THE WINNER'S CURSE IN LLM PIPELINE SELECTION: HOW CORRELATED CANDIDATE POOLS BREAK BEST-OF-N REPORTING, AND HOW TO CORRECT IT

Anonymous authors
Paper under double-blind review

ABSTRACT

Best-of-N pipeline selection, running N candidate prompts on an m-item dev set and reporting the winner's dev score, is the default way an LLM practitioner picks and reports a prompt. That dev score is optimistic: the winner was chosen in part because it got lucky on the dev items, and its true accuracy on fresh data is lower. We measure this winner's curse directly on four correctness matrices spanning three tasks (SST-2, Subj, AG News) and two models (Qwen2.5-1.5B-Instruct, Llama-3.2-3B-Instruct), sweeping N from 1 to 100 and m from 20 to 200. Bias grows with N and shrinks roughly as $1/\sqrt{m}$, reaching 0.088 accuracy points at $N = 100$, $m = 100$ on Subj. We then characterize why real candidate pools inflate this bias beyond the independent-candidates textbook case: two effective-candidate-count estimators, a spectral participation ratio and a moment-matching inversion of the observed selection gap, agree that a 105-to-111-candidate pool behaves like roughly 1.5 to 3.5 independent draws. Breaking that count down by candidate provenance turns up a result opposite our own working hypothesis: LLM-generated (APE-style) candidates carry far less effective diversity ($N_{\text{eff}}/K \approx 0.017$–$0.041$) than either hand-written seed prompts or paraphrase/mutation candidates ($N_{\text{eff}}/K \approx 0.14$–$0.24$ for both), consistently across all three tasks. We compare six ways to report a best-of-N winner (no correction, a naive dev split, two extreme-value union-bound corrections keyed to K or to N_{eff}, cross-fitting, and a bootstrap plug-in) on bias, mean squared error, and 90% interval coverage; correcting against N_{eff} dominates the independent-candidates union bound on every axis and matches or beats every other method on mean squared error, while the naive K-based correction overshoots badly enough to be worse than reporting nothing at all. Resampling the stored matrices at the exact (N, m) of a real, live EvoPromptGA trajectory reproduces its measured bias on all three tasks, inside a 90% predictive band in every case, which supports treating a one-time correctness matrix plus free resampling as a faithful stand-in for repeated live search. Finally, we derive budget-allocation curves for a fixed evaluation budget $B = N \cdot m$: on tasks already near their accuracy ceiling, the optimal split spends almost the whole budget on m; on a task with real headroom left, it keeps paying to search wider even as N_{eff}/K stays in the low single digits.

1 INTRODUCTION

A practitioner comparing 100 candidate prompts on a 50-item dev set, then shipping whichever one scored highest, is running an experiment with 100 implicit hypotheses and reporting only the result of the one that happened to win. The dev score of that winner is not a neutral readout of its quality. Some fraction of why it won is that it happened to answer this particular set of 50 items well, not that it is the best prompt in the pool. Deploy it, or re-score it on a fresh set of items, and its accuracy typically comes in lower than the number that got reported. This is the winner's curse, well known in genomics (Cawley & Talbot, 2010), econometrics (Andrews et al., 2024), and A/B testing (Berman et al., 2026), and it applies without modification to best-of-N prompt and pipeline selection: search over N candidates, evaluate each on m dev items, report the argmax's dev score as the method's accuracy.

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

This paper measures that bias directly rather than assuming its size, characterizes why real candidate pools make it worse than the textbook independent-candidates case, and turns both measurements into something a practitioner can act on, in four contributions:

1. **Measurement.** We build a per-item correctness matrix once per (task, model) pair and sweep the entire (N, m) grid by resampling it, decoupling the cost of covering the grid from the cost of running the model. Across three tasks and two models, selective bias grows with N, shrinks with m, and reaches 0.088 accuracy points on our hardest task at $N = 100, m = 100$.
2. **Mechanism.** A candidate pool of near-paraphrases behaves like far fewer than K independent draws. Two effective-candidate-count estimators agree on this, and breaking the estimate down by provenance shows that LLM-generated (APE-style) candidates carry the least effective diversity of the three provenance strata, the opposite of what we expected.
3. **Correction.** We compare six ways to report a best-of-N result on bias, mean squared error, and coverage, and show that correcting against the effective candidate count, estimated from the dev matrix a practitioner already has, beats every alternative we test on mean squared error with no extra data.
4. **Allocation.** Given a fixed budget split between candidate count and dev-set size, the right split depends on how much true performance a task still has to find, not a fixed ratio.

