CROSS-REPOSITORY GUIDANCE FOR CODING AGENTS: A SINGLE-TRIAL REPLICATION ADDENDUM

A PREPRINT

Asa Shepard
Williams College
as66@williams.edu

May 31, 2026

ABSTRACT

A companion paper (Shepard, 2026) showed that probe-and-refine tuning on a target repository produces `AGENTS.md` guidance that significantly outperforms an unguided agent on SWE-bench Verified. That procedure assumes access to the target repository. Here we ask a narrower question: can comparable guidance be assembled *without* ever running the tuning loop on the target, by mining edits from a pool of structurally similar external repositories? We mine probe-and-refine artifacts from 50 external Python repositories (none in the SWE-bench 12), filter the resulting 760 edits down to a 496-edit "generic" pool, and for each of the 12 target repos compose a $\leq$3000-character guidance file from the edits authored against the most structurally similar source repos. On SWE-bench Verified (N=500, single trial, Qwen3.5-35B-A3B at 200 steps), this cross-repo guidance resolves 45.4 % of instances vs. 39.6 % for no context (McNemar $p=6.4\times10^{-4}$) and 48.0 % for per-target probe-refined guidance (McNemar $p=0.118$, n.s.). We do not claim cross-repo guidance is as good as probe-refined: our single trial is underpowered for that comparison, and the 13-instance gap is in the direction we would expect. The result that survives is more modest: cross-repo guidance reliably outperforms no context, and the probe-refined condition is not far enough ahead at N=500 to call the difference at this trial count. We document the pipeline so it can be re-run, and flag the additional trials that would be needed to give the indistinguishability claim real weight.

1 Introduction

The companion paper (Shepard, 2026) establishes that iteratively refined repository guidance, produced by probe-and-refine tuning on a target repository, raises a Qwen3.5-35B-A3B coding agent's resolve rate on SWE-bench Verified from $\sim$25 % (no context) to $\sim$33 % (probe-refined). A natural follow-up question is whether the gain depends on tuning against the target repository at all. If most of what probe-and-refine teaches the agent is the *kind* of operational discipline an experienced contributor would apply (reproduce-first workflows, evidence-before-patch quality gates, navigation-by-trace rather than by symbol search), then guidance distilled from a different but structurally similar codebase should also help. That is the experiment reported here.

Probe-and-refine tuning, described in full in the companion paper, iteratively refines a per-repository guidance file through synthetic bug-fix probes: at each of 3–5 iterations, the procedure generates a batch of synthetic tasks, attempts and judges each, and mechanically applies the resulting diagnostic edits to the artifact. The output is a $\leq$3000-character `AGENTS.md` file containing operational rules (reproduce-first debugging workflows, subsystem-specific navigation instructions, and output quality gates) that the agent loads at the start of every trajectory. Unlike per-task context, this artifact is produced once per repository and reused across all instances.

We took every edit produced by probe-and-refine across 50 external Python repositories, kept only the edits judged transferable (no file paths, no project-specific names), and for each of the 12 SWE-bench target repositories assembled a guidance file from the edits authored against the most structurally similar source repos. The target repository is never seen by the tuning loop. We then evaluated three conditions on SWE-bench Verified at 200 steps: `no_context`, `cross_repo` (this work), and `probe_refined` (the companion paper's per-target condition, whose tuning loop ran on

Figure 1: The cross-repo guidance pipeline. The probe-and-refine procedure from the companion paper is applied to 50 external repositories the agent will never be evaluated on; edits are filtered for transferability, ranked by source-target structural similarity, and assembled into a target-specific `AGENTS.md` that respects the same 3000-character budget used in the companion paper.

Table 1: Edit pool after Phase 2. Of 760 edits extracted across all probe-and-refine iterations on 50 external repos, 65 % are kept as generic (transferable) and 35 % are dropped as specific to their source repository.

