Clustering-based Prompt Optimization for LLM Evaluation

Jinyoung Kim
NAVER Corp.
Republic of Korea
jin.y.kim@navercorp.com

Sun Kim
NAVER Corp.
Republic of Korea
sunkim.us@navercorp.com

Inho Kang
NAVER Corp.
Republic of Korea
once.ihkang@navercorp.com

Abstract

LLM evaluators ("AI judges") increasingly stand in for human raters in search-quality pipelines, but their agreement with humans is fragile to how the scoring rubric is worded. We study how to optimize that rubric under a fixed budget of expert labels and evaluation calls, comparing three methods on two evaluator tasks from a commercial search vertical (300 dual-rated queries each): an *intuitive linear loop* (read the worst disagreements, edit, re-score, repeat), the automated optimizer *GEPA*, and our proposed *cluster + batch-validate* loop (the *cluster loop*). The cluster loop runs in a single iteration: it clusters *all* baseline disagreements into consensus failure modes, builds one atomic single-rule variant per mode, and selects with a noise-aware bootstrap test.

The cluster loop yields the strongest held-out rubric on both tasks: Vertical Candidate +6.00 vs. +0.67 (linear), Vertical Quality +41.67 vs. +9.52 (linear) and +16.67 (GEPA). It is the only method robustly positive on both. For teams building human-aligned LLM evaluators, the results reduce to a recipe: aggregate *all* disagreements once; ship one named rule per failure mode; accept edits only on a noise-aware multi-seed bootstrap. The recipe applies wherever an LLM judge must track human judgment under a tight label budget.

Keywords

LLM evaluation, prompt optimization, human–AI agreement, rubric calibration, bootstrap selection

1 Introduction

LLMs are increasingly used as evaluators—scoring search-result quality, candidate plausibility, agent traces—at a scale and cost humans cannot match. The premise is that an LLM evaluator, prompted with the right scoring rubric, can stand in for a human rater. But the rubric is fragile: small wording changes shift agreement with humans by tens of percentage points. In our experiments, adding a *single* rule to a hand-written baseline moved the held-out agreement composite by +41.67 points on one task (§4.5). The gap between a hand-written rubric and a well-calibrated one is thus large enough to decide whether the evaluator is usable at all.

Human experts remain the ground truth; the LLM evaluator's job is only to reproduce their judgments, so the main optimization target is *human–AI agreement*. There are several reasons why general-purpose prompt optimizers are a poor fit. Methods such as DSPy/MIPRO [3, 7] and GEPA [1] tune a prompt so a downstream task is solved more accurately, and they assume evaluation is cheap and abundant—every candidate prompt can be scored on many examples at low cost. Evaluator-rubric calibration violates both assumptions: expert labels are scarce (a few hundred dual-rated queries), and each evaluation call is itself an expensive LLM inference. The binding constraint is not search-space size but *supervision and inference budget*. We therefore treat it as a data-efficiency problem: extract the most reliable rubric improvement per human label and per evaluation call.

The methodology question we address: *given a fixed budget of train queries and expert labels, how do we optimize the rubric prompt to maximize human–AI agreement on a never-shown held-out set, while spending as few evaluation calls as possible?*

We compare three methods on two evaluator tasks from the same vertical: the *intuitive linear loop* (the natural human debugging cycle: read the worst disagreements, edit, re-score, repeat), the automated optimizer *GEPA* [1], and our *cluster + batch-validate* loop (hereafter the *cluster loop*), which aggregates all disagreements once, proposes one atomic rule per consensus failure mode, and selects with a noise-aware bootstrap test. On both tasks the cluster loop produces a stronger held-out rubric than either alternative, at a fraction of the iteration count and, against GEPA, of the evaluation budget.

Contributions. §2 positions each contribution against the adjacent lines of work: task-accuracy prompt search, LLM-as-judge analysis, qualitative error analysis, and evaluator criteria alignment.

(1) **Budget-constrained rubric optimization for human–AI agreement.** We iterate the *rubric prompt* for *human–AI agreement* where the binding constraint is supervision and inference budget, not search-space size; the method is therefore engineered around *reusing every label and evaluation call*, not around exploring a large space.
(2) **Atomic, individually-attributable edits from failure-mode discovery.** We cluster *all* baseline (v0) disagreements into consensus failure modes and emit *one atomic single-rule* variant per mode, batch-validated in parallel. Because each edit is a single named rule anchored to one failure mode and validated on its own, every gain *decomposes* to a specific rule a practitioner can read, audit, and selectively adopt.
(3) **Finding: one structured pass beat iterative search.** On two structurally different tasks the cluster loop is the *only robustly-positive method*—Vertical Candidate held-out +6.00 vs. the linear loop's flat +0.67, Vertical Quality +41.67 vs. +9.52 (Table 5)—beating both GEPA's 10–20 reflective-mutation rounds and the ∼3–4 human-debugging iterations of the linear loop. The lever is not iteration depth but aggregating the *full* error set once and selecting on a noise-aware multi-seed bootstrap.

