CAN SATURATED BENCHMARKS BE RECOVERED BY LLM-AS-A-META-JUDGE?

Anonymous authors
Paper under double-blind review

ABSTRACT

Widely used language-model benchmarks are increasingly saturated, with frontier systems receiving near-tied scores that standard metrics cannot resolve. Rather than constructing harder alternatives, we ask whether existing benchmarks can regain informativeness through improved evaluation of the same candidate outputs. Therefore, we present *S*eeded *E*limination with *A*daptive *L*LM-as-a-Meta-Judge (SEAL), which combines a seeded tournament with evolving checklist criteria anchored to fixed task-level principles. To validate the recovered rankings, we construct a human-verified dense reference (HVR) by auditing exhaustive pairwise judgment traces against task and response evidence. Experiments span multiple saturated benchmarks such as code generation, mathematical reasoning, knowledge-intensive question answering, and tool calling. SEAL achieves the highest or joint-highest Spearman agreement with HVR across all settings, averaging 0.733 compared with 0.623 for exhaustive pairwise judging. On code generation, it uses 57.5% fewer judge calls and 65.8% fewer tokens than exhaustive comparison. Beyond the tournament, reusing evolved rubrics in pointwise scoring improves HVR agreement, raising mean correlation from 0.565 to 0.647. Together, these results demonstrate its effectiveness in adapting both comparisons and criteria to recover task-grounded ranking signal from saturated benchmarks.

1 INTRODUCTION

Benchmarks provide a common basis for measuring progress in large language models (LLMs) and comparing their capabilities (Wang et al., 2018; Liang et al., 2023). As model capabilities advance rapidly, however, many benchmarks lose their ability to distinguish leading systems. A systematic study of 60 LLM benchmarks finds that nearly half exhibit saturation (Akhtar et al., 2026). Near-ceiling or near-tied scores consequently provide diminishing guidance for model comparison. Developing harder benchmarks is one response, but requires new task construction and validation. This raises a complementary question: *can we recover the discriminative value of existing benchmarks by improving how their outputs are evaluated?*

This possibility arises because benchmark resolution depends on not only the questions but also the evaluation pipeline. Consider a mathematics task that asks for both a numerical answer and a supporting derivation. Two responses may reach the same correct answer, yet only one provides a valid derivation; the other relies on an unjustified intermediate step. An answer-only metric would treat them equally despite this difference in how well they satisfy the task. LLM-based evaluation can examine the reasoning behind such outcomes (Zheng et al., 2023). Pairwise comparison, for example, allows a judge to contrast the two derivations directly rather than score each response in isolation. Changing the evaluation pipeline can thus expose task-relevant distinctions that a coarse outcome leaves unresolved.

Changing the comparison format alone, however, leaves a central difficulty intact: the judge must still discover which differences matter. When evaluation uses only broad criteria, this depends on the judge's ability to infer task-specific requirements and apply them consistently; limited reasoning and evaluator biases can distort the resulting preferences (Zheng et al., 2023; Wang et al., 2024). Moreover, a distinction identified in one comparison is not automatically carried into the next. Even a saved rationale explains a local decision without necessarily providing a criterion that later evalu-

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.

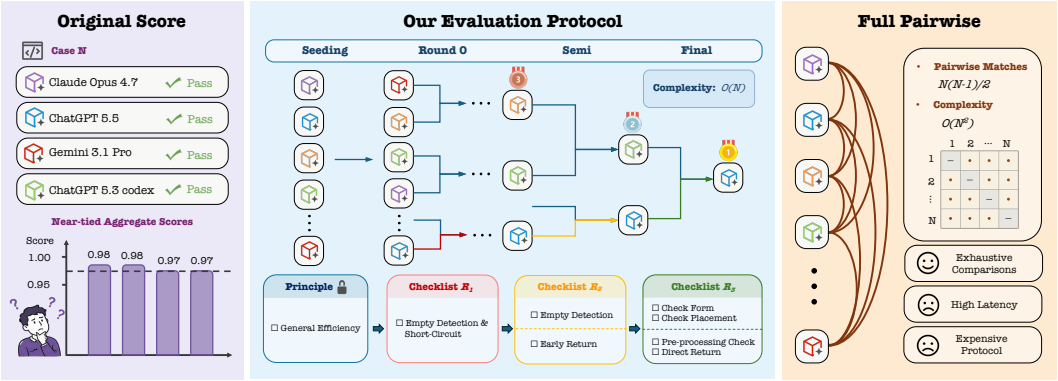

Figure 1: SEAL combines seeded elimination with checklist evolution under fixed task-level principles. Later rounds refine the criteria used to compare surviving candidates.

ations can reuse. Repeating comparisons can therefore repeat the work of discovering distinctions, while exhaustive pairwise judging incurs $\binom{N}{2}$ comparisons per task.