Quantity	Count
External repos mined	50
Total edits extracted (deduplicated)	760
classified SPECIFIC (regex)	113
classified by LLM judge	647
GENERIC edits (kept)	496
SPECIFIC edits (dropped)	264

the target repository). A single trial is sufficient to answer the qualitative question (does cross-repo transfer work at all?) but cannot resolve the quantitative question of whether cross-repo and probe-refined guidance perform identically. We report what the trial does and does not establish, and identify the replication that would be required to strengthen the second claim.

2 Methodology

The pipeline (Figure 1) has four phases: mining, filtering, similarity matching, and assembly. Phases 1–4 run once per release of the external-repo pool; the resulting 12 `AGENTS.md` files are then reused for every issue in their target repository.

1. Mine. We selected 50 active Python repositories outside the SWE-bench 12, stratified across eight strata (web framework, data modeling, scientific computing, CLI/devtools, testing, async/networking, infrastructure, miscellaneous libraries), with target counts roughly proportional to the SWE-bench repository mix. For each external repo we ran the probe-and-refine procedure from the companion paper unchanged: 5 iterations, 10 synthetic probes per iteration, attempt-and-judge against expected behaviors, mechanically applied edits, $\leq$3000-character budget. Every iteration emits an `iteration_N_edits_selected.json` listing the rules that were merged into the guidance at that step; we keep these as our raw material. The mining loop is the only LLM-heavy phase, and it is amortized: a repo mined for one experiment can supply edits to any future experiment.

2. Filter. Across the 50 repos, the iteration artifacts contain 760 distinct edits after deduplication. We classify each edit as GENERIC (transferable advice that does not reference a specific codebase) or SPECIFIC (mentions a file path, a project-specific symbol, or a project-specific run command). A regex pre-pass catches the obvious cases (file paths matching `\w+/\w+\.py`, the source repo's name appearing in the edit body, project-specific test commands like `tox -e ...`); a second pass uses Qwen3.5-35B-A3B as a single-shot judge on the ambiguous remainder. The pass yields 496 generic edits and 264 specific edits (Table 1). The specific pool is dropped: using a sympy-named edit to guide a django agent would be actively misleading.

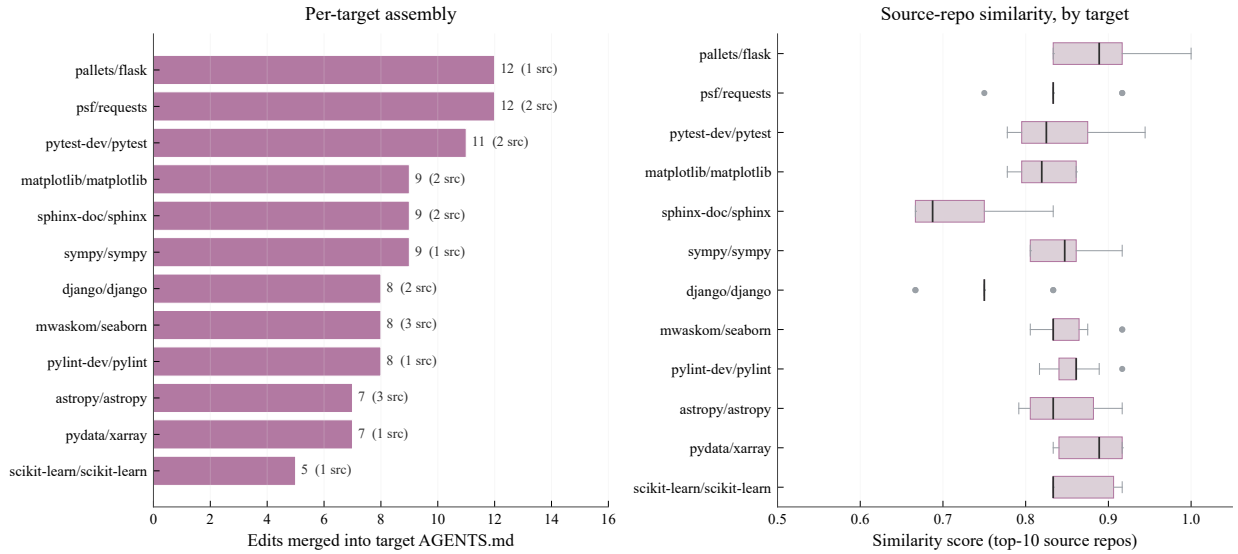

Figure 2: Per-target assembly outcomes. *Left:* number of edits merged into each target `AGENTS.md`, with the number of distinct source repositories those edits came from in parentheses. *Right:* similarity-score distribution over the top-10 most similar source repositories selected for each target. Targets cluster around 0.83–0.89; `sphinx-doc/sphinx` is the worst-matched target, with a top-10 mean of 0.72.