2 Related Work

Two adjacent lines bracket this work. *Task-accuracy prompt search* optimizes prompts so a downstream task is solved more accurately: DSPy/MIPRO [3, 7], the reflective-mutation optimizer GEPA [1],

Table 1: Positioning relative to adjacent lines of work. Our contribution targets human–AI agreement (like the LLM-as-judge line) but *iterates the prompt* via structured failure-mode discovery and atomic, individually-audited edits.

Family	Method	Target	Discovery	Edit unit
Task-accuracy prompt search	DSPy/MIPRO [3, 7], GEPA [1], ProTeGi [8]	answer correctness	gradient-free search / reflective mutation / textual gradients	whole instruction body per component
LLM-as-judge analysis	G-Eval [5], Prometheus [4], JudgeLM [11]	judge–human agreement	fixed prompt / supervised fine-tune	n/a
Qualitative error analysis	Qualitative-judge [2], QualEval [6]	report fidelity (generator errors)	per-instance analysis → cumulative clustering	n/a (report; ungated fixes)
Evaluator criteria alignment	EvalGen [9]	judge–human agreement	generate-then-filter candidate assertions	assertion selection (no text edit)
Iterative rubric opt. (baseline)	linear loop	judge–human agreement	top-K worst → bundled patch	bundled per iteration
Iterative rubric opt. (ours)	**cluster + batch-validate**	**judge–human agreement**	**clustering + multi-seed proposer**	**atomic, one rule per cluster**

and ProTeGi/APO's textual gradients [8]. Their target is answer correctness, and candidate scoring—even when economized, as in ProTeGi—presumes task labels that can be re-queried at will. *LLM-as-judge analysis* shares our target, agreement between an automated judge and humans, but typically holds the scoring prompt fixed and studies or fine-tunes the model: G-Eval [5], Prometheus [4], JudgeLM [11], and the broader MT-Bench line [10].

Two further lines are closer in mechanism. Closest to our discovery step, *LLM-as-a-qualitative-judge* [2] produces per-instance error descriptions and clusters them into a named issue-type report via a sequential cumulative algorithm, and QualEval [6] derives and quantifies attribute categories over model outputs. Both cluster *task errors of the system under evaluation* and terminate in a report; improvements driven by the report are ungated—manual pipeline revision in Chirkova et al.'s case study, prompt feedback without statistical acceptance testing in QualEval. We differ on three points. *What* is clustered: judge–human disagreements, not generator errors. *How*: parallel multi-seed whole-pool clustering with an $(N-1)$-of-N co-membership consensus filter, rather than an order-dependent single pass. *What for*: each consensus family emits one atomic rubric rule that survives only through pre-registered statistical gates and a single-shot held-out test.

EvalGen [9] is closest in framing: it aligns LLM-assisted evaluators with human grades under a limited grading budget, generating candidate binary assertions per criterion and selecting those with the highest alignment (harmonic mean of coverage and false-failure rate) on the graded sample. It differs on three points. *Operation*: generate-then-filter assembly of new assertions, rather than disagreement-driven repair of an existing ordinal rubric. *Discovery*: no failure-mode analysis of *why* the evaluator disagrees. *Selection*: alignment scored on the graded sample alone. That selection protocol—a single point estimate on the selection sample, with no resampling or held-out verification—is exactly the one whose unreliability at $n \approx 50$ we quantify in §4.5. Its central qualitative finding, *criteria drift*, independently motivates our setting: scoring criteria must be revised in response to observed outputs, which our loop operationalizes under pre-registered statistical gates.

We sit at the intersection of these lines (Table 1): like the judge and criteria-alignment lines we optimize for human agreement, like the prompt-search line we iterate the prompt text, and like the qualitative error-analysis line we cluster errors—but the clustering primitive is shared lineage, and the contribution here is the closed, budget-constrained calibration loop built around it: *structured failure-mode discovery* yielding *atomic, individually-validated* edits.

3 Methods

Terminology. We fix the following terms, used throughout. A *rubric* is the scoring prompt given to the evaluator; the hand-written starting rubric is the *baseline* (v0). The *evaluator* is the LLM under test that applies the rubric to a query and emits a score; the *proposer* is a separate LLM that suggests rubric edits. A *disagreement* is a query where the evaluator's score differs from both human raters. The *composite* is our agreement metric (§4.2). We partition queries into a *train* split (drives edits), a *validation* split (chooses among accepted rubrics), and a never-touched *held-out* split (the single-shot final test). An *atomic rule* is one rubric edit; a *bundled patch* is several edits applied together.

All three methods take the same input (a baseline rubric, a fixed train split, dual human labels) and produce the same kind of output (a winning rubric, evaluated once on the held-out split). They differ along three axes, which we use to describe each method below: how disagreements are *aggregated* (reflective minibatch / top-K list / clustered consensus), what *edit unit* is produced (whole instruction-body per component / bundled patch / atomic single-rule per cluster), and how candidates are *selected* (minibatch strict-improvement / single train Δ / noise-aware bootstrap). Figure 1 contrasts the three flows.

