ripwire

The ripgrep of AI context.

A zero-runtime-dependency C++23 CLI that maps any codebase into a ranked, deterministic call graph for coding agents — and puts a tripwire on every claim it emits.

rip — the speed half

Structural answers in tens of milliseconds, from a call graph it built itself.

wire — the honesty half

Unprovable totals ship labelled as floors. A zero means none found — never none exists.

Apache-2.0 · single binary · hermetic build (proven with the network off) · 21 vendored tree-sitter grammars

// the problem

Your agent reads the whole library to answer one question

Blind grep, then whole-file reads. Most of what enters the context window is never used — it is paid for anyway, on every step of every task.

And bigger context is not better context: irrelevant text actively degrades the answer. The job is selection — a small, ranked, structural map beats a pile of open files.

0.108 s

median answer, warm index — round 4, all arms one machine one day, N = 60

−39.4%

token ceiling vs the un-routed baseline — LocBench cost ledger, N = 243

Speed caveat travels with the number: ripwire answers from a warm, pre-built index, and round 4 measured its index cost too — 0.31 s, tabulated beside the competitors' rather than omitted.

bench/headtohead/r4-2026-08-06/ · bench/locbench/ — every figure on this deck names its instrument

// one binary, the whole pipeline

Crawl → parse → resolve → rank → emit. Deterministic, end to end.

crawl	parse	resolve	rank	emit
the tree, no VCS needed	tree-sitter, 21 vendored grammars	references → call graph	Personalized PageRank	minified XML, one line

byte-identical	two runs, same bytes — a gate on every push, not a tendency; warm equals cold
zero runtime deps	CMake + a C++23 compiler; builds with the network off — vendored everything
the languages	Rust · C++ · ObjC/C++ · C · Metal · CUDA · Python · Go · Swift · TypeScript · JavaScript · Java · Ruby · PHP · Lua · Bash · C# · JSON · TOML · YAML · Markdown — 21 vendored grammars; markdown headings are real symbols
agent-native	an MCP server and 179 long flags behind one `--help` that is always the authority

// every feature, one screen

Seven families of questions it answers

understand cold	*What is this repo, and what matters in it?*	`--for` `--tree` `--lego` `--exemplar` `--recall` `--pack-task` `--token-budget`
navigate	*Who calls this? Safe to change or delete? Which tests?*	`--callers` `--callees` `--uses` `--impact` `--path` `--connect` `--affected` `--situ` `--test-gate` `--from-trace` `--pattern` `--safe-delete`
detail ladder	*Show me more — but only where it pays.*	`--detail` `--pack-signatures` `--outline` `--expand` `--compress`
quality & risk	*Where is the risk, and did I just add some?*	`--quality-panel` `--quality-delta` `--dmm` `--readability` `--ensemble` `--context-ratio` `--nonlocal-state` `--field-affinity` `--hotspots` `--lint` `--clones`
self-diagnosis	*Is my setup actually working?*	`--doctor` `--skipped`
security	*Is this agent skill file safe to install?*	`--scan-skill` `--scan-skills`
knobs & modes	*Shape, format, cache, budget.*	`--json` `--format` `--mcp`

`--help` is generated from the binary's own flag table — 179 long flags; docs/COMMANDS.md carries an entry for every one of them, 161 with a recorded invocation and its output

// reach for it at the moment, not the manual

The reflexes: which verb fires when

Landing cold on a task

`--for="<task>"` · `--pack-task`

ranked, budgeted context in one call

Stack trace in hand

`--from-trace`

frames mapped to symbols, innermost body included

“Is it safe to change X?”

`--impact` · `--uses`

the whole blast radius, not just 1-hop callers

Just edited a symbol

`--edit-check`

did the contract change, who breaks — ~ms, warm

About to write a helper

`--exemplar` · `--grep`

imitate the house pattern; catch the duplicate early

Landing several branches

`--merge-scout`

pairwise conflict sites + a landing order

Before you call it done

`--quality-delta` · `--test-gate`

only what you made worse; exactly which tests to run

Handing the area on

`--handoff` · `--note-add`

the verified-vs-heuristic packet the next agent needs

and --pr-context when you are reviewing someone else's diff — per-file blast radius, tests-to-run, hotspot flags

// new since 2026-08-15 — each figure re-derived on the binary that ships

What 2026-08-15 to 2026-08-23 added

`--pattern` *search by code SHAPE, not by node kinds*	foo($X, ...) across 13 grammar objects / 11 languages. On this repo VERIFY($X) finds 101 hits over 625 eligible files, each with its enclosing symbol. A pattern no served grammar resolves REFUSES (exit 1) instead of reporting hits=0.
`--safe-delete` *“can I delete this?” in one call*	callers + transitive radius + every read/write/import site + how much of that radius a test reaches. risk= names what was FOUND — never a go/no-go verdict.
`--impact importers=` *the second, weaker reach*	the files that include/require a file defining SYM, with their own cap pair — NEVER summed into reaches=, because files and symbols are different units. IngestResult: reaches=0, importers=70.
`compact --for bundles` *bodies were 52.7% of every conceptual byte*	the subtoken+body route now ships the ranked map plus one-hop edge context, disclosed as bundle=compact bodies=0. 15-query class B: 184,857 B → 95,256 B, −48.5%, all 11 decisive markers still present. --auto-bodies is a permanent opt-out.
`corroborated callers` *shared= on --pack-task rows*	a neighbour reached by four of the bundle's anchors sorts above one reached by a single anchor — omitted at 1, which is what every 1-hop row satisfies anyway.
`per-rule --lint roll-up` *a capped view stops hiding whole rules*	on this repository 14 of the 31 rules that fired contribute ZERO shown rows — 368 findings whose only evidence is their count=. A rule no corpus language registers carries applicable=0, so its zero is inertness, not a measurement.

