DECOMPOSED PROMPT OPTIMIZATION FOR RELIABLE LLM-AS-JUDGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative LLM-as-judge systems for safety guardrails, reward models, automatic evaluators, and niche domains are typically configured by a natural-language prompt - a flexible lever for making a weak judge stronger without modifying its weights. We study how far this lever goes for small frozen judges, and find that unconstrained prompt optimization i.e. rewriting the prompt as a single free-text block, buys surprisingly little. The missing ingredient in here is *decomposition*: a judge prompt splits into two functionally distinct components: *criterion* (what condition to detect), and *scoring schema* (how to map a judgment to a decision), and making this decomposition *explicit* to the optimizer is what makes automatic prompt optimization pay off. Across four judge models (8B–31B) and six tasks we find that (i) reflective optimization improves every judge, most where it is weakest (up to $+0.44$ F1); (ii) treating the prompt as two explicit components beats optimizing it as one monolithic block on every judge (up to $+0.13$ F1); and (iii) *which* component to search and in what order, is the most effective way to *spend* a compute budget: phasing converts extra search into the largest gains on the hardest structured tasks (up to $0.58 \rightarrow 0.96$ F1) once initial search saturates. Applying this recipe to a real-world financial-compliance policy that fixed-taxonomy guards cannot express, a frozen 8B judge, optimized only through its prompt, improves F1 from 0.70 to 0.92 while baseline guards stall at 0.67–0.71.

1 INTRODUCTION

Lately, using large language models (LLMs) to score another model's response, as a safety guardrail, a reward model, or an automatic evaluator, has become a standard system feature (Zheng et al., 2023; Gu et al., 2024; Inan et al., 2023; Padhi et al., 2024), and small general LLMs are attractive for the task: fast, readily available, and reconfigurable on the fly through a natural-language prompt. In many practical use cases these judges are *inline* components (a guardrail labels every request (Rebedea et al., 2023), a reward model every rollout), so the judge's cost and latency are multiplied by the traffic of the system it guards. Owing to this sheer volume, model size becomes a first-order constraint, and a small frozen judge steered entirely through its prompt is the natural fit: one model can be retargeted to a new policy by merely editing text rather than through expensive retraining (Belcak et al., 2025). Out of the box, though, these LLMs make unreliable judges: they frequently fail to emit even a parseable verdict (Geng et al., 2025), and merely scaling within a model family often does not fix the issue, consistent with evidence that even frontier judges falter on harder tasks (Tan et al., 2025) and that scaling a judge's capability faces fundamental limits (Dorner et al., 2025). Since a judge is a frozen model with a configurable prompt, the prompt, not the weights, is the lever for making a weak judge stronger. Yet how to write it is largely left to trial and error, and even automatic optimizers fall short, treating it as a single opaque block of text to be rewritten.

In this article we argue that naive rewriting is the wrong operation, because the judge prompt has latent *structure*. It decomposes into a *criterion* c (what condition to detect) and a *scoring schema* s (how to map a judgment onto a decision). Making this decomposition explicit to the optimizer, rather than searching one undifferentiated free-text block, is what makes reflective optimization reliably pay off (Figure 1). We test this by optimizing four different models *as the judge* (8B–31B, two families), under regimes that vary only in how the prompt is decomposed (Table 1): reflective optimization improves every judge, most where it is weakest, and treating the prompt as two explicit components beats optimizing it as one monolithic block.

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.

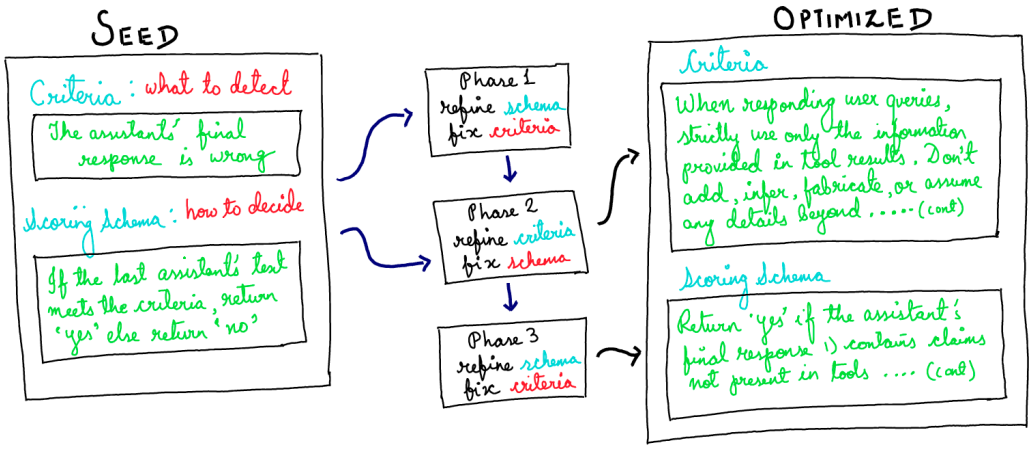

Figure 1: The judge prompt has two components that optimization refines. A judge is steered entirely by its prompt, which splits into a criterion and a scoring schema. The run show is one of the regimes we study: a three phase schedule rewriting one component at a time, while the other held fixed.

In the paper, our contributions are as follows:

- We decompose the LLM-judge prompt into two semantic components: a criterion (what condition to detect) and a scoring schema (how to map a judgment onto a decision), and define a taxonomy of optimization regimes over them: criterion-only, schema-only, combined (one free-text block), and explicit two-component search.
- Across four judges and six tasks, reflective optimization improves every judge, most where it is weakest and explicit decomposition beats monolithic free-text optimization on every judge with headroom. This underlines that the lever is prompt's structure, not model or search schedule.
- As cheaper single-shot search saturates, we use phasing to trade a compute surplus for accuracy, delivering the largest gains on the hardest structured tasks (agentic-trajectory safety $0.58 \rightarrow 0.96$).
- As structure, not scale, is the lever, a single specialized 8B judge, optimized only through its decomposed prompt, beats GEPA-optimized general models of its own generation on $11/12$ tasks.
- On a real adversarial BFSI compliance benchmark, we show that a 8B judge, optimized only through its prompt, reaches 0.92 F1 where deployed safety guardrails linger around 0.67–0.71.

2 Related Work

Using an LLM to score another model's response has actively been studied with respect to evaluation, reward modeling and safety guardrails (Zheng et al., 2023; Malik et al., 2025; Inan et al., 2023; Padhi et al., 2024). Such judges are often unreliable in ways that even scaling does not help. They often carry position, verbosity, and self-preference biases (Wang et al., 2024; Zeng et al., 2024b), and on other hard tasks they collapse even for frontier models (Tan et al., 2025). When verification does help, the gain comes from letting the verifier reason and not from making it larger (Zhang et al., 2024a). And scaling a judge's data or capability instead runs into fundamental statistical limits (Dorner et al., 2025). This is what brings attention to the prompt rather than the weights. Prompting has a ceiling of its own, though: it *surfaces* capability the pretrained model already holds rather than installing new knowledge (Zhou et al., 2023a), and prompt-based self-improvement cannot cross that boundary (Huang et al., 2024), so an efficient prompt and a strong model are complementary in nature, a distinction we make precise in Section 3. As the tasks in our experiments span from reward modeling (Malik et al., 2025) through grounding and hallucination (Niu et al., 2024) to instruction and scope following (Zhou et al., 2023b), and because one decomposed judge serves both pointwise and pairwise decisions, the question we take up is not which model sits behind the prompt but how that prompt is *structured and optimized*.

Table 1: Cross-judge study: held-out test F1, averaged over six tasks (five seeds each), for four judge models under each optimization regime. The strongest regime for each judge is in **bold**; Δ_{opt} is that best minus zero-shot; $\Delta_{\text{j-c}}$ is joint minus combined, positive for every judge, so explicit decomposition beats the monolithic free-text block regardless of the judge.

Judge	zero-shot	criteria-only	schema-only	combined	joint	Δ_{opt}	$\Delta_{\text{j-c}}$
`granite-4.1-8b`	0.144	**0.583**	0.319	0.388	0.516	+0.438	+0.128
`granite-4.1-30b`	0.425	**0.625**	0.581	0.574	0.617	+0.200	+0.043
`gemma-4-31b`	0.705	**0.839**	0.826	0.836	**0.839**	+0.135	+0.003
`granite-guardian-4.1-8b`	0.720	0.726	0.718	0.729	**0.733**	+0.013	+0.004

Optimizing a prompt for a fixed model, while keeping its weights untouched, has lately been quite popular. APE (Zhou et al., 2023c) and OPRO (Yang et al., 2023) search instructions with an LLM, while ProTeGi/APO (Pryzant et al., 2023) uses textual gradients, TextGrad (Yuksekgonul et al., 2024) backpropagates natural-language feedback, and EvoPrompt (Guo et al., 2024) and Promptbreeder (Fernando et al., 2024) mutate and recombine whole prompts under evolutionary search; MIPRO (Khattab et al., 2023; Opsahl-Ong et al., 2024) compiles and tunes multi-stage LM programs. GEPA (Agrawal et al., 2025), the one we adopt in this work, pairs reflective rewriting with pareto-front evolutionary search. What unites these is the fact that they all treat the prompt under consideration as a single undifferentiated string, while our claims operate with explicit components.