3.1 GEPA

GEPA [1] is a fully-automated reflective prompt evolver, which we run as a strong baseline. It maintains a *population* of candidate rubrics and evolves it over 10–20 rounds:

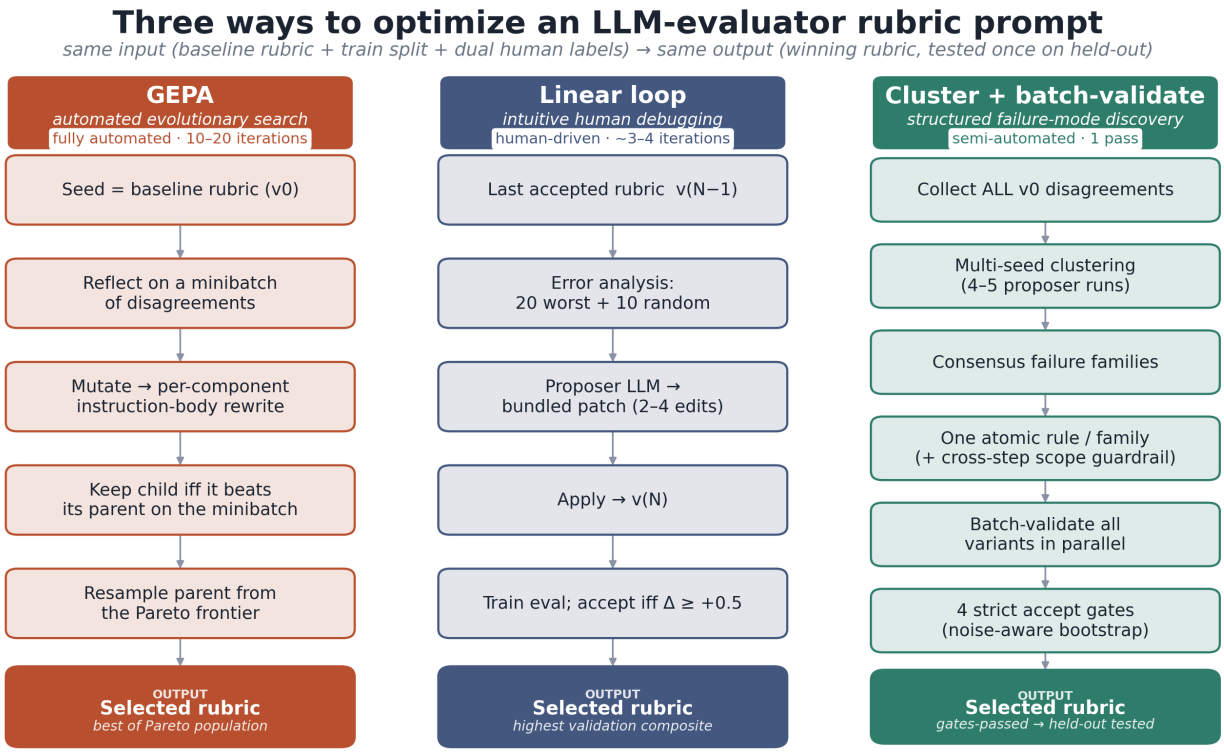

Figure 1: The three methods across the design axes of §3: aggregation, edit unit, selection. Held-out results in Table 5.

(1) **Initialization.** The population starts as {v0}, evaluated on the train split.
(2) **Parent selection.** Sample a parent from the *Pareto frontier*—the candidates that are best on at least one training instance—which preserves diversity rather than collapsing onto a single leader.
(3) **Reflective mutation.** A round-robin selector picks one program component; a reflection LLM reads a minibatch of the parent's scored disagreements (query, score, reasoning trace) and rewrites that component's *entire instruction body*. Our rubric is a single-component program, so each mutation rewrites the whole rubric.
(4) **Minibatch accept gate.** The child is scored on the same minibatch and admitted only if it strictly beats its parent there.
(5) **Full evaluation and frontier update.** Admitted children are scored on the full train split, and the Pareto frontier is updated.
(6) **Final candidate.** After the round budget, the best-scoring candidate is returned as the final rubric.

On our three axes: aggregation = reflective minibatch sampling, edit unit = *whole instruction-body per component* (not an atomic edit), selection = minibatch strict-improvement + Pareto. One structural caveat: GEPA's per-component granularity is unavailable on a single-component rubric, so its edit unit is maximally coarse in our setting—a disadvantage we report rather than correct. The one control that matters for a fair comparison is the evaluator: GEPA must optimize against the *same* judge as the other methods (we show in §4.5 that a near-floor task LM yields no signal at all).