And two languages: PHP and Lua, each shipped with its floor stated rather than implied — PHP's run-time dispatch ($fn(), call_user_func, __call) names its callee at run time and is a declared floor; a Lua corpus reports no inheritance edges, because metatable inheritance has no syntax to read.

// new since 2026-08-23 — each figure measured, dated, and re-derivable

What 2026-08-23 to 2026-08-30 added

`--slice-flow`

cross-statement data-flow slicing (ARISE, arXiv:2605.03117)

The paper's own slicer — reaching-definition edges, a seed plus a direction, a bounded BFS that stops at function boundaries — implemented and MEASURED on a registered fix-commit protocol: flow rows lift function-level added-line recall 0.163 → 0.198 at 25% of --expand's whole-body bytes (whose recall is 1.0 by construction). 7 commits / 38 instances, cpp only — a thin corpus, reported as thin.

`--slice, first numbers`

the 2026-08-28 registered contract, finally measured

Per-variable line-recall 0.726, hit-all rate 0.632 — a FLOOR under a noisy relevance oracle: every inspected miss was the protocol's word-regex matching identifiers inside comments and string literals, occurrences the slicer correctly refuses to call variable uses. No inspected instance showed a real occurrence the slice dropped.

`~1350× on --expand`

the secret-redaction sweep was quadratic in LINE length

One selector in a 2.1 MB / 768-line minified yarn bundle NEVER completed — killed at 1,343.9 s of user CPU with an empty output file; 196.7 s is the only clean lower bound — and now answers in 0.46 s warm. The 20 KB single-line fixture: 23.02 s → 0.017 s. A pure memoization, so an output no-op: 120/120 invocations byte-identical; gated on behaviour, the timings a ledger row and never a red-CI threshold.

sources: docs/EVALS.md §--slice-flow (registered protocol + per-instance ledger) · bench/PROFILE.md 2026-08-30 — no number travels without its caveat

// new since 2026-08-30 — the window this deck did not cover until now

What 2026-08-30 to 2026-09-06 added

.gitignore, honoured by default

named for ripgrep, whose defining default this is

The crawl walked ignored files; in a git work tree it now consults git's own rules and skips what the repository already declared uninteresting. This root, three checkouts under an ignored bench/external/: files= 8,674 → 1,522, cold 2.81 s → 0.52 s, warm 0.63 s → 0.10 s — and --no-ignore restores the previous walk exactly. Nothing drops silently: ignored_files= is exact, ignored_dirs= says the walk STOPPED there so those contents are UNKNOWN rather than zero, and both are ABSENT when the rules dropped nothing — so a tree with nothing ignored is byte-identical to what it produced before.

--html --color-by

the map, rendered — no server, no CDN

One self-contained HTML file: a force-directed call graph you click to recentre. --color-by=lang|community|cx|churn|tested sets the INITIAL colour only — the page embeds all five and keeps a live selector, so switching lens costs no second run. One five-stop blue-to-orange scale rather than the usual green-to-red, so reading it never depends on telling red from green.

--eval-retrieval, re-sampled

the sampler was measuring the corpus, not the ranker

For fourteen months the gold set was the first 150 doc-commented symbols in PATH order — and bench/ sorts before src/, so adding a docstring to a benchmark harness moved a published number without touching ranking code. One 60-symbol probe, one corpus, only its PATH varied: 0.105 and 0.356 MRR from spelling alone. Now exhaustive — population=3011 scored=3011 rule=exhaustive — and EVERY retrieval figure this project publishes was re-derived under it.