Measuring the bias itself, contribution 1 above, is not new on its own: SIREN (Xu et al., 2026), published in May 2026, names and corrects this exact winner's curse for adaptive prompt and program search. Section 2 details what it covers and what it does not; in short, it corrects the bias under a fixed, already-selected shortlist and states explicitly that open-ended search over a correlated candidate pool is future work. That is precisely contributions 2 through 4 of this paper: a candidate pool built by an actual prompt-generation procedure is not a bag of K independent draws, effective-candidate-count estimation quantifies exactly how far from independent it is, and that estimate feeds directly into both a better correction and a budget-allocation rule.

2 Related Work

SIREN. Xu et al. (2026) name the winner's curse in adaptive LLM benchmarking directly and propose a selection-aware repeated-split reporting protocol that freezes a post-search shortlist, separates the split used for selection from the split used for held-out evaluation, and quantifies uncertainty with an item-level Gaussian multiplier bootstrap, validated on MMLU-Pro tuning runs. SIREN scopes its estimator to a fixed-shortlist regime with smooth, stabilized selection and states that fully open-ended adaptive search remains future work; the analysis here treats the whole (N, m) grid, including small N and small m, without that smoothness assumption. Its correction is also agnostic to the internal structure of the candidate pool: it does not ask how correlated the candidates are with one another, which is the question Sections 5 and 6 answer, and it does not derive guidance on how to split a fixed budget between candidate count and dev-set size, which is Section 7.

Prompt optimization. APE (Zhou et al., 2023) is best-of-N selection in its original form: an LLM proposes a pool of instruction candidates, each is scored on a dev set, and the argmax is reported, exactly the procedure this paper studies the bias of. EvoPrompt (Guo et al., 2024) replaces one-shot argmax selection with a population evolving under LLM-driven crossover and mutation; its selection-plus-crossover loop is what Section 8's live trajectory reimplements. OPRO (Yang et al., 2024) iterates an LLM-as-optimizer trajectory and reports, in its own overfitting analysis, a train/test gap consistent with a winner's curse, without measuring it across an (N, m) grid or correcting for it. GAAPO (Sécheresse et al., 2025) reports a similar population-size-dependent train/test gap in a genetic-algorithm prompt optimizer. Akinwande et al. (2023) give a PAC-Bayes account of why prompt generalization gaps are often smaller than expected, a theoretical complement to the empirical bias measurement in Section 4. promptolution (Zehle et al., 2025) and DSPy (Khattab et al., 2023) are modular tooling for building and benchmarking prompt optimizers; neither's reported experiments measure bias across a candidate-count-by-dev-size grid.

Selection bias and the winner's curse outside NLP. Cawley and Talbot (2010) document the same phenomenon under nested cross-validation and model selection in classical machine learning. Dwork et al. (2015) formalize the reusable holdout as a defense against this kind of adaptive

reuse of a fixed evaluation set. Berk et al. (2013) derive simultaneously valid post-selection confidence intervals by purchasing coverage across every possible selection outcome, the same idea behind Section 6's union-bound corrections. Andrews, Kitagawa, and McCloskey (2024) and Zrnic and Fithian (2024) develop valid inference on the winning arm under arbitrary dependence between candidates; SIREN builds on the former, and neither has been applied to a correlated LLM prompt pool. Berman, Zhang, and Zhao (2026) evaluate a similar slate of corrections, plug-in, cross-fitting, and sample splitting among them, against bias, mean squared error, and coverage in a randomized-experiment setting, largely the same targets and method slate as Section 6, outside any LLM context. Nikolopoulos (2026) estimates an effective-multiplicity count for a correlated pool of backtested trading strategies with a spectral participation-ratio estimator essentially identical to our Estimator A, in a different domain with no LLM candidates. Miller (2024) argues for treating LLM evaluations as the statistical experiments they are, the standard this paper's bootstrap intervals and pre-registered sanity checks try to meet. Li and Talwalkar (2019) document a related search-budget-dependent optimism gap in neural architecture search, a different search domain with the same cause.

3 Problem Setup

A candidate pool holds K prompts or pipelines. Each candidate k has a true accuracy μ_k on the task's full item distribution. A best-of-N procedure draws $N \leq K$ candidates from the pool, evaluates each on m dev items, and reports the dev score of the argmax. Writing $\hat{\mu}_k^{(m)}$ for a candidate's dev-set accuracy over those m items and $k^* = \arg\max_k \hat{\mu}_k^{(m)}$ over the N sampled candidates, the two quantities this paper measures are the *selective bias*

$$B = \mathbb{E}\left[\hat{\mu}_{k^*}^{(m)} - \mu_{k^*}\right]$$

and the *regret*

$$R = \mathbb{E}\left[\max_{k \in N} \mu_k - \mu_{k^*}\right].$$

B is how inflated the number a practitioner reports is; R is how much true accuracy noisy selection actually costs, since k^* need not be the best of the N sampled candidates even by their true accuracy.