3. Similarity. Each of the 50 source repos and 12 target repos is summarized as a fixed-length vector of eight categorical features extracted from the same tree-sitter probe used by the companion paper's `static_kb` layer: test framework (pytest / tox / unittest / nose / other), linter overlap (Jaccard over {flake8, black, ruff, isort, mypy, pylint, pyright}), repository size bucket, docstring style, import style, type-hint coverage bucket, hub-density bucket, and presence of a top-level test directory. The features carry weights $(3, 2, 2, 1, 1, 1, 1, 1)$, summing to 12; the similarity score is the matched-feature weight divided by 12, so it lies in $[0, 1]$. The choice to use eight categorical features at all (rather than embeddings, AST kernels, or a learned scoring function) is the contribution we want to evaluate; the specific weights $(3, 2, 2, 1, \ldots)$ reflect a plausible ordering (test framework matters more than docstring style) but were not tuned, and we make no claim that this particular weighting is optimal. The hypothesis the experiment tests is the weaker one: *some* simple, hand-readable similarity score over surface-structural features is enough to drive useful edit transfer. We discuss feature ablation as a natural follow-up in Section 5. For each target we keep the top-10 most similar source repos. The resulting per-target similarity distributions (Figure 2, right) range from `sphinx` (top-10 mean 0.72) to `flask` (0.89, with two perfect 1.0 matches against `pallets/click` and `pallets/jinja`).

4. Assemble. For each target, we start from the same v0 KB scaffold the companion paper uses (tree-sitter–derived structural lines: hubs, entry points, import chains, test directories, linter configs). We then walk the 496-edit generic pool ranked by source-repo similarity, deduplicate by normalized content, route each edit to the appropriate `AGENTS.md` section, and insert until the 3000-character budget is reached. The result is a guidance file that has the same structural skeleton as the companion paper's `static_kb`, populated with operational rules the model originally wrote against a different but similar codebase. The 12 assembled files contain 5–12 edits drawn from 1–3 source repos each (Figure 2, left), with total length 2893–2994 characters, right at the 3000-character ceiling.

5. Evaluate. The downstream coding agent, scaffold, model, and benchmark match the companion paper, with one known difference: the present experiment was run two months later on an updated point release of sglang (used to host Qwen3.5-35B-A3B), which was not pinned across the two campaigns. Absolute resolve rates therefore shift relative to the companion paper's numbers (no-context here is 39.6 % vs. 25.5 % there), but within-trial paired contrasts, per-repo orderings, and the coverage-not-precision mechanism (Section 3) all replicate. The comparable signal across the two papers is the relative effect of guidance, not the absolute resolve rate. The benchmark is SWE-bench Verified (N=500), the context budget is 16k effective tokens, the agent step budget is 200, and a single-shot fallback fires if the agent does not produce a patch. Each instance is run under three conditions, with the only variable being the contents of the `AGENTS.md` file the scaffold loads at the start of the trajectory: `no_context` (none), `cross_repo` (assembled by the procedure above), and `probe_refined` (the companion paper's per-repo artifact; its tuning loop ran on the target repository). This is a single trial, run once. The companion paper's four-trial replication establishes that trial-level