--plan-lanes

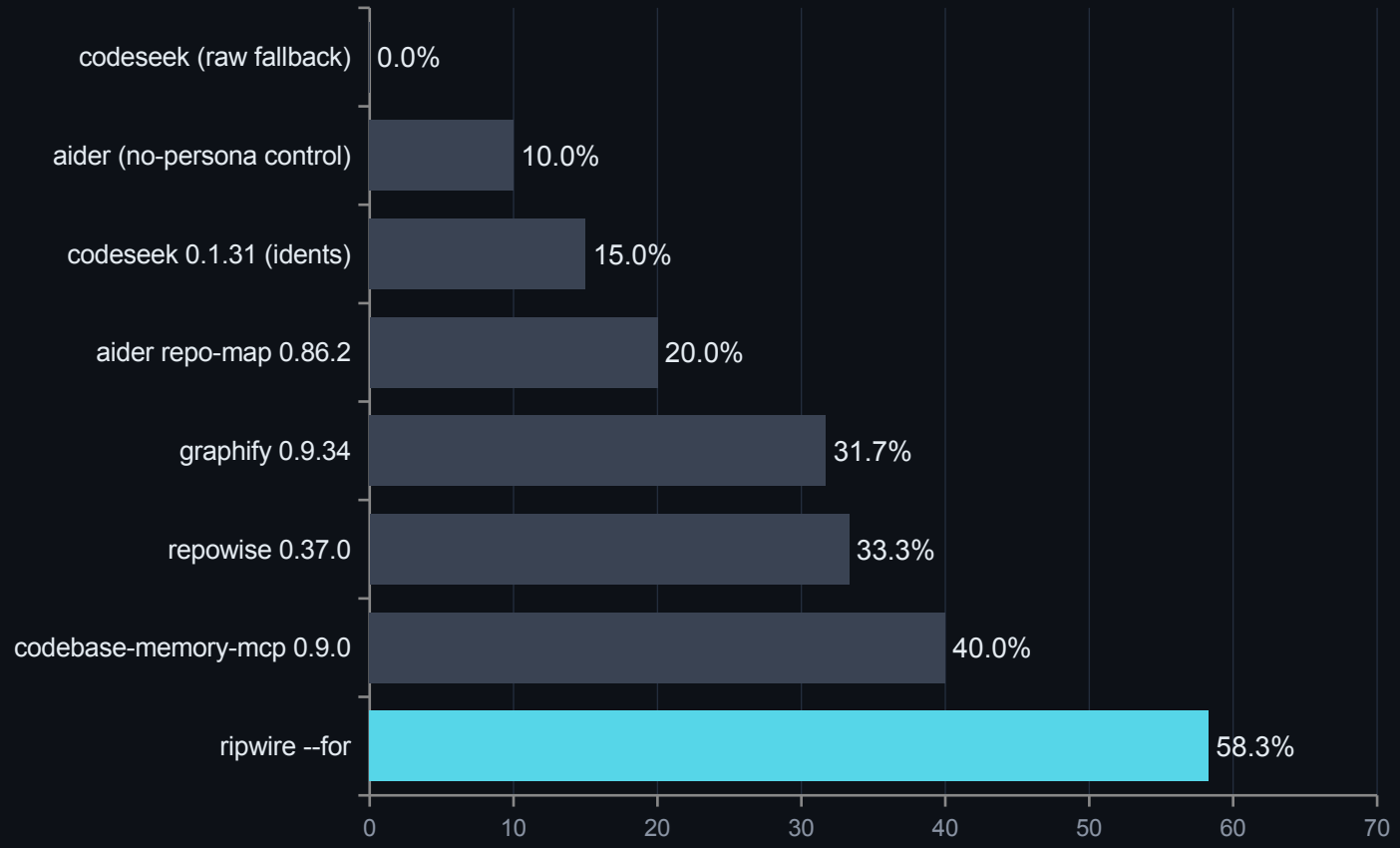
which lanes would collide, BEFORE a line is written

Where --merge-scout says "these branches already conflict", this says "these lanes WOULD conflict if assigned this way" — no ref to resolve, no re-ingest. Each lane now also carries an advisory execution profile (model + reasoning effort) that labels its own evidence basis="structural-only" and versions its policy separately from the schema, so a recommendation cannot quietly claim more than the structure it read.

CHANGELOG.md [Unreleased] carries the ignore-default measurement and its ledger in bench/PROFILE.md · the sampler negative is docs/EVALS.md §7, published as a counterexample against our own published numbers

// one harness, one machine, one day — N = 60 paired, zero exclusions

Head-to-head: every competitor, one table



1.46×

the margin over the best competitor — 58.3% against 40.0% strict file@10

And the index it was measured with
ripwire 0.31 s · codebase-memory-mcp 1.24 s · codeseek 3.37 s · graphify 7.82 s · repowise 34.0 s, whose worst single index was **352 s**. Cold, nothing to an answer: 0.213 s.

What re-running cost us, published: aider moved 18.3% → 20.0% on its own tie-break nondeterminism, and we printed the higher number. Paired, ripwire loses 2 instances to codebase-memory-mcp, 2 to repowise, and 1 each to graphify, aider and the aider control.

bench/headtohead/r4-2026-08-06/ — harness, per-instance JSONL and recipe committed · replaces the two non-comparable tables r1 and r2 needed

// scored against a third-party oracle, never against each other

Graded by a compiler: zero silent misses

Both tools were scored against a scip-clang compiler-grade index of this repository — 68 answers, 34 blind-authored queries × --uses and --callers.

0

true silent misses — pre-fix and post-fix alike. Every imperfect answer either carried a discriminating self-flag (amb=, defs>1, external="1") or was an oracle-scope artifact where ripwire was right and the compiler could not see the file.

What a compiler-grade index costs

	ripwire	Serena 1.6.2.dev0 (clangd 19.1.2)	
warm query	**49.8 ms**	3,760 ms	~70×
cold start	**208 ms**	8.7 s	~40×
index build	**none**	~8 min	—
errors	**0 / 101**	7 / 61	—

Read honestly in both directions. The LSP is more precise — site-level 0.929/0.941 against ripwire's 0.914/0.941 after the fix round, a gap of roughly 1.5 points with recall now equal. ripwire also pays ~7× the bytes per call, because its output carries self-describing legends. What it cannot do is answer a macro query at all — clangd's documentSymbol has no macro entries, which is 3 of its 7 errors — or start in less than eight minutes.