Correctness matrices and free resampling. For each (task, model) pair we build a $K \times M$ binary correctness matrix C, $C[k, i] = 1$ iff candidate k answers item i correctly, in a single pass over the model: every candidate is scored on every one of $M = 600$ class-stratified items with constrained, single-token, temperature-0 decoding restricted to the task's verbalizer tokens. The 600 items split into a 300-item dev pool and a disjoint 300-item truth pool, drawn once, published as index lists, and reused by every experiment in this paper. Every subsequent quantity, every (N, m) cell, every correction method, every bootstrap replicate, is a resample of that one stored matrix: draw N of the K rows and m of the 300 dev columns without touching the model again, take the dev-column argmax, and look up its truth-pool score. This is what makes an exhaustive sweep of the (N, m) grid, six correction methods, and thousands of bootstrap draws affordable on a free-tier GPU budget: the GPU cost is fixed at one pass per (task, model) pair, and every additional analysis is free.

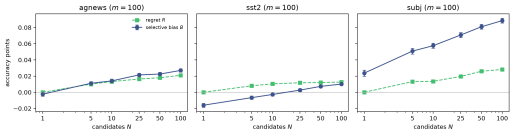
Tasks and models. The three core tasks, SST-2, Subj, and AG News, are the same task set EvoPrompt (Guo et al., 2024) benchmarks against, chosen so measured bias sits next to a published optimizer's reported gains without an apples-to-oranges caveat (Section 9). The primary model is Qwen2.5-1.5B-Instruct, built for all three core tasks; Llama-3.2-3B-Instruct is a secondary model, built for SST-2 only, as a cross-model check on the mechanism findings in Section 5. Each candidate pool mixes three provenance strata: 10 hand-written seed prompts, APE-style candidates generated by prompting an LLM to propose new instructions (Zhou et al., 2023), and paraphrase/mutation candidates produced by perturbing existing candidates.

4 Measuring the Winner's Curse

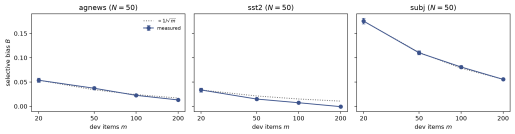
Table 1 summarizes the four correctness matrices. Subj has by far the lowest median accuracy (0.513) and the widest spread (0.387 to 0.637), leaving more true performance for a search procedure to find; AG News and SST-2 already sit close to a ceiling near 0.86 to 0.94.

Table 1: Per-(task, model) summary. K is candidates after near-duplicate pruning (Section 10); dev and truth pools are 300 items each, disjoint. Split offset is the fixed dev/truth difficulty gap (Section 3), measured at $N = 1$ where no selection occurs. N_{eff} (spectral) is Estimator A pooled over all K candidates (Section 5).

Task	Model	K	Dev	Truth	Acc. min	Acc. median	Acc. max	N_{eff} (spectral)
AG News	Qwen2.5-1.5B	111	300	300	0.707	0.860	0.887	1.5
SST-2	Qwen2.5-1.5B	105	300	300	0.503	0.907	0.940	2.5
SST-2	Llama-3.2-3B	106	300	300	0.593	0.892	0.937	3.0
Subj	Qwen2.5-1.5B	111	300	300	0.387	0.513	0.637	3.5



(a) vs. candidate count N, fixed $m = 100$



(b) vs. dev-set size m, fixed $N = 50$

Figure 1: Selective bias B and regret R for the three core tasks (Qwen2.5-1.5B-Instruct). (a) growth with candidate count N. (b) decay with dev-set size m; dotted line is a $1/\sqrt{m}$ reference. Error bars are bootstrap 90% intervals.

Figure 1 (left) shows bias growing with N on all three tasks, tracking regret closely on AG News and SST-2 and pulling ahead of it on Subj, where the gap between the two curves is the part of the reported number that reflects lucky dev-set noise rather than any real accuracy the extra search found. Figure 1 (right) shows bias falling as m grows, close to the $1/\sqrt{m}$ rate a Gaussian selection-noise model predicts, though SST-2 falls faster than that reference and Subj falls slower. Figure 2 lays out the full grid: bias on Subj climbs to 0.189 at the smallest tested dev size ($N = 100, m = 20$) and still sits at 0.088 (90% interval $[0.086, 0.091]$) at $N = 100$, $m = 100$, roughly three times AG News's 0.027 ($[0.025, 0.029]$) and nine times SST-2's 0.010 ($[0.009, 0.012]$) at the identical grid cell. SST-2's bias is occasionally negative at small N; that is not noise crossing zero by chance but the fixed dev/truth split offset in Table 1 (-0.0151 for SST-2/Qwen), the fact that the two disjoint 300-item pools are not exactly equally difficult, which persists at every N and is a property of the split, not of selection.