The work closest to ours gives the judge an internal structure, an explicit rubric of a criterion plus score descriptions, and shows that a small evaluator prompted that way can compete against larger judges (Kim et al., 2024a; Ye et al., 2024; Kim et al., 2024b; 2025). Additionally, G-Eval (Liu et al., 2023) evaluates with a hand-written criterion inside its chain-of-thought prompt that is never optimized, while the checklist and unit-test-style methods push the idea to binary yes/no items (Lee et al., 2025; Saad-Falcon et al., 2024). In all these cases the decomposition is either written with a human in the loop or optimized automatically as a single block of free text (similar to our *combined* regime). It is worth noting that specialized guardrail models serve similar goals in this landscape: some are shipped as fixed-taxonomy classifiers (Inan et al., 2023; Zeng et al., 2024a; Han et al., 2024; Padhi et al., 2024), while others are governed by written rules (Zhang et al., 2024b), cast as reasoning-based guards (Liu et al., 2025), or conditioned on arbitrary user-defined policies (Hoover et al., 2025). This bring-your-own-criteria feature is what we put under our study lens. To our knowledge no prior work treats the criterion/schema decomposition as the optimization unit, in this setup or otherwise.

3 Problem Setup

3.1 Decomposition into two components

A judge is a frozen LLM, M, that maps an input x and a prompt θ to a verdict $\hat{y} = M(x;\theta) \in \{\texttt{yes}, \texttt{no}\}$. The model here is never fine-tuned, and inference runs at temperature zero, so that outputs are deterministic given θ: the prompt is the only lever. We decompose this prompt into two functionally distinct free-text components,$\theta = (c,\ s)$, where the criterion c states what condition to detect and the scoring schema s states how to map a judgment onto the label. We make this distinction explicit by modeling the judge as a composition of an evidence step and a decision step,

$$M\big(x;(c,s)\big) \ \approx \ d_s\big(e_c(x)\big),$$

where the criterion c shapes what the model attends to and extracts from x (the assessment e_c), and the schema s maps that assessment onto the decision label (the decision rule d_s). Note that this is a modeling lens, not a claim of two forward passes: at inference the model still sees a single block $B(c,s)$ that concatenates the two under fixed section headers. The point of this decomposition is not two forward passes but two separately addressable optimization coordinates, each with a specific objective; the monolithic baseline optimizes B as one undifferentiated span, leaving the (c,s) coordinates implicit. It is important to note that most judgments studied in the literature are binary, and we take $\hat{y} \in \{\texttt{yes}, \texttt{no}\}$ in this article as well, together with our reward and metrics. The decomposition, however, is not tied to binary labels: the schema s maps a judgment onto whatever label space a task defines, ordinal or categorical.

Table 2: Six optimization modes. Every mode edit only the judge prompt $\theta = (c, s)$ of the model, differing in *which coordinates* optimizer may touch and *how* the per-run budget B (judge evaluations) is spent. $\mathcal{C}, \mathcal{S}$ are the criterion and schema text spaces, $\mathcal{B}$ the rendered block and $\theta_0 = (c_0, s_0)$ the seed. *combined* searches $\mathcal{B}$ as undifferentiated span (the monolithic baseline), while *joint* edits criterion and schema as two independently addressable coordinates $\mathcal{C} \times \mathcal{S}$ under a single budget.

Mode	Search space	Per step / schedule	Cost
Single-budget regimes			
criteria-only	$\mathcal{C} \times \{s_0\}$	Rewrite c; schema frozen at seed.	B
schema-only	$\{c_0\} \times \mathcal{S}$	Rewrite s; criterion frozen at seed.	B
combined	$\mathcal{B}$ (one span)	Rewrite the merged block as *one* span; c, s not separately addressable (monolithic baseline).	B
joint	$\mathcal{C} \times \mathcal{S}$	Round-robin $c, s, c, \ldots$; both coordinates co-evolve (explicit decomposition).	B
Phased schedules (rerun the whole search each phase)			
two-phase	$\mathcal{C} \times \mathcal{S}$ (seq.)	Phase 1 optimizes s (c frozen); phase 2 optimizes c (s fixed at phase 1's best).	$2B$
three-phase	$\mathcal{C} \times \mathcal{S}$ (seq.)	Sequential $s \to c \to s$; each phase optimizes one coordinate, others frozen.	$3B$

Given a labeled validation set, optimization searches over (c, s) to maximize a class-balanced reward, $R(\theta) = \frac{1}{2}(\mathrm{TPR} + \mathrm{TNR})$, the mean of the per-class accuracies (equivalently balanced accuracy), so that the majority class can not dominate on the imbalanced data typical of guardrail tasks; we report test F1, and return to this reward-metric gap in 5.

Across all our experiments, we use GEPA (Agrawal et al., 2025) as our search procedure, a reflective optimizer that proposes edits from natural-language feedback on mislabeled examples, aiming to tighten the rule for false positives and broaden it for false negatives. The reflection step that proposes each edit is carried out by a separate model, held fixed within a run; we sweep four such teachers, `qwen3-next-80b-a3b-thinking`, `llama-3.3-70b-instruct`, `nemotron-3-super-120b-a12b-fp8`, and `granite-4.1-30b` (abbreviated qwen, llama, nemo, g30 in the appendix), and report the best per cell; the cross-judge study of 4 fixes `qwen3-next-80b-a3b-thinking` as the teacher. We compare six optimization modes (Table 2) that differ only in what the optimizer can edit and how the budget is spent. Let $\mathcal{C}$ and $\mathcal{S}$ be the criterion and schema text spaces, respectively, and let $(c_0, s_0) = \theta_0$ be the seed state. Each single-budget regime restricts where the search may move,

$$\Theta_{\text{crit}} = \mathcal{C} \times \{s_0\}, \quad \Theta_{\text{schema}} = \{c_0\} \times \mathcal{S}, \quad \Theta_{\text{joint}} = \mathcal{C} \times \mathcal{S}, \quad \Theta_{\text{comb}} \subset \mathcal{B},$$

where Θ_{comb} searches the rendered block $\mathcal{B}$ as one undifferentiated span, the monolithic baseline that rubric/criteria optimizers use. It is important to note the contrast between Θ_{comb} (with no explicit markers) and Θ_{joint} (edit c and s as two components in round-robin, i.e. explicit decomposition). Note $\Theta_{\text{crit}} \cup \Theta_{\text{schema}} \subseteq \Theta_{\text{joint}}$, so joint search in principle can match the better single component; whether it does at a finite budget is an empirical question (5). A *k-phase* schedule performs block-coordinate search: phase i optimizes one coordinate for B evaluations with the other component frozen at the previous phase's selection (two-phase evolves $s \to c$; three-phase $s \to c \to s$). As *budget* counts judge evaluations, such a schedule costs $\mathrm{Cost}(k\text{-phase@}B) = kB$ evaluations. This *compute-accounting* identity is the crux of 5: comparing modes at a fixed per-phase B compares them at *unequal* total cost of $\{B, 2B, 3B\}$, whereas the honest comparison holds the total $T = \sum_i B_i$ fixed.

3.2 Performance gains are bounded by headroom

As the judge is frozen and optimization only operates on prompts, every regime is confined to the behaviors that *some* prompt can induce in M. Let $F^\star$ be the best F1 any prompt can elicit from the model (its *elicitation ceiling*), let θ_0 be the seed prompt, and let $H = F^\star - F_1(\theta_0)$ be the judge's *headroom*. Because each regime only searches for a better prompt, it can never beat the ceiling, so its gain over the seed, G, is bounded by the headroom, $G \leq H$. Writing the *regret* $\rho \geq 0$ for the F1

Table 3: zero-shot vs. optimized, every scorer. Held-out test F1 (mean over 5 seeds), twelve tasks: *0-shot* = seed criterion + default schema, *best* = strongest GEPA config (mode·teacher·budget, $\{200, \ldots, 3200\}$); columns differ only in the scoring model, **bold** = best optimized cell per row. *best* columns are grid maxima, optimistically biased by ≈ 0.008 (7).

	`guardian-8b`		`granite-8b`		`granite-30b`		`gemma-12b`		`gemma-26b`		`gemma-31b`	
Dataset	0-shot	best	0-shot	best	0-shot	best	0-shot	best	0-shot	best	0-shot	best
ragtruth	0.646	0.707	0.000	0.585	0.000	0.619	0.211	0.708	0.457	0.743	0.634	**0.798**
assebench	0.508	0.705	0.142	0.686	0.259	0.669	0.430	0.738	0.513	0.764	0.511	**0.770**
atbench	0.577	0.955	0.043	0.784	0.463	0.784	0.476	0.948	0.341	0.956	0.564	**0.974**
toolcall	0.770	0.868	0.000	0.648	0.116	0.712	0.703	0.882	0.636	0.898	0.751	**0.914**
reward-bench-2	0.559	0.584	0.558	0.619	0.457	0.606	0.669	0.684	0.635	0.675	0.679	**0.734**
beavertails	0.783	**0.796**	0.528	0.713	0.645	0.727	0.709	0.765	0.691	0.766	0.730	0.782
hallumix	0.849	0.899	0.120	0.768	0.844	0.886	0.682	0.874	0.722	0.893	0.886	**0.907**
opir	0.812	0.841	0.440	0.788	0.753	0.808	0.723	0.827	0.731	0.830	0.818	**0.859**
pairwise	0.767	0.772	0.695	0.695	0.654	0.713	0.768	**0.774**	0.766	**0.774**	0.766	0.766
off-topic	0.954	**0.987**	0.016	0.947	0.905	0.978	0.812	0.952	0.921	0.975	0.965	0.973
toxigen	0.875	0.882	0.650	0.876	0.814	0.873	0.767	0.871	0.705	0.856	0.798	**0.885**
wildguardmix	0.900	0.925	0.588	0.907	0.782	0.891	0.765	0.927	0.799	0.931	0.876	**0.942**
mean	0.750	0.827	0.315	0.751	0.558	0.772	0.643	0.829	0.660	0.838	0.748	**0.859**

that the best admissible prompt would still add on top of what the search actually found, the realized gain splits cleanly into $G = H - \rho$: what the seed leaves on the table, minus what the optimizer fails to recover.