The number that moved the wrong way, published with the rest: over-hedging rose 18.6% → 22.6% across the fix round. Closing loss buckets added self-flags faster than it removed imperfect answers. That direction is deliberate — calibrated to warn too often rather than too rarely — but it is a cost, and it is on the slide.

bench/headtohead/r9-2026-08-09/RESULTS.md — C++ only, 34 scorable queries, one corpus (this repository)

// 48 paired questions, zero exclusions, both arms exit 0 on all 48

Against a graph-database context server: 27–7–14

class	n	ripwire	GitNexus	tie	bytes ÷ it
symbol lookup	12	6	2	4	1.35×
conceptual search	15	8	1	6	1.23×
callers / blast radius	12	7	2	3	0.39×
task orientation	9	6	2	1	0.06×
all 48	48	**27**	**7**	14	0.159×

Its seven wins are named one by one, and its first-pass numbers do not exist. Four are cost losses on an answer ripwire gets right — a by-name lookup it serves in ~1 KB where ripwire spends 3× to also hand back the body. Three are ranking misses on webpack, and two of those share one unfixed mechanism: a symbol whose NAME matches the query beats the implementation that carries it in its body. Under the improve-first rule the round's FIRST pass was never published — its product was a loss list, and what is on this slide is the state after that list was worked. One of those fixes met its pre-registered band and was reverted anyway for failing a separate standing requirement; **the loss it would have fixed is still counted against us above.**

The cost of the whole sweep

309,348 B against 1,945,231 B. Warm median 197 ms against 1,082 ms. Index 0.25–0.45 s / 6.6–16.5 MB against 23–52 s / 391–623 MB — written into the repository, not $TMPDIR.

What it does better: compact one-hop context at 0.7–1.0 KB, a depth-LABELLED blast radius, per-stage timing on every query, and two-command onboarding into eight agents.

Failure modes, both directions

needed a 2nd call	0	8/48
empty radius, real callers	0	1
malformed through a pipe	0	7/48

The published numbers use the file-redirected form — the configuration favourable to it.

docs/EVALS.md §2 — GitNexus 1.6.9 (npm latest at 2026-08-22) vs ripwire at 7eb638e; django, webpack and this repository, all tree-hash verified clean · ratios are on class TOTALS, the medians form is printed there too because the two disagree

// the rule: numbers are held back until the loss list is worked

Every head-to-head round, and the ones that went against us

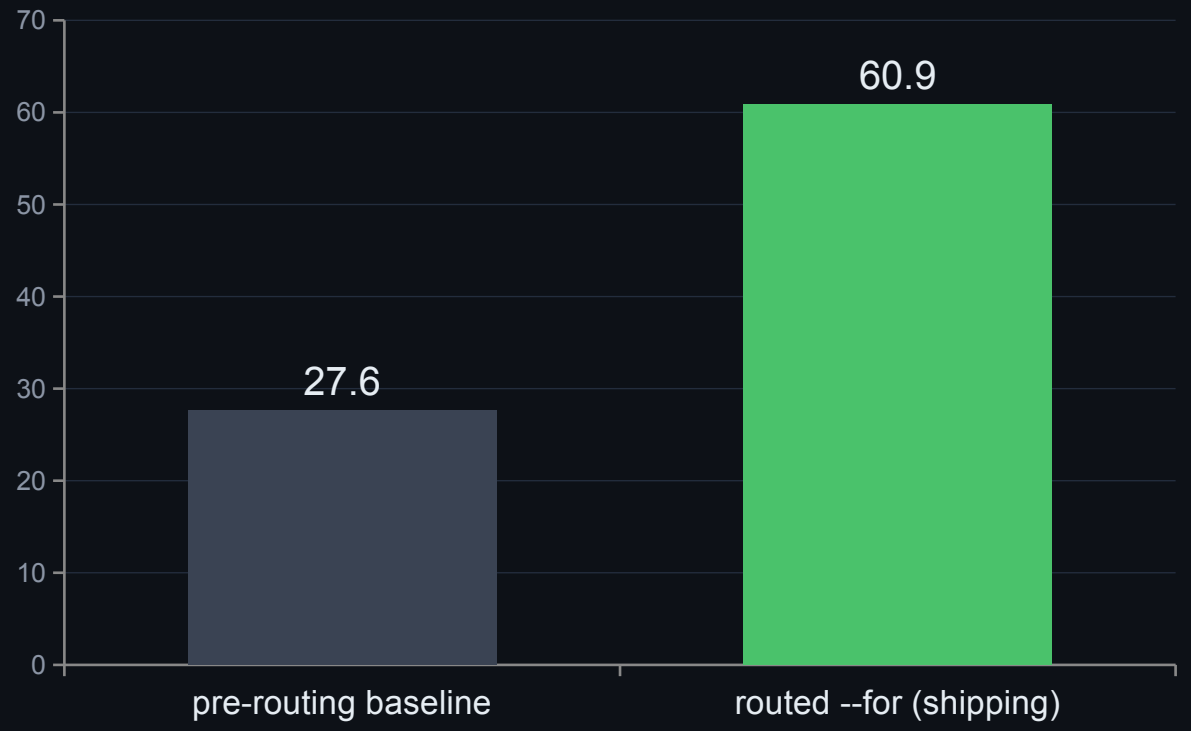
r1	2026-07-13/14	Aider repo-map · codebase-memory-mcp · graphify	*superseded by r4*
r2	2026-08-03	repowise · codeseek	*superseded by r4*
r3	2026-08-03	headroom — a compression layer, so a different instrument	passed code through byte-identical
r4	2026-08-06/08	all five competitors, one harness, one day	the table on the previous slide
r6	2026-08-04/05	agent-in-the-loop: the Codex CLI pilot	localization parity; +80% tokens, classified
r7	2026-08-08	CodeGraph, loss-first — we looked for our losses	fix REJECTED by its own preregistered band
r8	2026-08-08	aider again, at equal token budget	loss questions traced to one root cause
r9	2026-08-09	Serena / scip-clang oracle	0 silent misses; the fix list it produced
r10	2026-08-22	GitNexus 1.6.9 — a graph-database context server	27-7-14 — its own slide, two back

