yue Pan, Earl T. Barr, Federica Sarro, Zhaoyang Chu, and He Ye. Contextbench: A benchmark for context retrieval in coding agents. *arXiv preprint arXiv:2602.05892*, 2026.

Bowen Qin and Yi Xie. Agent retrieval bench: Evaluating repository context retrieval for coding agents. *arXiv preprint arXiv:2607.24882*, 2026.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.

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 *NeurIPS*, 2024.

Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. Repocoder: Repository-level code completion through iterative retrieval and generation. In *EMNLP*, pp. 2471–2484, 2023. doi: 10.18653/v1/2023.emnlp-main.151.

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.

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.

Sheng Zhang, Yifan Ding, Shuquan Lian, Shun Song, and Hui Li. Coderag: Finding relevant and necessary knowledge for retrieval-augmented repository-level code completion. In *EMNLP*, pp. 23278–23288, 2025. doi: 10.18653/v1/2025.emnlp-main.1187.

APPENDIX

The appendix documents every experiment in enough detail to rerun it from the released archive. Appendix A gives the frozen Delphi configuration; B the matched baseline ladder and the verbatim prompts it shares with Delphi; C the exposure ledger and the round chronology; D the complete repository-retrieval tables and additional figures; E the candidate $\times$ reranker factorial; F the branch-level ablation on the fresh draw; G the sequential analysis of the three SWE-bench draws; H the development-time ablations and the rejected ideas; I the documentation track; J the determinism probes; K the executable pilot; L the hosted engine accounting; M compute and cost; and N the artifact release.

A FROZEN SYSTEM CONFIGURATION

All confirmatory Delphi numbers come from a single frozen build (its revision identifier is recorded in the archive), served natively against PostgreSQL 14 with pgvector. Every scored response carried the following attested configuration (Table 3): embedding model `text-embedding-3-small`; vector mode exact scan (approximate search disabled for scoring); hybrid candidate branches vector, BM25, exact symbol, exact path, path affinity, and trigram with reciprocal-rank fusion weights $0.50/0.25/0.20/0.15/0.15/0.05$ and k=60; 50 fused candidates; cross-encoder `cross-encoder/ms-marco-MiniLM-L-6-v2` over the top 30 with blend α=0.4 (code-aware reranker disabled); quoted-path demotion when the query envelope carries evidence keys; listwise reranking of the top 20 by `gpt-4o` at temperature 0, seed 1042, 280-character excerpts, 300-token replies; hypothetical-document expansion by `gpt-4o-mini` (temperature 0, seed 1042, 220-token replies) gated on prose-like queries and feeding only the vector branch; persistent stage cache; API top-k=20; file-diverse BM25 and path-token branches disabled; agent quality mode with tests, docs, examples, configs, and manifests indexed and generated source retained under 500,000-byte and NUL-content guards. Context packing for BCY@8k uses ARB's official packing routine at an 8,000-token budget. The factorial's no-reranker cells (Appendix E) are the same build with the cross-encoder and the listwise stage disabled; the branch ablations (Appendix F) additionally override the fusion weights through the build's per-deployment weight setting. Every response of every ablated cell attests the configuration actually served, including the weights, and nothing else differs.

Table 3: Frozen serving configuration of Delphi. Secrets and local paths omitted.

Setting	Value
Embedding provider and model	OpenAI, `text-embedding-3-small` (1,536 dimensions)
Vector search	exact scan (HNSW index present but not consulted)
Candidate branches	vector, BM25, exact symbol, exact path, path affinity, trigram
Fusion	reciprocal rank, k=60, weights $0.50/0.25/0.20/0.15/0.15/0.05$
Fused candidates / rerank window	50 / 30
Cross-encoder	`cross-encoder/ms-marco-MiniLM-L-6-v2`, blend α=0.4
Code-aware reranker	disabled
Query expansion	enabled; `gpt-4o-mini`, temperature 0, seed 1042, gated on prose-like queries
Listwise reranking	enabled; `gpt-4o`, top 20, temperature 0, seed 1042
LLM stage cache	persistent (archived) with a 2,048-entry in-memory tier
Quoted-path demotion	enabled when the query envelope carries evidence keys
File-diverse BM25 / path-token branches	disabled
API top-k	20
Worker threads	4
Rate limits	disabled for local scoring

Attestation. Every search response from the frozen build carries a configuration object listing the branches, weights, reranker, listwise, expansion, vector mode, and k actually used to serve it. The evaluation runner compares this object with the run's declared configuration on every response and aborts the run on the first mismatch; each run summary records the verification outcome and the set of observed configurations, which is a singleton in every reported run.

Index audits. Before the 98-instance draw was scored, an audit walked every provisioned snapshot and compared the file count at the base commit (after the same size and binary guards) with the

indexed file, chunk, and embedding counts in the database: 98 snapshots, 182,415 files, 1,079,743 chunks, 1,079,743 embeddings, zero mismatches. The fresh 60-instance draw passed the same audit: 60 snapshots, 111,367 files, 666,837 chunks, 666,837 embeddings, zero mismatches. A gold-searchability preflight then confirmed that every gold path of every case is present in the index; on the 98-instance draw this preflight surfaced the one patch-created gold file (§3).

B MATCHED BASELINE LADDER

Corpus and chunks. Every rung operates on the same materialized snapshot as Delphi and the lexical comparators, chunked with the official ARB routine (one whole-file chunk plus one chunk per top-level symbol). Non-UTF-8 and binary blobs are skipped by the same guard Delphi uses.

Dense. Chunks are embedded with `text-embedding-3-small` (1,536 dimensions) in token-aware batches (at most 200,000 tokens or 512 texts per request; texts above 8,000 tokens are truncated with the `cl100k_base` tokenizer); vectors are cached keyed by model and content hash, so a chunk is embedded once regardless of how many rungs read it. Queries are embedded with the same model; similarity is cosine; a file's score is the maximum over its chunks; the top 50 files are returned.

Hybrid. ARB's official BM25 chunk ranker over the same chunks, max-pooled to files, fused with the dense file ranking by reciprocal-rank fusion, $\mathrm{RRF}(f) = \sum_r 1/(60 + \mathrm{rank}_r(f))$ with equal weights. Nothing is tuned.

+ Delphi's rerankers. The top 50 fused files are narrowed to 30 with Delphi's cross-encoder (`ms-marco-MiniLM-L-6-v2`, scored on the query and the first 4,000 characters of the file's best-matching chunk, its sigmoid blended at α=0.4 with the fused score normalized by the top fused score, as in Delphi), then the top 20 are reordered by Delphi's listwise reranker (`gpt-4o`, temperature 0, seed 1042, 280-character excerpts). The system prompt and the reply parser are imported from Delphi's listwise-reranking module at the frozen revision.

+ Delphi's expansion. Delphi's hypothetical-document expansion (`gpt-4o-mini`, temperature 0, seed 1042, 220-token reply) is applied under Delphi's own gate (prose-like queries only) and the expansion text is appended to the dense query; BM25 sees the original query, as in Delphi. The prompt is imported from Delphi's query-expansion module.

Hybrid + expansion (factorial cell). For the factorial of Appendix E, a fifth conventional system applies the expansion to the hybrid's dense query and returns the fused ranking without learned reranking, so that expansion is held constant across the four cells.

Caching and latency. LLM replies are cached keyed by model, prompt, and seed, so a repeat of a rung is bitwise identical and free. Reported latencies are wall-clock per query on a warm process excluding index construction, as for Delphi, and include hosted API round trips.

Listwise reranker system prompt (verbatim, shared with Delphi).

```
You rank candidate source files by how directly each one
answers a developer's request about a codebase.  You are
given a request and a numbered list of candidates, each with
its repository path and an excerpt.  Return the candidate
numbers ordered best first, as a JSON array of integers, and
nothing else.  Example:  [4,1,9,2] Include every candidate
number exactly once.  Judge by what the request actually
asks for.  If it asks which test covers some behaviour, a
test file is the answer and the implementation is not.  If
it asks where something is implemented, the reverse holds.
Prefer the file that would have to be opened or edited to
satisfy the request over one that merely mentions the same
words.
```

Query-expansion system prompt (verbatim, shared with Delphi).

```
You write short, plausible code snippets that would answer a
developer's question about an unfamiliar codebase.  Reply
with code only:  no prose, no explanation, no fences.
Invent idiomatic function, class, and variable names for
the language implied by the question.  Being wrong about the
specifics is fine; using the vocabulary real code would use
is what matters.  Keep it under 15 lines.
```

C EXPOSURE LEDGER AND CHRONOLOGY

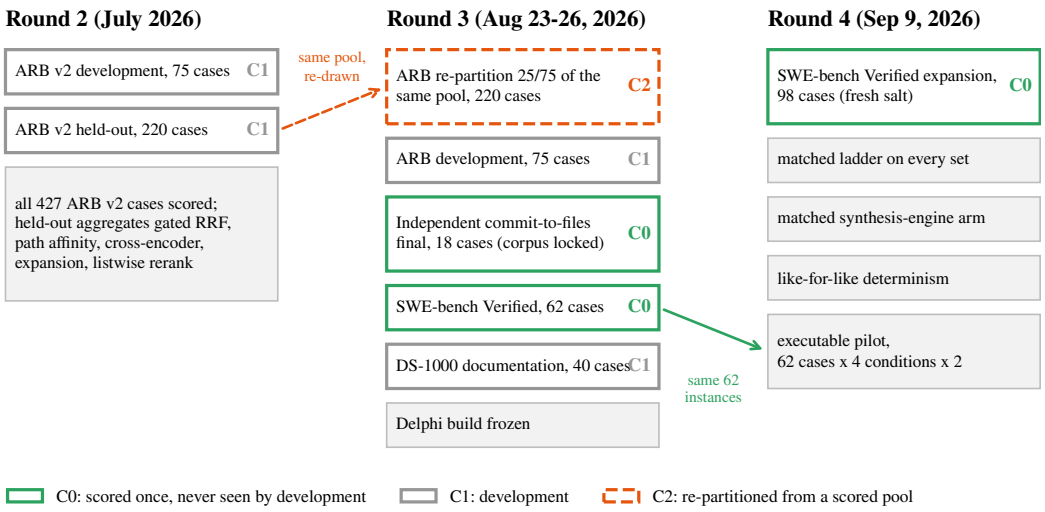

Figure 3: Evaluation rounds, the sets each round scored, and their exposure class. Round 2 scored every case of the ARB v2 pool and used the held-out aggregates to accept components; round 3 re-drew a 220-case partition from that pool (C2) and introduced the first C0 sets; round 4 added the matched ladder, the 98-instance draw, the matched synthesis-engine arm, like-for-like determinism, the executable pilot on the round-3 62-instance set, and, after the confirmatory runs, the factorial, the seed-interface check, and the fresh 60-instance draw for the branch ablation.

Table 4: Case-level exposure ledger. Every case ID of every set is listed in the machine-readable ledger in the archive with the evidence for its class.

Set	Cases	Class	Supports
ARB round-3 partition (positive workflows)	220	C2	development/validation evidence; not a confirmatory test of the frozen stack
ARB round-3 development	75	C1	development
Independent commit-to-files final	18	C0	confirmatory
Independent commit-to-files development	48	C1	development
SWE-bench Verified, round 3	62	C0	confirmatory
SWE-bench Verified, round-4 draw	98	C0	confirmatory
SWE-bench Verified, fresh draw (branch ablation)	60	C0	confirmatory (third SWE-bench draw); branch ablation
DS-1000 documentation subset	40	C1	development

Decisions and the evidence that informed them. Query-time embedding alignment, RRF fusion, the path-affinity branch, the cross-encoder blend, expansion, and listwise reranking were accepted on round-2 development (75) and round-2 held-out (220) aggregates over the ARB v2 pool that round 3 re-partitioned (Appendix H); quoted-path demotion on round-2 per-workflow rates over all 427 cases; exact scan, single-flight, the persistent cache, and total SQL ordering on round-3

development and determinism probes; generated-source indexing on the gold-path preflight over the independent and ARB development sets; the File-Okapi rejection on both development sets. No decision was informed by any C0 set: the independent final corpus was locked before any query was issued; each SWE-bench draw was scored once per system after the build was frozen.