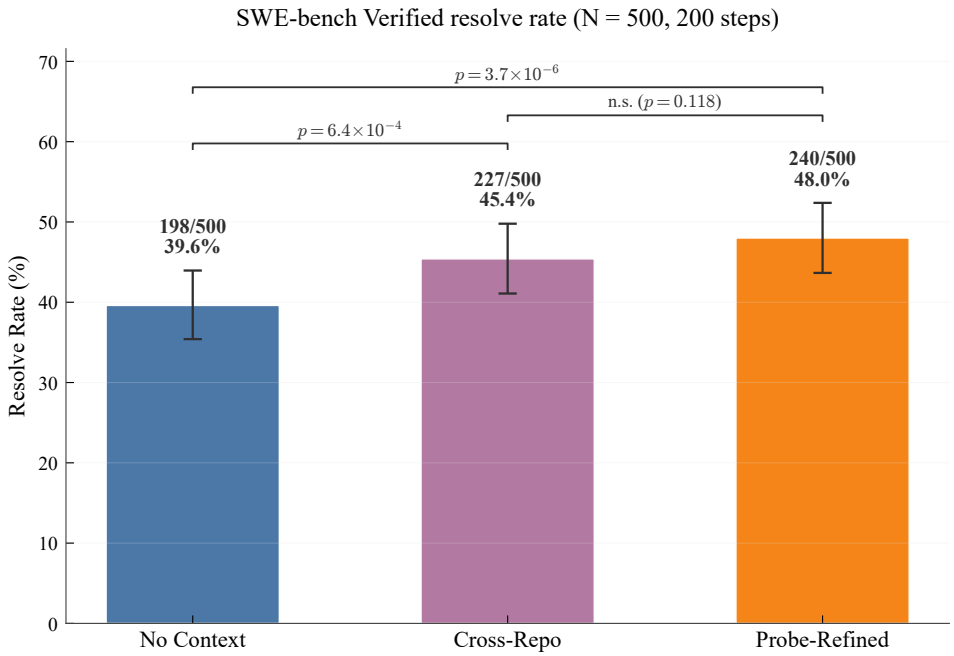

Figure 3: Resolved instances per condition on SWE-bench Verified (N=500, 200 steps). Error bars are 95 % Wilson confidence intervals on the proportion. Brackets show exact two-sided McNemar p-values on the within-trial paired observations. Cross-repo guidance is significantly better than no context; the gap to probe-refined guidance does not reach significance at this trial count.

Table 2: Headline results on SWE-bench Verified (N=500, 200 steps, Qwen3.5-35B-A3B, single trial).

Condition	**Resolved**	**Rate**	**95 % Wilson CI**
`no_context`	198 / 500	39.6 %	[35.4 %, 44.0 %]
`cross_repo` (this work)	227 / 500	45.4 %	[41.1 %, 49.8 %]
`probe_refined` (per-repo)	240 / 500	48.0 %	[43.7 %, 52.4 %]
Pairwise McNemar (exact, two-sided)			
probe-refined vs. no context	$p = 3.7\times10^{-6}$		
cross-repo vs. no context	$p = 6.4\times10^{-4}$		
probe-refined vs. cross-repo	$p = 0.118$ (n.s.)		

variance for resolve rate at N=500 is small relative to instance-level variance (the trial random-effect variance was estimated at zero in a hierarchical model on 6,000 observations); however, that result does not transfer mechanically to the cross-repo condition, which has a different guidance distribution and may have different run-to-run variance.

3 Results

The headline result is summarized in Figure 3 and Table 2. Cross-repo guidance resolves 227 of 500 instances (45.4 %, 95 % Wilson CI [41.1, 49.8]), a 5.8 pp improvement over the unguided baseline (198 / 500, 39.6 %, [35.4, 44.0]); the McNemar exact two-sided test on the paired observations gives $p = 6.4\times10^{-4}$. Probe-refined guidance resolves 240 / 500 (48.0 %, [43.7, 52.4]), a further 2.6 pp above cross-repo and 8.4 pp above no context. The McNemar test for cross-repo vs. probe-refined does not reach significance at $\alpha = 0.05$ ($p = 0.118$); the test for probe-refined vs. no context is highly significant ($p = 3.7\times10^{-6}$).