r7 in full, because it is the point: the fix was built, gated green, measured — predicted 12/22 against a pre-registered band of 10–14, measured 5/22 — and reverted rather than tuned. Published as a negative in docs/EVALS.md §7.

bench/headtohead/ (r1-r4, r9) · bench/r7/ · bench/r8/ — harnesses committed even for rounds that rejected their own fix · r5 left no head-to-head record in this tree, so it is not listed · r10's method and pins are published in docs/EVALS.md §2, its pre-fix numbers nowhere

// held-out, repo-disjoint, frozen dataset

Routing the ranker: +33 points, measured properly



+33.33pp — paired delta, 243 held-out instances in 78 repository clusters

+25.00pp — clustered-bootstrap 95% lower bound — the honest floor

−39.4% — token ceiling for that gain

+3.4% — warm latency for that gain

bench/locbench/ — czlll/Loc-Bench_V1, 560 rows, JSON SHA-256 pinned in dataset.lock; metric code = LocAgent's strict Acc@k

// compiled, not interpreted — and measured on public corpora

Fast enough to call on reflex

0.213 s

cold — nothing to an answer: parse, rank, reply, no cache (r4)

0.108 s

warm query, median — against aider's 2.920 s (r4)

0.31 s

index build — against repowise's 34.0 s, worst case 352 s (r4)

49.8 ms

warm query on this repo — against a clangd LSP's 3,760 ms (r9)

Where the second goes

Cold profiles are tree-sitter parse plus tag-query capture — not graph math. Resolve and PageRank together are a small fraction of the run, and the ranking was never the bottleneck.

Which is why the wins came from removing work, not micro-optimising it: demand-driven side captures, five whole-AST passes fused into one pre-order walk, a calibrated per-worker parse reserve.

Two opt-in faster builds

LTO is on by default. PGO is a script away and buys 14–25% cold.

Both are opt-in performance, not opt-in correctness: **the output stays byte-identical across build flavours** — the determinism gate runs on both, and CI builds a plain flavour too, because NDEBUG once compiled a whole class of degrade-path checks out of existence.

every card above comes from a committed harness — bench/headtohead/r4-2026-08-06/ and r9-2026-08-09/ — not from a private corpus

// how it got there — the pre-release performance sprint

Work removed before work optimized

file ingest

FILE* whole-file reads; CSV counting became a byte scan — iostream out of the hot path

capture gating

value-use capture runs only for the verbs that actually need it, not on every run

AST pass fusion

five whole-AST side-capture passes share ONE pre-order walk instead of five

parallel parse

query compile overlaps parse scheduling; files schedule by size — ids stay deterministic

radix + S+tree

numeric keys for edges and scores; a bounded-scratch hot map whose fixed cap reports its own overflow

no std::map

unordered_dense, reserve() before ingest — no std::unordered_map on any hot path

Why this slide carries no numbers. The sprint's own headline pair was measured in June 2026 on a large private C++ corpus that is not publicly reproducible, and the deck does not quote figures a reader cannot re-derive — the previous slide's numbers all come from committed harnesses. What survives the omission is the ***shape***: the graph math was never the bottleneck, and every win came from doing less work rather than doing the same work faster. Same map contract throughout — byte-identical output before and after each of these changes.

the mechanisms are in bench/PROFILE.md and the commit history; the private-corpus figures they produced are deliberately not quoted

// docs/EVALS.md §5, pinned on this repository 2026-08-08 (the --for row re-pinned 2026-08-22)