The decomposition here separates the two levers cleanly. Prompt optimization, the focus of our article, can only shrink the regret ρ toward 0 and drive G up to H; it can not raise H. Raising H means raising the ceiling $F^\star$, which requires a more capable frozen model or fine-tuning it, not a better prompt. The ceiling $F^\star$ is itself fixed by the frozen model's task knowledge, acquired in pretraining and surfaced by the prompt rather than instilled by it, the same view under which alignment tuning mostly exposes latent ability rather than teaching new knowledge (Zhou et al., 2023a). On a task whose decision the base model has not internalized (its scores ranking little better than chance, so $F^\star$ is low), even a zero-regret optimizer ($\rho = 0$) tops out at the low ceiling, and the judge stays suboptimal no matter how the prompt is drafted, consistent with evidence that prompting-based self-improvement can not cross a model's underlying capability boundary (Huang et al., 2024). This failure mode is distinct from the small-headroom case, where the ceiling is high but already reached, whereas here a knowledge gap in the base model holds the ceiling itself down, as is the case for applications in niche domains. Prompt optimization, irrespective of whether it is decomposed or not, helps most on the tasks the model can already perform but is merely prompted poorly for, and least where the requisite knowledge is simply absent. Decomposition and phasing are therefore devices for reducing the regret within a fixed capacity, which is why reaching for a bigger model and optimizing the prompt are complementary, not substitutable (Dorner et al., 2025): the former buys the headroom, the latter spends it.

The bound also caps the advantage of decomposition. Empirically we find that no regime underperforms the seed, so every gain lies in $[0, H]$: two regimes can then differ by at most H, and in particular the decomposition advantage (*joint* minus *combined*) is at most H, vanishing as $H \to 0$. Two predictions follow, both borne out in 4: optimization gains, and the decomposition advantage specifically, are largest for judges that start far from ceiling and vanish for judges that start near it. Explicit decomposition therefore buys reachability within a budget, not additional capacity, i.e., a judge with no headroom can't be moved, so a near-null decomposition effect on an already-strong judge is what the bound predicts, not an anomaly.

4 Does Explicit Decomposition Help Every Judge?

To understand whether explicit decomposition helps across the board, we fix the reflection teacher and search budget and vary only the *judge model* and the *decomposition regime*. The judges span two families and a $4\times$ size range: three general-purpose LLMs (`granite-4.1-8b`, `granite-4.1-30b`, `gemma-4-31b`) and the purpose-built guardrail model `granite-guardian-4.1-8b`. For each judge we run all six optimization modes (Ta-

ble 2) on six tasks (agentic-trajectory safety, agent safety/security, tool-augmented response faithfulness, scope control, grounding, and toxicity; described in Appendix A), five seeds each, and report the held out test F1.[1] For each task we draw a single labeled pool from its public release and partition it ourselves, stratified by label, into disjoint train / validation / test splits (at most 140 / 160 / 140 examples), so no example appears in more than one split. GEPA optimizes on train, selects on validation, and every reported number is scored on the untouched test partition, to which the optimizer is blind. Splits are seeded per task, so every judge and every regime is evaluated on *identical* examples, and "test" denotes our held-out partition rather than a dataset's canonical benchmark test set. Table 1 reports test F1 for the four single-budget regimes (criterion-only, schema-only, combined, joint), together with the two summary contrasts the analysis below turns on: Δ_{opt} (best regime minus zero-shot) and $\Delta_{\text{j-c}}$ (joint minus combined). The *decomposed* bar of Figure 2 aggregates the five decomposition regimes, including the two phased schedules.

4.1 Explicit decomposition beats a monolithic block

Averaged over the six tasks, the best regime improves F1 over the zero-shot prompt by $+0.44$ for `granite-4.1-8b` ($0.14 \rightarrow 0.58$), $+0.20$ for `granite-4.1-30b` ($0.43 \rightarrow 0.63$), $+0.13$ for `gemma-4-31b` ($0.70 \rightarrow 0.84$), and a marginal $+0.01$ for the already-strong guardian ($0.72 \rightarrow 0.73$). The gain shrinks monotonically as the zero-shot judge gets stronger, because reflective optimization can only recover the headroom a judge leaves unused, and the strongest judge starts with almost none on these tasks. One of our core observations is that treating the prompt as two separate, explicit components (the *joint* regime) beats optimizing it as one free-text block (the *combined* regime). The gain is clearest on the judges where there is headroom. The joint regime adds +0.128 and +0.043 F1 on `granite-4.1-8b` and `granite-4.1-30b` respectively, both above the measured run-to-run noise floor (0.0126–0.0144 F1). The improvement is largest where the judge is weakest and has the most to gain from added structure. On the two near-ceiling judges the gap stays positive but falls within that floor; we read this as a predicted null rather than evidence of a counterexample. With almost no headroom ($H \approx 0$), the headroom bound (3.2) already pins the decomposition gap to near zero. Across all four judges the advantage never reverses sign, and it shrinks as zero-shot strength rises. This reflects a property of the judge prompt's *structure* that holds across models, not something particular to the guardian we started from. A check on two further non-granite judges supports this cross-family reading rather than leaning on the near-ceiling null of `gemma-4-31b`. On `gemma-4-26b`, a mixture-of-experts judge with genuine headroom (zero-shot F1 around 0.60), the joint regime beats combined by $+0.045$ F1, clearing the noise floor as it does on the granite judges mentioned earlier. Its smaller sibling `gemma-4-12b` (zero-shot 0.56) is the boundary case. Its *average* gap is a null ($+0.005$), but that average is driven entirely by a single long-context agentic-safety task, where the smallest judge does better with one integrated instruction than with the separated structure. On the other five tasks it recovers the same positive decomposition gain ($+0.027$, pushing it above the floor).

Explicit decomposition therefore asks for two things: headroom and enough capacity to act on separated components. This marks a boundary on where the effect lands, consistent with the headroom bound. The transferable takeaway is only this coarse one: which specific component or schedule wins varies by judge and task (5), so we do not single out a dominant one.

4.2 The optimized criteria are semantic

Over the course of our experiments we noticed that a few tasks can be separated by a shallow construction cue rather than property they are meant to probe and that a reflection-driven optimizer can occasionally latch onto that cue instead of real signal, producing a brittle, suboptimal criterion. This raises the question of whether our gains merely reflect these dataset-specific shortcuts.

We test this on the two tasks that carry a removable construction cue on which GEPA also leads to a large gain. We re-score the criterion and schema GEPA actually discovered, without further optimization, on data where that cue and nothing else is removed, while every gold label is preserved.

[1] Five of the six tasks use identical data draws and split sizes across every table. The scope-control task uses a class-balanced 600-example draw in the cross-judge study and an unbalanced draw of same split sizes in the component and compute-matched tables (Tables 4, 13); its rows are therefore not example-for-example comparable across those tables (seed-regime F1 0.942 vs. 0.954), though no headline claim depends on it.

Table 4: Component transplant (judge `granite-guardian-4.1-8b`, best teacher; mean over 5 seeds; **bold** = best of the three). Each component is paired with the other's *seed* to isolate it (4: from scratch). No single component is universally the lever; on the coupled tasks the two are *super-additive* (interaction $+0.12$ to $+0.16$), the criterion with the seed schema falling *below* it.

Dataset	seed	crit-transplant	schema-transplant	both
atbench	0.577	0.522	0.886	**0.954**
assebench	0.508	0.582	0.603	**0.703**
toolcall	0.770	0.642	0.839	**0.868**
ragtruth	0.646	0.648	**0.707**	0.705
opir	0.812	0.805	**0.846**	0.844
hallumix	0.849	0.880	0.882	**0.897**
off-topic	0.954	0.798	0.982	**0.987**
wildguardmix	0.900	0.900	**0.925**	0.923
toxigen	0.875	0.877	0.875	**0.882**
beavertails	0.783	**0.798**	0.777	0.794
pairwise	0.767	0.769	0.769	**0.772**
reward-bench-2	0.559	0.572	0.573	**0.586**

We find that the optimized artifact barely moves, at most 0.008 at the median, whereas a cost-free rule keyed on the same cue collapses to the constant-class score or to zero: the optimizer learns a semantic notion of refusal and of tool use, not a literal string.