Reading the McNemar p-values alongside the Wilson CIs. McNemar's test answers a within-trial question: of the instances where exactly one condition resolves and the other does not, is the split far enough from 50/50 to reject equal expected resolve rates? For our three contrasts, the underlying 2×2 tables are shown in Table 3 and the discordant-pair counts (62/20, 49/20, 36/23) are what drive the reported p-values. For probe-refined vs. cross-repo, the discordant split

Table 3: McNemar 2×2 contingency tables on the 500 paired observations. "X-only" and "Y-only" are the discordant cells whose asymmetry the McNemar test scores. Both X-only and Y-only counts are integers out of 500.

Contrast (X vs. Y)	**both pass**	**X only**	**Y only**	**both fail**	p
probe_refined vs. no_context	178	62	20	240	3.7×10^{-6}
cross_repo vs. no_context	178	49	20	253	6.4×10^{-4}
probe_refined vs. cross_repo	204	36	23	237	0.118

of 36/23 yields $p = 0.118$; that is consistent with two distinct stories the data cannot distinguish: (a) the two conditions are genuinely close in expectation and probe-refined wins by a small margin our $N{=}500$ paired sample cannot detect, or (b) probe-refined is meaningfully better and we have under-sampled. A back-of-envelope power calculation suggests that distinguishing a 2.6 pp difference at 80 % power would require roughly an additional 3–4 paired trials at $N{=}500$, depending on the discordant-pair rate.

The Wilson 95 % CIs answer a related but coarser question: is the resolve-rate point estimate robust under independent resampling? The Wilson CI for cross-repo (41.1–49.8 %) overlaps slightly with the upper tail of no-context (35.4–44.0 %), and overlaps more substantially with probe-refined (43.7–52.4 %). The cross-repo CI sits almost entirely inside the probe-refined CI. The CI overlap pattern and the McNemar p-value pattern point in the same direction: cross-repo is clearly better than no-context, and not clearly different from probe-refined. What McNemar adds is the within-trial pairing: every CI overlap that looks borderline becomes more interpretable once we know how many instances each condition gets uniquely right. For cross-repo vs. no-context, only 20 of 500 instances flipped from a cross-repo win to a no-context win; the asymmetry against the 49 that went the other way is what the small p-value reflects.

What *is* replicated. The cross-repo > no-context contrast is on solid footing. A 5.8 pp gap on $N{=}500$ paired observations with $p = 6.4\times10^{-4}$ would survive a Bonferroni correction for the three contrasts and is comparable in direction and magnitude to the static-KB → no-context gap reported in the companion paper (2.8 pp mean across four trials, $p = 0.004$ in the hierarchical model). The cross-repo gap is, in fact, larger than the static-KB gap, despite the cross-repo guidance being assembled from edits that never touched the target repository. We read this not as evidence that cross-repo guidance is *better* than the static KB (the comparison is single-trial and the conditions differ in more than one variable), but as evidence that the localization-aid interpretation in the companion paper extends to guidance whose source is not the target.

The mechanism is coverage, not precision. The harness disposition (Figure 4) replicates the companion paper's central mechanism finding. Per-patch precision (resolved / evaluable) is essentially flat across conditions: 52.1 % for no-context, 54.0 % for cross-repo, 54.4 % for probe-refined. What changes is how often the agent reaches an evaluable patch at all: 380, 420, and 441 of 500 respectively. The biggest single shift is in apply-error patches, malformed diffs the harness cannot apply, which drop from 51 under no-context to 13 under cross-repo (and 17 under probe-refined). Some of the cross-repo edits explicitly tell the agent to verify diff syntax and reject empty patches, and that quality-gate effect is visible directly in the disposition counts.