3.2 Linear loop (intuitive baseline)

This is the natural way a human debugs a rubric, formalized. Per iteration N, the loop runs six steps:

(1) **Error analysis.** From the most recent accepted rubric v_{N-1}, take the top-K worst-disagreement train cases (query + AI score + dual human scores + AI reasoning trace) plus a small random in-distribution sample as a regression guard ($K = 30$: 20 worst + 10 random).
(2) **Proposer call.** A proposer LLM (Claude Sonnet 4.6) reads v_{N-1}, the error slice, the running leaderboard, and a persistent failure-memory store, and returns a *bundled patch*—typically two to four edits applied together.
(3) **Patch application.** A human reviews and approves each proposed patch before it is applied—the only human step in the loop besides the labels themselves. Approved edits produce v_N with a per-version changelog.
(4) **Train evaluation.** v_N is run by the evaluator on the full train split (~50 on Vertical Candidate, ~75 on Vertical Quality), in parallel.

(5) **Accept gate.** Accept iff train Δ vs. $v_{N-1} \geq +0.5$; otherwise reject and record the failure (with a diagnosis) for the next iteration.
(6) **Validation pass.** After all iterations, run a single validation pass per accepted version; the winner has the highest validation composite.

The linear loop's final rubric (*v4*) emerged from four attempted iterations on Vertical Candidate, two accepted and two rejected; on Vertical Quality, three accepted iterations. Two failure modes this loop surfaced—*cross-step bleed* (an edit to one rubric step silently shifting an off-target step) and *bundled-edit attribution opacity* (a four-edit patch whose gain cannot be attributed to any single edit)—motivate the alternative.

3.3 Cluster + batch-validate loop (proposed)

The cluster loop runs once end-to-end:

(1) **Disagreement pool.** Collect *every* v0 train disagreement. If the train pool is small, augment with additional captured queries sharing the same evaluator and rubric, leaving held-out untouched.
(2) **Multi-seed clustering.** Issue four to five parallel proposer calls with two input reshuffles and two to three prompt reframings (axis = error semantics, candidate position, rubric step). Each call returns 3–5 named failure-mode clusters, each with claimed-fix queries and a draft atomic rule.
(3) **Consensus families.** Compute pairwise co-membership across runs. A query pair in the same cluster in $(N-1)$ of N runs forms a stable consensus family; single-run-only members are discarded as proposer noise. Both datasets yielded 4–5 families with no noise-only assignments.
(4) **One atomic variant per family.** Build a candidate rubric = v0 + one rule (anchored at the correct step) + a one-line cross-step scope guardrail restricting the rule to its target step (so an edit cannot silently shift an off-target step).
(5) **Batch-validate.** Run every variant on train in parallel (∼5 wall-clock minutes per variant). Compute per-variant composite Δ, per-dimension bleed deltas, and a bootstrap distribution of Δ from 200 random sub-samples of the train slice.
(6) **Strict accept gates.** Accept iff *all four* gates of Table 2 hold. Canaries are train queries v0 already scores correctly, retained as a regression check; all gate thresholds were fixed before any variant was validated.
(7) **Held-out test of winners only.** Held-out is untouched until all four gates pass; it is a single-shot test, never a selection signal.

Why LLM-based clustering. We cluster with an LLM proposer rather than embeddings over error descriptions because each cluster arrives with the artifacts the next step consumes (a draft rule and its claimed-fix queries). Evidence that LLM-produced per-instance issue descriptions track human annotations [2] supports the choice, though we have not run a head-to-head against embedding-based clustering (see Future Work).

If multiple variants pass, we compose the value-positive, non-overlapping subset. In this study one variant per dataset passed all gates: the consensus-family variants we label **C1** (Vertical Candidate) and **F3** (Vertical Quality); all later results refer to these. Composition was therefore not invoked (§6).

Table 2: The four acceptance gates (step 6). A variant is accepted only if it passes *all four*; every threshold was fixed before any variant was validated.

gate	criterion	failure mode caught
(a) train gain	Δ vs. v0 ≥ +0.5	no-op rules
(b) bootstrap	$P(\Delta{>}0) \geq 95\%$ (200 sub-samples)	gains that are sampling noise
(c) off-target guard	every off-target dimension within ±2 pp	cross-step bleed
(d) canary	≥1 v0-correct canary stays correct	regressions on already-correct queries

What the gates do and do not check. The gates bound two conflict surfaces: gate (c) catches *cross-dimension* conflicts (an edit silently shifting an off-target step), and gate (d) catches same-dimension regressions on queries the baseline already scored correctly—but only within canary coverage. One risk is explicitly *not* checked: a new rule may contradict existing baseline text *on the same dimension* over inputs the train pool under-represents (e.g., a baseline rule correct on 90% of inputs, patched by a cluster rule for the failing 10%, with noisy human labels blurring the boundary between the two regimes). Atomicity and step-anchoring shrink this surface, and because each accepted edit is a single named rule, a conflict discovered later can be reverted alone; a systematic rule-interaction check (e.g., pairwise canaries over rule-overlap inputs) remains future work (§6).

Rule lifecycle in deployment. The unit of change is the named rule: every accepted variant is a versioned rubric with a per-version changelog, the gates decide acceptance without human intervention, and a human audits the named rules after the fact. Because gains decompose per rule, rollback is per-rule rather than per-version: a rule that later proves harmful is removed without disturbing the rest of the rubric. The append-only failure memory records rejected proposals with diagnoses so they are not re-proposed. Interactions between a previously accepted rule and a newly proposed one are covered only by re-running the gates at proposal time (each new candidate is validated on top of the current rubric); adversarial pairwise interactions beyond that are the open problem noted above.