4.3 Specialization over scale

If a decomposed prompt improves the realized F1 of every judge, the natural reading would be to optimize whichever large model is at hand. However, our full-suite results point the other way. Expanding from the six core tasks to all twelve, with every scorer at its best configuration (Table 3; full per-dataset results in Appendix B), we see the same pattern as on the core tasks: our method helps most where the judge is weakest, i.e., optimization rescues the general judges that start weakest, several of which degenerate to near-zero zero-shot F1. A single frozen 8B guardian carries the same decomposition while covering both pointwise and pairwise tasks in one model. The guardian also holds its own against same-generation general models: once GEPA-optimized, it out-scores `granite-4.1-8b` and `-30b` on 11 of 12 tasks, the only exception being reward-bench-2. We read this as *parameter-efficiency and unified coverage*, not dominance: scaling the general model to 30B does not close the gap, yet a newer family of similar size to that scale-up (`gemma-4-31b`) is the best scorer on $9/12$ once optimized. The important transferable lesson is that the decomposition travels down to a small, deployable judge, not that a small judge beats every large one.

5 Component, Schedule, and Compute

In the previous section we established a robust but coarse claim: decomposing the prompt beats leaving it monolithic. Two stronger readings of that result do not hold up, and this section takes each apart. The first is that a single component, the schema, is the lever, while the second is that elaborate phased schedules beat plain joint search.

5.1 Where the gains come from

For the judge models, with configurable and optimized components, which component gives the highest gain is task-dependent, not fixed. The per-task transplant view isolates each component's contribution by pairing one optimized component with the seed of the other. Table 4 presents this over twelve datasets on the guardian, best teacher per cell. The schema transplant wins on some tasks (e.g., ragtruth, opir, wildguardmix), while the criterion transplant wins on others (e.g., beavertails). The same non-uniformity shows up cross-judge (Figure 2) as an interaction with judge strength: for the two weaker judges, crit-only is the strongest single-component regime. Similarly, joint search does not reliably beat the better single component. Its average edge over the best single is small, and for two of the four judges it is negative (`granite-4.1-8b` -0.067, `granite-4.1-30b`

Table 5: On the guardian, the criterion carries the discrimination gain. Each applies *only* the optimized criterion or *only* the schema (other frozen; 5 seeds). ΔAUROC = threshold-invariant ranking gain, ΔF1 re-thr. = re-thresholding the *seed's own* scores. The criterion's ΔAUROC ≈ 4× the schema's; both vanish near ceiling (3 bound).

Task	Seed AUROC	ΔAUROC (crit)	ΔAUROC (schema)	ΔF1 re-thr.
AsseBench	0.648	+0.092	−0.033	+0.164
ATBench	0.635	+0.086	+0.067	+0.078
Tool-call faithfulness	0.666	+0.020	+0.006	+0.096
RAGTruth	0.795	+0.004	+0.014	+0.053
Off-topic	0.998	−0.000	−0.000	+0.044
ToxiGen	0.953	−0.005	−0.001	−0.020
Mean	0.782	+0.033	+0.009	+0.069

−0.008). The coupling between criterion and schema is task- and judge-dependent, hence which component to search is a tuning decision rather than a principled one.

The same transplants (Table 4) also show *how* the two combine: on coupled structured tasks a meaningful share of the gain needs both optimized together, as if the criterion and schema jointly encode the decision boundary. This is what makes decomposition pay off even with no universal lever. To see *what each does*, we re-scored the guardian's seed with the optimized criterion alone, then the optimized schema alone. This separates two axes: discrimination (threshold-invariant ranking, AUROC) and the decision rule (where the threshold sits); on the guardian they map cleanly onto the two components (Table 5). The criterion raises AUROC about 4× as much as the schema (mean +0.033 vs. +0.009), while the schema leaves ranking untouched and recovers +0.069 F1 by moving the threshold alone. Both gains are bounded by headroom (3.2): on near-ceiling tasks (off-topic, ToxiGen; seed AUROC ≥ 0.95) neither moves, and the criterion's AUROC gain shrinks as the seed's rises (Pearson −0.62, all judges and tasks). This split is a tendency, not a rule (on ATBench the schema sharpens ranking too), but it explains why the criterion is the more common lever: most of the discrimination headroom lives there.

5.2 The phasing improvement is mostly a compute artifact

A per-budget table invites a tidy story: the modes line up crit ≈ schema < joint < two-phase < three-phase, so more phases must help. But that ranking is an artifact of how compute is counted. A k-phase run at budget B spends the full budget *per phase*, kB evaluations in all, so a fixed-B comparison pits the modes at *unequal* compute (B, $2B$, $3B$). Fix total evaluations instead and the gap closes: on the structured tasks, joint at 3200 and two-phase at 1600 each spend ≈ 3200 evaluations and reach statistically indistinguishable F1 (0.732 vs. 0.735). Whatever edge the phased modes show in a per-budget table is bought with compute, not structure (Table 12).

5.3 But phasing is how to spend a compute surplus

Matched compute settles the ranking question, but not the deployment one: compute is not always scarce, and that is where phasing earns its keep. From 1600 to 3200 evaluations joint search stops improving (mean F1 0.738 → 0.732, within the noise floor), while the phased schedules keep climbing: two-phase reaches 0.80 at 6400 evaluations, three-phase 0.81 at 4800 (Table 13). We never ran joint past 3200 evaluations, so we do not claim it beats a larger single joint budget. The narrower point is enough: once joint has saturated, phasing is a productive use of surplus compute, a search-time analogue of the test-time compute scaling that lifts LLM judges (Chan et al., 2025). It pays off most on the hardest structured tasks, where the paper's largest absolute gains come from phased runs whose criterion is hardest to write by hand (agentic-trajectory safety 0.58 → 0.96 under a three-phase schema schedule; the FinProof compliance policy 0.70 → 0.92, 6). The claim is not that phasing is structurally better at equal cost (it is not), only that once joint search flattens, phasing is the right *use* of extra compute.

Table 6: FinProof BFSI attack detection. Held-out test F1/precision/recall and benign false-positive rate (positive class = attack), all rows on the same public V1 split (248 attacks + 120 benign). A frozen 8B guardian improves from 0.70 zero-shot to a five-seed mean of 0.92 under the three-phase recipe (best single deployed prompt 0.957; a low-FPR prompt holds 0.95 at a 3.3% benign FPR).

Model	Adaptation	F1	Prec.	Rec.	FPR
Our judge (frozen granite-guardian-4.1-8b, prompt-only)					
granite-guardian-4.1-8b	zero-shot	0.700	0.993	0.540	0.008
granite-guardian-4.1-8b	GEPA (3-phase, mean)	0.915±0.030	–	–	–
granite-guardian-4.1-8b	GEPA (best deployed)	**0.957**	0.975	0.940	0.050
granite-guardian-4.1-8b	GEPA (low-FPR prompt)	0.952	0.983	0.923	0.033
GEPA-optimized general judges (identical protocol, best config, 5-seed mean)					
granite-4.1-8b	GEPA (best)	0.922±0.038	–	–	–
granite-4.1-30b	GEPA (best)	0.898±0.019	–	–	–
gemma-4-31b	GEPA (best)	0.947±0.028	–	–	–
Fixed-taxonomy guardrails, zero-shot (same test split)					
ShieldGemma-9B	zero-shot	0.670	1.000	0.504	0.000
LlamaGuard-3-8B	zero-shot	0.712	1.000	0.552	0.000
WildGuard-7B	zero-shot	0.674	1.000	0.508	0.000

6 When the Policy Is Not in the Taxonomy

The cross-judge study asks whether decomposition helps each judge; it never asks whether a decomposed prompt lets a general judge take on a policy that deployed safety guards *cannot represent at all*. We test that on FinProof, a third-party adversarial benchmark for banking/finance/insurance (BFSI) guardrails whose decision boundary is a custom regulatory policy. The task is to classify a user prompt as an *attack* (an attempt to elicit a regulatory violation: unlicensed investment advice, KYC/AML bypass, fabricated regulatory claims) versus a *benign* customer inquiry. The benign set mirrors the attack categories (same finance topic, different intent), so the distinction is regulatory *intent*, not keywords. We follow the benchmark's native train/dev/test split and report on the held-out test set (368 examples). A 8B judge, optimized only through its prompt and never fine tuned, improves F1 from 0.70 (zero-shot) to a five-seed mean of 0.92 under a three-phase search with a reasoning teacher; the best single deployed prompt reaches 0.96 (Table 6). The gain is almost entirely recall: for the best deployed prompt it rises from 0.54 to 0.94 (0.92 at a stricter false-alarm setting) at held precision (0.975–0.983). Fixed-taxonomy safety guards, run by us on the same split, cannot approach it (ShieldGemma-9B 0.67, LlamaGuard-3-8B 0.71, WildGuard-7B 0.67) because they cannot be *told* the custom BFSI policy: the gap is expressiveness, not scale. General judges, which unlike those guards *can* take the policy, reproduce the parameter-efficiency picture of 4 once optimized under the identical protocol: on the five-seed mean they bracket the guardian without displacing it. The guardian is also tunable: at a 2% false-alarm budget it catches 91% of attacks (vs 62% zero-shot), with AUROC 0.99.