Post-hoc analyses on C0 sets. After the confirmatory runs, the C0 sets were used for declared explanatory analyses: the candidate × reranker factorial (Appendix E), which adds ablated cells of the frozen build and of the conventional ladder on the 98-instance draw and on a re-indexed independent corpus, and the seed-interface check on the 62 pilot instances (Appendix K). The 60-instance draw was taken for the branch ablation and scored once per configuration. None of these changed any shipped configuration or any confirmatory number; all are labelled explanatory wherever they appear. Should any inform a future change to Delphi, the sets involved become C1 for that change and cannot serve as its confirmatory test.

Why re-drawing does not un-expose. A case scored in round 2 contributed to the aggregate that accepted or rejected a component. Any partition drawn later from the same pool inherits that influence at the aggregate level, whatever the seed. What a fresh draw does exclude is per-case tuning, and only if per-case outputs were never inspected; the round-2 per-case outputs are not in the archive, so this cannot be demonstrated from the record, and we do not claim it.

What the ledger cannot control. The foundation models inside every arm (OpenAI embeddings, `gpt-4o`, `gpt-4o-mini`, `gpt-5.4-mini`, and the models behind the hosted engines) were trained on public code that very likely includes these repositories and possibly the fixing commits. This affects every arm and every rung equally within an experiment, but it means absolute numbers are not estimates of performance on unseen code.

D COMPLETE REPOSITORY-RETRIEVAL RESULTS

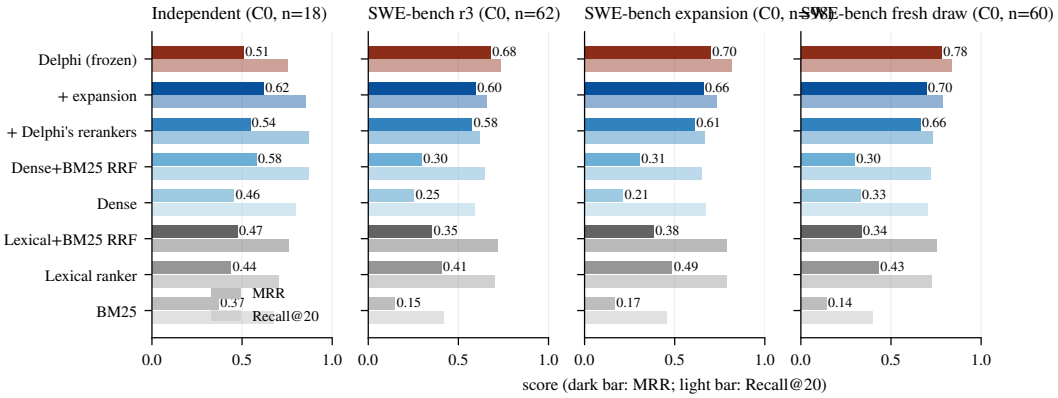

Figure 4: The C0 sets, every system, scored once. Dark bars are MRR, light bars Recall@20. Grey: lexical comparators; blue: the matched ladder, darker as components are added; red: the frozen Delphi stack.

Exploratory split of the SWE-bench issues. To test the workflow hypothesis outside the C2 set we split the 62 round-3 issues by whether the issue text contains a Python traceback or a path ending in `.py`. Delphi's MRR lead over the full conventional stack was largest on issues with neither (n=35), and smaller on issues with a traceback or a path (n=27). The split was decided after the rankings existed and is reported as exploratory; it does not support the hypothesis that Delphi's structural branches are what separate it on issue-style queries. The branch ablation of Appendix F is the designed test.

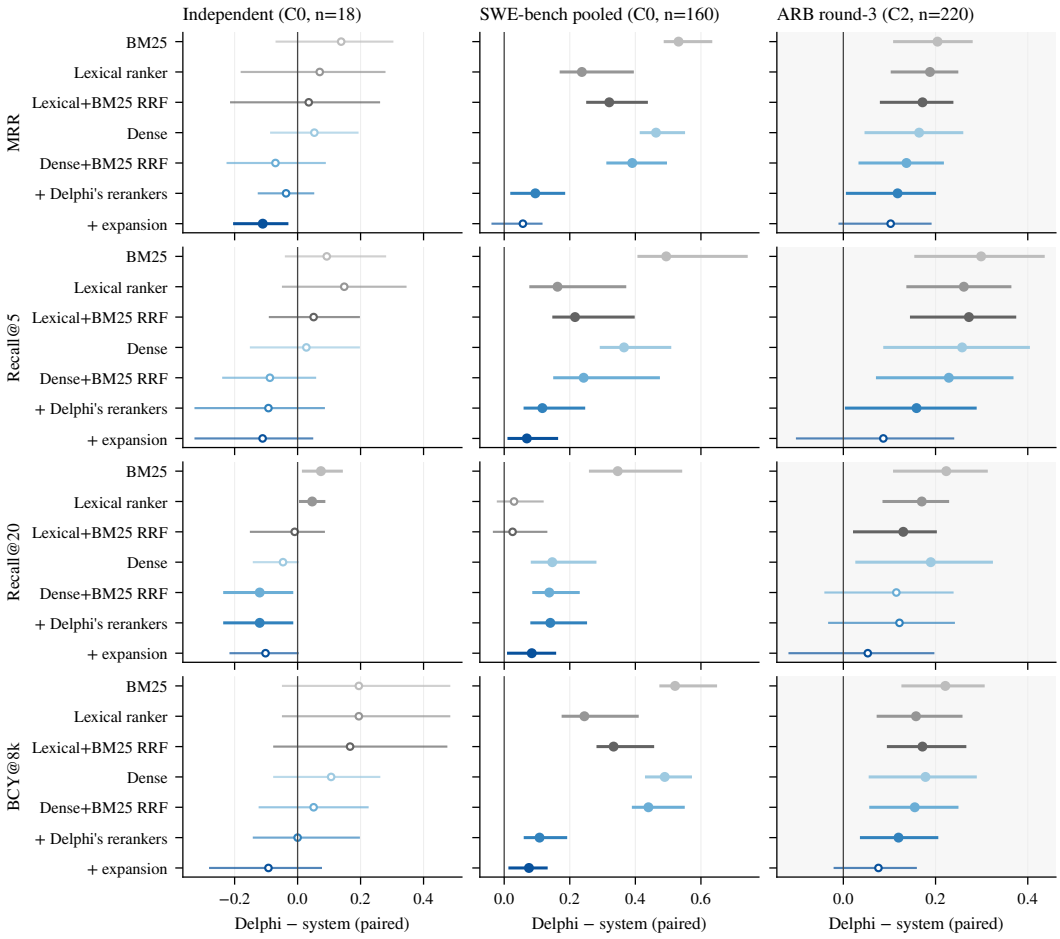

Figure 5: Paired Delphi−system differences on all four metrics with repository-cluster bootstrap 95% intervals (20,000 resamples, seed 1042). Filled markers: interval excludes zero. Left: independent final (C0, n=18, 6 clusters). Centre: pooled round-3 and round-4 SWE-bench draws (C0, n=160, 12 clusters). Right, shaded: ARB round-3 partition (C2, n=220, 19 clusters), validation only.

Table 5: Independent commit-to-files final (C0): 18 cases, 6 repositories, scored once. Any-gold is the number of cases with a gold file in the top 20; latency is the median query time in seconds. Bold marks the best value in each metric column. Δ rows are Delphi minus the named system with repository-cluster bootstrap 95% intervals; * excludes zero. The last ladder rung (+ expansion) is the full conventional stack.

System	MRR ↑	R@5 ↑	R@20 ↑	BCY@8k ↑	any-gold ↑	latency (s) ↓
Delphi (frozen)	0.508	0.551	0.750	0.454	15/18	3.7
Dense (same embedding)	0.455	0.523	0.796	0.347	15/18	0.5
Dense+BM25 RRF	0.579	0.639	**0.870**	0.403	16/18	0.2
+ Delphi's rerankers	0.545	0.644	**0.870**	0.454	16/18	2.3
+ expansion	**0.619**	**0.662**	0.852	**0.546**	16/18	4.1
Lexical+BM25 RRF	0.473	0.500	0.759	0.287	15/18	1.6
Lexical ranker	0.438	0.403	0.704	0.259	14/18	1.1
BM25	0.370	0.458	0.676	0.259	14/18	1.2
*Paired Delphi−system deltas, repository-cluster bootstrap 95% intervals; * interval excludes zero*						
Δ vs. Dense (same embedding)	+0.053 [−0.084, +0.190]	+0.028 [−0.148, +0.194]	−0.046 [−0.139, +0.000]	+0.106 [−0.074, +0.259]	15 vs 15	p=1
Δ vs. Dense+BM25 RRF	−0.070 [−0.222, +0.086]	−0.088 [−0.236, +0.056]	−0.120* [−0.231, −0.019]	+0.051 [−0.120, +0.222]	15 vs 16	p=1
Δ vs. + Delphi's rerankers	−0.037 [−0.123, +0.049]	−0.093 [−0.324, +0.083]	−0.120* [−0.231, −0.019]	+0.000 [−0.139, +0.194]	15 vs 16	p=1
Δ vs. + expansion	−0.111* [−0.200, −0.034]	−0.111 [−0.324, +0.046]	−0.102 [−0.213, +0.000]	−0.093 [−0.278, +0.074]	15 vs 16	p=1
Δ vs. Lexical+BM25 RRF	+0.035 [−0.211, +0.259]	+0.051 [−0.088, +0.194]	−0.009 [−0.148, +0.083]	+0.167 [−0.074, +0.472]	15 vs 15	p=1
Δ vs. Lexical ranker	+0.070 [−0.177, +0.276]	+0.148 [−0.046, +0.343]	+0.046* [+0.009, +0.083]	+0.194 [−0.046, +0.481]	15 vs 14	p=1
Δ vs. BM25	+0.138 [−0.067, +0.301]	+0.093 [−0.037, +0.278]	+0.074* [+0.019, +0.139]	+0.194 [−0.046, +0.481]	15 vs 14	p=1

Table 6: SWE-bench Verified localization, round-3 draw (C0): 62 instances, 12 repositories, scored once. Columns as in Table 5.