We address this gap by treating the evaluation criteria themselves as an evolving output of the pipeline. Explicit rubrics already structure LLM evaluation (Liu et al., 2023; Kim et al., 2024); our question is how to refine them as comparisons reveal what a broad rubric fails to resolve. The key idea is to convert useful distinctions into concrete checklist items, retain them across rounds, and apply them to subsequent comparisons. For the mathematics example, an unjustified intermediate step motivates an explicit check that each step in the derivation follows from the stated assumptions or preceding steps. The checklist thus records concrete, task-grounded reasons for distinguishing otherwise similar outputs. We realize this idea in SEAL (Figure 1). A listwise judge seeds a single-elimination tournament, concentrating comparisons on increasingly competitive candidates. Each match uses fixed task-level principles and an evolving checklist. A meta-judge refines the checklist from comparison evidence, while structured principle votes determine advancement. The principles anchor the evaluation objectives; evolution turns them into more specific checks. Together, the tournament and evolving rubrics address where to spend the judging budget and how to retain the distinctions it uncovers.

Retaining these criteria also makes their contribution testable beyond the tournament. We freeze each task's evolved rubric and reuse it in pointwise scoring. Comparing this setting with rubric-free and fixed-rubric scoring asks whether comparison-derived criteria improve individual-response evaluation, separating rubric refinement from the schedule that produced it.

Moreover, recovering ranking signal requires more than separating previously tied candidates: the recovered distinctions must also be justified by the task and response evidence, since automated protocols may share the same judgment errors or unsupported preferences. To validate the resulting rankings, we construct a *human-verified dense reference* (HVR) from pairwise traces. Human reviewers retain supported judgments, correct unsupported preferences, and assign criterion-level ties when the task admits multiple equally reasonable interpretations. The verified decisions form a common reference for assessing all evaluation protocols. **In summary, our main contributions are as follows**:

- We propose an evaluation protocol SEAL that recovers ranking signal from saturated benchmarks by combining seeded elimination with rubric evolution under fixed task-level principles. The protocol turns distinctions uncovered during comparison into explicit, reusable criteria while keeping benchmark tasks and candidate outputs unchanged.
- We construct a human-verified reference for validating evaluation rankings. Human review of exhaustive pairwise traces grounds comparison decisions in task and response evidence, corrects unsupported preferences, and assigns criterion-level ties when underspecified requirements admit multiple reasonable interpretations.
- Across various benchmark domains, SEAL achieves the highest or joint-highest ranking agreement with the human-verified reference among all the baselines. Reusing evolved rubrics further improves the pointwise evaluation, supporting their value beyond the tournament. Cost analysis on HumanEval further shows a 65.8% token reduction relative to exhaustive pairwise evaluation.

2 SEAL: SEEDED ELIMINATION WITH ADAPTIVE LLM-AS-A-META-JUDGE

Let x denote a benchmark task and $\mathcal{Y}_x = \{y_1, \ldots, y_N\}$ the outputs from a fixed set of systems. A native benchmark metric maps each output to a coarse task-level outcome. We call an evaluation region *saturated* when these outcomes provide insufficient separation within the candidate pool, for example when many systems receive the same result or top-tier aggregate scores fall within a narrow band (Akhtar et al., 2026). Our goal is to recover additional ranking signal while retaining the original tasks, candidate outputs, and native outcomes.

SEAL produces a task-level ranking and a task-specific checklist. A listwise judge first assigns seed tiers, which initialize a single-elimination bracket. Pairwise matches use fixed task-level principles and the current checklist. After each round, an LLM meta-judge refines the checklist from comparison evidence for subsequent matches. Once the tournament finishes, task-level ranks are aggregated across the benchmark. Figure 1 summarizes how the comparison schedule and checklist refinement work together.

2.1 SEEDED ELIMINATION

For each saturated task, SEAL first applies a listwise judge to assign candidate outputs to coarse seed tiers. The seeding prompt groups semantically equivalent answers and requests substantive differences rather than an exact order within each tier. These seeds initialize a single-elimination bracket, spreading likely strong candidates across the tournament and reducing early collisions among close competitors. This design follows the broader idea that informative subsets of pairwise comparisons can support ranking without exhaustive enumeration (Jamieson & Nowak, 2011; Wauthier et al., 2013).

Each match compares two outputs using the task, fixed principles, and the current checklist. Seeding supplies a scheduling heuristic; pairwise judgments determine advancement. Early rounds use broad checklist items to separate obvious differences. As stronger candidates survive, their remaining differences tend to become more specific, motivating finer criteria in later rounds. The bracket therefore determines which outputs are compared, while checklist evolution determines which task-grounded distinctions those comparisons inspect.

Seeding does not directly determine the final ranking. Outputs assigned to the same tier can be separated when direct comparison reveals a substantive difference. Evidence is retained from every match, including matches involving early eliminations. Candidates eliminated in the same round are ordered by their accumulated principle-level vote margin, rather than by their initial seed alone. This preserves information beyond the identity of each match winner when constructing the task-level ranking.