7 Conclusion and Limitations

In the article we treat a judge's prompt as two separate optimizable components, a criterion and a scoring schema, and ask what reflective search over them buys. Across every judge that we tested, decomposing the prompt rather than optimizing it as a single block is the robust lever, even though the component that carries the gain shifts between various tasks and judges. For guardian, this yields genuine gains on 8 of 12 tasks. As presented throughout the article, several caveats bound our claims. First, re-running the same prompts shifts the guardian's F1 by 0.0268, and selection compounds it. This is often optimistically biased, and similar overoptimization has been noted for reward models (Gao et al., 2023). The deeper limit is that prompt optimization can chase the headroom but cannot create the capability a frozen model lacks: when a knowledge gap holds the elicitation ceiling $F^\star$ down, no amount of decomposition, phasing, or budget produces a reliable judge. These limits, however, a practitioner can see coming: a cheap seed pass confirms the headroom before any budget is spent (3), and our controls confirm the gains are semantic rather than surface shortcuts, so that decomposition can be applied only where it genuinely pays off.

AI USE STATEMENT

AI-assisted coding tools were used to help implement parts of the experiments for this work and LLMs were used to post-polish the text for improved clarity and grammar. All AI-assisted code was verified and tested by the authors and all AI-edited text was reviewed by the authors.

ETHICS STATEMENT

Optimizing guardrail prompts is dual-use. The same procedure that hardens a judge could be used to probe its weaknesses. We study only publicly released safety benchmarks; there are no human subjects and no PII involved. The FinProof rubric encodes a jurisdiction-specific (BFSI) notion of a compliance violation and should not be read as a universal standard.

REFERENCES

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. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025. URL https://arxiv.org/abs/2507.19457.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic AI. *arXiv preprint arXiv:2506.02153*, 2025. URL https://arxiv.org/abs/2506.02153.

Chi-Min Chan, Chunpu Xu, Jiaming Ji, Zhen Ye, Pengcheng Wen, Chunyang Jiang, Yaodong Yang, Wei Xue, Sirui Han, and Yike Guo. J1: Exploring simple test-time scaling for LLM-as-a-judge. *arXiv preprint arXiv:2505.11875*, 2025. URL https://arxiv.org/abs/2505.11875.

Gabriel Chua, Shing Yee Chan, and Shaun Khoo. A flexible large language models guardrail development methodology applied to off-topic prompt detection. *arXiv preprint arXiv:2411.12946*, 2024. URL https://arxiv.org/abs/2411.12946.

Florian E. Dorner, Vivian Y. Nastl, and Moritz Hardt. Limits to scalable evaluation at the frontier: LLM as judge won't beat twice the data. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/4264ee4376776907c0b87ed70b959585-Abstract-Conference.html.

Deanna Emery, Michael Goitia, Freddie Vargus, and Iulia Neagu. HalluMix: A task-agnostic, multi-domain benchmark for real-world hallucination detection. *arXiv preprint arXiv:2505.00506*, 2025. URL https://arxiv.org/abs/2505.00506.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544. PMLR, 2024. URL https://proceedings.mlr.press/v235/fernando24a.html.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866. PMLR, 2023. URL https://proceedings.mlr.press/v202/gao23h.html.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. JSONSchemaBench: A rigorous benchmark of structured outputs for language models. *arXiv preprint arXiv:2501.10868*, 2025. URL https://arxiv.org/abs/2501.10868.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A survey on LLM-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024. URL https://arxiv.org/abs/2411.15594.

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. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9156b0f6dfa9bbd18c79cc459ef5d61c-Abstract-Conference.html.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. WildGuard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of LLMs. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0f69b4b96a46f284b726fbd70f74fb3b-Paper-Datasets_and_Benchmarks_Track.pdf.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3309–3326, 2022. doi: 10.18653/v1/2022.acl-long.234. URL https://aclanthology.org/2022.acl-long.234/.

Monte Hoover, Vatsal Baherwani, Neel Jain, Khalid Saifullah, Joseph Vincent, Chirag Jain, Melissa Kazemi Rad, C. Bayan Bruss, Ashwinee Panda, and Tom Goldstein. DynaGuard: A dynamic guardian model with user-defined policies. *arXiv preprint arXiv:2509.02563*, 2025. URL https://arxiv.org/abs/2509.02563.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8b4add8b0aa8749d80a34ca5d941c355-Abstract-Conference.html.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*, 2023. URL https://arxiv.org/abs/2312.06674.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. BeaverTails: Towards improved safety alignment of LLM via a human-preference dataset. *arXiv preprint arXiv:2307.04657*, 2023. URL https://arxiv.org/abs/2307.04657.

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. *arXiv preprint arXiv:2310.03714*, 2023. URL https://arxiv.org/abs/2310.03714.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations (ICLR)*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/803485352e61e3ebf41221e4776c9fd4-Abstract-Conference.html.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4334–4353, 2024b. doi: 10.18653/v1/2024.emnlp-main.248. URL https://aclanthology.org/2024.emnlp-main.248/.

Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, Sue Hyun Park, Hyeonbin Hwang, Jinkyung Jo, Hyowon Cho, Haebin Shin, Seongyun Lee, Hanseok Oh, Noah Lee, Namgyu Ho, Se June Joo, Miyoung Ko, Yoonjoo Lee, Hyungjoo Chae, Jamin Shin, Joel Jang, Seonghyeon Ye, Bill Yuchen Lin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. The BiGGen bench: A principled benchmark for fine-grained evaluation of language models with language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL)*, pp. 5877–5919, 2025. doi: 10.18653/v1/2025.naacl-long.303. URL https://aclanthology.org/2025.naacl-long.303/.

Yukyung Lee, JoongHoon Kim, Jaehee Kim, Hyowon Cho, Jaewook Kang, Pilsung Kang, and Najoung Kim. CheckEval: A reliable LLM-as-a-judge framework for evaluating text generation using checklists. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 15771–15798, 2025. doi: 10.18653/v1/2025.emnlp-main.796. URL https://aclanthology.org/2025.emnlp-main.796/.

Dongrui Liu, Qihan Ren, Chen Qian, Shuai Shao, Yuejin Xie, Yu Li, Zhonghao Yang, Haoyu Luo, Peng Wang, Qingyu Liu, Binxin Hu, Ling Tang, Jilin Mei, Dadi Guo, Leitao Yuan, Junyao Yang, Guanxu Chen, Qihao Lin, Yi Yu, Bo Zhang, Jiaxuan Guo, Jie Zhang, Wenqi Shao, Huiqi Deng, Zhiheng Xi, Wenjie Wang, Wenxuan Wang, Wen Shen, Zhikai Chen, Haoyu Xie, Jialing Tao, Juntao Dai, Jiaming Ji, Zhongjie Ba, Linfeng Zhang, Yong Liu, Quanshi Zhang, Lei Zhu, Zhihua Wei, Hui Xue, Chaochao Lu, Jing Shao, and Xia Hu. AgentDoG: A diagnostic guardrail framework for AI agent safety and security. *arXiv preprint arXiv:2601.18491*, 2026. URL https://arxiv.org/abs/2601.18491.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. ToolACE: Winning the points of LLM function calling. *arXiv preprint arXiv:2409.00920*, 2024. URL https://arxiv.org/abs/2409.00920.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2511–2522, 2023. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Yue Liu, Hongcheng Gao, Shengfang Zhai, Yufei He, Jun Xia, Zhengyu Hu, Yulin Chen, Xihong Yang, Jiaheng Zhang, Stan Z. Li, Hui Xiong, and Bryan Hooi. GuardReasoner: Towards reasoning-based LLM safeguards. *arXiv preprint arXiv:2501.18492*, 2025. URL https://arxiv.org/abs/2501.18492.

Hanjun Luo, Shenyu Dai, Chiming Ni, Xinfeng Li, Guibin Zhang, Kun Wang, Tongliang Liu, and Hanan Salam. AgentAuditor: Human-level safety and security evaluation for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3dc85735f6e2fcf093e67b134fa00d21-Abstract-Conference.html.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing reward model evaluation. *arXiv preprint arXiv:2506.01937*, 2025. URL https://arxiv.org/abs/2506.01937.

Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, KaShun Shum, Randy Zhong, Juntong Song, and Tong Zhang. RAGTruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 10862–10878, 2024. doi: 10.18653/v1/2024.acl-long.585. URL https://aclanthology.org/2024.acl-long.585/.

NVIDIA. Nemotron-Cascade-RM-Training. Hugging Face dataset nvidia/Nemotron-Cascade-RM-Training, 2025. URL https://huggingface.co/datasets/nvidia/Nemotron-Cascade-RM-Training.

Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9340–9366, 2024. doi: 10.18653/v1/2024.emnlp-main.525. URL https://aclanthology.org/2024.emnlp-main.525/.