Three sanity checks gate every figure in this section: bias must be non-decreasing in N, non-increasing in m, and, once the fixed split offset is subtracted out, statistically indistinguishable from zero at $N = 1$, where no selection occurs. All three checks pass on every committed matrix, within a tolerance scaled to each grid cell's own resampling standard error rather than a single fixed threshold, since the $N = 1$, $m = 20$ corner of the grid is an order of magnitude noisier than the $N = 100$, $m = 200$ corner.

5 Mechanism: Correlated Candidate Pools

A pool of K near-paraphrases of the same instruction succeeds and fails on nearly the same items, so it carries far less selection-relevant diversity than K independent candidates would. We estimate the effective candidate count N_{eff} two ways.

Estimator A (spectral). Center each candidate's per-item correctness row by its own mean, correlate the K centered rows across items, and take the eigenvalues λ_i of that $K \times K$ correlation matrix. The participation ratio

$$N_{\text{eff}} = \frac{\left(\sum_i \lambda_i\right)^2}{\sum_i \lambda_i^2}$$

equals K for a perfectly independent pool and falls toward 1 as the pool collapses onto a single shared factor.

Estimator B (moment-matching). Invert the extreme-value relation between candidate count and the expected size of the best dev-set overshoot. Given the empirically observed $\mathbb{E}[\max_k(\hat{\mu}_k - \mu_k)]$ at dev size m and the per-item Bernoulli noise scale σ, solve numerically for the candidate count

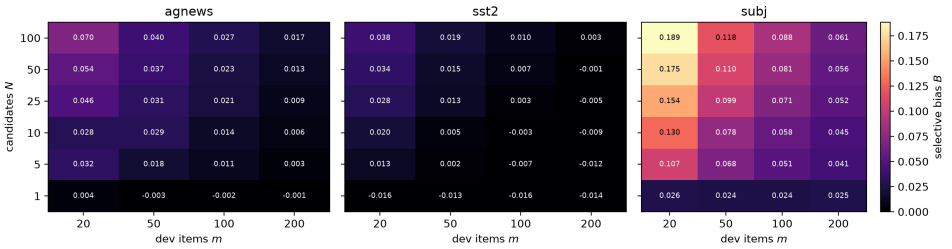


Figure 2: Selective bias B across the full (N, m) grid, one panel per task.

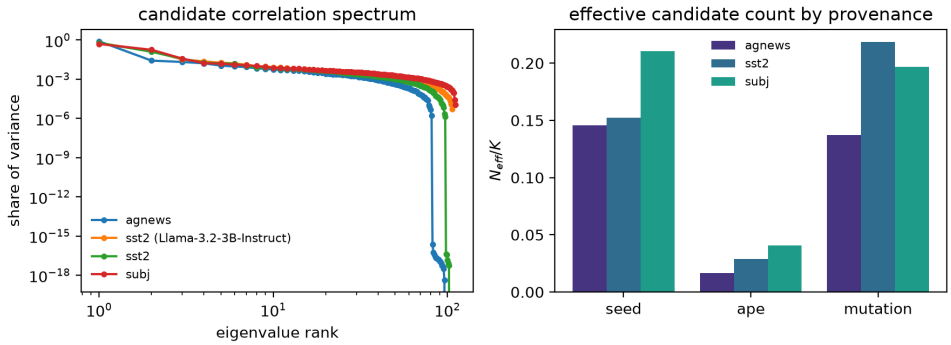


Figure 3: Left: eigenvalue spectrum of the candidate correlation matrix, one curve per (task, model), log-log axes. Right: N_{eff}/K (Estimator A) by candidate provenance, for the three core tasks (Qwen2.5-1.5B-Instruct).

whose expected maximum of that many i.i.d. Gaussian draws matches what was observed, using the exact Monte Carlo expected maximum rather than the asymptotic $\sigma\sqrt{2\ln n}$ approximation, which overstates the expected maximum by roughly a third at N in the range this paper tests.

Figure 3's left panel shows why: every spectrum collapses by three to four orders of magnitude within the first handful of eigenvalues, then flattens into a long, nearly flat tail. Table 1's last column reports the pooled result: a 105-to-111-candidate pool behaves like 1.5 to 3.5 independent draws, one to three percent of K.