4 Experiment

We now instantiate all three methods on two production evaluator tasks, holding the evaluator, splits, and metric fixed.

4.1 Datasets and raters

Both datasets are quality-audit batches of real search-engine user queries from a single live surface of a commercial search vertical (a non-English market), spanning head/torso/tail queries over entity, property, list, and empty-tail intents. The unit of evaluation is the rank-1 candidate card returned by the production retrieval pipeline. As a running example: for a person-name query such as "*[actor name] profile*," the pipeline returns a person-profile card at rank

1, and the evaluator must judge how well that card answers the query—the same judgment the human raters make.

Two trained analysts (H1, H2) independently labeled every query blind to the AI score; they differ systematically in calibration—H1 is strict (top score on 59% of queries), H2 lenient (82%), which surfaces different evaluator failure modes. H1 and H2 agreed exactly on 76% of in-scope queries and were within one point on 97%; we treat them as two independent ground-truth sources rather than averaging, since there is no third rater to break ties.

4.2 Tasks, splits, and metric

Vertical Candidate (*Cand.*) scores a single 0/1/2 on the rank-1 card. Vertical Quality (*Qual.*) is scored by a four-step rubric (S1→S4); we target the final step, S4 (*vertical quality*), an ordinal {*Wrong*, *Partial*, *Good*} judgment of the card. The upstream steps gate card eligibility and exposure; they enter the study only through eligibility filtering (see the *N/A-folding* normalization below) and the off-target regression guards (§3.3). Table 3 gives the splits and pools. Two details the table cannot show: Vertical Candidate's 50 *unused* queries are not idle—they are pooled into the multi-seed validation shuffles, so all 300 dual-rated queries inform model selection; and Vertical Quality's held-out is 149 usable queries after one was dropped at capture time (we use 149 throughout). Vertical Candidate's held-out set is the fixed second half of the pool (ids 251–400), disjoint from all training.

Metric: unit and computation. For each query the evaluator emits one ordinal score (Vertical Candidate: 0/1/2 on the rank-1 card; Vertical Quality: S4 mapped *Wrong*/*Partial*/*Good* → 0/1/2). We compare it against *each* of the two human raters separately, so every query contributes two (AI, human) pairs. Across all pairs we compute exact-match accuracy exact% $\in [0, 100]$ and the mean absolute ordinal error MAE $\in [0, 2]$, and combine them into the *composite*

$$\text{composite} \;=\; \text{exact\%} \;-\; 0.5 \times \text{MAE} \times 100,$$

expressed in **percentage points** on a $[-100, 100]$ scale (higher is better; +100 = exact agreement with both raters). Every number we report—in Tables 4 and 5—is this composite, either as an *absolute* value or as a *gain* $\Delta = \text{composite}_{\text{method}} - \text{composite}_{\text{v0}}$ on the same set; e.g. Vertical Quality F3's held-out 75.00 vs. v0's 33.33 is reported as $\Delta = +41.67$.

We prefer this composite to raw exact-match accuracy or Cohen's κ because the scores are *ordinal* (*Wrong*<*Partial*<*Good*): exact-match ignores how far off a miss is, and κ corrects for chance but discards ordinal distance, whereas the MAE term penalizes a two-step miss more than a one-step miss while the exact% term still rewards perfect agreement. The method ranking is not an artifact of this choice: under quadratic-weighted κ, Spearman ρ, and composite weights 0.25–0.75, the cluster winner remains first on both tasks (e.g., Vertical Quality held-out QWK: v0 0.41, linear 0.47, GEPA 0.51, F3 0.83).

For Vertical Quality we apply an *N/A-folding* normalization: human cases labeled S1=Y + S4=N/A are folded to *Wrong*, mirroring the AI's *Missing*→*Wrong* fold, so both sides use the same three-class S4 space. A query contributes an S4 pair only when *both* sides emit a scorable value; queries that are not card-eligible (S1=N, hence human S4=N/A) drop out entirely, which shrinks the effective sample far below the nominal query count on Vertical Quality (see Limitations). On Vertical Candidate every query is scored; when the frozen capture contains no rank-1 candidate the scorer assigns 0.

4.3 Agents, evaluator, and proposer

Three agents, three roles (version-pinned, 2026-05). The pipeline assigns three distinct agents to three roles. (1) A *capture agent*, Claude Code (Anthropic), fetches each query's vertical card once and freezes it to a static artifact that every later step reuses. Capture runs as its own agent because the Codex sandbox cannot reach the internal capture endpoints, so the data-capture and judging models are deliberately separate. (2) The *evaluator* (the judge under test) is Codex (OpenAI, gpt-5.4)[1]. (3) The *proposer* is Claude Sonnet 4.6. Because the capture artifacts are frozen, every evaluation is a pure function of (rubric version, captured card) and is reused across re-scorings.