Inkit Padhi, Manish Nagireddy, Giandomenico Cornacchia, Subhajit Chaudhury, Tejaswini Pedapati, Pierre Dognin, Keerthiram Murugesan, Erik Miehling, Martín Santillán Cooper, Kieran Fraser, Giulio Zizzo, Muhammad Zaid Hameed, Mark Purcell, Michael Desmond, Qian Pan, Zahra Ashktorab, Inge Vejsbjerg, Elizabeth M. Daly, Michael Hind, Werner Geyer, Ambrish Rawat, Kush R. Varshney, and Prasanna Sattigeri. Granite guardian. *arXiv preprint arXiv:2412.07724*, 2024. URL https://arxiv.org/abs/2412.07724.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7957–7968, 2023. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, pp. 431–445, 2023. doi: 10.18653/v1/2023.emnlp-demo.40. URL https://aclanthology.org/2023.emnlp-demo.40/.

Jon Saad-Falcon, Rajan Vivek, William Berrios, Nandita Shankar Naik, Matija Franklin, Bertie Vidgen, Amanpreet Singh, Douwe Kiela, and Shikib Mehri. LMUnit: Fine-grained evaluation with natural language unit tests. *arXiv preprint arXiv:2412.13091*, 2024. URL https://arxiv.org/abs/2412.13091.

Ihor Stepanov. OpirSafetyBench. Hugging Face dataset Ihor/OpirSafetyBench, 2026. URL https://huggingface.co/datasets/Ihor/OpirSafetyBench.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/9e720fce64f91114c49cfd640d821da3-Abstract-Conference.html.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 9440–9450, 2024. doi: 10.18653/v1/2024.acl-long.511. URL https://aclanthology.org/2024.acl-long.511/.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023. URL https://arxiv.org/abs/2309.03409.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. FLASK: Fine-grained language model evaluation based on alignment skill sets. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/f41b4a6b202adcd8e150a9d4f124d8f6-Abstract-Conference.html.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024. URL https://arxiv.org/abs/2406.07496.

Wenjun Zeng, Yuchi Liu, Ryan Mullins, Ludovic Peran, Joe Fernandez, Hamza Harkous, Karthik Narasimhan, Drew Proud, Piyush Kumar, Bhaktipriya Radharapu, Olivia Sturman, and Oscar Wahltinez. ShieldGemma: Generative AI content moderation based on Gemma. *arXiv preprint arXiv:2407.21772*, 2024a. URL https://arxiv.org/abs/2407.21772.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large language models at evaluating instruction following. In *International Conference on Learning Representations (ICLR)*, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/afc8b034823271816d14f7c1aefe1dff-Abstract-Conference.html.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024a. URL https://arxiv.org/abs/2408.15240.

Zhexin Zhang, Yida Lu, Jingyuan Ma, Di Zhang, Rui Li, Pei Ke, Hao Sun, Lei Sha, Zhifang Sui, Hongning Wang, and Minlie Huang. ShieldLM: Empowering LLMs as aligned, customizable and explainable safety detectors. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10420–10438, 2024b. doi: 10.18653/v1/2024.findings-emnlp.610. URL https://aclanthology.org/2024.findings-emnlp.610/.

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. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. LIMA: Less is more for alignment. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ac662d74829e4407ce1d126477f4a03a-Abstract-Conference.html.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023b. URL https://arxiv.org/abs/2311.07911.

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)*, 2023c. URL https://arxiv.org/abs/2211.01910.

A Datasets

In Table 7 we list the twelve datasets, their type and their seeded split sizes.

B Per Dataset Results

Table 8 gives complete guardian results (zero-shot vs. GEPA at budget 200 and at the best budget) for every dataset, visualized per dataset in Figure 3; Table 9 the six-configuration ablation at budget 200; Table 10 the per-mode budget curves; Table 11 the full configuration $\times$ teacher grid; Table 12 the per-dataset mode comparison at matched *total* evaluations; Table 13 the compute-accounting view of phasing on the hardest structured tasks, with each regime laid out by its actual judge-evaluation count (k-phase $= kB$).

Table 7: The 12 guardrail datasets. Type: PO=prompt-only, PR=prompt–response, DOC=grounded, A/B=pairwise; *Pool* is the upstream split our train/val/test partition draws from (3). Seeded splits of at most 140/160/140, identical across scorers. †Pool is the source's evaluation split, sub-partitioned like any other pool.

Dataset	Task	Type	Source	Pool
toolcall-halluc.	tool-augmented response faithfulness	PR	(Liu et al., 2024)	train
off-topic	system-prompt scope	PO	(Chua et al., 2024)	train
pairwise	A/B preference	A/B	(NVIDIA, 2025)	train
hallumix	hallucination	DOC	(Emery et al., 2025)	train
atbench	agentic-trajectory safety	PR	(Liu et al., 2026)	test†
reward-bench-2	preference (instruction following)	A/B	(Malik et al., 2025)	test†
ragtruth	RAG groundedness	DOC	(Niu et al., 2024)	train
wildguardmix	prompt harm	PO	(Han et al., 2024)	train
beavertails	response safety	PR	(Ji et al., 2023)	train
toxigen	implicit toxicity	PO	(Hartvigsen et al., 2022)	train
assebench	agent safety/security	PR	(Luo et al., 2025)	test
opir	safety classification	PO	(Stepanov, 2026)	train

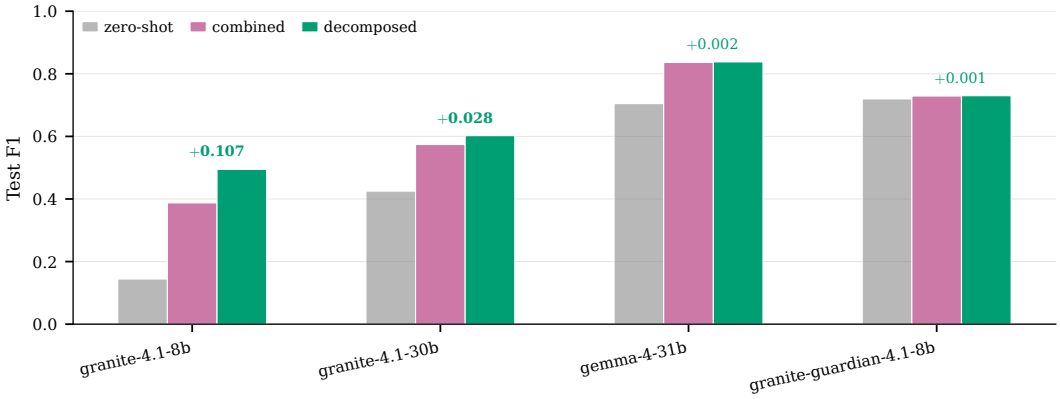

Figure 2: Decomposing the judge prompt helps every judge, most where it is weakest. Macro-mean held-out test F1 over 6 tasks and five seeds for 4 judges spanning 8B–31B and two families (ordered by increasing zero-shot strength). Three bars per judge: the zero-shot seed, the monolithic *combined* prompt, and *decomposed*, the mean over the 5 regimes (crit-only, schema-only, joint, two-phase, three-phase). *(i)* Optimizing a frozen judge's prompt lifts every one, from 0.144 to 0.583 for the weakest (`granite-4.1-8b`) down to a near-null $0.720 \rightarrow 0.733$ for the guardian. *(ii)* Making the decomposition explicit (criterion and scoring schema as separate components) beats the same guidance in one block for every judge (annotated Δ decomposed−combined: +0.107, +0.028, +0.002, +0.001 F1). The sharper matched-compute form, *joint* alone keeping a positive gap over *combined* for all four judges, is in the text.

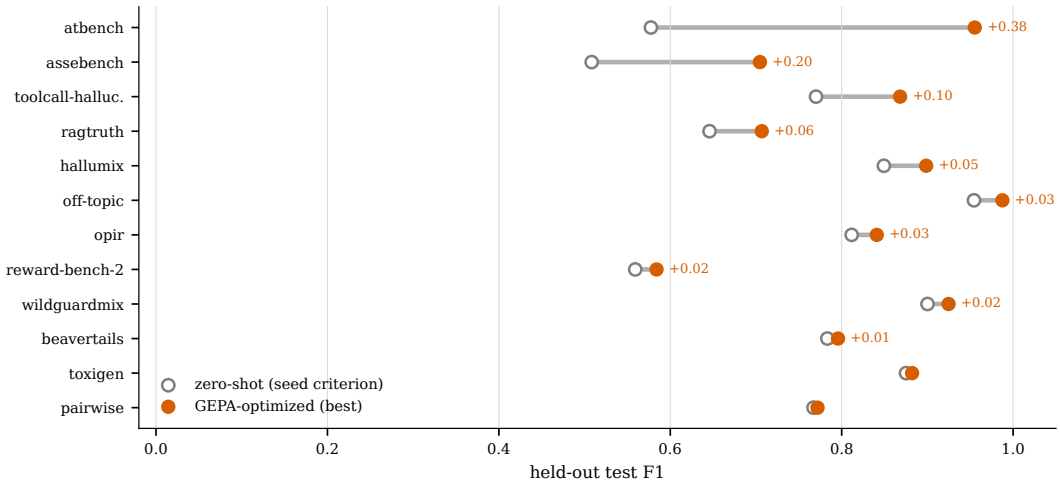

Figure 3: Per-dataset guardian test F1, zero-shot seed vs. GEPA@best, visualizing the guardian columns of Table 8: large gains on the structured tasks, near-null movement where the seed is already near ceiling.