System	MRR ↑	R@5 ↑	R@20 ↑	BCY@8k ↑	any-gold ↑	latency (s) ↓
Delphi (frozen)	**0.680**	**0.704**	**0.737**	**0.638**	48/62	3.6
Dense (same embedding)	0.254	0.324	0.591	0.210	39/62	0.5
Dense+BM25 RRF	0.296	0.452	0.648	0.234	43/62	2.0
+ Delphi's rerankers	0.575	0.598	0.616	0.533	41/62	3.7
+ expansion	0.597	0.623	0.657	0.573	44/62	6.2
Lexical+BM25 RRF	0.354	0.500	0.719	0.306	47/62	13.2
Lexical ranker	0.406	0.509	0.702	0.387	46/62	11.0
BM25	0.149	0.230	0.416	0.145	29/62	10.7
*Paired Delphi−system deltas, repository-cluster bootstrap 95% intervals; * interval excludes zero*						
Δ vs. Dense (same embedding)	+0.426* [+0.328, +0.492]	+0.380* [+0.262, +0.583]	+0.146 [+0.000, +0.288]	+0.427* [+0.340, +0.500]	48 vs 39	p=0.064
Δ vs. Dense+BM25 RRF	+0.384* [+0.275, +0.464]	+0.252* [+0.152, +0.442]	+0.089 [−0.038, +0.182]	+0.404* [+0.288, +0.517]	48 vs 43	p=0.3
Δ vs. + Delphi's rerankers	+0.104 [−0.065, +0.210]	+0.106 [+0.000, +0.263]	+0.121 [−0.033, +0.259]	+0.105 [+0.000, +0.190]	48 vs 41	p=0.14
Δ vs. + expansion	+0.083 [−0.107, +0.203]	+0.081 [−0.056, +0.251]	+0.080 [−0.105, +0.217]	+0.065 [−0.062, +0.171]	48 vs 44	p=0.42
Δ vs. Lexical+BM25 RRF	+0.325* [+0.216, +0.453]	+0.204* [+0.099, +0.455]	+0.018 [−0.073, +0.075]	+0.331* [+0.211, +0.552]	48 vs 47	p=1
Δ vs. Lexical ranker	+0.274* [+0.194, +0.412]	+0.195* [+0.107, +0.403]	+0.035 [−0.042, +0.111]	+0.251* [+0.157, +0.433]	48 vs 46	p=0.75
Δ vs. BM25	+0.530* [+0.450, +0.662]	+0.474* [+0.352, +0.726]	+0.321* [+0.189, +0.572]	+0.493* [+0.395, +0.669]	48 vs 29	p=0.00055

Table 7: SWE-bench Verified, round-4 draw (C0): 98 instances drawn under a fresh salt and never used in any round, 11 repositories, scored once from the frozen build after the index audit and the gold-searchability preflight. Columns as in Table 5.

System	MRR ↑	R@5 ↑	R@20 ↑	BCY@8k ↑	any-gold ↑	latency (s) ↓
Delphi (frozen)	**0.700**	**0.745**	**0.820**	**0.685**	83/98	6.4
Dense (same embedding)	0.214	0.389	0.673	0.156	69/98	0.8
Dense+BM25 RRF	0.306	0.509	0.651	0.223	67/98	2.0
+ Delphi's rerankers	0.611	0.622	0.667	0.576	69/98	3.8
+ expansion	0.660	0.684	0.733	0.603	76/98	6.2
Lexical+BM25 RRF	0.383	0.521	0.789	0.350	81/98	4.2
Lexical ranker	0.487	0.603	0.793	0.445	82/98	2.2
BM25	0.168	0.238	0.457	0.146	48/98	2.2
*Paired Delphi−system deltas, repository-cluster bootstrap 95% intervals; * interval excludes zero*						
Δ vs. Dense (same embedding)	+0.486* [+0.434, +0.622]	+0.356* [+0.281, +0.496]	+0.147* [+0.083, +0.330]	+0.529* [+0.453, +0.656]	83 vs 69	p=0.0043
Δ vs. Dense+BM25 RRF	+0.395* [+0.299, +0.552]	+0.236* [+0.133, +0.513]	+0.168* [+0.116, +0.307]	+0.463* [+0.391, +0.636]	83 vs 67	p=0.0015
Δ vs. + Delphi's rerankers	+0.089* [+0.022, +0.227]	+0.123* [+0.067, +0.277]	+0.153* [+0.099, +0.298]	+0.110* [+0.062, +0.238]	83 vs 69	p=0.0043
Δ vs. + expansion	+0.041 [−0.028, +0.112]	+0.061* [+0.023, +0.148]	+0.087* [+0.042, +0.167]	+0.082* [+0.030, +0.152]	83 vs 76	p=0.17
Δ vs. Lexical+BM25 RRF	+0.317* [+0.229, +0.466]	+0.224* [+0.141, +0.415]	+0.031 [−0.045, +0.205]	+0.335* [+0.247, +0.472]	83 vs 81	p=0.79
Δ vs. Lexical ranker	+0.213* [+0.135, +0.404]	+0.142* [+0.050, +0.396]	+0.027 [−0.035, +0.163]	+0.241* [+0.150, +0.439]	83 vs 82	p=1
Δ vs. BM25	+0.533* [+0.453, +0.668]	+0.507* [+0.405, +0.750]	+0.362* [+0.279, +0.554]	+0.539* [+0.455, +0.712]	83 vs 48	p=5.5e − 10

Table 8: SWE-bench Verified, fresh draw for the branch ablation (C0): 60 instances drawn under a third salt from the 338 never used, 10 repositories, scored once. Columns as in Table 5.

System	MRR ↑	R@5 ↑	R@20 ↑	BCY@8k ↑	any-gold ↑	latency (s) ↓
Delphi (frozen)	**0.782**	**0.796**	**0.839**	**0.746**	53/60	8.0
Dense (same embedding)	0.330	0.508	0.704	0.283	43/60	0.7
Dense+BM25 RRF	0.299	0.501	0.719	0.200	44/60	1.9
+ Delphi's rerankers	0.663	0.726	0.731	0.621	45/60	4.5
+ expansion	0.696	0.765	0.786	0.626	48/60	4.7
Lexical+BM25 RRF	0.336	0.501	0.756	0.287	48/60	4.0
Lexical ranker	0.432	0.560	0.728	0.410	46/60	2.0
BM25	0.142	0.212	0.397	0.104	25/60	2.0
*Paired Delphi−system deltas, repository-cluster bootstrap 95% intervals; * interval excludes zero*						
Δ vs. Dense (same embedding)	$+0.452^{*}$ [+0.333, +0.658]	$+0.287^{*}$ [+0.181, +0.476]	$+0.135^{*}$ [+0.029, +0.411]	$+0.463^{*}$ [+0.313, +0.570]	53 vs 43	p=0.0063
Δ vs. Dense+BM25 RRF	$+0.483^{*}$ [+0.405, +0.672]	$+0.294^{*}$ [+0.175, +0.533]	$+0.119^{*}$ [+0.027, +0.318]	$+0.546^{*}$ [+0.466, +0.769]	53 vs 44	p=0.022
Δ vs. + Delphi's rerankers	$+0.119^{*}$ [+0.007, +0.376]	+0.069 [−0.035, +0.286]	$+0.108^{*}$ [+0.013, +0.322]	$+0.125^{*}$ [+0.026, +0.324]	53 vs 45	p=0.039
Δ vs. + expansion	+0.086 [−0.017, +0.323]	+0.031 [−0.068, +0.257]	+0.053 [−0.020, +0.226]	$+0.119^{*}$ [+0.026, +0.295]	53 vs 48	p=0.23
Δ vs. Lexical+BM25 RRF	$+0.446^{*}$ [+0.376, +0.636]	$+0.294^{*}$ [+0.179, +0.537]	$+0.083^{*}$ [+0.037, +0.190]	$+0.458^{*}$ [+0.364, +0.713]	53 vs 48	p=0.23
Δ vs. Lexical ranker	$+0.350^{*}$ [+0.285, +0.493]	$+0.236^{*}$ [+0.148, +0.382]	$+0.111^{*}$ [+0.046, +0.186]	$+0.336^{*}$ [+0.266, +0.476]	53 vs 46	p=0.065
Δ vs. BM25	$+0.640^{*}$ [+0.533, +0.863]	$+0.583^{*}$ [+0.484, +0.817]	$+0.442^{*}$ [+0.325, +0.686]	$+0.642^{*}$ [+0.538, +0.859]	53 vs 25	p=5.8e − 08

Table 9: ARB round-3 partition (C2, validation evidence only): 220 cases, 19 repositories. Columns as in Table 5. Reported for completeness; no confirmatory claim rests on it.

System	MRR ↑	R@5 ↑	R@20 ↑	BCY@8k ↑	any-gold ↑	latency (s) ↓
Delphi (frozen)	**0.332**	**0.476**	**0.648**	**0.296**	163/220	4.3
Dense (same embedding)	0.168	0.218	0.458	0.118	115/220	0.5
Dense+BM25 RRF	0.195	0.247	0.533	0.141	137/220	0.5
+ Delphi's rerankers	0.214	0.317	0.526	0.176	135/220	2.1
+ expansion	0.229	0.389	0.595	0.220	148/220	4.2
Lexical+BM25 RRF	0.160	0.204	0.518	0.124	131/220	1.9
Lexical ranker	0.144	0.215	0.478	0.138	123/220	1.4
BM25	0.128	0.177	0.425	0.075	113/220	1.3
*Paired Delphi−system deltas, repository-cluster bootstrap 95% intervals; * interval excludes zero*						
Δ vs. Dense (same embedding)	$+0.164^{*}$ [+0.049, +0.257]	$+0.258^{*}$ [+0.090, +0.401]	$+0.190^{*}$ [+0.029, +0.321]	$+0.178^{*}$ [+0.058, +0.286]	163 vs 115	p=5.9e − 08
Δ vs. Dense+BM25 RRF	$+0.137^{*}$ [+0.036, +0.215]	$+0.229^{*}$ [+0.074, +0.366]	+0.115 [−0.039, +0.237]	$+0.155^{*}$ [+0.060, +0.246]	163 vs 137	p=0.0022
Δ vs. + Delphi's rerankers	$+0.118^{*}$ [+0.009, +0.198]	$+0.159^{*}$ [+0.007, +0.286]	+0.122 [−0.031, +0.240]	$+0.120^{*}$ [+0.039, +0.203]	163 vs 135	p=0.0011
Δ vs. + expansion	+0.103 [−0.008, +0.189]	+0.087 [−0.100, +0.238]	+0.053 [−0.117, +0.195]	+0.076 [−0.019, +0.157]	163 vs 148	p=0.082
Δ vs. Lexical+BM25 RRF	$+0.172^{*}$ [+0.083, +0.235]	$+0.272^{*}$ [+0.148, +0.372]	$+0.130^{*}$ [+0.024, +0.200]	$+0.172^{*}$ [+0.098, +0.263]	163 vs 131	p=0.00017
Δ vs. Lexical ranker	$+0.188^{*}$ [+0.106, +0.246]	$+0.261^{*}$ [+0.140, +0.361]	$+0.170^{*}$ [+0.088, +0.226]	$+0.158^{*}$ [+0.076, +0.255]	163 vs 123	p=2.4e − 06
Δ vs. BM25	$+0.204^{*}$ [+0.111, +0.277]	$+0.299^{*}$ [+0.157, +0.433]	$+0.223^{*}$ [+0.111, +0.310]	$+0.221^{*}$ [+0.129, +0.303]	163 vs 113	p=5e − 09

Table 10: Pooled SWE-bench Verified localization, 160 C0 instances (Tables 6 and 7), 12 repositories. Paired Delphi−system deltas with repository-cluster bootstrap 95% intervals; * excludes zero. The last ladder rung (+ expansion) is the full conventional stack. Intervals adjusted for the number of looks are in Table 17.