Per-repository breakdown. SWE-bench Verified is dominated by Django (231 of 500 instances), so we report per-repo counts to confirm the effect is not Django-only (Table 4). The cross-repo > no-context ordering holds in 7 of the 9 repos with at least 8 instances; matplotlib is the only repo where cross-repo strictly underperforms no-context (13 vs. 14), and on scikit-learn cross-repo and probe-refined both lose 2 instances relative to no-context. On Django specifically, cross-repo and probe-refined tie at 112; the residual gap to probe-refined is concentrated in sympy (33 vs. 37), sphinx (17 vs. 21), astropy (6 vs. 8), and xarray (11 vs. 14). This is the pattern the companion paper's localization analysis predicts: target-specific tracing instructions help most on the scientific and numerical repositories with idiosyncratic internal layouts, and these are exactly the repos where cross-repo guidance, which by construction cannot name target-specific files, falls short.

4 Discussion

What this experiment shows and what it does not. The narrowest claim the data support is: a procedure that mines and transfers operational guidance from external repositories produces an `AGENTS.md` that meaningfully helps a Qwen3.5-35B-A3B coding agent on a target repository it never saw during tuning, at a cost roughly indistinguishable in deployment from the no-context baseline (the assembled file is a few thousand characters at the front of the prompt), and the gain is mechanistically the same kind of gain the companion paper reports: more well-formed evaluable patches,

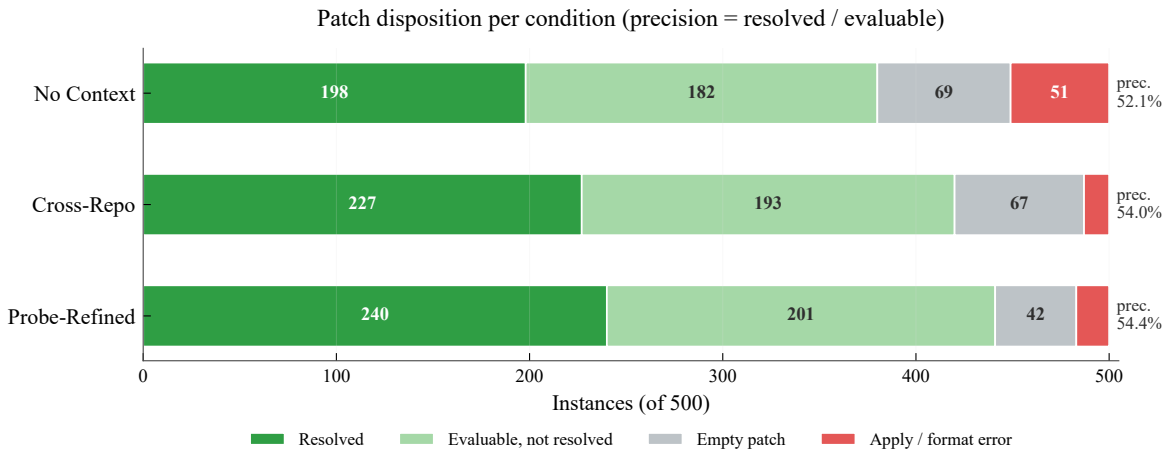

Figure 4: Patch disposition per condition (the four buckets are mutually exclusive and sum to $N{=}500$). Both guided conditions move instances out of the apply-error bucket and into the evaluable buckets while leaving per-patch precision (right column, resolved / evaluable) approximately constant at 52–54 %.

Table 4: Per-target-repo resolved counts. Repos are sorted by instance count; `n` is the per-repo SWE-bench Verified count.

Repository	**n**	`no_context`	`cross_repo`	`probe_refined`
django/django	231	94	112	112
sympy/sympy	75	27	33	37
sphinx-doc/sphinx	44	12	17	21
matplotlib/matplotlib	34	14	13	11
scikit-learn/scikit-learn	32	19	17	19
astropy/astropy	22	5	6	8
pydata/xarray	22	11	11	14
pytest-dev/pytest	19	11	12	12
pylint-dev/pylint	10	2	2	2
psf/requests	8	2	3	3
mwaskom/seaborn	2	0	0	0
pallets/flask	1	1	1	1
Total	500	198	227	240

not better per-patch precision. What we do *not* show is that cross-repo guidance is as good as probe-refined guidance. A single trial with a 13-instance gap in the expected direction is consistent with the two conditions being equivalent in expectation, but it is also consistent with probe-refined being meaningfully better; the companion paper reports per-trial SDs of 1.4–2.2 pp at $N{=}500$, and a 2.6 pp gap is exactly the kind of difference single-trial measurements cannot adjudicate. The cross-repo $\approx$ probe-refined reading should be treated as "not yet ruled out by these data," not "established."