For each query the evaluator receives the query text, the rank-1 vertical card, and the current rubric, and emits a final score, a reasoning trace, and the rubric's intermediate-step outputs. It is called 10-way parallel; one full pass takes 5–10 wall-clock minutes (~US$1–3). The proposer makes one non-interactive call per iteration, with deterministic input: the current rubric, the error/disagreement slice, the leaderboard, and an append-only failure-memory store of rejected proposals with one-line diagnoses.

End-to-end, one cluster-loop run (~US$25 for 4–5 variant passes + proposer calls) is comparable in evaluator spend to the linear loop's full run (whose per-iteration cost includes a train pass plus per-accepted-version validation passes), far below GEPA's 10–20 rounds, and yields a stronger held-out prompt on both tasks.

4.4 Validation results

Table 4 reports validation composites for the methods that use the validation split as their model-selection signal; GEPA is absent by design, since it selects internally (minibatch strict-improvement + Pareto, §3.1) and its final candidate goes directly to the held-out test. On Vertical Candidate, **single-seed validation is actively misleading**: it inflated the linear loop to +10.00 (an extreme sampling artifact) and gave C1 a *negative* −1.96 on one seed but +2.22 on another. We therefore report the multi-seed mean over 8 stratified val shuffles, which ranks the methods exactly as the held-out test does: C1 is robustly positive (+7.75±4.53, $P(\Delta > 0) = 100\%$) while the linear loop is weak (+1.25±3.06, $P = 75\%$). This is the empirical case for replacing single-seed validation with multi-seed selection.

On Vertical Quality, a post-hoc multi-seed re-validation (8 stratified 75-query shuffles of the train+val pool, same evaluation window) confirms the single-seed selection: linear +32.9±17.3, F3 +46.8±9.7 vs. v0, both $P(\Delta > 0) = 100\%$, with F3 ahead in 7 of 8 shuffles. Only 28–32 pooled pairs are scorable per shuffle, so per-seed estimates are coarse; Table 4 reports the seed-42 absolute composites.

[1]The evaluator and the proposer are proprietary, version-pinned API models accessed via their vendor CLIs in 2026-05; no model paper exists to cite, so we record the exact version strings for reproducibility. Both are called with vendor-default decoding parameters (no temperature or sampling overrides).

Table 3: The two evaluator tasks, both drawn from the same commercial search vertical and dual-rated by two human experts. "Effective held-out pairs" counts the (AI, human) pairs that enter the composite after S4-scorability filtering (36–42 scorable pairs on Vertical Quality; see Limitations).

	Vertical Candidate	Vertical Quality
Unit scored	rank-1 vertical card	the query's vertical card
Target label	ordinal 0/1/2	S4 ordinal {Wrong/Partial/Good}
Rubric structure	single-step	4-step (S1→S2→S3→**S4**)
Human raters	2 (dual-rated)	2 (dual-rated)
Dual-rated queries	300	300
Train / val / held-out	50 / 50 / 150	75 / 75 / 150 (149 usable)
Effective held-out pairs (pooled, 2 raters)	300 (23 queries imputed 0)	36–42 (S4-scorable)
Extra pool	50 unused (→ multi-seed val pool)	+450 single-rated (cluster step only)
Normalization	—	N/A-fold (S1=Y, S4=N/A → Wrong)

Table 4: Validation *composite* (percentage points, §4.2; absolute, with Δ vs. v0 in parentheses). Vertical Candidate rows are multi-seed means over 8 stratified val shuffles; Vertical Quality shows seed-42 absolute composites, and a post-hoc multi-seed re-validation (text) confirms the same ranking.

dataset	v0	linear (v4)	cluster+batch
Cand. (multi-seed val)	52.06	53.31 (+1.25)	C1: **59.81** (+7.75)
Qual. (seed-42 val)	28.12	61.76 (+33.64)	F3: **70.00** (+41.88)

Table 5: Single-shot held-out results: absolute *composite* (percentage points, §4.2) and gain Δ over the v0 baseline, with 95% bootstrap CIs (200 sub-samples). Vertical Candidate uses all 150 queries (300 pooled pairs); Vertical Quality rests on 36–42 scorable pairs (see Limitations), hence its wide CIs.

dataset	method	comp. (Δ vs v0)	95% CI	$P>0$
Cand.	v0	40.33	—	—
Cand.	linear	41.00 (+0.67)	[−2.4, +4.6]	55%
Cand.	GEPA	42.00 (+1.67)	[−2.9, +6.4]	79%
Cand.	**cluster (C1)**	**46.33** (+6.00)	[+3.0, +11.5]	**100%**
Qual.	v0	33.33	—	—
Qual.	linear	42.86 (+9.52)	[−6.2, +30.8]	84%
Qual.	GEPA	50.00 (+16.67)	[−1.0, +33.3]	96%
Qual.	**cluster (F3)**	**75.00** (+41.67)	[+16.7, +70.0]	**100%**

4.5 Held-out results

Held-out is never read during proposing or selection; numbers below are single-shot evaluations on the reserved 150 (Vertical Candidate) / 149 (Vertical Quality) queries, with 95% bootstrap CIs over 200 sub-samples.