Delphi − system	MRR	R@5	R@20	BCY@8k	any-gold (p)
Dense (same embedding)	$+0.463^{*}$ [+0.418, +0.547]	$+0.365^{*}$ [+0.296, +0.505]	$+0.147^{*}$ [+0.086, +0.277]	$+0.490^{*}$ [+0.434, +0.568]	131 vs 108 (p=0.00043)
Dense+BM25 RRF	$+0.390^{*}$ [+0.316, +0.492]	$+0.242^{*}$ [+0.154, +0.470]	$+0.138^{*}$ [+0.090, +0.226]	$+0.440^{*}$ [+0.394, +0.546]	131 vs 110 (p=0.0011)
+ Delphi's rerankers	$+0.095^{*}$ [+0.023, +0.181]	$+0.116^{*}$ [+0.064, +0.242]	$+0.141^{*}$ [+0.084, +0.248]	$+0.108^{*}$ [+0.064, +0.188]	131 vs 110 (p=0.0011)
+ expansion	+0.057 [−0.036, +0.114]	$+0.069^{*}$ [+0.015, +0.160]	$+0.084^{*}$ [+0.013, +0.154]	$+0.076^{*}$ [+0.017, +0.128]	131 vs 120 (p=0.08)
Lexical+BM25 RRF	$+0.320^{*}$ [+0.255, +0.434]	$+0.216^{*}$ [+0.151, +0.393]	+0.026 [−0.031, +0.128]	$+0.334^{*}$ [+0.286, +0.453]	131 vs 128 (p=0.69)
Lexical ranker	$+0.237^{*}$ [+0.174, +0.391]	$+0.163^{*}$ [+0.081, +0.368]	+0.030 [−0.019, +0.117]	$+0.245^{*}$ [+0.180, +0.406]	131 vs 128 (p=0.66)
BM25	$+0.532^{*}$ [+0.491, +0.630]	$+0.494^{*}$ [+0.411, +0.738]	$+0.346^{*}$ [+0.263, +0.538]	$+0.521^{*}$ [+0.478, +0.645]	131 vs 77 (p=2.7e − 12)

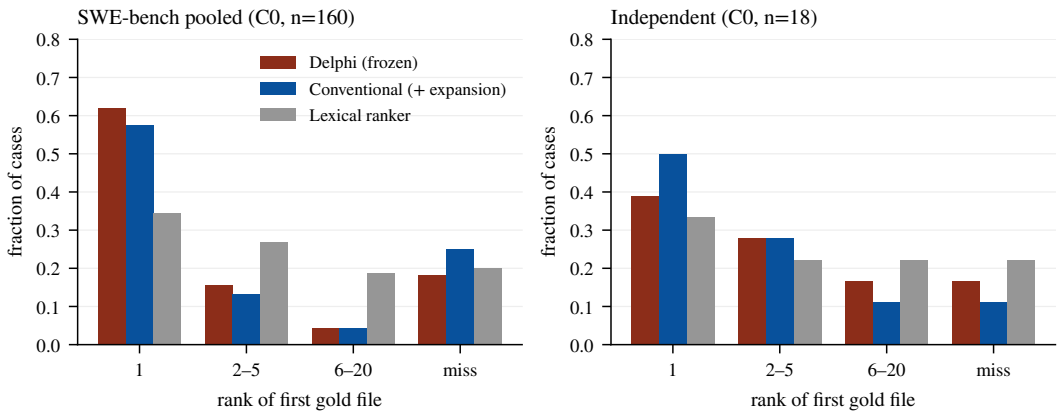

Figure 6: Rank of the first gold file for Delphi, the full conventional stack, and the lexical ranker. Left: 160 pooled SWE-bench instances. Right: the 18-case independent final. "Miss" is no gold file in the top 20.

Table 11: Pooled SWE-bench Verified (160 C0 instances, first two draws) by repository. Means of per-instance metrics; repositories ordered by instance count. Repository-cluster intervals in Table 10 resample these twelve clusters.

Repository	n	Delphi		Conventional (+ expansion)		Lexical ranker	
		MRR	R@20	MRR	R@20	MRR	R@20
`django/django`	73	0.691	0.792	0.618	0.696	0.518	0.779
`sympy/sympy`	24	0.729	0.779	0.663	0.739	0.608	0.773
`sphinx-doc/sphinx`	14	0.615	0.643	0.607	0.607	0.227	0.679
`matplotlib/matplotlib`	11	0.682	0.818	0.530	0.727	0.223	0.500
`scikit-learn/scikit-learn`	9	0.815	0.889	0.623	0.722	0.396	0.889
`astropy/astropy`	7	0.548	0.643	0.762	0.786	0.275	0.762
`pydata/xarray`	7	0.929	1.000	0.571	0.500	0.307	0.857
`pytest-dev/pytest`	6	0.708	1.000	0.843	1.000	0.722	0.833
`psf/requests`	3	0.444	0.667	1.000	1.000	0.399	1.000
`pylint-dev/pylint`	3	0.333	0.333	0.083	0.167	0.067	0.167
`mwaskom/seaborn`	2	0.750	1.000	1.000	0.750	0.667	1.000
`pallets/flask`	1	1.000	1.000	1.000	1.000	0.143	1.000

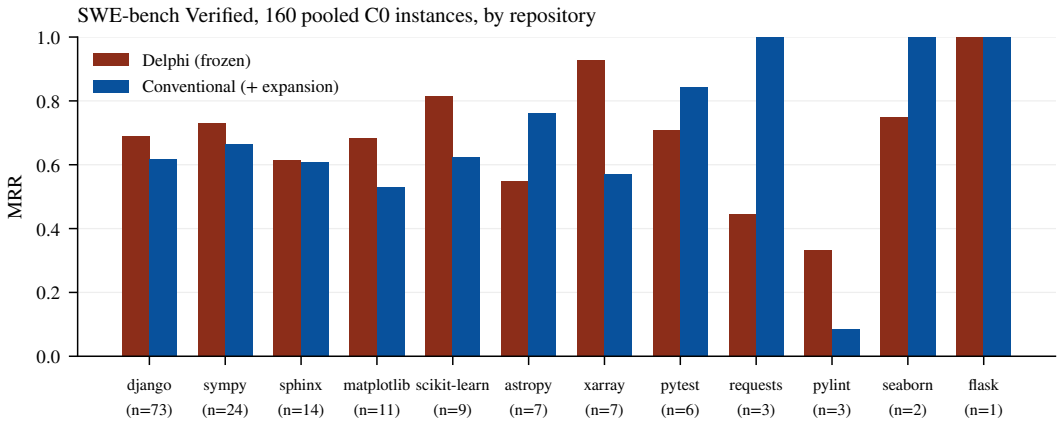

Figure 7: Per-repository MRR on the 160 pooled SWE-bench instances, Delphi against the full conventional stack. Delphi leads on the two largest clusters (`django`, `sympy`) and trails on several small ones; the repository-cluster bootstrap in Table 10 is what turns this heterogeneity into the reported intervals.

Table 12: ARB round-3 partition (C2) by workflow. The concentration of Delphi's lead in `trace2code` and `edit2ripple`, whose queries carry file paths and symbol names that Delphi's exact-path, path-affinity, and symbol branches match directly, is a hypothesis for a designed test; the partition is validation evidence.

Workflow	n	Delphi		Conventional (+ expansion)		Lexical ranker	
		MRR	R@20	MRR	R@20	MRR	R@20
`trace2code`	38	0.579	0.895	0.149	0.368	0.177	0.711
`edit2ripple`	44	0.383	0.650	0.208	0.642	0.250	0.616
`comment2context`	55	0.303	0.567	0.260	0.691	0.172	0.542
`code2test`	83	0.211	0.588	0.257	0.610	0.055	0.255

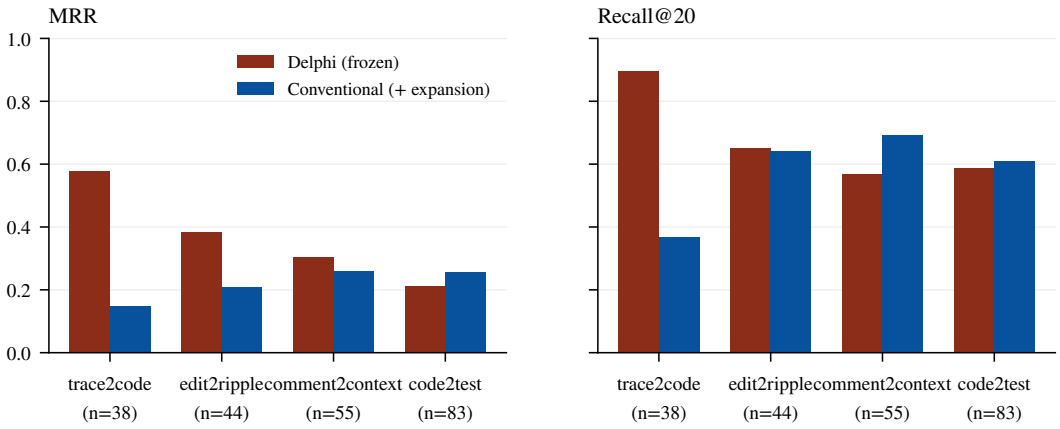

Figure 8: ARB round-3 partition (C2) by workflow, Delphi against the full conventional stack. Queries in `trace2code` carry stack traces with file paths and in `edit2ripple` carry diffs with paths and symbols; `comment2context` and `code2test` carry prose or code without paths.

Table 13: Median query latency in seconds per system and set (wall clock on a warm process, index construction excluded, hosted API round trips included). The lexical comparators' round-3 SWE-bench latencies were recorded in round 3 and are about five times their round-4 values on the same repositories; the ranker code is identical, so the difference is harness conditions, and we do not compare those two columns.

System	Independent	SWE-bench r3	SWE-bench exp.	ARB (C2)
BM25	1.23	10.66	2.19	1.27
Lexical ranker	1.11	11.03	2.20	1.37
Lexical+BM25 RRF	1.57	13.22	4.19	1.93
Dense	0.49	0.51	0.75	0.50
Dense+BM25 RRF	0.20	2.02	2.04	0.53
+ Delphi's rerankers	2.25	3.72	3.78	2.14
+ expansion	4.06	6.20	6.18	4.22
Delphi (frozen)	3.67	3.63	6.36	4.31

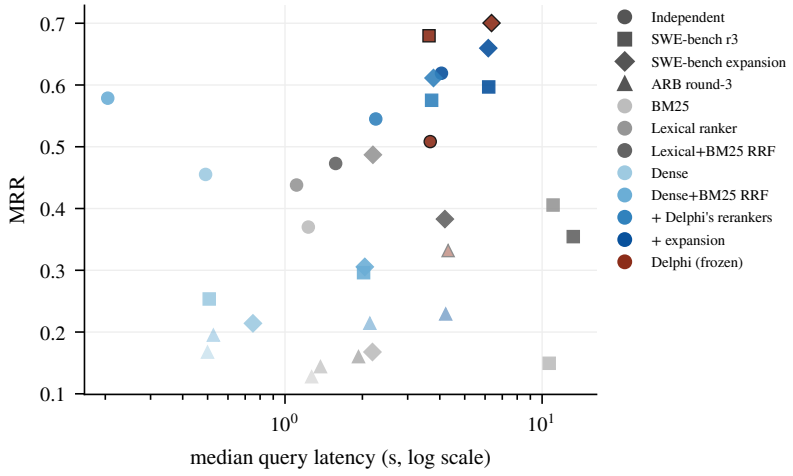

Figure 9: Median query latency against MRR for every system and set (marker shape: set; colour: system; ARB shown faded). The two hosted-LLM rungs and Delphi occupy the same latency band, 2–6 seconds per query; the dense rung is an order of magnitude faster and, on the independent set, competitive.

E CANDIDATE GENERATION × RERANKING FACTORIAL

Design. Two factors, expansion held constant (on, under Delphi's gate). *Candidate generation*: conventional (dense+BM25 reciprocal-rank fusion over ARB chunks, Appendix B) or Delphi's (all six branches, tuned fusion weights, quoted-path demotion, Delphi's chunker and index). *Learned reranking*: none, or Delphi's cross-encoder blend followed by its listwise `gpt-4o` stage. The conventional cells are the ladder's hybrid+expansion system and the full conventional stack; the Delphi-with-reranking cell is the confirmatory run; the Delphi-without-reranking cell is the frozen build served with both rerankers disabled and every other setting unchanged, attested on every response. On the 98-instance draw the factorial was scored after the confirmatory runs and is explanatory (Appendix C); on the fresh 60-instance draw all cells were scored in one pass. The Delphi-without-reranking cell requires the frozen index; it was scored on the 98-instance draw, whose database was retained, on the fresh draw, and on the independent set, whose 18 snapshots were re-indexed for this purpose (the re-indexed build with rerankers is compared with the confirmatory run in Table 15); the round-3 SWE-bench and ARB databases were not retained, so those sets carry only the conventional half.