2.2 FIXED PRINCIPLES AND PAIRWISE JUDGMENTS

The rubric has two layers. *Principles* define stable evaluation dimensions, such as functional correctness and interface compatibility for code, or function selection and argument correctness for tool use. *Checklist items* translate these principles into concrete questions about the current task. The seed checklist contains one item per principle; subsequent checklists refine their coverage without redefining the principles themselves. Appendix A provides the domain-specific principles and prompt templates.

For each match, the judge returns structured left, right, or tie votes for the principles and checklist items, together with confidence and principle-level rationales. The pairwise prompt expresses the signed principle margin as

$$m(A, B) = \sum_{k=1}^{K} c_k v_k, \qquad v_k \in \{-1, 0, +1\}, \quad c_k \in [0, 1], \tag{1}$$

where negative and positive votes favor A and B, respectively. The backend derives the match outcome from structured votes instead of an unconstrained overall score. Checklist judgments provide supporting evidence rather than additional independent terms in this margin. Consequently, splitting a broad item into several checks does not mechanically increase its principle's voting weight.

The distinction between principles and checklist items also constrains what refinement should reward. A code checklist can ask whether the response handles every input explicitly allowed by the specification; a reasoning checklist can ask whether each step supports the conclusion. Neither should introduce a preference solely for an unspecified convention or for additional explanation when the task requires only an answer. The objective is a more precise application of the original requirements. Checklist length alone is not evidence of improved evaluation.

2.3 LLM-AS-A-META-JUDGE

The meta-judge does not directly assign final scores or rankings; instead, it refines the checklist used in subsequent matches. This differs from meta-judging for self-improving alignment, where meta-judgments improve reward or preference-learning signals (Wu et al., 2025). After each tournament round, SEAL summarizes match evidence and asks the meta-judge which fixed principles remain under-resolved by the current evidence. It then proposes finer checklist items grounded in those principles and tailored to the remaining candidate pool.

Checklist refinement uses three complementary operations. *Split* decomposes a broad item into more specific checks; *adversarial addition* introduces a probe for a distinction exposed by comparison; and *merge* consolidates overlapping items. Each resulting item remains associated with a fixed principle. For adversarial addition, the prompt includes the task, competing responses, existing checklist, and loser-side principle evidence. The proposed item describes an observable distinction and illustrates the quality difference with short response snippets. Including the current checklist helps avoid proposing a criterion that is already represented.

These operations implement the original rank-resolution principle: different regions of a ranking require different rubric granularity. A broad criterion about specification coverage may suffice for an incomplete response, whereas two otherwise complete responses may require an explicit check of a particular argument, boundary condition, or logical branch. The meta-judge turns such distinctions into checklist items that later comparisons can apply directly, rather than requiring the judge to rediscover them in each match.

Adapting only the checklist layer keeps the evaluation objectives stable across rounds. The saved history records which criteria were available at each stage and how they were refined. This makes it possible to trace a preference back to the task requirement and the response evidence that motivated the corresponding item. The overall procedure thus produces both a ranking and an inspectable record of the criteria used to obtain it.

2.4 AGGREGATION, BUDGET, AND RUBRIC REUSE

After all matches for a task finish, SEAL converts the bracket into a full task-level ranking. Tournament outcomes establish the elimination order; candidates eliminated in the same round are ordered by their accumulated principle-level vote margin across the matches they played. For rank r_{xm} among $N_x > 1$ candidates, the normalized Borda score is

$$b_{xm} = 1 - \frac{r_{xm} - 1}{N_x - 1}. \tag{2}$$

Benchmark-level scores average these values across saturated tasks, with the mean principle-level margin resolving remaining ties. Rank normalization makes task-level orderings comparable before aggregation.

For N candidates, exhaustive pairwise judging, denoted FULLPAIR, requires $\binom{N}{2}$ comparisons. SEAL instead uses one seeding call and $N - 1$ tournament matches, together with meta-judge refinement calls. The comparison backbone is linear in the number of candidates; the additional refinement budget is spent on improving the criteria used for close comparisons. Calls, tokens, and elapsed time characterize different aspects of this budget and are distinguished in the experimental analysis.

Finally, we freeze each task's evolved checklist and score responses independently with the saved criteria, without further evolution. Keeping tasks and outputs fixed isolates reuse across evaluation formats: criteria discovered in pairwise comparison are applied to individual-response scoring.

3 HUMAN-VERIFIED DENSE REFERENCE

Recovering distinctions is useful only if the resulting preferences follow task evidence. Agreement between automated protocols cannot establish this when judges share biases or overlook the same requirement. We therefore construct a human-verified dense reference (HVR) to validate evaluation rankings. FullPair provides exhaustive comparison traces under fixed principles and the seed checklist. Human reviewers examine these traces against the original tasks and responses, retaining justified decisions and correcting unsupported preferences; FullPair's automated rankings remain an evaluated baseline.