Table 2 breaks the pooled estimate down by provenance and turns up a result opposite our own working hypothesis going into this study. We expected hand-written seed prompts, being deliberately varied by a human, to carry the most effective diversity, and mutation candidates, being small perturbations of existing text, to carry the least. Instead, APE-generated candidates have the lowest N_{eff}/K on every one of the three core tasks, by roughly a factor of five to eight below both seed and mutation candidates, and the ordering (APE lowest, seed and mutation both several times higher) is identical across AG News, SST-2, and Subj. The SST-2 cross-model check on Llama-3.2-3B-Instruct reproduces the same ordering (N_{eff}/K: seed 0.194, APE 0.034, mutation 0.241), so this is not an artifact of one model's scoring behavior. A plausible mechanism: an LLM asked to propose many candidate instructions from a small set of exemplars tends to restate the same underlying instruction in different words, which varies the text substantially while barely moving which items the resulting prompt gets right or wrong, whereas the mutation and hand-written strata start from more behaviorally distinct starting points. We report this as a genuine empirical finding rather than adjust the candidate-generation procedure to match our prior expectation; Section 10 notes that the mutation stratum, at K between 8 and 12 per task, is small enough that its own N_{eff} estimate is noisier than APE's or seed's.

6 Bias Correction

We compare six ways to report a best-of-N result, each seeing only the $N \times m$ dev matrix a practitioner would actually have:

Table 2: N_{eff} (Estimator A, spectral) by candidate provenance, Qwen2.5-1.5B-Instruct.

Task	Provenance	K	N_{eff}	N_{eff}/K
AG News	seed	10	1.46	0.146
AG News	APE	91	1.52	0.017
AG News	mutation	10	1.37	0.137
SST-2	seed	10	1.52	0.152
SST-2	APE	87	2.50	0.029
SST-2	mutation	8	1.75	0.219
Subj	seed	10	2.10	0.210
Subj	APE	89	3.61	0.041
Subj	mutation	12	2.36	0.197

Table 3: Correction methods, averaged over the full (N, m) grid and the three core tasks (Qwen2.5-1.5B-Instruct). Coverage target is 90%.

Method	Bias	|Bias|	MSE	Coverage
No correction	0.031	0.031	0.0029	74.8%
Naive split	0.001	0.002	0.0032	88.4%
Union bound, K	−0.051	0.051	0.0043	84.3%
Union bound, N_{eff}	0.003	0.005	**0.0014**	93.7%
Cross-fitting	−0.005	0.005	0.0016	98.0%
Bootstrap plug-in	0.009	0.009	0.0017	84.4%

- **No correction.** Report the dev score of the argmax. Current common practice.
- **Naive split.** Select on half the dev items, report the held-out half's score for the selected candidate. Unbiased by construction, at the cost of selecting on a noisier half-size dev set and reporting an interval sized to that same half.
- **Union bound over K.** Subtract the expected maximum of N i.i.d. Gaussian dev-noise draws from the dev score, and widen the interval by a Bonferroni correction over N, the textbook correction under the assumption that every sampled candidate is an independent draw.
- **Union bound over N_{eff}.** Identical machinery, but with N replaced by an in-sample Estimator A computed from the same dev matrix, so the method needs no data beyond what a practitioner already has.
- **Cross-fitting.** Average the held-out score of the full-data argmax over 50 independent random half-splits of the dev set, which damps the split-to-split noise a single naive split is exposed to.
- **Bootstrap plug-in.** Bootstrap-resample the dev items, re-run selection on each resample, and use the average gap between the resampled winner's score and its full-dev score as a plug-in estimate of the bias to subtract off.

Table 3 and Figure 4 make the trade-offs explicit rather than picking a single scalar to rank methods by. Union bound over K overcorrects hard enough to be actively counterproductive: it assumes every one of the N sampled candidates is an independent draw, which Section 5 shows is false by roughly two orders of magnitude, and the result is a method whose mean squared error (0.0043) is worse than doing nothing at all (0.0029). Across the four (task, model) pairs and the full grid, its squared error differs from the uncorrected baseline, after Holm–Bonferroni correction across all 240 (task, model, N, m, method) comparisons, in all 48 of its own cells, and the difference runs the wrong way in most of them. Naive split is nearly unbiased (mean bias 0.001) but pays for it with a worse mean squared error (0.0032) than four of the other five methods, since halving the dev set roughly doubles the standard error of both selection and reporting; it also undercovers (88.4% against a 90% target). Bootstrap plug-in undercorrects and undercovers similarly (84.4%). Cross-fitting comes closest to eliminating bias entirely (mean bias -0.005) and over-covers safely (98.0%), at the cost of the widest intervals of any method we test. Union bound over N_{eff} comes closest to