Table 14: Factorial cell means. Bold marks the best value per metric within a set.

Set / candidates	Rerankers	MRR ↑	R@5 ↑	R@20 ↑	BCY@8k ↑
Independent commit-to-files (C0, n=18)					
conventional	none	0.615	**0.694**	**0.852**	0.454
conventional	Delphi's	**0.619**	0.662	**0.852**	**0.546**
Delphi	none	0.516	0.588	0.750	0.505
Delphi	Delphi's	0.580	0.551	0.750	0.454
Delphi, confirmatory index	Delphi's	0.508	0.551	0.750	0.454
SWE-bench Verified round 3 (C0, n=62)					
conventional	none	0.307	0.444	0.673	0.234
conventional	Delphi's	0.597	0.623	0.657	0.573
Delphi	Delphi's	**0.680**	**0.704**	**0.737**	**0.638**
SWE-bench Verified expansion (C0, n=98)					
conventional	none	0.319	0.503	0.733	0.248
conventional	Delphi's	0.660	0.684	0.733	0.603
Delphi	none	0.507	0.645	0.806	0.408
Delphi	Delphi's	**0.700**	**0.745**	**0.820**	**0.685**
SWE-bench Verified fresh draw (C0, n=60)					
conventional	none	0.295	0.518	0.769	0.183
conventional	Delphi's	0.696	0.765	0.786	0.626
Delphi	none	0.570	0.653	**0.839**	0.469
Delphi	Delphi's	**0.782**	**0.796**	**0.839**	**0.746**
ARB round-3 partition (C2, n=220)					
conventional	none	0.221	0.290	0.596	0.159
conventional	Delphi's	0.229	0.389	0.595	0.220
Delphi	Delphi's	**0.332**	**0.476**	**0.648**	**0.296**

Reading. On issue queries the two factors are partial substitutes: on the 98-instance draw Delphi's candidates lead by $+0.188$ MRR before reranking and by $+0.041$ after it, because the rerankers add $+0.341$ to the conventional pool and $+0.193$ to Delphi's; the Recall@20 lead ($+0.073$ before, $+0.087$ after) is not affected by reranking, which cannot promote files outside its window. The fresh 60-instance draw gives the same reading with larger effects: candidates $+0.275$ MRR $[+0.195, +0.402]$, $+0.069$ Recall@20 $[-0.018, +0.253]$, and $+0.286$ BCY before reranking; rerankers $+0.401$ MRR on conventional candidates and $+0.212$ on Delphi's; after reranking $+0.086$ MRR $[-0.017, +0.323]$, $+0.053$ Recall@20 $[-0.020, +0.226]$, $+0.119$ BCY $[+0.026, +0.295]$. On commit-to-files queries the rerankers add $+0.004$ MRR to conventional candidates given expansion (-0.034 without it, Table 5) and $+0.065$ $[-0.044, +0.176]$ to Delphi's on the same re-indexed corpus; Delphi's candidate pool trails the conventional hybrid by -0.099 MRR $[-0.190, -0.028]$ and -0.102 Recall@20 $[-0.213, +0.000]$ before reranking, and Recall@20 is unchanged by reranking for both pools. On the ARB partition the rerankers add $+0.008$ to conventional candidates.

Table 15: Factorial contrasts: paired differences with repository-cluster bootstrap 95% intervals (20,000 resamples, seed 1042); * excludes zero. "Rerankers" rows are the conditional effect of adding Delphi's two rerankers to the named candidates; "candidates" rows are the conditional effect of Delphi's candidate generator against the conventional one at the named reranking level. Recall@20 is unchanged by reranking for conventional candidates by construction (a 30-candidate window reordered into a top 20).

Set	Contrast	ΔMRR	ΔR@20	ΔBCY@8k
Independent commit-to-files	rerankers, conventional candidates	$+0.004$ $[-0.093, +0.119]$	$+0.000$ $[+0.000, +0.000]$	$+0.093$ $[-0.111, +0.306]$
	rerankers, Delphi candidates	$+0.065$ $[-0.044, +0.176]$	$+0.000$ $[+0.000, +0.000]$	-0.051 $[-0.190, +0.125]$
	Delphi − conventional candidates, no rerankers	-0.099^* $[-0.190, -0.028]$	-0.102 $[-0.213, +0.000]$	$+0.051$ $[-0.060, +0.176]$
	Delphi − conventional candidates, with rerankers	-0.039 $[-0.151, +0.046]$	-0.102 $[-0.213, +0.000]$	-0.093 $[-0.278, +0.074]$
	Delphi (confirmatory index) − conventional, with rerankers	-0.111^* $[-0.200, -0.034]$	-0.102 $[-0.213, +0.000]$	-0.093 $[-0.278, +0.074]$
	Delphi re-indexed − confirmatory index (same configuration)	$+0.072^*$ $[+0.005, +0.159]$	$+0.000$ $[+0.000, +0.000]$	$+0.000$ $[+0.000, +0.000]$
SWE-bench Verified round 3	rerankers, conventional candidates	$+0.289^*$ $[+0.219, +0.471]$	-0.016 $[-0.053, +0.079]$	$+0.339^*$ $[+0.246, +0.517]$
	Delphi − conventional candidates, with rerankers	$+0.083$ $[-0.107, +0.203]$	$+0.080$ $[-0.105, +0.217]$	$+0.065$ $[-0.062, +0.171]$
SWE-bench Verified expansion	rerankers, conventional candidates	$+0.341^*$ $[+0.256, +0.468]$	$+0.000$ $[+0.000, +0.000]$	$+0.355^*$ $[+0.290, +0.500]$
	rerankers, Delphi candidates	$+0.193^*$ $[+0.161, +0.285]$	$+0.014$ $[+0.000, +0.043]$	$+0.277^*$ $[+0.221, +0.373]$
	Delphi − conventional candidates, no rerankers	$+0.188^*$ $[+0.036, +0.317]$	$+0.073^*$ $[+0.005, +0.159]$	$+0.160^*$ $[+0.072, +0.310]$
	Delphi − conventional candidates, with rerankers	$+0.041$ $[-0.028, +0.112]$	$+0.087^*$ $[+0.042, +0.167]$	$+0.082^*$ $[+0.030, +0.152]$
SWE-bench Verified fresh draw	rerankers, conventional candidates	$+0.401^*$ $[+0.297, +0.508]$	$+0.017$ $[+0.000, +0.061]$	$+0.443^*$ $[+0.355, +0.542]$
	rerankers, Delphi candidates	$+0.212^*$ $[+0.119, +0.419]$	$+0.000$ $[+0.000, +0.000]$	$+0.276^*$ $[+0.158, +0.488]$
	Delphi − conventional candidates, no rerankers	$+0.275^*$ $[+0.195, +0.402]$	$+0.069$ $[-0.018, +0.253]$	$+0.286^*$ $[+0.161, +0.413]$
	Delphi − conventional candidates, with rerankers	$+0.086$ $[-0.017, +0.323]$	$+0.053$ $[-0.020, +0.226]$	$+0.119^*$ $[+0.026, +0.295]$
ARB round-3 partition	rerankers, conventional candidates	$+0.008$ $[-0.011, +0.031]$	-0.001 $[-0.014, +0.010]$	$+0.061^*$ $[+0.042, +0.094]$
	Delphi − conventional candidates, with rerankers	$+0.103$ $[-0.008, +0.189]$	$+0.053$ $[-0.117, +0.195]$	$+0.076$ $[-0.019, +0.157]$

Re-indexing shift. The independent corpus had to be re-indexed for its Delphi cells. Re-running the frozen configuration on the new index with an empty stage cache reproduced the confirmatory run's Recall@20 and BCY for every case but changed MRR on 5 of 18 cases ($+0.072$ $[+0.005, +0.159]$ in aggregate); only 1 of 18 ordered top-20 lists and 7 of 18 top-20 sets were identical. The sources are re-embedding (which jitters similarities at the fifth decimal and reorders nearties among 10^5 chunks) and fresh seeded LLM calls for expansion and listwise reranking, which are not guaranteed identical across sessions. The confirmatory independent numbers are those of the original index; the factorial's within-index contrasts use the re-indexed cells, and the confirmatory-index contrast is shown alongside them in Table 15.

F BRANCH-LEVEL ABLATION ON THE FRESH DRAW

Design. The fresh 60-instance SWE-bench Verified draw (Table 8) was taken for this experiment, with its size, salt, and preprocessing rules fixed before any scoring, and no system had seen any of its cases. All configurations use the frozen build with both learned rerankers disabled and expansion on, so only candidate generation varies. The baseline is the full six-branch generator with the tuned fusion weights ($0.50/0.25/0.20/0.15/0.15/0.05$ for vector, BM25, exact symbol, exact path, path affinity, trigram). Each leave-one-out ablation sets one branch's fusion weight to zero through the build's per-deployment weight setting, leaving the other weights unchanged; a further configuration keeps only the vector and BM25 branches at equal weights ($0.5/0.5$), which is the conventional two-branch fusion computed inside Delphi's own chunker and index and therefore separates the effect of Delphi's extra branches and weights from the effect of its chunking and indexing. Every configuration was scored once; every response attests the served weights.

Results. Tables 1 and 16. The full generator scores 0.570 MRR, 0.653 Recall@5, 0.839 Recall@20, and 0.469 BCY without rerankers. Leave-one-out effects are small for every non-vector branch (exact symbol -0.025 MRR $[-0.072, +0.015]$, -0.033 Recall@20; trigram -0.009; exact path -0.001; path affinity -0.001; BM25 -0.002 MRR with Recall@20 $+0.021$), and large for the vector branch (-0.188 MRR $[-0.265, -0.090]$, -0.126 Recall@20 $[-0.176, -0.029]$). Removing all four structural branches at once costs -0.060 MRR $[-0.096, -0.005]$ at Delphi's vector/BM25 weights ($0.50/0.25$), and moving those two branches to equal weights costs a further -0.061 (-0.121 $[-0.160, -0.070]$ in total); the vector branch alone scores 0.450, so BM25 at equal weight adds nothing to it and at the tuned weight adds $+0.060$. The equal-weight two-branch configuration is the conventional ladder's fusion design computed inside Delphi's index; it scores 0.449 MRR and 0.775 Recall@20 against 0.295 and 0.769 for the ladder's hybrid+expansion cell over ARB chunks. The candidate-pool advantage on this draw ($+0.275$ MRR before reranking) therefore decomposes into chunking, indexing, and fusion implementation ($+0.154$ at identical design),

vector-heavy weighting ($+0.061$), and the structural branches jointly ($+0.060$). The branches are redundant with one another: any one can be removed at a cost under 0.03, but removing all four is visible. Recall@20 moves with the structural branches ($0.775 \rightarrow 0.839$) and not with the weights. All intervals are from 60 instances in 10 repository clusters and the joint effects, not the single-branch ones, are the resolved findings.

Table 16: Branch-ablation cell means on the fresh draw. Bold marks the best value per metric.

Configuration (rerankers off)	MRR ↑	R@5 ↑	R@20 ↑	BCY@8k ↑
All six branches, tuned weights	**0.570**	0.653	0.839	0.469
− exact symbol	0.545	0.644	0.806	0.453
− exact path	0.569	0.653	0.839	0.469
− path affinity	0.569	0.653	0.831	0.469
− trigram	0.561	0.653	0.839	0.453
− BM25	0.568	**0.685**	**0.860**	**0.514**
− vector	0.382	0.501	0.713	0.264
vector + BM25 only, equal weights	0.449	0.619	0.775	0.336
vector + BM25 only, Delphi's weights (0.50/0.25)	0.510	0.628	0.775	0.403
vector only	0.450	0.535	0.792	0.389

G SEQUENTIAL ANALYSIS OF THE THREE SWE-BENCH DRAWS

The three SWE-bench Verified draws were scored in sequence (62 in round 3; 98 in round 4; 60 for the branch ablation), each with its size and preprocessing rules fixed before scoring and each scored once per system. Because the second and third draws were taken after earlier results were known, a pooled interval computed as if the 220 instances were one prespecified sample understates the uncertainty of a reader who allows that the sequence could have stopped after any draw. Table 17 therefore reports, for Delphi against the full conventional stack and against the lexical ranker, each draw on its own, the pooled estimate after each draw, the sign of the point estimate in each draw, and intervals at 95% and at $1 - 0.05/3$ (three looks). Our reading rule is stated in §3: the 98-instance draw is the prespecified replication of the 62-instance comparison, the 60-instance draw is a third replication, and pooled estimates are descriptive.