Human Review and Coverage. Seven reviewers verified the pairwise records using task requirements, candidate responses, and available answers or execution evidence. Comparisons involving each model were distributed across reviewers as randomly as practicable to limit the concentration of individual preferences. Each pair received one human review. The final reference retains 12,236 of 12,406 available pairs (98.63%): 4,410 for code generation, 4,185 for mathematical reasoning, 3,056 for knowledge-intensive question answering, and 585 for function calling. The 170 exclusions comprise 36 assumption-dependent decisions and 134 unresolved cases. Appendix C.3 reports the data card and release construction.

Task-grounded Preferences. Review checks whether the criterion follows from the task, whether the cited behavior occurs in the response, and whether that behavior supports the preference. Passing native tests is relevant but does not settle requirements those tests omit. If the question admits multiple equally reasonable interpretations, a criterion distinguishing only those interpretations receives a tie; supported differences in other dimensions remain active. Wholly inapplicable criteria are excluded. Insufficient evidence remains unresolved rather than being converted into a tie. Each record preserves the original decision, verified label, rationale, and response binding.

Validation Role. The verified comparisons provide a common reference for assessing benchmark rankings, with equal aggregate scores assigned average ranks. Labels are reserved for evaluation rather than checklist evolution or pointwise scoring. Because reviewers inspect existing traces, HVR is an audit-based reference rather than an independent blind preference collection. Its coverage and criterion-level evidence make the basis for comparison explicit, while leaving reviewer agreement and the effects of trace anchoring as questions for further validation.

4 EXPERIMENTS

Our experiments ask whether the revised pipeline recovers rankings supported by human verification, whether evolved criteria retain value outside the tournament, and what evaluation budget this requires. We first examine ranking agreement, then the mechanism and reuse of rubric evolution, and finally cost and stability. FullPair is an automated baseline throughout; HVR supplies the validation reference.

Benchmarks and Models. We evaluate eight strong candidate models per domain on HumanEval (Chen et al., 2021), GSM8K (Cobbe et al., 2021), MMLU (Hendrycks et al., 2021), and BFCL-v2 (Patil et al., 2025). They cover code generation, mathematical reasoning, knowledge-intensive question answering, and function calling. The tournament and rubric-reuse runs contain 162, 150, 110, and 22 tasks, respectively. BFCL-v2 uses the single-call `live_simple` subset. Appendix B lists candidate pools and detailed leaderboard records; Appendix C documents reference and reuse coverage.

Baselines and Metrics. ORIGINAL uses the native metric; POINTWISE scores responses independently; FIXEDRUB adds a fixed machine-generated rubric; LISTWISE ranks all outputs in one call; FLATBRK uses seeded elimination without checklist evolution; and FULLPAIR compares all available pairs with the seed checklist. These settings vary scoring format, comparison schedule, and evaluation criteria (Liu et al., 2023; Zheng et al., 2023; Kim et al., 2024). We measure benchmark-ranking agreement with HVR using Spearman's ρ, assigning average ranks to ties and giving the four benchmarks equal weight in the mean. Resolution gain describes numerical score separation rather than judgment validity and is reported separately in Appendix C.2. Judge settings, principles, and prompts appear in Appendix A.

4.1 RANKING AGREEMENT WITH HUMAN VERIFICATION

Protocol	HumanEval	GSM8K	MMLU	BFCL-v2	Mean
Original	0.524	-0.299	0.667	0.405	0.324
Pointwise	**0.952**	-0.072	0.810	0.571	0.565
FixedRub	0.190	0.263	**0.929**	0.500	0.471
Listwise	0.571	0.252	0.690	0.476	0.497
FlatBrk	0.905	0.108	0.905	0.500	0.605
FullPair	0.881	0.275	**0.929**	0.405	0.623
SEAL	**0.952**	**0.359**	**0.929**	**0.690**	**0.733**

Table 1: Spearman agreement with HVR. Mean weights the four benchmarks equally. Bold marks the highest agreement in each column, including ties.

SEAL achieves the highest or joint-highest agreement on all four benchmarks (Table 1), with a mean of 0.733 versus 0.623 for FullPair and 0.605 for FlatBrk. It ties Pointwise on HumanEval and FixedRub and FullPair on MMLU, while leading on GSM8K and BFCL-v2. Exhaustive comparison therefore does not automatically produce the ranking most consistent with human verification: broader coverage still depends on the quality of each judgment.

The improvements are largest where the native ranking is less informative. On GSM8K, Original has negative agreement (-0.299) and Pointwise remains slightly negative (-0.072); SEAL reaches 0.359, exceeding FullPair's 0.275. On BFCL-v2, agreement rises from 0.405 for Original and FullPair to 0.690. The remaining variation across domains is substantial: GSM8K correlation is still moderate, whereas HumanEval and MMLU admit several competitive evaluators. These results support adapting evaluation to the task and candidate pool, rather than assuming that one scoring format is uniformly sufficient.

The distinction between score spread and validity is consequential. Listwise has greater resolution gain than SEAL in every domain, but lower HVR agreement throughout (Appendix C.2). Conversely, SEAL improves BFCL-v2 agreement without widening scores relative to Original. Recovering useful signal requires the direction of a preference to follow task evidence; numerical separation alone is insufficient.