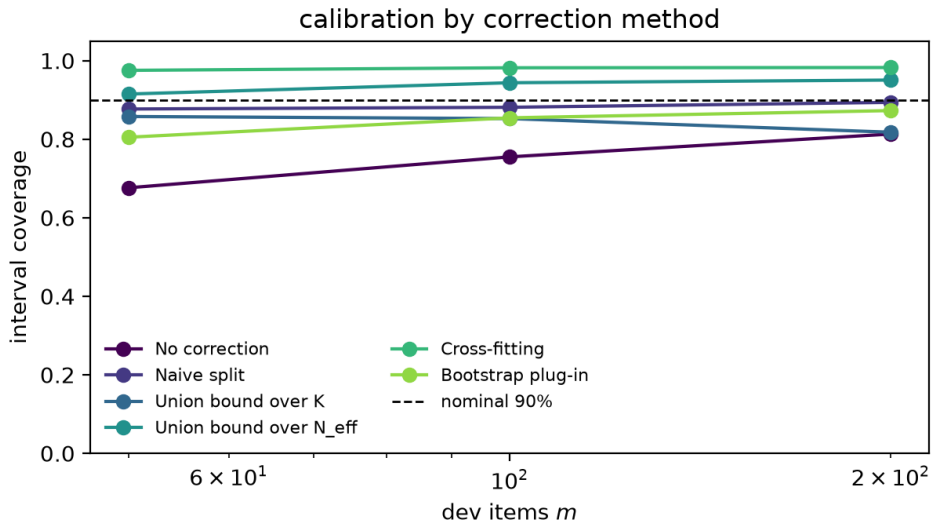


Figure 4: 90% interval coverage by correction method, as dev-set size m grows, averaged across the three core tasks and the N dimension of the grid. Dashed line is the nominal 90% target.

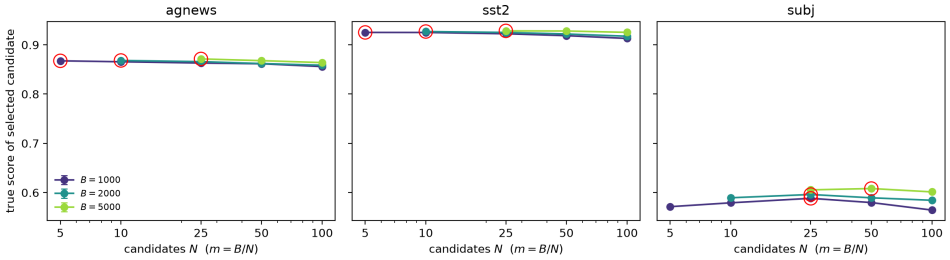


Figure 5: True accuracy of the selected candidate as a function of N (with $m = B/N$ held fixed to the labeled budget), for three budgets. Circled points mark the true-accuracy optimum at each budget.

the nominal target on every axis at once: mean bias near zero (0.003), the lowest mean squared error of all six methods (0.0014), and coverage (93.7%) close to nominal without cross-fitting's degree of conservatism, all from a quantity computed directly from the dev matrix a practitioner already has, no extra held-out data or repeated re-fitting required. We recommend it as the default correction when a spectral N_{eff} estimate is available, matching the pattern Berman et al. (2026) report for a structurally similar method slate outside any LLM context: no single correction dominates on every target, and the right choice depends on which of bias, mean squared error, or coverage the reporting context cares about most.

7 Budget Allocation

Given a fixed evaluation budget $B = N \cdot m$, how should a practitioner split it between searching wider (more candidates N) and measuring more precisely (a larger dev set m)? We sweep $B \in \{1000, 2000, 5000\}$ over the same (N, m) grid and, for each budget, locate the split that maximizes the *true* accuracy of the resulting selection, not its reported dev score.

Figure 5 shows the same budget producing a different optimal split on different tasks. On AG News and SST-2, both already close to their accuracy ceiling, the optimum sits at the smallest feasible N and the largest tested m at every one of the three budgets: $(N, m) = (5, 200) \rightarrow (10, 200) \rightarrow (25, 200)$ as B grows from 1000 to 5000 on both tasks. On Subj, still far from any ceiling, the optimum instead moves toward more candidates: $(25, 40) \rightarrow (25, 80) \rightarrow (50, 100)$ over the same budget range, spending a larger share on search even though Section 5 shows Subj's pool is exactly

Table 4: Live EvoPromptGA trajectory vs. pool-replay prediction, $N_{\text{live}} = 20$, $m = 100$, one seed per task.