Table 17: Sequential view of the SWE-bench draws: paired Delphi−system mean differences with repository-cluster bootstrap intervals at 95% and adjusted for three looks (98.3%); * excludes zero at the adjusted level. Pooled rows accumulate the draws in the order they were taken.

Comparator	Draw	MRR	R@5	R@20	BCY@8k
Full conventional stack	Round-3 draw (62)	$+0.083$ $[-0.174, +0.249]$	$+0.081$ $[-0.103, +0.307]$	$+0.080$ $[-0.150, +0.268]$	$+0.065$ $[-0.111, +0.211]$
	Round-4 draw (98)	$+0.041$ $[-0.054, +0.143]$	$+0.061^*$ $[+0.013, +0.178]$	$+0.087^*$ $[+0.032, +0.198]$	$+0.082^*$ $[+0.017, +0.180]$
	Fresh draw (60)	$+0.086$ $[-0.039, +0.392]$	$+0.031$ $[-0.087, +0.323]$	$+0.053$ $[-0.028, +0.267]$	$+0.119$ $[-0.003, +0.359]$
	Pooled after two draws (160)	$+0.057$ $[-0.068, +0.133]$	$+0.069$ $[-0.005, +0.190]$	$+0.084$ $[-0.006, +0.176]$	$+0.076$ $[-0.006, +0.145]$
	Pooled after three draws (220)	$+0.065$ $[-0.019, +0.152]$	$+0.059^*$ $[+0.008, +0.182]$	$+0.076^*$ $[+0.025, +0.160]$	$+0.087^*$ $[+0.037, +0.162]$
Lexical ranker	Round-3 draw (62)	$+0.274^*$ $[+0.162, +0.446]$	$+0.195^*$ $[+0.087, +0.442]$	$+0.035$ $[-0.074, +0.143]$	$+0.251^*$ $[+0.136, +0.470]$
	Round-4 draw (98)	$+0.213^*$ $[+0.119, +0.438]$	$+0.142^*$ $[+0.039, +0.453]$	$+0.027$ $[-0.050, +0.204]$	$+0.241^*$ $[+0.126, +0.481]$
	Fresh draw (60)	$+0.350^*$ $[+0.272, +0.528]$	$+0.236^*$ $[+0.123, +0.415]$	$+0.111^*$ $[+0.024, +0.215]$	$+0.336^*$ $[+0.246, +0.515]$
	Pooled after two draws (160)	$+0.237^*$ $[+0.163, +0.417]$	$+0.163^*$ $[+0.073, +0.396]$	$+0.030$ $[-0.035, +0.145]$	$+0.245^*$ $[+0.170, +0.438]$
	Pooled after three draws (220)	$+0.268^*$ $[+0.215, +0.407]$	$+0.183^*$ $[+0.108, +0.349]$	$+0.052^*$ $[+0.011, +0.129]$	$+0.270^*$ $[+0.216, +0.415]$

H DEVELOPMENT ABLATIONS AND NEGATIVE RESULTS

Round-3 preregistered ablations (C1 development sets). Exact vector scan replaced approximate (HNSW) search: $+0.020$ Recall@20 $[0.000, 0.063]$ on the ARB development set, with approximate search additionally implicated in run-to-run variance (Appendix J); accepted. Query expansion off versus on: expansion stayed on. Gemini embedding swap in place of `text-embedding-3-small`: MRR -0.054 $[-0.102, -0.011]$; rejected. File-level Okapi BM25 branch (whole-file BM25 as a seventh fused branch): the single approved configuration failed three preregistered gates (MRR interval floor -0.050 and BCY floor -0.037 against a -0.01 gate on each; warm median latency ratio 1.34 against a 1.20 gate) despite a non-negative Recall@20 delta

and zero any-gold losses; rejected and shipped default-off. Two development findings moved code: review-comment queries retrieved the file the comment already quoted, so paths present in the query envelope are demoted after the cross-encoder (quoted-path demotion); and a global exclusion of generated protobuf sources had silently dropped a gold file, so agent-mode indexing now retains generated source under the existing size and binary guards.

Round-2 components, accepted on the ARB v2 pool (now C2). Aligning the query-time embedding model with the index moved workflow-macro MRR from 0.055 to 0.173; reciprocal-rank fusion replaced max-normalized score fusion; the path-affinity branch made file paths searchable; the `ms-marco-MiniLM` cross-encoder at k=30, α=0.4 moved MRR $0.176 \rightarrow 0.193$; hypothetical-document expansion moved held-out MRR $0.228 \rightarrow 0.241$ and Recall@20 $0.552 \rightarrow 0.579$; listwise reranking moved held-out MRR $0.241 \rightarrow 0.285$, Recall@5 $0.355 \rightarrow 0.419$, and BCY@8k $0.376 \rightarrow 0.446$. These are the decisions that make the round-3 ARB partition C2.

Round-2 ideas measured and rejected. Recorded so that the negative space of the design is visible. Four of these looked like wins on the 75-case development split and lost on the 220-case held-out split; development differences under about 0.03 were not trustworthy at that size, and run-to-run variance with an LLM in the loop was about 0.02.

- *File-evidence aggregation* (best chunk plus damped support from other chunks): Recall@20 $0.544 \rightarrow 0.467$ at w=0.5; many gold files match on exactly one chunk.
- *Deeper candidate pools* (50, 100, 150 per branch): MRR $0.197/0.195/0.194$; candidate supply was not the constraint.
- *Deeper rerank window* (k=30, 60, 100): Recall@20 $0.560/0.552/0.444$; the cross-encoder promotes plausible files from the tail.
- *Pure cross-encoder ordering* (α=1.0): MRR $0.193 \rightarrow 0.154$, the worst configuration tested.
- *Larger code-aware reranker* (`bge-reranker-base`, 278M parameters): lost on every metric and ran 7× slower (21.3 s vs. 2.9 s).
- *Fusion weight rebalancing* (five configurations): defaults at or near the optimum; lexical-heavy clearly worse once embeddings were aligned.
- *Reverse-dependency branch* at a global weight: `edit2ripple` Recall@5 $0.238 \rightarrow 0.381$ but overall MRR $0.193 \rightarrow 0.163$; gated on query intent, the harm and the gain both vanished (held-out MRR $+0.004$, Recall@5 -0.007) because the listwise stage already promotes dependents.
- *File path prepended to rerank passages*: development Recall@5 $0.327 \rightarrow 0.333$, held-out $0.355 \rightarrow 0.346$.
- *Wider listwise pool* (20 to 40 candidates): $+0.016$ Recall@20 for -0.016 MRR and -0.008 Recall@5 and $+1.2$ s.
- *Cascade listwise* (second pass over the top 6 with 1,200-character excerpts): $+0.001$ MRR for -0.036 Recall@5 and -0.017 Recall@20.
- *Listwise window and excerpt tuning* (k=12, 700 characters): development Recall@5 0.411 vs. 0.391, held-out 0.378 vs. 0.419.

The diagnostic that these attempts shared: on the held-out split 70.5% of cases had a gold file in the top 20 and 54.1% in the top 5, so the remaining gap was inside the reranker's window, and no change to coverage, fusion, pool size, or prompt shape moved it.

I DOCUMENTATION TRACK

Cases. Forty DS-1000 problems (development subset, C1), eight each from NumPy, pandas, Matplotlib, scikit-learn, and TensorFlow, each annotated with a gold API identifier that a correct answer must name. Identifier hit is a case-insensitive substring match of the gold identifier against the scored text.

Arms. *Documentation engine* (hosted): library resolution followed by documentation retrieval; two modes were recorded, "oracle" (the library ID chosen by us) and "quality-guided" (the engine's own top-ranked library), both compacted to the 8,000-token budget. *Delphi*: the frozen build indexing the same libraries' documentation sources, k=5 and k=20, compacted to the same budget. *Synthesis engine* (hosted): its native answer mode, recorded once per case with its five

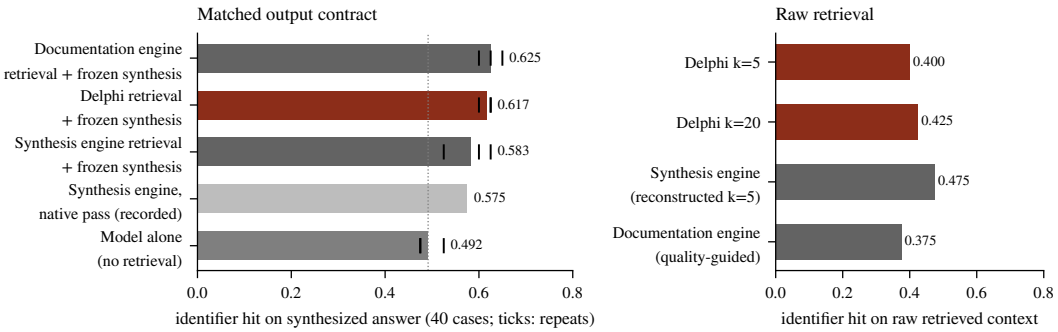

Figure 10: Documentation track, 40 development cases (C1). Left: every engine's retrieved evidence—including the synthesis engine's own recorded sources—through one frozen synthesis stage (`gpt-5.4-mini`, one prompt, 8,000-token budget), three repeats (ticks); dotted: the no-retrieval floor. Right: raw retrieved context scored directly.

cited source chunks; its raw-retrieval mode returned no sources during the recorded round. *Reconstructed synthesis-engine retrieval*: the five source chunks extracted from each recorded response (mean 2,754 tokens per case), packed with the identical routine and budget. *Frozen synthesis stage*: `gpt-5.4-mini`, temperature 0, 1,600 output tokens, the system prompt below, three repeats per arm. *Control*: the same model and repeats with the control prompt and no retrieved context.

Synthesis system prompt (verbatim).

```
You are a documentation-grounded coding assistant.
Answer the coding question directly and concretely.
Name the exact library APIs, functions, methods, and
parameters that solve the task, using their precise
identifiers.
Ground your answer in the retrieved documentation when it is
relevant, and cite the source paths you used.
If the retrieved documentation is missing or unhelpful,
still answer from your own knowledge of the library and say
the documentation did not cover it.
Do not use Markdown code fences.
```

The user turn is `<coding_question>...</coding_question>` followed by `<retrieved_documentation>...</retrieved_documentation>`.

Control system prompt (verbatim).

```
You are a coding assistant.
Answer the coding question directly and concretely.
Name the exact library APIs, functions, methods, and
parameters that solve the task, using their precise
identifiers.
Answer from your own knowledge of the library.
Do not use Markdown code fences.
```

Raw retrieval. Identifier hit on the raw retrieved context: Delphi k=5 0.400, k=20 0.425; documentation engine 0.375 in both the quality-guided and the oracle-ID mode; reconstructed synthesis-engine sources 0.475. The synthesis engine's raw retrieval is therefore ahead of Delphi by 0.05–0.075 on this set, a difference that is not resolved after the frozen synthesis stage (Table 18).