The cluster-loop final is the only robustly-positive method on *both* tasks ($P(\Delta > 0) = 100\%$). It beats the linear final by +5.33 absolute on Vertical Candidate, where linear is flat (+0.67, $P = 55\%$). On Vertical Quality it wins by +32.14 absolute: F3 > the linear final (v4) in 200/200 sub-samples, and on the small scorable subset of Table 3 this gap corresponds to a handful of query-level flips. F3 raised v0's held-out composite from 33.33 to 75.00 using a single narrowing rule: *"AI briefing panels, generic web blocks, third-party community wikis, and other unofficial information blocks are NOT independent vertical cards; only official structured entity boxes (person profile, securities card, movie/TV card, etc.) clearly visible on screen count as vertical cards."* The Vertical Quality v0 baseline over-credited non-vertical elements as vertical cards; F3 eliminates that with one rule at the correct step. The gain is thus concentrated: it repairs one definitional flaw of the baseline, and Limitations discusses how that bounds its generality.

Table 6: Metric-robustness check on the Vertical Quality held-out set: *binary* agreement (S4 collapsed to *Good* vs. rest), computed from the same single-shot evaluations as Table 5. *n* = pooled (AI, human) pairs over both raters; P/R = precision/recall of the *Good* class; bootstrap as in Table 5.

method	acc. (Δ vs v0)	95% CI	$P>0$	P/R (*Good*)	*n*
v0	72.2	—	—	75/82	36
linear	81.0 (+8.7)	[+1.3, +19.3]	98%	85/85	42
GEPA	76.2 (+4.0)	[−4.3, +13.1]	78%	86/80	42
cluster (F3)	**90.0** (+17.8)	[+3.3, +31.7]	**100%**	**100/83**	40

As a metric-robustness check, Table 6 re-scores the same held-out evaluations on a binary collapse of S4 (*Good* vs. rest), which removes the composite's MAE term entirely: the ranking is preserved and F3's margin sharpens (+17.8 pp, no false-*Good* predictions). Given the small scorable subset, we read this as corroboration of the composite result rather than an independent claim.

Automated baseline (GEPA).. Run against the *same* Codex evaluator on Vertical Quality, the automated reflective optimizer GEPA [1] reaches held-out +16.67 ($P(\Delta > 0) = 96\%$): it beats the linear loop (+9.52) and is in fact more robust than it, but remains ∼2.5× below the cluster loop. Tellingly, GEPA earned its S4 gain by rewriting the whole rubric and shifting upstream steps (S1 eligibility +31 pp), whereas the cluster loop added a single step-anchored rule under a ±2 pp off-target guardrail—competitive but blunt automation versus surgical, audited structure. Optimizing the same rubric against a near-floor task LM (gpt-4.1-mini) instead yields +0.00: the evaluator must be strong before *any* search finds signal, so a fair automated baseline requires the production judge.

On Vertical Candidate, by contrast, GEPA's gain is *not* robust (+1.67, $P(\Delta > 0) = 79\%$), and neither is the linear loop's (+0.67, $P = 55\%$); only the cluster loop is robustly positive on *both* tasks ($P = 100\%$). Automated search is task-dependent here; structured failure-mode discovery is not.

4.6 Sanity checks

Cross-step bleed: all proposed winners include the scope guardrail; off-target deltas vs. v0 are within ±2 pp, and v0-correct canaries remain correct. **Multi-seed (Vertical Candidate):** the multi-seed ranking of §4.4 matches the fixed held-out basis (C1 +6.00 on ids 251–400).

5 Limitations

These results come with deliberate scope and protocol caveats.

External validity. The evidence base is two tasks from a single search vertical, scored by one evaluator (Codex, gpt-5.4); external validity to other domains and to other evaluator models is untested (we list cross-evaluator transfer as future work). The two tasks also share a capture/evaluation pipeline and an overlapping two-analyst rater pool, so they are not fully independent replications, and "ground truth" here is two-rater agreement rather than a large panel.

Effective sample size. Held-out sets are small, and on Vertical Quality the *effective* sample is far smaller than the nominal one: of the 298 pooled (AI, human) pairs its 149 held-out queries could contribute (two raters per query), only 56 survive human-side S4-scorability filtering (most queries are not card-eligible), and requiring a scorable AI output too leaves the 36–42 pooled pairs behind every reported composite (Tables 3 and 6). Because S4-scorability also depends on the AI's own upstream output, the scorable pair set varies slightly across methods; each composite is computed over its own scorable pairs. Restricting to the 29 held-out queries scorable under all four versions preserves the ranking (F3 70.6 vs. GEPA 47.1, v0 38.2, linear 29.4); in a paired query-level comparison F3 beats the linear final on 4 queries and loses none (13 ties).

Imputation. On Vertical Candidate all 150 queries are scored, but 23 of them carry an imputed 0 because the frozen capture contains no rank-1 candidate; the same 23 queries are imputed for every method, so the imputation cancels in all Δs while deflating absolute composites uniformly.