4.2 WHAT DOES CHECKLIST REFINEMENT CAPTURE?

Relative to FlatBrk, SEAL improves agreement by 0.047, 0.251, 0.024, and 0.190 on HumanEval, GSM8K, MMLU, and BFCL-v2. This comparison holds the seeded-elimination design fixed and supports the adaptive evaluation pipeline. Because evolution also consumes additional judging effort, it is not an equal-budget isolation of rubric quality. The reuse experiment below provides a complementary test by applying the resulting criteria in a different scoring format.

From a Missed Requirement to an Explicit Check. HumanEval/124, `valid_date`, requires the literal format `mm-dd-yyyy`. Two saved responses, from GPT-5 and Qwen3-Coder-Plus, pass the supplied native tests. However, their behavior differs on `2-29-2020`: GPT-5 rejects the one-digit month, whereas Qwen accepts it after splitting the string and converting the fields to integers. This difference follows from an explicit format requirement, independently of preferences about coding style. Re-executing the supplied tests and this additional input verifies the distinction.

The rubric history makes the corresponding refinement concrete. The seed asks broadly about specification details that visible tests miss. After the first round, an evolved item explicitly requires two month digits, two day digits, and four year digits. The final checklist specifies a ten-character string with hyphens in the required positions. The progression is therefore task requirement $\rightarrow$ concrete response behavior $\rightarrow$ reusable validation item. It shows how a broad correctness principle can become an executable question for later judgments; it does not assume that every added item improves the ranking.

Keeping Refinement within the Task. HumanEval/92 provides a complementary boundary case. Its integer and sum requirements support checking all inputs and all sum relationships, but an evolved preference between exact-type checks and `isinstance` relies on conventions about sub-

classes and Booleans that the task does not settle. Under HVR, that criterion supplies no preference while the explicit requirements remain active. Together, the cases distinguish a useful refinement from an unsupported addition without reducing evaluation to native test outcomes.

4.3 Reusing Evolved Rubrics in Pointwise Evaluation

The second output of the tournament is a set of task-specific evaluation criteria. We freeze each task's final checklist and give it to a pointwise judge, which scores each response independently without further evolution. Pointwise + ER thus reuses comparison-derived criteria on the same tasks and responses. Table 2 compares this setting with plain Pointwise and FixedRub.

Freezing the checklist separates criterion discovery from its application. For example, an item requiring two-digit date fields can be checked against a single program, even if the omission was first identified by comparing two programs. At reuse time, the evaluator receives the resulting criterion without needing another opponent or a new tournament. The comparison therefore asks whether the saved checklist carries useful information into an independent scoring format, without additional criterion updates at that stage.

Rubric setting	HumanEval	GSM8K	MMLU	BFCL-v2	Mean
Pointwise	**0.952**	-0.072	0.810	0.571	0.565
FixedRub	0.190	**0.263**	**0.929**	0.500	0.471
Pointwise + ER	0.905	0.199	0.874	**0.608**	**0.647**
Δ over Pointwise	-0.047	+0.271	+0.064	+0.037	+0.081

Table 2: Frozen evolved rubrics (ER) reused in pointwise scoring. Values are Spearman agreement with HVR; the final row shows the change from plain Pointwise. Means and changes are computed before rounding.

Evolved rubrics improve pointwise agreement on three benchmarks and decrease it on HumanEval. The largest gain is on GSM8K ($+0.271$), followed by MMLU ($+0.064$) and BFCL-v2 ($+0.037$); HumanEval decreases by 0.047. Mean agreement increases from 0.565 to 0.647. Pointwise + ER also exceeds FixedRub on average (0.647 versus 0.471), although FixedRub leads on GSM8K and MMLU. The benefit is therefore domain-dependent, rather than a consequence of supplying any explicit rubric.

These results support reuse across evaluation formats: distinctions accumulated during pairwise judging can help assess a single response later. Since the same tasks and responses contribute to rubric formation and reuse, the experiment establishes this within-pool reuse rather than generalization to unseen candidates or tasks. The application stage completes 3,550 scoring calls; rubric acquisition is a separate cost. Appendix C preserves the full scores and coverage records, including the 443 of 444 final checklists that differ from their seeds.

4.4 Agreement and Evaluation Budget

Figure 2 visualizes the trade-off between mean agreement and estimated mean total token consumption across the four benchmarks. The cheaper protocols reduce token usage substantially, but their ranking quality remains limited. Listwise and FixedRub require the fewest tokens, yet achieve only about 0.50 mean agreement, while Pointwise improves agreement to roughly 0.57 at a moderate increase in cost. FlatBrk further improves agreement to about 0.60, but still leaves a considerable gap to SEAL. SEAL occupies the high-agreement, low-cost region of the trade-off. By combining a tournament backbone with adaptive checklist refinement, it concentrates additional computation on comparisons that remain difficult to resolve rather than exhaustively evaluating all candidate pairs. As a result, SEAL achieves the highest mean agreement ($\rho \approx 0.73$) with an estimated mean token consumption of only about 9M, compared with roughly 26M for FullPair. Overall, these results show that the adaptive tournament substantially improves token efficiency.