Table 8: Zero-shot vs. GEPA (test F1, mean over 5 seeds; columns differ only in the scoring model). *@200*: best mode/teacher at budget 200; *best*: best over budgets $\{200, \ldots, 3200\}$ (winning mode · budget noted; 2ϕ/3ϕ=two/three-phase). Δ_b =best−@200, rows sorted by it; *best* is a grid maximum, optimistically biased by ≈0.008 (7).

	Zero-shot F1			GEPA F1		
Dataset	granite-8b	granite-30b	guardian-8b	@200	best (mode, budget)	Δ_b
atbench	0.043	0.463	0.577	0.584	**0.955** (3ϕ, 1600)	**+0.371**
assebench	0.142	0.259	0.508	0.543	**0.705** (3ϕ, 3200)	+0.162
toolcall-halluc.	0.000	0.116	0.770	0.796	**0.868** (2ϕ, 1600)	+0.072
opir	0.440	0.753	0.812	0.813	**0.841** (3ϕ, 3200)	+0.028
hallumix	0.120	0.844	0.849	0.873	**0.899** (3ϕ, 1600)	+0.026
off-topic	0.016	0.905	0.954	0.964	**0.987** (3ϕ, 1600)	+0.024
reward-bench-2	0.558	0.457	0.559	0.561	**0.584** (3ϕ, 1600)	+0.023
ragtruth	0.000	0.000	0.646	0.685	**0.707** (2ϕ, 3200)	+0.022
wildguardmix	0.588	0.782	0.900	0.909	**0.925** (3ϕ, 1600)	+0.015
pairwise	0.695	0.654	0.767	0.767	**0.772** (3ϕ, 400)	+0.005
toxigen	0.650	0.814	0.875	0.879	**0.882** (joint, 1600)	+0.003
beavertails	0.528	0.645	0.783	0.796	**0.796** (3ϕ, 200)	+0.000

Table 9: Best-teacher test F1 (mean±std/5 seeds) per configuration: the first four columns are the four modes at budget 200; *joint400/joint600* are budget-matched joint controls spending 400/600 evaluations, not budget-200 results. Sample std, 5 seeds.

Dataset	crit	joint200	twophase	threephase	joint400	joint600
toolcall-hallucination	0.778±0.04	0.796±0.03	0.794±0.05	0.792±0.04	0.808±0.04	0.829±0.03
off-topic	0.960±0.02	0.964±0.02	0.963±0.02	0.962±0.02	0.977±0.01	0.978±0.02
pairwise	0.767±0.03	0.767±0.03	0.767±0.03	0.765±0.03	0.767±0.03	0.766±0.03
hallumix	0.873±0.05	0.867±0.04	0.872±0.05	0.863±0.04	0.889±0.02	0.883±0.01
atbench	0.577±0.05	0.575±0.05	0.578±0.05	0.584±0.05	0.699±0.16	0.601±0.13
reward-bench-2	0.559±0.05	0.561±0.06	0.559±0.05	0.559±0.05	0.563±0.04	0.572±0.06
ragtruth	0.646±0.04	0.665±0.03	0.671±0.04	0.685±0.05	0.672±0.04	0.678±0.05
wildguardmix	0.900±0.02	0.902±0.03	0.903±0.03	0.909±0.03	0.904±0.02	0.911±0.02
beavertails	0.777±0.04	0.780±0.03	0.784±0.04	0.796±0.03	0.792±0.03	0.771±0.04
toxigen	0.876±0.02	0.875±0.02	0.879±0.02	0.876±0.03	0.871±0.03	0.880±0.02
assebench	0.511±0.07	0.508±0.08	0.515±0.12	0.543±0.10	0.508±0.07	0.462±0.13
opir	0.795±0.02	0.813±0.03	0.799±0.04	0.806±0.03	0.801±0.06	0.823±0.05

Table 10: Best-teacher test F1 (mean±std/5 seeds) per mode vs. budget, all 12 datasets. Budget 200 is the main-sweep anchor (*joint=joint200* of Table 9); budgets 400–3200 come from the dedicated budget sweep. Sample std, 5 seeds.

Dataset	Mode	b200	b400	b800	b1600	b3200
toolcall-hallucination	crit	0.778±0.04	0.778±0.04	0.818±0.03	0.820±0.05	0.821±0.06
	joint	0.796±0.03	0.808±0.04	0.851±0.03	0.851±0.04	0.867±0.03
	twophase	0.794±0.05	0.834±0.04	0.857±0.04	0.868±0.03	0.860±0.03
	threephase	0.792±0.04	0.823±0.05	0.854±0.04	0.861±0.03	0.868±0.04
off-topic	crit	0.960±0.02	0.983±0.02	0.983±0.02	0.982±0.02	0.984±0.02
	joint	0.964±0.02	0.972±0.01	0.981±0.02	0.982±0.02	0.984±0.02
	twophase	0.963±0.02	0.979±0.01	0.983±0.01	0.981±0.02	0.982±0.02
	threephase	0.962±0.02	0.981±0.02	0.981±0.02	0.987±0.01	0.983±0.02
pairwise	crit	0.767±0.03	0.767±0.03	0.758±0.04	0.763±0.04	0.758±0.04
	joint	0.767±0.03	0.771±0.03	0.763±0.04	0.771±0.03	0.765±0.04
	twophase	0.767±0.03	0.764±0.03	0.763±0.04	0.765±0.03	0.763±0.03
	threephase	0.765±0.03	0.772±0.04	0.765±0.04	0.767±0.03	0.761±0.04
hallumix	crit	0.873±0.05	0.887±0.02	0.888±0.01	0.889±0.01	0.889±0.02
	joint	0.867±0.04	0.894±0.02	0.886±0.02	0.885±0.01	0.891±0.01
	twophase	0.872±0.05	0.890±0.02	0.895±0.02	0.889±0.02	0.891±0.01
	threephase	0.863±0.04	0.886±0.02	0.892±0.02	0.899±0.02	0.895±0.02
atbench	crit	0.577±0.05	0.575±0.05	0.583±0.05	0.595±0.06	0.572±0.05
	joint	0.575±0.05	0.575±0.04	0.573±0.04	0.803±0.13	0.678±0.17
	twophase	0.578±0.05	0.660±0.03	0.655±0.12	0.684±0.15	0.909±0.08
	threephase	0.584±0.05	0.647±0.08	0.802±0.16	0.955±0.02	0.919±0.06
reward-bench-2	crit	0.559±0.05	0.561±0.06	0.556±0.06	0.554±0.05	0.571±0.06
	joint	0.561±0.06	0.560±0.07	0.568±0.05	0.581±0.08	0.572±0.08
	twophase	0.559±0.05	0.565±0.06	0.572±0.07	0.572±0.05	0.572±0.07
	threephase	0.559±0.05	0.567±0.06	0.580±0.06	0.584±0.05	0.580±0.07
ragtruth	crit	0.646±0.04	0.648±0.04	0.646±0.04	0.645±0.04	0.659±0.06
	joint	0.665±0.03	0.671±0.04	0.698±0.05	0.705±0.02	0.695±0.04
	twophase	0.671±0.04	0.692±0.04	0.697±0.05	0.701±0.03	0.707±0.04
	threephase	0.685±0.05	0.696±0.05	0.703±0.03	0.703±0.05	0.698±0.05
wildguardmix	crit	0.900±0.02	0.901±0.03	0.898±0.03	0.899±0.02	0.907±0.01
	joint	0.902±0.03	0.909±0.01	0.916±0.02	0.907±0.01	0.906±0.01
	twophase	0.903±0.03	0.907±0.02	0.908±0.02	0.914±0.02	0.913±0.01
	threephase	0.909±0.03	0.907±0.01	0.915±0.02	0.925±0.02	0.921±0.02
beavertails	crit	0.777±0.04	0.772±0.04	0.789±0.02	0.773±0.04	0.785±0.05
	joint	0.780±0.03	0.782±0.04	0.781±0.04	0.787±0.05	0.780±0.05
	twophase	0.784±0.04	0.771±0.04	0.781±0.03	0.783±0.02	0.772±0.04
	threephase	0.796±0.03	0.768±0.05	0.769±0.02	0.782±0.04	0.795±0.06
toxigen	crit	0.876±0.02	0.875±0.03	0.878±0.03	0.868±0.02	0.873±0.02
	joint	0.875±0.02	0.874±0.03	0.864±0.02	0.882±0.02	0.862±0.04
	twophase	0.879±0.02	0.875±0.02	0.875±0.03	0.872±0.02	0.868±0.02
	threephase	0.876±0.03	0.876±0.03	0.876±0.03	0.870±0.02	0.877±0.01
assebench	crit	0.511±0.07	0.510±0.08	0.481±0.11	0.518±0.11	0.513±0.17
	joint	0.508±0.08	0.522±0.16	0.578±0.11	0.497±0.07	0.580±0.10
	twophase	0.515±0.12	0.533±0.12	0.462±0.12	0.604±0.15	0.675±0.08
	threephase	0.543±0.10	0.549±0.15	0.588±0.13	0.697±0.12	0.705±0.13
opir	crit	0.795±0.02	0.807±0.03	0.807±0.05	0.800±0.05	0.810±0.04
	joint	0.813±0.03	0.803±0.05	0.799±0.06	0.833±0.02	0.841±0.02
	twophase	0.799±0.04	0.823±0.05	0.822±0.04	0.817±0.04	0.830±0.03
	threephase	0.806±0.03	0.832±0.02	0.837±0.02	0.823±0.04	0.841±0.02

Table 11: Per-dataset configuration × teacher grid at budget 200 (test F1 mean±std/5 seeds). Teacher abbreviations qwen/llama/nemo/g30 are defined in 3. Sample std, 5 seeds.