DS-1000 execution (unresolved, not negative). A separate arm executes generated code against the DS-1000 tests with a frozen generator (40 cases $\times$ 3 repeats): no retrieval 0.575; documentation engine 0.567 (-0.008 [-0.142, 0.125]); Delphi retrieval+synthesis 0.592 ($+0.017$ [-0.125, 0.158]); synthesis engine 0.642 ($+0.067$ [-0.075, 0.200]). Every interval spans effects from substantial harm

Table 18: Matched-output-contract documentation comparison, 40 development cases (C1) $\times$ 3 repeats through one frozen synthesis stage. Deltas are case-cluster bootstrap 95% intervals against the no-retrieval control. The synthesis engine's native single pass is shown for reference only.

Arm	Identifier hit	Per-repeat	Δ vs. control
Documentation engine retrieval + frozen synthesis	0.625	0.650/0.600/0.625	$+0.133$ $[+0.008, +0.258]$
Delphi retrieval + frozen synthesis	0.617	0.600/0.625/0.625	$+0.125$ $[+0.000, +0.250]$
Synthesis engine retrieval + frozen synthesis	0.583	0.625/0.525/0.600	$+0.092$ $[-0.008, +0.200]$
Synthesis engine, native synthesis (recorded)	0.575	single pass	—
Model alone (no retrieval)	0.492	0.475/0.475/0.525	—

Table 19: Engine-versus-engine paired differences under the matched contract (case-cluster bootstrap 95% intervals; wins/losses/ties over 40 cases). Every interval includes zero.

Comparison	Δ	95% CI	W/L/T
Delphi $-$ synthesis engine (matched)	$+0.033$	$[-0.092, +0.158]$	7/5/28
Delphi $-$ synthesis engine (native)	$+0.042$	$[-0.092, +0.183]$	6/6/28
Delphi $-$ documentation engine	-0.008	$[-0.117, +0.092]$	6/5/29
Documentation engine $-$ synthesis engine (matched)	$+0.042$	$[-0.083, +0.167]$	10/7/23
Synthesis engine matched $-$ native	$+0.008$	$[-0.050, +0.075]$	5/4/31

to substantial benefit; the experiment does not show that retrieval fails to help, it shows that 40 cases cannot tell.

J DETERMINISM

Probes. Eight ARB development queries repeated ten times each against the pre-freeze stack (mean exact top-10 rate 30.6%), with paired traces of every stage. Three sources were isolated: the listwise and expansion stages were uncached and unseeded; the fusion SQL broke ties by insertion order; and approximate nearest-neighbour search returned different neighbour sets under concurrent load. Fixes: total ordering in every SQL stage (score, then path, then chunk ID), single-flight sharing of concurrent identical hosted calls, an explicit LLM seed, and a persistent stage cache keyed by model, prompt, and seed. After the fixes: 100% exact repeats in-process across two concurrent 8×10 packets and two 75-case repeats; a cold restart moved $7/8$ lists until the cache was warm, after which $8/8$ survived two restarts.

Embedding stability. `text-embedding-3-small`: $N{=}20$ repeats of the same texts differed at the fifth decimal and never changed top-ten membership; Gemini embeddings were bitwise identical across repeats.

Like-for-like measures (Table 2, Figure 11). Ten documentation cases $\times$ ten runs per engine, all $\binom{10}{2} \times 10 = 450$ run pairs. *Retrieved set exact*: the set of retrieved units is identical (units: retrieved chunks for Delphi and the documentation engine; the set of cited URLs for the synthesis engine, whose retrieval is otherwise unobservable). *Retrieved order exact*: the ordered list is identical. *Context byte-exact*: the delivered text is identical. *Identifier-hit agreement*: the scored outcome agrees across the pair. Because the observable units differ in granularity, the retrieved-set columns are reported per engine and are not used to rank engines against one another. A comparison of Delphi's and the documentation engine's byte-exact context rate with the synthesis engine's byte-exact *answer* rate measures different quantities and is the comparison Table 2 replaces.

K EXECUTABLE PILOT

Closed-tool seeding (replication of ARB). ARB's seed intervention holds a closed-tool agent fixed and varies its initial context. We replicated it with `gpt-5.4-mini`, tools `list_dir`/`grep`/`read_file`/`submit`, and 40 paired development cases per arm, adding a seed

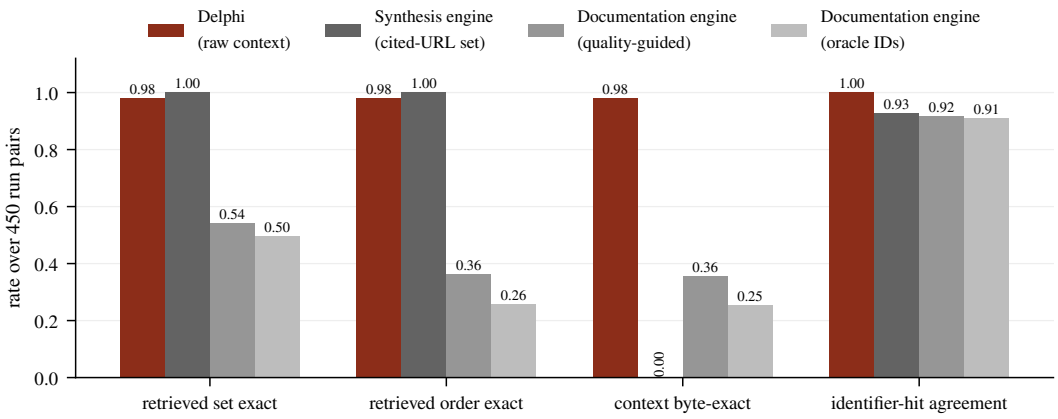

Figure 11: Like-for-like repeatability over 450 run pairs per engine (Table 2). The synthesis engine's cited-source set never changes while its answer text always does; Delphi's retrieved text is byte-exact in 98% of pairs and its scored outcome in 100%.

from the pre-freeze Delphi. File-F1: no seed 0.046; random files 0.013; BM25 seed 0.093; lexical seed 0.119; Delphi seed 0.234; oracle 0.858. The Delphi seed beats no-seed by $+0.188$ $[0.055,\ 0.294]$ and BM25 by $+0.142$ $[0.059,\ 0.208]$; against lexical the interval includes zero $[-0.002,\ 0.190]$. Seeds change which files a frozen agent submits, ordered by seed quality, as ARB also found.

Protocol (preregistered before any run). Tasks: the 62 round-3 SWE-bench Verified instances (C0). Agent: mini-SWE-agent 2.4.6, text-based action format, `gpt-5.4-mini`, official `x86_64` instance images run under emulation on an Apple-silicon host. Conditions: *no seed*; *random* (five candidate files drawn with seed 1042 per instance, gold excluded); *Delphi* (top five of the frozen build's recorded ranking); *conventional* (top five of the full conventional stack's recorded ranking). Seed block: identical wording, up to five paths with the first 40 lines of each file at the base commit, capped at 3,000 tokens. Budgets: step limit 50 with a 2.0 USD cost cap; a tighter secondary budget was preregistered and not run because agents used 7–8 steps of 50. Outcome: verified repair under the official SWE-bench harness (v5.0.2) against the SWE-bench Verified release; secondary outcomes API cost, steps, and submission rate. Analysis: paired by instance; instance-cluster and repository-cluster bootstrap 95% intervals (20,000 resamples, seed 1042); exact McNemar on discordant pairs. An interval including zero at n=62 is reported as unresolved. Repeats: two independent trajectories per instance and condition (496 trajectories), pooled as per-instance means.

Seed block (verbatim shape).

```
<retrieved_context>
A code search engine retrieved the following repository
files as potentially relevant to this issue, best match
first.  Treat them as hints and verify them with your own
tools; the fix may involve other files.
### path/to/first_file.py (lines 1-40)
...first 40 lines of the file at the base commit ...
### path/to/second_file.py (lines 1-40)
...
</retrieved_context>
```

When the 3,000-token budget would be exceeded, a file's head is dropped and only its path is listed; when even the path does not fit, the block ends. The random condition uses the same block with five files drawn uniformly from the snapshot's non-gold files.

Reading the repeats. In the first repeat the Delphi seed resolved 31 instances against 25 for no seed, an apparent $+0.097$ that a single-repeat analysis would have reported with an instance-cluster interval excluding zero ($[+0.016,\ +0.194]$); in the second repeat both resolved 25. The conventional

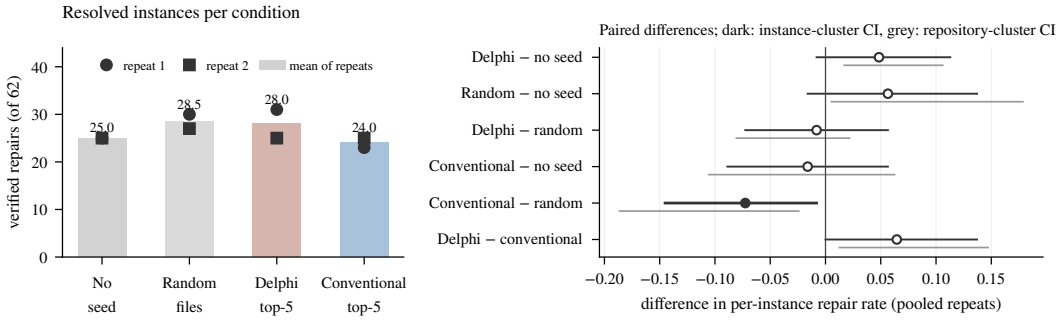

Figure 12: Executable pilot: 62 C0 SWE-bench Verified instances, mini-SWE-agent with `gpt-5.4-mini`, step budget 50, two trajectories per instance and condition. Left: verified repairs per condition and repeat. Right: paired differences in per-instance repair rate with instance-cluster (dark) and repository-cluster (grey) intervals; the filled marker is the only interval excluding zero.

Table 20: Pooled pilot results: per-instance means over the two repeats. * instance-cluster interval excludes zero.

Condition	Resolved (mean of 2 repeats, of 62)	Rate	Mean cost (USD)	Mean steps
No seed	25.0	0.403	0.0190	8.0
Random files (control)	28.5	0.460	0.0251	8.3
Delphi top-5 (frozen)	28.0	0.452	0.0212	7.2
Conventional ladder top-5	24.0	0.387	0.0232	7.5
Paired differences in per-instance resolved rate: instance-cluster 95% CI [repository-cluster CI]; instances favouring a/b				
Delphi − no seed	$+0.048$ $[-0.008, +0.113]$ $[[+0.017, +0.106]]$; 6/2			
Delphi − random	-0.008 $[-0.073, +0.056]$ $[[-0.081, +0.022]]$; 6/8			
Delphi − conventional	$+0.065$ $[+0.000, +0.137]$ $[[+0.013, +0.147]]$; 12/5			
Conventional − no seed	-0.016 $[-0.089, +0.056]$ $[[-0.106, +0.062]]$; 5/8			
Conventional − random	-0.073^{*} $[-0.145, -0.008]$ $[[-0.186, -0.024]]$; 3/9			
Random − no seed	$+0.056$ $[-0.016, +0.137]$ $[[+0.005, +0.179]]$; 9/5			

seed resolved 23 in the first repeat and 25 in the second. Within a condition, the two repeats disagree on 8 (no seed), 7 (random), 10 (Delphi), and 8 (conventional) of 62 instances. Pooled over repeats, the only interval excluding zero is the conventional seed's deficit against the random control. Sampling variance across trajectories of the same instance is therefore of the same order as the between-condition differences, and any single-trajectory pilot of this size should be read as inconclusive regardless of its point estimate.

Seed audit: what the block contained. The seed names files and shows their first 40 lines. Against the gold patches: Delphi's seed named at least one gold file on 47 of 62 instances (50 seeded gold files in total); the 40-line head covered a region the gold patch edits for 9 of those 50 files (on 9 instances); the median first edited line in a seeded gold file is 200 and 48% of first edits lie beyond line 200. The conventional stack's seed named a gold file on 42 instances (44 files), with the head covering an edit for 9. The pilot therefore tested file hints with file headers, not the engine's task-relevant evidence. The random control's 310 files were 93 tests, 77 Python sources, 47 documentation files, and 93 other files; 4 (1.3%) shared a directory with a gold file and 18 (5.8%) a top-level package; 3 instances received any same-directory file.