Table 3 summarizes protocol call counts for eight candidates. Table 4 reports a HumanEval budget estimate in which SEAL uses 65.8% fewer tokens than FullPair. Prompt sizes, representative response lengths, and output-token assumptions determine these estimates. The 1,926 recorded

judging calls average 11.89 per task versus 28 for FullPair; they exclude a complete accounting of evolution and local-critique requests. Calls and estimated tokens therefore characterize the evaluation budget, without measuring wall-clock latency. Tables 4a and 4b separate token consumption from proxy dollar costs. Rubric acquisition and later reuse should be accounted for separately.

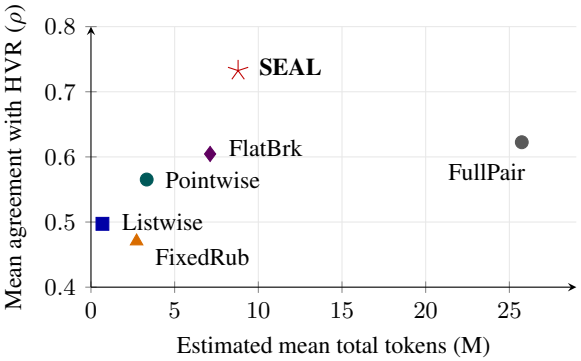

Figure 2: Four-benchmark mean agreement and estimated mean total tokens. SEAL lies near the high-agreement, low-cost region by combining a tournament backbone with adaptive checklist refinement.

Method	Calls / task	Backbone
Original	0	$O(1)$
Listwise	1	$O(1)$
Pointwise	8	$O(N)$
FixedRub	$8 + 1/M$	$O(N)$
FlatBrk	8	$O(N)$
FullPair	28	$O(N^2)$
SEAL	$\sim 11.9^{\dagger}$	$O(N)$

Table 3: Calls for eight candidates. M tasks share one fixed rubric. †: recorded judging and refinement calls.

Protocol	Calls	Input	Output	Total
Listwise	162	0.69	0.32	1.00
Pointwise	1,296	4.36	0.49	4.86
FixedRub	1,297	2.81	1.17	3.98
FlatBrk	1,296	8.60	1.79	10.39
FullPair	4,536	30.99	6.58	37.57
SEAL	1,926	10.63	2.20	12.83

(a) Reported calls and estimated token totals (M).

Protocol	Cost	Cost / 1k	Rel.
Listwise	6.74	42	0.04×
Pointwise	17.10	106	0.10×
FixedRub	25.46	157	0.15×
FlatBrk	47.62	294	0.27×
FullPair	173.79	1,073	1.00×
SEAL	58.67	362	0.34×

(b) Proxy USD costs; /1k extrapolates to 1,000 tasks.

Table 4: HumanEval budget estimates for 162 tasks. Dollar estimates use a fixed proxy of USD 2/M input and USD 17/M output tokens, rather than provider billing.

4.5 Stability under Task Sampling and Repeated Runs

Stability asks whether a ranking changes under task sampling or repeated execution, separately from whether it agrees with HVR. The reported 80-task subsampling experiment uses 30 seeds on HumanEval, GSM8K, and MMLU (Table 5). Its mean correlations with the full-task ranking are 0.97, 0.95, and 0.98. Four full HumanEval runs (Table 6) additionally yield six pairwise comparisons, with mean correlation 0.972, minimum 0.952, and identical top-ranked models in all six comparisons.

Benchmark	Full (sampled)	Agreement	Top-1 recovery
HumanEval	162 (80)	0.97 (min 0.90)	100%
GSM8K	150 (80)	0.95 (min 0.92)	100%
MMLU	110 (80)	0.98 (min 0.91)	100%

Table 5: Reported 80-task subsamples over 30 seeds, relative to the full-task ranking.

Runs	Mean ρ	Min ρ	Top-1 match
4	0.972	0.952	6/6

Table 6: Reported HumanEval run-to-run agreement across six comparisons.

These analyses indicate consistency under the tested perturbations, but they are not independent measurements of ranking validity. A stable judge can repeatedly prefer the same unsupported distinction; human verification addresses that separate concern. Likewise, aggregate correlations do not quantify how an early elimination affects later comparisons. The results motivate retaining both

tournament traces and checklist histories so that such local decisions remain available for examination.

5 Related Work

Benchmark Saturation. GLUE and SuperGLUE measure model progress (Wang et al., 2018; 2019), while BIG-bench and HELM broaden capability coverage (Srivastava et al., 2023; Liang et al., 2023). Dynamic benchmark construction collects new challenges as models improve (Kiela et al., 2021). Saturation and reuse studies examine static metrics and recovering discrimination (Ott et al., 2022; Akhtar et al., 2026; Etzine et al., 2025). SEAL keeps questions and outputs fixed while refining evaluation criteria and comparisons.