Dataset	cfg	qwen	llama	nemo	g30
toolcall-hallucination	crit	0.770±0.03	0.777±0.05	0.770±0.03	0.778±0.04
	joint200	0.796±0.03	0.787±0.04	0.771±0.03	0.772±0.03
	twophase	0.776±0.03	0.794±0.05	0.770±0.03	0.792±0.04
	threephase	0.791±0.05	0.774±0.04	0.770±0.03	0.792±0.04
	joint400	0.789±0.03	0.780±0.06	0.808±0.04	0.790±0.03
	joint600	0.815±0.05	0.784±0.02	0.829±0.03	0.777±0.03
off-topic	crit	0.953±0.02	0.953±0.02	0.957±0.02	0.960±0.02
	joint200	0.954±0.02	0.953±0.02	0.954±0.02	0.964±0.02
	twophase	0.953±0.02	0.963±0.02	0.959±0.02	0.960±0.02
	threephase	0.954±0.02	0.961±0.02	0.960±0.02	0.962±0.02
	joint400	0.969±0.02	0.967±0.02	0.977±0.01	0.972±0.03
	joint600	0.978±0.02	0.967±0.02	0.975±0.02	0.978±0.02
pairwise	crit	0.755±0.04	0.767±0.03	0.766±0.03	0.765±0.03
	joint200	0.757±0.04	0.767±0.03	0.767±0.03	0.758±0.04
	twophase	0.767±0.03	0.763±0.03	0.767±0.03	0.761±0.03
	threephase	0.759±0.04	0.765±0.03	0.762±0.03	0.765±0.03
	joint400	0.749±0.05	0.767±0.03	0.762±0.03	0.766±0.03
	joint600	0.766±0.03	0.760±0.04	0.764±0.04	0.759±0.03
hallumix	crit	0.857±0.04	0.862±0.04	0.860±0.04	0.873±0.05
	joint200	0.860±0.03	0.854±0.03	0.860±0.04	0.867±0.04
	twophase	0.850±0.03	0.867±0.04	0.860±0.04	0.872±0.05
	threephase	0.854±0.03	0.857±0.04	0.856±0.04	0.863±0.04
	joint400	0.874±0.02	0.889±0.02	0.875±0.01	0.888±0.02
	joint600	0.876±0.03	0.879±0.02	0.878±0.02	0.883±0.01
atbench	crit	0.575±0.05	0.570±0.05	0.575±0.05	0.577±0.05
	joint200	0.572±0.05	0.575±0.05	0.572±0.05	0.572±0.04
	twophase	0.572±0.05	0.578±0.05	0.574±0.04	0.572±0.05
	threephase	0.584±0.05	0.558±0.07	0.543±0.06	0.495±0.19
	joint400	0.544±0.03	0.569±0.05	0.699±0.16	0.575±0.05
	joint600	0.574±0.05	0.591±0.05	0.601±0.13	0.572±0.05
reward-bench-2	crit	0.551±0.07	0.559±0.05	0.559±0.05	0.559±0.05
	joint200	0.561±0.06	0.550±0.06	0.559±0.05	0.559±0.05
	twophase	0.559±0.05	0.559±0.05	0.553±0.06	0.559±0.05
	threephase	0.552±0.06	0.551±0.06	0.559±0.05	0.553±0.06
	joint400	0.556±0.05	0.561±0.06	0.563±0.04	0.551±0.05
	joint600	0.556±0.05	0.549±0.06	0.572±0.06	0.555±0.05
ragtruth	crit	0.646±0.04	0.646±0.04	0.646±0.04	0.644±0.04
	joint200	0.659±0.04	0.665±0.03	0.662±0.05	0.653±0.04
	twophase	0.659±0.03	0.666±0.04	0.661±0.04	0.671±0.04
	threephase	0.649±0.04	0.670±0.04	0.656±0.04	0.685±0.05
	joint400	0.655±0.04	0.668±0.04	0.661±0.04	0.672±0.04
	joint600	0.649±0.04	0.678±0.05	0.671±0.04	0.655±0.06
wildguardmix	crit	0.882±0.03	0.896±0.02	0.900±0.02	0.900±0.02
	joint200	0.901±0.03	0.895±0.02	0.902±0.03	0.894±0.02
	twophase	0.900±0.02	0.897±0.02	0.894±0.02	0.903±0.03
	threephase	0.907±0.04	0.899±0.03	0.909±0.03	0.896±0.02
	joint400	0.900±0.02	0.895±0.02	0.904±0.02	0.900±0.02
	joint600	0.900±0.02	0.893±0.02	0.905±0.03	0.911±0.02
beavertails	crit	0.748±0.06	0.769±0.04	0.770±0.07	0.777±0.04
	joint200	0.761±0.04	0.780±0.03	0.761±0.05	0.764±0.04
	twophase	0.753±0.06	0.776±0.05	0.784±0.04	0.776±0.04
	threephase	0.796±0.03	0.776±0.05	0.761±0.05	0.767±0.04
	joint400	0.792±0.03	0.759±0.04	0.766±0.06	0.782±0.03
	joint600	0.765±0.06	0.761±0.04	0.765±0.03	0.771±0.04
toxigen	crit	0.869±0.02	0.876±0.02	0.866±0.03	0.869±0.02
	joint200	0.863±0.03	0.875±0.02	0.863±0.03	0.874±0.02
	twophase	0.854±0.03	0.879±0.02	0.876±0.02	0.873±0.02
	threephase	0.872±0.02	0.863±0.04	0.863±0.02	0.876±0.03
	joint400	0.871±0.03	0.865±0.02	0.868±0.02	0.866±0.03
	joint600	0.880±0.02	0.873±0.03	0.863±0.03	0.869±0.02
assebench	crit	0.429±0.12	0.511±0.07	0.511±0.07	0.464±0.12
	joint200	0.491±0.09	0.506±0.08	0.459±0.10	0.508±0.08
	twophase	0.497±0.08	0.489±0.09	0.515±0.12	0.471±0.10
	threephase	0.406±0.12	0.506±0.09	0.543±0.10	0.453±0.09
	joint400	0.406±0.12	0.508±0.07	0.505±0.10	0.454±0.13
	joint600	0.458±0.10	0.462±0.13	0.443±0.12	0.417±0.13
opir	crit	0.764±0.06	0.772±0.06	0.773±0.05	0.795±0.02
	joint200	0.757±0.07	0.798±0.05	0.813±0.03	0.799±0.03
	twophase	0.769±0.07	0.778±0.05	0.777±0.03	0.799±0.04
	threephase	0.793±0.05	0.806±0.03	0.797±0.06	0.802±0.03
	joint400	0.801±0.06	0.794±0.04	0.794±0.04	0.792±0.04
	joint600	0.823±0.05	0.779±0.05	0.804±0.06	0.797±0.05

Table 12: No optimization mode dominates at matched compute (test F1, best teacher per (dataset, mode); judge `granite-guardian-4.1-8b`). All six modes are given a *common* budget of 3200 evaluations (two-phase at b1600, three-phase at b1067), so every column spends the same total. **Bold** = best in its row; *phasing is not a free win once compute is matched* (Table 10).

Dataset	crit-only	schema-only	joint	combined	2-phase	3-phase
toolcall-hallucination	0.821	0.851	0.867	**0.869**	0.868	0.866
off-topic	**0.984**	0.975	**0.984**	0.979	0.981	0.980
pairwise	0.758	0.768	0.765	**0.769**	0.765	0.765
hallumix	0.889	0.867	**0.891**	0.879	0.889	0.886
atbench	0.572	**0.861**	0.678	0.809	0.684	0.678
reward-bench-2	0.571	0.564	0.572	0.567	0.572	**0.575**
ragtruth	0.659	0.698	0.695	0.699	**0.701**	0.696
wildguardmix	0.907	**0.916**	0.906	0.905	0.914	0.912
beavertails	**0.785**	0.776	0.780	0.766	0.783	0.766
toxigen	0.873	0.869	0.862	0.876	0.872	**0.883**
assebench	0.513	0.620	0.580	**0.684**	0.604	0.525
opir	0.810	**0.842**	0.841	0.802	0.817	0.834

Table 13: Phasing is a compute effect. Test F1 with judge `granite-guardian-4.1-8b` by *actual judge evaluations*: a k-phase run at budget B costs kB. At matched compute ($\approx$3200) joint (b3200) and two-phase (b1600) are indistinguishable (0.732 vs. 0.735); joint saturates by ~ 1600, phasing gaining only with *more* compute.

Regime	Budget	Evaluations	Test F1
joint	800	800	0.700
joint	1600	1600	0.738
joint	3200	3200	0.732
two-phase	800	1600	0.699
two-phase	1600	3200	0.735
two-phase	3200	6400	0.796
three-phase	800	2400	0.757
three-phase	1600	4800	0.808