Task	Bias (live)	Replay μ (σ)	90% band	z
SST-2	-0.030	$+0.002$ (0.025)	$[-0.040, +0.040]$	-1.27
Subj	$+0.080$	$+0.069$ (0.041)	$[\ 0.000, +0.137]$	$+0.27$
AG News	$+0.037$	$+0.019$ (0.029)	$[-0.030, +0.067]$	$+0.59$

Table 5: Measured selective bias B at $N = 50$, Qwen2.5-1.5B-Instruct, against Zhou et al. (2023)'s reported APE gain over a human-written baseline prompt, text-davinci-002. "Explained" is B divided by the reported gain.

Task	m	Bias	Gain (APE)	Explained
AG News	50	0.037	0.051	73%
AG News	100	0.023	0.051	44%
AG News	200	0.013	0.051	26%
SST-2	50	0.015	0.042	35%
SST-2	100	0.007	0.042	17%
SST-2	200	-0.001	0.042	-2%
Subj	50	0.110	0.085	130%
Subj	100	0.081	0.085	95%
Subj	200	0.056	0.085	65%

as redundant, in relative terms, as the other two tasks' pools. The right split is not a fixed ratio between N and m. It tracks how much true performance a task still has left to find: on a near-ceiling task, additional candidates barely move the achievable accuracy, so the budget is better spent measuring a modest pool precisely; on a task with real headroom, continued search keeps paying for itself even though it costs more selection regret, because the true accuracy gained from finding a better candidate outweighs that cost.

8 Validity of the Pool-Replay Methodology

Every result so far comes from resampling a static, pre-built correctness matrix, not a live search. That is a fair stand-in for a real optimizer only if a live trajectory's selection behavior looks like drawing N candidates uniformly at random from the fixed pool, which we check directly rather than assume.

For each core task we ran one EvoPromptGA-style trajectory (population 10, 10 generations) against the live model, a lightweight, faithful-in-spirit reimplementation of EvoPrompt's selection-plus-LLM-crossover loop (Guo et al., 2024) rather than a port of its codebase, and recorded which unique candidates it touched ($N_{\text{live}} = 20$ across all three tasks) and how it scored them on the same 100-item dev slice used throughout ($m = 100$). We then resampled the corresponding static matrix 1,000 times at the matched $(N = 20, m = 100)$ cell and compared the live run's actual bias to the spread of those resampled draws.

Table 4 reports the result: on all three tasks, the live trajectory's bias falls inside the empirical 90% predictive band of the replay draws, with z-scores between -1.27 and $+0.59$. A single live trajectory is one draw, not an average, so it belongs against the spread of individual replay draws, not the tighter uncertainty of the replay *mean*; judged the wrong way, an ordinary single run would look like an outlier by construction, regardless of task or model. Three tasks and one seed each is evidence for the pool-replay methodology's fidelity, not proof of it, and we treat it that way in Section 10.

9 Comparison to Published Gains

We compare against Zhou et al. (2023)'s reported APE gains only; extending this comparison to additional published optimizers is left to future work. The two studies use different base models

(Qwen2.5-1.5B-Instruct and Llama-3.2-3B-Instruct here, text-davinci-002 in Zhou et al.), so Table 5's "explained" column is not a claim that Zhou et al.'s specific reported number is wrong. It is a demonstration that a winner's curse of the size Section 4 measures, on a task and pool of this scale, is large enough to account for a substantial share, and on Subj at the smallest dev size tested, more than the entirety, of a typical published prompt-optimizer gain over a baseline. A gain this close in size to the measured selection bias is not, by itself, evidence the underlying method does not work; it is evidence that reporting the bare dev score of a best-of-N winner is not enough to tell the two apart.

10 LIMITATIONS

Pool-replay validity. Section 8 tests one live trajectory per task, three data points total. All three land inside the predicted band. That is evidence for the pool-replay methodology, not proof it holds under every optimizer, seed, or task the static-matrix analysis covers.

SST-2/Qwen candidate count. The SST-2 pool (106 candidates: 10 seed, 88 APE, 8 mutation) was scored against both models. Two APE candidates, a near-paraphrase pair differing by one word, produced byte-identical correctness on all 600 items under Qwen2.5-1.5B-Instruct. We pruned one before analysis, leaving $K = 105$ for SST-2/Qwen; the SST-2/Llama matrix keeps the full $K = 106$. The published pool file flags the pruned candidate rather than deleting it.

Provenance split. Generation targeted a 45/10/45 APE/seed/mutation split. Deduplication at temperature 1.5 removed most near-identical mutation candidates and left roughly 88/10/8 instead. Table 2's mutation stratum, K between 8 and 12 per task, is correspondingly small, and its N_{eff} estimate is the noisiest of the three strata. We report the pools as built rather than re-running generation to hit the original target.