LLM Judges and Adaptive Criteria. MT-Bench and Chatbot Arena support judge-based comparison (Zheng et al., 2023; Chiang et al., 2024). G-Eval generates evaluation steps from supplied criteria and uses structured scoring to assess generated text (Liu et al., 2023). Prometheus instead trains an open evaluator to produce feedback and scores conditioned on a rubric and reference answer (Kim et al., 2024). Both make evaluation criteria explicit, but improving their application is distinct from discovering which task-specific checks a close comparison needs. Evaluator biases further motivate validating preferences against evidence rather than treating a judge's rationale as sufficient justification (Wang et al., 2024; Dubois et al., 2024). Autorubric makes criterion design and judge configuration reusable and auditable, integrating calibration and reliability diagnostics (Rao & Callison-Burch, 2026). Meta-Rewarding improves judges through meta-judgments (Wu et al., 2025), and GenRubric trains rubric generators through reinforcement learning (Chen et al., 2026). SEAL evolves checklists during evaluation under fixed principles without retraining the judge. It validates rankings against HVR, and tests whether evolved rubrics transfer to pointwise scoring.

Efficient Ranking. Active ranking recovers orderings from selected comparisons (Jamieson & Nowak, 2011; Wauthier et al., 2013). Recent LLM evaluation methods use Swiss-style matchmaking to reduce round-robin costs (Xu et al., 2025) or knockout tournaments for iterative assessment (Sandan et al., 2025). SEAL couples seeded elimination with checklist refinement, so later matches reuse criteria discovered in earlier comparisons. The comparison schedule and the evaluation criteria address complementary choices: which responses to compare, and which distinctions those comparisons should assess. Our experiments therefore evaluate both ranking quality and resource use, using human-reference agreement, rubric reuse, stability, and estimated token consumption.

6 Conclusion

We presented SEAL, an evaluation protocol that recovers ranking signal from saturated benchmarks while keeping their tasks and candidate outputs fixed. Seeded elimination concentrates comparisons on increasingly competitive candidates, while a meta-judge converts task-specific distinctions into explicit checklist items under fixed evaluation principles. This process allows evidence uncovered in one comparison to inform subsequent judgments, making rubric refinement an accumulated product of evaluation. Across four domains, SEAL achieves the highest or joint-highest agreement with a human-verified dense reference, with mean agreement of 0.733 compared with 0.623 for exhaustive pairwise judging. These results show that exhaustive comparison alone does not guarantee more accurate rankings: the criteria applied to those comparisons also matter. The HumanEval budget estimate further indicates 65.8% lower token consumption than pairwise evaluation. Beyond the tournament, reusing frozen evolved rubrics in pointwise evaluations raises mean agreement from 0.565 to 0.647. This finding provides evidence that comparison-derived criteria retain value when the same responses are evaluated independently.

Together, these findings highlight rubric evolution as a way to recover the discriminative value of saturated benchmarks. In SEAL, each comparison provides evidence for refining what subsequent judgments should examine: broad criteria become more specific as the evaluation encounters distinctions among strong candidates. Fixed task-level principles anchor this evolution, while explicit checklist updates preserve the resulting criteria across rounds. This perspective treats evaluation as a process that develops its own criteria, extending the usefulness of existing benchmarks through standards that become more informative as comparisons accumulate.

AI USE STATEMENT

Generative AI tools were used to assist with drafting and polishing the manuscript. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the paper's content. The use of language models as components of the proposed evaluation protocol is described in the methodology and experimental settings.

ETHICS STATEMENT

This work studies the evaluation of language models using existing benchmark tasks and candidate responses. To reduce the risk of treating evaluator biases as quality differences, we construct a human-verified reference by auditing automated judgments against task requirements and response evidence. Criteria that admit multiple equally reasonable interpretations are treated as ties.

REPRODUCIBILITY STATEMENT

The paper and appendices document the experimental settings and implementation details, including benchmark subsets, candidate models, judge configurations, evaluation prompts, and rubric evolution procedures. We also describe the construction and verification of the human-verified dense reference (HVR). The corresponding reference data will be publicly released to support reproduction of our evaluation results and further research on benchmark evaluation.