Ten everyday moments, and what each one costs

Ask it	Command	ripwire	naive read	savings
Orient me in this repo	`ripwire .`	**~5.6K**	~20K–25K	**3.6×–4.5×**
Where is X handled?	`--for="…"`	**~2.1K**	~4.9K–20K	**2.3×–9.3×**
What do I already know?	`--recall="…"`	**~15K**	~445K	**29.2×**
Set me up for this task	`--pack-task="…"`	**~2.1K**	~16K–80K	**7.7×–37.7×**
Show me this one function	`--expand=SYM --top-k=0`	**~260–16.5K**	~43K–174K	**2.6×–670×**
Who calls this function?	`--callers=SYM`	**~580**	~40K–52K	**69.2×–89.1×**
Is it safe to change this?	`--impact=SYM + --uses=SYM`	**~1.3K**	~18K	**14.4×**
I have a stack trace	`--from-trace=FILE`	**~1.4K**	~124K–298K	**86.9×–208.6×**
I changed these files — tests? radius?	`--situ`	**~410**	~3K–132K	**7.3×–324.2×**
Review this PR/diff	`--pr-context=REF`	**~1.9K**	~4.8K–51K	**2.6×–27.5×**

Figures are ~tokens (≈ bytes/4). Every row is scored same-correct-answer-or-it-doesn't-count — both sides were checked, not assumed — and every ratio is a **range**: the cheap end and the honest end of what an agent would actually read, never the single most flattering number. Absolute counts belong to the corpus AT THE PIN — .gitignore became a crawl default on 2026-09-03, which moved what a re-run reads; the commands, not the constants, are what reproduces.

// the cost lever

Smaller answers that are also better answers

74.7%

fewer element bytes at top-50 — the --pack-signatures ladder, root-neutralised, re-derived on this repo every run

This figure survived its own audit: three corrections deep, re-derived root-neutrally after the original was shown to depend on how the corpus path was spelled — three spellings of one root read 18.6 points apart before that subtraction. top-50 is the quotable number because the payload is top-50 whatever --top-k says.

Ask for a symbol by NAME and the router uses the name-exact ranker:

```
MRR  0.745 → 0.960
recall@1  61.1% → 91.3%
```

name-shaped queries on src/, all 3,011 doc-commented symbols scored, re-pinned 2026-09-05

And the pollution metric — fixture and generated paths contaminating results — is driven to 0.0% on the ranking lane while recall went UP, not down.

docs/EVALS.md §4-§5 — showcasecapturecheck re-derives the byte-reduction triple; `ripwire src --eval-retrieval` re-derives the ranking bars, scoring every doc-commented symbol rather than a sample

// the half no other tool ships

The tripwire: honesty as an output contract

counts_floor="1"

call edges come from source text — dynamic dispatch adds no edge, so every count that cannot claim completeness is labelled a floor, in the output itself

a zero is a measurement

zero means none found, never none exists — and absent ≠ 0

truncation is disclosed

every cap, page seam and budget cut is named in the header before you read a row

refusals teach

a refused query names the flag, the problem, and a working example — never a bare error

```
$ ripwire . --callers=rankGraphTeleport

<callers of="rankGraphTeleport"
  defs="1" count="6"
counts_floor="1">
<s t="fn" n="runEval" .../>
<s t="fn" n="rankGraph" .../>
…
</callers>
```

The map grades itself before it answers. This repository's own src/: files=153 symbols=5122 edges=14182 ambiguous=5982 unresolved=1598.

docs/EVALS.md §8 lists the numbers this project refuses to publish, each with its reason

// the honesty marks stop being a promise and become a measurement

Four levers accepted, one rejected on purpose

macro edges	`kParserVer 48`	function-like #define invocations land role="macro" edges on disclosed-degraded symbols
fn-pointer bindings	`kParserVer 49`	calls through unambiguous local function-pointer bindings now resolve
using-declarations	`kParserVer 51`	re-exports emit role="import" references — r9 loss bucket 1
shadow suppression	`kParserVer 52`	a local variable shadowing a function name stops stealing its use-sites — r9 loss bucket 2
RTA-lite	`reverted`	instantiation-filtered CHA cone: refuted TWICE — the instantiated set crossed namespaces, and narrowed calls LOST their amb= mark

The two r9 levers, against the compiler oracle: site precision 0.9046 → 0.9136, recall 0.9285 → 0.9412.

The fn-pointer lever raised unresolved= by 1,039. Disclosure, not regression: call sites the resolver now RECOGNIZES but refuses to resolve are counted instead of silently ignored.

`bench/resolverround/RESULTS.md — per-lever attribution; RTA-lite's rejection is written up in docs/EVALS.md §7`

// how it stays true

Proven, not promised

548 gate scripts
the suite runs on every push — plus determinism, cache-transparency and golden contracts; the gate count itself is gated against the runner's own loop

byte-identical, always
two runs over the same tree produce the same bytes; warm equals cold. Enforced in CI, twice — Release AND a plain flavour, because NDEBUG once blinded a whole class of checks

differential refactoring
a refactor must prove it changed nothing observable: two binaries, hundreds of argv vectors, stdout + stderr + exit codes byte-identical

held-out labels, authored blind
eval labels were written by reading source before the ranker ever ran on them — so the eval is allowed to say the ranker is wrong. It has.

sanitizer wall
ASan + UBSan + integer + float-cast, no-recover; TSan separately; leak suppression is one pinned file

its own output is audited
the showcase capture is regenerated and adversarially re-read as a claims corpus — the methodology ships in docs/METHODOLOGY.md

// the reason to believe any number on this deck

Claims you can trust, because we publish what failed

548

gate scripts named by test/regression.sh — and the COUNT itself is gated against the runner's own loop, so it cannot go stale quietly