The APE/N_{eff} finding's robustness. Section 5's finding, that APE candidates carry less effective diversity than seed or mutation candidates, holds across all three core tasks and both models tested on SST-2. It rests on one generation configuration per task, one temperature, one generator model; we did not test other prompts, temperatures, or generator models.

Compute scale. Every result here comes from models up to 3B parameters, run on a free-tier Kaggle T4 budget of 30 GPU-hours per week. A 7B model stayed out of reach within that budget. Whether the bias magnitudes, correction-method ranking, or allocation guidance hold at larger scale remains untested.

Table 5's scope. Section 9's published-gains comparison covers one prompt optimizer (APE) and one baseline number per task. It makes no claim about how selection bias compares to gains reported by other prompt-optimization methods.

11 CONCLUSION

Reporting the dev score of a best-of-N winner without correction overstates its true accuracy by an amount that grows with how many candidates were searched and shrinks with how many dev items were used to judge them, reaching 0.088 accuracy points at $N = 100$, $m = 100$ on our highest-headroom task. That bias is worse than the textbook independent-candidates case predicts because real candidate pools are not independent: two effective-candidate-count estimators agree that a 105-to-111-candidate pool behaves like 1.5 to 3.5 independent draws, and LLM-generated candidates carry less of that diversity than hand-written or paraphrase-mutated candidates, the opposite of what we expected. Correcting against that effective count, rather than against the raw candidate count or not correcting at all, gets closer to the true accuracy on every metric we test while requiring nothing beyond the dev matrix a practitioner already has, and a live optimizer trajectory's bias lands inside what the static-matrix methodology predicts on every task we checked. The upshot is threefold: report a corrected number, not the winner's raw dev score; check how correlated your candidate pool is before trusting a union bound over its nominal size; and split a fixed budget by how much true performance the task still has to find, not a fixed rule of thumb. The correctness matrices, correction package, and figure-regeneration scripts here are released, so extending this analysis to more tasks, models, or optimizers costs one GPU pass, not a re-implementation.

AI USE STATEMENT

We used generative AI tools (Claude, Anthropic) for ICLR formatting: converting the manuscript from an IEEEtran two-column format to the ICLR LaTeX template, converting citations to the required natbib style, adjusting figure layout to fit the page limit, and tightening prose in the introduction, related work, limitations, validity, and conclusion sections for length. We did not use generative AI tools for any other part of this research. We checked every citation, figure, and numerical claim against the source manuscript and data after each edit. We take responsibility for the final content of this work, including text produced with AI assistance.

REPRODUCIBILITY STATEMENT

The correctness matrices, correction package, and figure-regeneration scripts described in Section 11 are available at an anonymized repository: `https://anonymous.4open.science/r/llm-pipeline-selection-05A4/`. The repository includes the per-item correctness matrices for all four (task, model) pairs, the resampling and bias-correction code used to produce every table and figure in this paper, and the scripts that regenerate each figure directly from the stored matrices.

REFERENCES

Victor Akinwande, Yiding Jiang, Dylan Sam, and J. Zico Kolter. Understanding prompt engineering may not require rethinking generalization, 2023.

Isaiah Andrews, Toru Kitagawa, and Adam McCloskey. Inference on winners. *The Quarterly Journal of Economics*, 139(1):305–358, 2024.

Richard Berk, Lawrence Brown, Andreas Buja, Kai Zhang, and Linda Zhao. Valid post-selection inference. *The Annals of Statistics*, 41(2):802–837, 2013.

Ron Berman, Walter W. Zhang, and Hangcheng Zhao. Valuing winners: When and how to correct for selection bias in randomized experiments, 2026.

Gavin C. Cawley and Nicola L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.08532.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines, 2023.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence (UAI)*, 2019. arXiv:1902.07638.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations, 2024.

Ioannis Nikolopoulos. Spurious predictability in financial machine learning, 2026.

Xavier Sécheresse, Jacques-Yves Guilbert-Ly, and Antoine Villedieu de Torcy. GAAPO: Genetic algorithmic applied to prompt optimization. *Frontiers in Artificial Intelligence*, 8:1613007, 2025. arXiv:2504.07157.

Yang Xu, Jiefu Zhang, Haixiang Sun, Zihan Zhou, Tianyu Cao, and Vaneet Aggarwal. Towards reliable LLM evaluation: Correcting the winner's curse in adaptive benchmarking, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.03409.

Tom Zehle, Timo Heiß, Moritz Schlager, Matthias Aßenmacher, and Matthias Feurer. promptolution: A unified, modular framework for prompt optimization, 2025.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2211.01910.

Tijana Zrnic and William Fithian. A flexible defense against the winner's curse, 2024.