Precision and concentration. The direction of every reported gain is robust ($P(\Delta > 0) = 100\%$ for both cluster winners), but the point estimates are imprecise—the Vertical Quality +41.67 carries a 95% CI of [+16.7, +70.0] and corresponds to a handful of query-level flips. That gain also concentrates in a single definitional failure mode of the baseline (vertical-card recognition); whether the method delivers comparable gains on baselines without one dominant, easily-articulated flaw is untested—Vertical Candidate's +6.00 is likely the more typical magnitude.

Protocol caveats. The cluster loop differs from the linear baseline on three axes at once (full-pool aggregation, atomic single-rule edits, multi-seed selection); we have not run component ablations, so the gain cannot be attributed among the three. Clustering used a single proposer model (Claude Sonnet 4.6); cluster stability across proposer models is unverified. The acceptance gates also do not test for contradictions between an accepted rule and existing baseline text on the same dimension over inputs the train pool under-represents (§3.3); canaries mitigate but do not close this gap. Our multi-seed *validation* resamples the query set with the prompt held fixed—it is not k-fold cross-validation (no per-seed retraining), and its slices overlap each method's training set by ∼10 queries, so it gauges metric stability rather than generalization; only the held-out set (ids 251–400, §4.2) supports the generalization claim.

Finally, the Vertical Candidate held-out was captured ∼18 days after annotation and re-captured after a transient outage, a possible distribution-shift confound we disclose rather than correct.

Ethics and normative-target considerations. Optimizing agreement with two raters who differ systematically in calibration (H1 strict, H2 lenient) sidesteps choosing between them; a deployment that adopts a single rater's calibration as its target makes a normative choice—what counts as a *Good* result—with user-facing consequences, so rubric edits should be auditable normatively as well as technically. The evaluation artifacts contain queries and rendered result cards only, with no user identifiers. Fairness, privacy, and misuse-resistance evaluation of the underlying search system is out of scope for this study.

6 Future Work

- **Interaction-aware composition.** When multiple variants pass, the current "compose the non-overlapping positive subset" rule ignores cross-edit interactions; an exploratory hypothesis-guided composition run outside the main protocol, given 10× more training data, still reached only +3.51 on the same held-out set—a cautionary case. Validating each candidate bundle on a held-out-like inner slice before promoting would close this gap.
- **Rule-interaction checks.** The gates validate each atomic rule against the baseline and off-target dimensions, but not rule-pair interactions on overlapping inputs (§3.3); pairwise canary suites over rule-overlap strata would close the contradictory-rule gap.
- **Feature-aware clustering.** Adding structural features (candidate position, query length, match-type code) as explicit cluster axes would surface failure modes a semantic-only reading misses; so would a head-to-head comparison of LLM-consensus clustering against embedding-based alternatives (k-means/HDBSCAN over error-description embeddings) on the same disagreement pool.
- **Cross-evaluator transfer.** Replicating C1/F3 on Claude or Gemini would test whether the discovered rules encode generalizable human-alignment heuristics or evaluator-specific tweaks.
- **Agentic-AI evaluation.** Porting the loop to judges that score agent traces and tool-use trajectories: clustering disagreements over trajectory-step judgments, with atomic rules anchored to individual decision points (tool choice, grounding, termination), is a direct analogue of step-anchored rubric rules.

References

[1] Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. 2026. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. In *International Conference on Learning Representations (ICLR).* Oral; arXiv:2507.19457.

[2] Nadezhda Chirkova, Tunde Oluwaseyi Ajayi, Seth Aycock, Zain Muhammad Mujahid, Vladana Perlić, Ekaterina Borisova, and Markarit Vartampetian. 2025. LLM-as-a-qualitative-judge: automating error analysis in natural language generation. *arXiv preprint arXiv:2506.09147* (2025).

[3] 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. 2024. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. In *International Conference on Learning Representations (ICLR)*. arXiv:2310.03714.

[4] Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2024. Prometheus: Inducing Fine-grained Evaluation Capability in Language Models. In *International Conference on Learning Representations (ICLR)*. arXiv:2310.08491.

[5] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. arXiv:2303.16634.

[6] Vishvak Murahari, Ameet Deshpande, Peter Clark, Tanmay Rajpurohit, Ashish Sabharwal, Karthik Narasimhan, and Ashwin Kalyan. 2024. QualEval: Qualitative Evaluation for Model Improvement. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. arXiv:2311.02807.

[7] Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. 2024. Optimizing Instructions and Demonstrations for Multi-Stage Language Model Programs. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. arXiv:2406.11695.

[8] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. arXiv:2305.03495.

[9] Shreya Shankar, J.D. Zamfirescu-Pereira, Björn Hartmann, Aditya G. Parameswaran, and Ian Arawjo. 2024. Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST)*. arXiv:2404.12272.

[10] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*. arXiv:2306.05685.

[11] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2023. JudgeLM: Fine-tuned Large Language Models are Scalable Judges. *arXiv preprint arXiv:2310.17631* (2023).