8

registered NEGATIVES — changes built, gated green, measured against a band written before the code, and reverted rather than tuned

0

numbers on this deck without a committed instrument behind them — §8 lists the ones this project refuses to publish at all

the fresh-clone arm	our own battery only ever ran inside a configured, long-lived worktree — one value of every AMBIENT input. It clones HEAD into a genuinely unconfigured checkout and re-runs the gates whose past failures were exactly that: a developer's git config, a fixture's inherited branch name, the checkout's own path length.
the merge-conservation gate	consistency is not conservation. A conflict resolution can drop a gate from the runner's loop while regenerating the pinned count from that SAME damaged list — both sides then agree at a wrong number. On a merge, the loop must equal the UNION of both parents', minus explicit tombstones.
the eight, by name	r7's prose-strip fix (predicted 12/22, band 10–14, measured 5/22) · RTA-lite, refuted twice · file-level evidence pooling, +0.00pp held-out · un-guarded query stemming · its IDF-guarded retry, VOIDED by its own verifier · query-term density weighting, both displacement guards tripped · --anchor and --cochange-boost, no confirmed lift · a name-coverage floor that MET its pre-registered band and was reverted anyway for a standing requirement.

A gate that cannot go red is a gate that proves nothing — which is why every arm on this deck ships with the mutation that makes it fail.

// own the losses — they are in the README too

Where it loses, published with the wins

The public C++ number is lower than the private one it replaced	SFML: strict file@10 28.7%, any@10 41.7%, first-hit MRR 0.21. Until 2026-08-07 this bullet compared against a ~89% any@10 private corpus that is no longer reproducible. That comparison is retired; the public number is the baseline going forward.
--grep costs more than it saves	+19.7% tokens / −11.2% — published next to the flag it indicts, and carrying the same private-corpus caveat as every figure in that source
PageRank is the wrong co-change ranker	3.8% recall@5 against 40.3% for plain lexical, and fusing the two made it worse — relatedness is lexical, importance is structural
Strict multi-file localization stays hard	single-file gold 73.4% vs multi-file 18.2% (held-out); the same cliff on every corpus — open headroom, said plainly

A tool that publishes its counterexamples is a tool whose wins you can believe.

// the single command for "is this code actually rotten?"

Six families of evidence, counted — never blended

structural	shape: complexity, size, nesting, params, readability rank	`--metrics`
lexical	identifier text: the naming rules	`--lint --naming-consistency`
confusion	syntax: the atoms-of-confusion rules	`--lint`
historical	git change frequency, measured PER FILE	`--hotspots`
colocation	how much you must READ outside this file	`--context-ratio`
state	this function's OWN BODY touching non-local mutable state	`--nonlocal-state`

Ranked by the COUNT of distinct families that fire — never a weighted composite, because a composite lets one loud family impersonate six.

4 of 6 is what the strict preset counts — the verb says so itself, families="6" enabled_n="4". Two families did not clear the stability ladder.

`--quality-panel[=strict|default|lenient]` — and `--quality-delta` for the narrower question: what did MY change make worse?

// a new lens does not ship because it sounds plausible

The calibration that disqualified a family

+0.168

the LARGEST cross-family correlation in the whole 6×6 matrix — widening the panel from four families to six did not raise it

27,889

eligible functions, five independent trees — two reported separately and never pooled, because one of them is this repository

4 of 6

families the strict preset actually counts — TWO were measured and held back from gating

The uncomfortable result, kept. The stability pass ran each family across a ladder of commits to ask whether it is steady enough to gate on. **colocation came out worse than historical on one ladder**, so neither gates: strict counts four of the six — a lens its own author wanted, measured, and demoted. The ladder was run precisely because that answer was possible; assuming it would pass is how a plausible axis becomes a permanent one.

--field-affinity was NOT made a family either — an exclusion by unit, not a failed measurement: it ranks struct FIELDS by co-access against declared layout, and the panel's unit is a function. Stated in the calibration rather than left for someone to notice.

docs/EVALS.md §9.9 — same harness and criteria as the original four families, run BEFORE the two new ones shipped enabled

// built for the agent's seat

One command wires it into your agent

```
$ ripwire wrap claude
```

claude · cursor · codex · windsurf · gemini · aider — or --all to detect every one you have installed

31 MCP verbs

16 read verbs mirroring the CLI, 12 flagship reflexes (impact, uses, edit_check, from_trace, connect …), 3 span-addressed edit verbs with a safety contract

lazy-body handles

read verbs return signatures and a stable handle; the agent fetches a body only when it decides it needs one — names by default, bytes on request

18 agent skills

moment-matched workflows (orient, navigate, change-check, quality-bar …) — wrap prints the recipe, skills/install.sh installs them

11 orchestrator loops

copy-paste prompts in prompts/: run the same audit, eval and head-to-head machinery that built this tool, on your own repository

wrap does not just print JSON: it probes what that agent already has, emits the config in that agent's own shape, and includes a use-when blurb so the agent knows which verb fires at which moment — not just that a server exists.

the MCP server exposes the same deterministic engine — one index, shared with the CLI, staleness-checked

// it does not only read

Span-addressed edits: the map writes back

Three verbs, one resolved span

```
--replace-symbol-body=TARGET
--insert-before-symbol=TARGET
--insert-after-symbol=TARGET
  --edit-payload=FILE|-
```

TARGET is a name, an @FILE:LINE seed — paste the location out of a diff hunk or a compiler error and it binds the innermost enclosing definition — or a freshness-pinned handle from --grep --handles. No line numbers to keep in your head, and no whole-file rewrite.