Why the gap to probe-refined might persist. The companion paper's localization analysis identifies a class of instances (roughly a third of the probe-refined-only consistent solves) where the fix lives in a non-obvious internal module that the agent reaches only when the guidance contains explicit subsystem-specific tracing instructions. By construction, the cross-repo procedure cannot produce edits naming target-specific files, because the filter step drops them. If this class accounts for the residual 2.6 pp gap, it would be expected to remain, and in fact would not shrink with additional source repos, no matter how large the pool gets. The per-repo breakdown (Table 4) is consistent with this: the residual probe-refined advantage is concentrated in sympy, sphinx, astropy, and xarray, the same scientific and numerical repositories the companion paper flagged as benefiting most from repo-specific tracing.

5 Limitations

Single trial. The most important limitation is the one we have already discussed: with one trial per condition, our power to distinguish the cross-repo and probe-refined conditions is poor. Three or four additional trials of all three conditions at $N{=}500$ would put the indistinguishability claim on the same footing as the four-trial replication in the companion paper. We did not run them.

Single model and scaffold. All conclusions are specific to Qwen3.5-35B-A3B in the same scaffold the companion paper used. The companion paper's Nemotron experiment shows that guidance calibrated for one model can actively harm a different model; the cross-repo guidance, having been mined with Qwen, inherits this model-fit dependency. Whether a multi-model mining pool would produce more transferable edits is open.

Similarity score is uninterrogated. The 8-feature categorical similarity score with weights $(3, 2, 2, 1, 1, 1, 1, 1)$ was chosen for explainability rather than expressiveness, and we make no claim that this particular feature set or weighting is optimal. The single trial reported here cannot tell us *which* of the eight features carries the signal: it is plausible that test-framework match alone (the highest-weighted feature) does most of the work, or that the score functions mostly as a proxy for "Python web/data library written in the last five years" and any rough match would do. A targeted feature-ablation experiment would re-run Phase 4 (assembly) with each feature in turn dropped or up-weighted, and re-evaluate at the same N and step budget. This is the most natural follow-up to the present work, and the most likely path to either a more reliable similarity score or evidence that the choice of features matters less than we assumed. The 50-repo pool is also small enough that for some targets the 10th-best match is not very similar (see `sphinx-doc/sphinx` in Figure 2, right, where the top-10 mean similarity is 0.72), so a separate axis of follow-up would scale the source pool.

Source-repo coverage is uneven. Of the 12 assembled guidance files, half draw from a single source repo and only two (`astropy/astropy` and `mwaskom/seaborn`) draw from three. This concentration is a side effect of the assembly procedure walking the edit pool ranked by similarity and stopping at the budget; the most-similar source for a given target tends to dominate. Whether forcing source diversity would help, hurt, or do nothing is not tested here.

6 Conclusion

The companion paper's probe-and-refine procedure can be re-purposed as a cross-repository transfer pipeline: mine probe artifacts from 50 external Python repos, filter to a generic pool, route to targets by structural similarity, and assemble within the same character budget. The resulting guidance significantly outperforms no context on SWE-bench Verified ($p = 6.4{\times}10^{-4}$) and is not measurably worse than per-target probe-refined guidance at $N{=}500$ in a single trial ($p = 0.118$). The second comparison is the more interesting of the two, and is also the one our data are least equipped to settle. We document the pipeline so the multi-trial replication that would settle it is straightforward to run.

References

Asa Shepard. Probe-and-Refine Tuning of Repository Guidance for Coding Agents. 2026.

SWE-bench. SWE-bench Verified. 2024. `https://www.swebench.com/verified.html`.

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.

Qwen Team. Qwen3.5-35B-A3B. 2026. `https://huggingface.co/Qwen/Qwen3.5-35B-A3B`.