Seed-interface check (exploratory). Two further arms were run on the same 62 instances with one trajectory each, after the preregistered pilot and outside it: *paths only* (Delphi's top five paths, no content; mean seeded statement 591 tokens) and *paths + best-matching chunk* (each file's highest-scoring ARB chunk for the issue text under BM25, symbol chunks preferred, capped at 60 lines; 1,538 tokens), against the pilot's *paths + file heads* (2,066 tokens). The chunk seed covered an edited region in 13 of the 50 seeded gold files, against 9 for the file heads. Table 23 compares the three interfaces with the first pilot repeat, all single-trajectory: paths only resolved 26 instances, paths + heads 31 (25 in the second pilot repeat), paths + chunks 24, no seed 25. Against no seed, paths only is $+0.016$ $[-0.097, +0.129]$ and paths + chunks -0.016 $[-0.097, +0.065]$; paths + chunks

Table 21: First repeat alone.

Condition	Resolved (of 62)	Rate	Mean cost (USD)	Mean steps
No seed	25.0	0.403	0.0162	7.7
Random files (control)	30.0	0.484	0.0263	9.0
Delphi top-5 (frozen)	31.0	0.500	0.0208	7.3
Conventional ladder top-5	23.0	0.371	0.0240	7.5
Paired differences in per-instance resolved rate: instance-cluster 95% CI [repository-cluster CI]; instances favouring a/b				
Delphi − no seed	$+0.097^{*}$ [+0.016, +0.194] [[+0.019, +0.244]]; 7/1			
Delphi − random	+0.016 [−0.065, +0.097] [[−0.057, +0.154]]; 4/3			
Delphi − conventional	$+0.129^{*}$ [+0.032, +0.242] [[+0.029, +0.317]]; 10/2			
Conventional − no seed	−0.032 [−0.113, +0.048] [[−0.147, +0.045]]; 2/4			
Conventional − random	-0.113^{*} [−0.210, −0.032] [[−0.208, −0.059]]; 1/8			
Random − no seed	+0.081 [−0.016, +0.177] [[−0.015, +0.200]]; 7/2			

Table 22: Second repeat alone.

Condition	Resolved (of 62)	Rate	Mean cost (USD)	Mean steps
No seed	25.0	0.403	0.0219	8.3
Random files (control)	27.0	0.435	0.0238	7.6
Delphi top-5 (frozen)	25.0	0.403	0.0216	7.2
Conventional ladder top-5	25.0	0.403	0.0224	7.5
Paired differences in per-instance resolved rate: instance-cluster 95% CI [repository-cluster CI]; instances favouring a/b				
Delphi − no seed	+0.000 [−0.097, +0.097] [[−0.097, +0.061]]; 5/5			
Delphi − random	−0.032 [−0.129, +0.065] [[−0.219, +0.040]]; 4/6			
Delphi − conventional	+0.000 [−0.097, +0.097] [[−0.074, +0.027]]; 4/4			
Conventional − no seed	+0.000 [−0.097, +0.097] [[−0.082, +0.094]]; 5/5			
Conventional − random	−0.032 [−0.129, +0.048] [[−0.189, +0.023]]; 3/5			
Random − no seed	+0.032 [−0.065, +0.129] [[−0.034, +0.194]]; 6/4			

trails the first file-head repeat by -0.113 $[-0.210, -0.016]$, but that repeat is the one whose $+0.097$ over no seed the second repeat did not reproduce, and against the second repeat the difference is -0.016. The check therefore does not identify an interface that helps or hurts at this size; it shows that the between-repeat spread of one interface (six instances) exceeds the between-interface spread (two instances), so a confirmatory study must fix the interface on development tasks and budget repeats before it can attribute anything to the seed's content. Cost and steps were similar across interfaces (\$0.021–0.024 and 7.0–7.7 steps per trajectory).

Table 23: Seed-interface check on the 62 pilot instances, one trajectory per instance (the no-seed and paths+heads rows are the first pilot repeat). Exploratory; not preregistered.

Seed interface (one trajectory per instance, 62 instances)	Resolved	Rate	Mean cost (USD)	Mean steps
No seed	25	0.403	0.0162	7.7
Delphi top-5, paths only	26	0.419	0.0209	7.0
Delphi top-5, paths + file heads (pilot design)	31	0.500	0.0208	7.3
Delphi top-5, paths + best-matching chunk	24	0.387	0.0242	7.7
Paired differences in resolved rate: instance-cluster 95% CI [repository-cluster CI]; instances favouring a/b				
Paths only − no seed	+0.016 [−0.097, +0.129] [[−0.083, +0.143]]; 7/6			
Paths + heads − no seed	$+0.097^{*}$ [+0.016, +0.194] [[+0.019, +0.244]]; 7/1			
Paths + chunks − no seed	−0.016 [−0.097, +0.065] [[−0.087, +0.053]]; 3/4			
Paths + chunks − paths + heads	-0.113^{*} [−0.210, −0.016] [[−0.260, −0.026]]; 2/9			
Paths only − paths + heads	−0.081 [−0.194, +0.032] [[−0.205, −0.025]]; 4/9			

Precision target for a confirmatory study. The per-instance paired differences in the pooled pilot (means over two repeats) have standard deviations of 0.23 (Delphi − no seed), 0.26 (Delphi − random), and 0.30 (random − no seed); single-trajectory differences have 0.34–0.37. For a 95% half-width of 0.05 on a paired repair-rate difference these imply 83–138 instances with two trajectories each (173–213 with one), before repository clustering widens the interval. We set the confirmatory design at 200 instances × 2 trajectories per condition (expected half-width about 0.04),

drawn from the Verified instances no round has used, with the seed interface fixed on development tasks beforehand and the stopping rule fixed in advance.

Seed quality and cost. Agents in every condition retained ordinary shell tools and could, and did, locate files the seed had not named. Mean cost per trajectory: no seed \$0.019, random \$0.025, Delphi \$0.021, conventional \$0.023; mean steps 8.0, 8.3, 7.2, 7.5 respectively. Total spend for the 496 preregistered trajectories was \$10.97. All but two of them terminated by submission; the two exceptions (random condition, first repeat) hit the step or cost limit and count as unresolved.

L HOSTED ENGINE ACCOUNTING

Documentation engine. Accessed through its public REST API with a project credential. Two retrieval modes were recorded (oracle library ID and quality-guided); the quality-guided mode is the fair comparison because it does not receive the gold library. Its retrieved documentation drifted across runs (54% retrieved-set exact), which is the engine's behaviour and not an artifact of our client.

Synthesis engine, documentation surface. Accessed through its hosted API. The native answer mode was recorded once per case in round 3 with its cited sources; the raw-retrieval mode returned no sources during that round. Round 4 reconstructed the retrieved context from the recorded responses rather than issuing new calls, so the matched arm reflects the engine's round-3 retrieval exactly. Determinism probes (ten cases $\times$ ten runs) were issued in round 3.

Synthesis engine, repository surface. Indexing the ARB development split requires 68 exact (repository, base commit) snapshots in the provider's account. Round 2 indexed 32 pairs through a sharded upload; whole-snapshot ingestion of the large repositories entered a "syncing" state that never reached a terminal status. Round 3 issued a minimal one-shard probe that remained syncing across a 40-minute deadline and a three-hour watch. Round 4 issued a fresh one-shard probe under a newly provisioned credential (which the account inventory shows resolves to the same account, with the earlier probes still syncing) that again timed out after 40 minutes. No repository query was ever scored against the engine, so no number is reported in either direction. We do not attribute the stall to the engine's design; it may be an account, quota, or ingestion-path issue that a different account would not encounter.

Credentials. Hosted runs used project credentials supplied by the authors; none of the recorded artifacts contain them, and re-running the hosted arms requires the reader's own.

M COMPUTE AND COST

All round-4 experiments ran on one Apple-silicon workstation with a local PostgreSQL instance, using hosted APIs for embeddings and LLM stages. The matched ladder embedded 233.5M new tokens across its runs on the first three sets (3,807 embedding requests, roughly \$4.70 at list price), plus the fresh draw's snapshots; its persistent embedding cache holds the vectors, and its LLM stage cache holds the listwise and expansion replies, so any rung, including the factorial's hybrid+expansion cell, can be re-executed bitwise without hosted calls. Delphi's own stage caches for the frozen build are archived separately; the factorial's and branch ablation's no-reranker cells reused cached expansions where the query had been expanded before. Re-indexing the 18 independent snapshots and indexing the 60 fresh snapshots embedded their chunks once each. The executable pilot ran 496 preregistered trajectories (4 conditions $\times$ 2 repeats $\times$ 62 instances) with four concurrent containerized workers under `amd64` emulation for a total API spend of \$10.97, plus 124 exploratory trajectories for the seed-interface check; harness evaluation ran the official SWE-bench images once per trajectory. The documentation matched arm reused recorded hosted responses and issued 120 frozen-synthesis calls. Round-3 costs (the Delphi runs, lexical comparators, determinism probes, and hosted documentation calls) are recorded in the archive's journals.

N ARTIFACT RELEASE

Everything below is released in a public repository, `https://github.com/aayambansal/delphi-benchmark`, which also contains the source of this paper, except that the agent trajectories and per-instance harness logs are published as a Hugging Face dataset, `https://huggingface.co/datasets/aayambansall/delphi-benchmark-traces`, together with the seeded datasets, harness reports, and pilot analyses. The frozen Delphi source is revision `91d76c1` of the public Delphi repository, `https://github.com/synthetic-sciences/delphi`.

- *Per-case results* for every system on every set, including development, rejected, factorial, and branch-ablation configurations: one record per case with the query, gold paths, ranked paths, per-case metrics, latency, and the served configuration, together with per-run summaries.
- *Paired analyses*: every bootstrap interval and McNemar count reported in the paper, the factorial contrasts, the branch-ablation contrasts, the sequential analysis, the like-for-like determinism analysis, and the documentation synthesis outputs for every arm and repeat.
- *The exposure ledger* with every case ID and the evidence for its class, the draw manifests with their salts and preprocessing rules, and the index audits.
- *Agent trajectories* (the Hugging Face dataset above): the seeded datasets for every condition and seed interface, all 620 mini-SWE-agent trajectories with every model call and action, the official harness evaluation reports and logs per condition and repeat, the seed audit, and the pooled, per-repeat, and seed-interface analyses.
- *Split manifests and case files* for every set.
- *Implementation*: the evaluation runner for every static system (Delphi, the matched ladder including the factorial cell, the lexical comparators), the paired-analysis, factorial, branch-ablation, and sequential code, the exposure-ledger builder, the corpus restorer (bare clones at recorded commits), the draw sampler with its preprocessing rules, the pilot dataset builder with its three seed interfaces, the seed audit and pilot analyzers, the documentation-track runners with the synthesis stage and the synthesis-engine context reconstruction, and the scripts that regenerate every table and figure in this paper.
- *Serving*: the frozen Delphi source, the native launch configuration with its declared ablation switches (Appendix A), the pilot run configuration, and the configurations of the round-3 ablations.
- *Journals*: the state, decision, hypothesis, and workstream records; the preregistered pilot protocol; an artifact inventory with SHA-256 of every table's inputs; the round-2 protocol, source and statistics locks, capability audits of the hosted engines, and the measured-and-rejected record (Appendix H).

Repository corpora (58 repositories, 425 snapshots) are not redistributed; a provided script re-clones them from public GitHub at the recorded commits. Hosted-engine artifacts are the recorded responses; re-running those arms needs the reader's own credentials.