The refusals are the feature

Freshness hash, lock, a re-check immediately before the rename, fsync, mode preserved, atomic rename. A target that resolves to more than one definition refuses. An empty payload refuses rather than being read as a deletion. Every refusal leaves the file byte-identical, and every success prints a JSON receipt whose span is the POST-EDIT byte range — where the payload now sits, not where it was aimed.

Preview the bytes before they exist

--edit-check=SYM --edit-payload=- --dry-run splices the payload over the span IN MEMORY, re-parses that one file, rebuilds the call graph over the re-derived tree, and answers the contract question about bytes nobody has written: did the signature change, and which callers does it break? Preview, then apply — no Read of the file in between.

Several edits, one transaction

--edit-plan=FILE is a versioned JSON multi-edit plan and its mode is EXPLICIT: exactly one of --dry-run or --apply, and neither is not a default. Every target, payload and span is preflighted before any write; overlapping edits refuse; payload paths are confined to the plan's own directory, so an absolute path or a '..' escape refuses and names what it resolved to.

Same engine on both seats: the three MCP edit verbs are these three flags, and the navigation the deck has no room for is the same story — --graph-query, --whereis, --stray-content, --at, --arch, --owners, --export, --help-task all sit in docs/COMMANDS.md with a recorded invocation and its output.

every claim on this slide is stated by `ripwire --help` and captured in docs/COMMANDS.md — test/deckcheck.sh proves each flag named here exists in the shipped binary

// standing on giants

The research inside — classic and current

42 repositories + 67 papers folded · and a labelled survey of 237 tools that contributed nothing, which says so — every row with the lesson taken and where it lives: docs/LINEAGE.md

1976	**Cyclomatic complexity — McCabe**	the metric behind --hotspots
1994	**Okapi BM25 — Robertson · Spärck Jones**	the lexical ranker (twice: subtoken+body, whole-name)
1998	**Personalized PageRank — Page · Brin**	the headline importance signal
2008	**Louvain modularity — Blondel et al.**	the module / community map
2009	**Reciprocal Rank Fusion — Cormack et al.**	deterministic signal fusion (--rank-by=rrf)
2011	**Readability model — Posnett · Hindle · Devanbu**	the lexical family of the quality panel
2019	**Delta Maintainability Model — di Biase et al.**	--dmm: one comparable number per change

TDAD · arXiv 2603.17973

a static test-map cut agent-caused regressions 6.08% → 1.82%; prose TDD instructions alone made agents WORSE → the shape of --test-gate

What-to-Retrieve · arXiv 2503.20589

retrieval selection beats retrieval volume for coding agents → the selection-over-dumping thesis

LocAgent / Loc-Bench (2025)

the localization metric (strict Acc@k) and the frozen 560-instance dataset every accuracy number here is scored on

scip-clang · Serena / clangd

the compiler-grade oracle round 9 graded both tools against — an outside referee, not a rival

tree-sitter

the incremental GLR parsing substrate — 21 grammars vendored, one parser per thread

counts gated in-repo (readmedriftcheck arm E derives them from LINEAGE.md's own tables) — the lineage is the design rationale, not decoration

// do not take any of it on trust

Every claim, and the command that re-derives it

179 long flags · 29 slides	`bash test/deckclaimcheck.sh`
every --flag named here exists	`bash test/deckcheck.sh`
74.7% fewer element bytes	`bash test/showcasecapturecheck.sh`
548 gate scripts	`bash test/manifestcheck.sh`
42 repos · 67 papers · 237 surveyed	`bash test/readmedriftcheck.sh`
the ten moments, any row	`ripwire . --callers=SYM | wc -c`
the head-to-head table	`bench/headtohead/r4-2026-08-06/`
the oracle round	`bench/headtohead/r9-2026-08-09/RESULTS.md`

The deck is generated, and the generator is gated. present/deck5_ripwire_build.js is scanned for fabricated flags and its slide count is derived from its own addSlide() calls — a deck that drifts from the binary fails the suite.

docs/EVALS.md is the source of truth; §7 is the counterexamples, §8 the claims this project refuses to make

// sixty seconds to the first ranked map

Quickstart

```
git clone <repo> && cd ripwire
cmake -S . -B build && cmake --build build -j     # hermetic: works with the network off
./build/ripwire --help
```

`ripwire .`	the ranked map — start here on any unfamiliar repo
`ripwire . --for="cache invalidation"`	the task lens: what to touch, ranked
`ripwire . --callers=someFunction`	who calls it — with the honesty marker attached
`ripwire . --quality-panel`	is this code actually rotten? six families, one shortlist
`ripwire . --test-gate`	before you commit: exactly which tests must run
`ripwire wrap claude`	wire it into your agent, in that agent's own config shape

ripgrep for the speed. **tripwire** for the honesty. Apache-2.0.