REFERENCES

Mubashara Akhtar, Anka Reuel, Prajna Soni, Sanchit Ahuja, Pawan Sasanka Ammanamanchi, Ruchit Rawal, Vilém Zouhar, Srishti Yadav, Chenxi Whitehouse, Dayeon Ki, Jennifer Mickel, Leshem Choshen, Marek Šuppa, Jan Batzner, Jenny Chim, Jeba Sania, Yanan Long, Hossein A. Rahmani, Christina Knight, Yiyang Nan, Jyoutir Raj, Yu Fan, Shubham Singh, Subramanyam Sahoo, Eliya Habba, Usman Gohar, Siddhesh Pawar, Robert Scholz, Arjun Subramonian, Jingwei Ni, Mykel J. Kochenderfer, Sanmi Koyejo, Mrinmaya Sachan, Stella Biderman, Zeerak Talat, Avijit Ghosh, and Irene Solaiman. When AI benchmarks plateau: A systematic study of benchmark saturation. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026. URL https://openreview.net/forum?id=YC1Otscjbs.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *Computing Research Repository*, arXiv:2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Yifan Chen, Haitao Li, Qingyao Ai, Fengbin Zhu, Tat-Seng Chua, Min Zhang, and Yiqun Liu. GenRubric: Self-evolving rubric generation for scalable LLM evaluation. *arXiv preprint arXiv:2608.29856*, 2026. URL https://arxiv.org/abs/2608.29856.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8359–8388. PMLR, 2024. URL https://proceedings.mlr.press/v235/chiang24b.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *Computing Research Repository*, arXiv:2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled AlpacaEval: A simple way to debias automatic evaluators. In *Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2404.04475v2. Revised version, March 2025.

Bryan Etzine, Masoud Hashemi, Nishanth Madhusudhan, Sagar Davasam, Roshnee Sharma, Sathwik Tejaswi Madhusudhan, and Vikas Yadav. Revitalizing saturated benchmarks: A weighted metric approach for differentiating large language model performance. In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pp. 511–523, Albuquerque, New Mexico, May 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.trustnlp-main.33. URL https://aclanthology.org/2025.trustnlp-main.33/.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Kevin G. Jamieson and Robert D. Nowak. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems*, volume 24, 2011. URL https://proceedings.neurips.cc/paper/2011/hash/6c14da109e294d1e8155be8aa4b1ce8e-Abstract.html.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4110–4124, Online, 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.naacl-main.324.

Seungone Kim, Jay Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Ryan S Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations*, pp. 29927–29962, 2024. URL https://openreview.net/forum?id=8euJaTveKw.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christopher Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=iO4LZibEqW.

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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153.

Simon Ott, Adriano Barbosa-Silva, Kathrin Blagec, Jan Brauner, and Matthias Samwald. Mapping global dynamics of benchmark creation and saturation in artificial intelligence. *Nature Communications*, 13:6793, 2022. doi: 10.1038/s41467-022-34591-0. URL https://www.nature.com/articles/s41467-022-34591-0.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Delip Rao and Chris Callison-Burch. Autorubric: A unifying framework for rubric-based LLM evaluation on non-verifiable tasks. In *Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2603.00077v3. Accepted at COLM 2026; camera-ready version.

Isik Baran Sandan, Tu Anh Dinh, and Jan Niehues. Knockout LLM assessment: Using large language models for evaluations through iterative pairwise comparisons. In *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM)*, pp. 121–128. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.gem-1.10/.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. URL https://dblp.org/rec/journals/tmlr/SrivastavaRRSAF23.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, 2018. Association for Computational Linguistics. URL https://aclanthology.org/W18-5446.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.nips.cc/paper/8589-superglue-a-stickier-benchmark-for-general-purpose-language-understanding-systems.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.511. URL https://aclanthology.org/2024.acl-long.511/.

Fabian L. Wauthier, Michael I. Jordan, and Nebojsa Jojic. Efficient ranking from pairwise comparisons. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 109–117. PMLR, 2013. URL https://proceedings.mlr.press/v28/wauthier13.html.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason E. Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with LLM-as-a-meta-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11537–11554. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.emnlp-main.583/.

Yi Xu, Laura Ruis, Tim Rocktäschel, and Robert Kirk. Investigating non-transitivity in LLM-as-a-judge. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 69583–69612. PMLR, 2025. URL https://proceedings.mlr.press/v267/xu25w.html.

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*, volume 36, 2023. URL https://arxiv.org/abs/2306.05685.

A Detailed Experimental Settings

This section provides implementation details for the evaluation protocol. We first describe the predefined task-level principles and seed checklists used to initialize all runs. We then summarize the judge and evolution model choices and provide the main prompt templates used for seeding, pairwise judging, and checklist evolution.

A.1 Predefined Principles and Seed Checklists

SEAL starts from a fixed two-layer rubric for each task type. The outer layer consists of task-level principles, which define broad quality dimensions such as correctness, robustness, efficiency, factuality, or schema compliance. Principles remain the fixed units of verdict aggregation, while checklist items supply more detailed judging probes. The original pairwise prompt aggregates confidence-weighted principle votes, as reproduced below. The inner layer is the seed checklist Γ_0, which contains one checklist item per principle. Each seed item operationalizes its matched principle as a concrete judging probe used in the initial pairwise comparisons; Table 7 summarizes their counts. The original automated judge comparisons use the same predefined principles; FLAT-BRK and FULLPAIR also keep the same seed checklist fixed throughout evaluation, while SEAL starts from this checklist and then adaptively refines checklist items through split, adversarial-add, and merge operations in later rounds. Generic seed items are initialization choices, not automatically applicable task requirements; HVR checks their relevance against the original question.

Benchmark	Task type	# Principles	# Seed items
HumanEval	`code_generation`	6	6
GSM8K	`math_reasoning`	4	4
MMLU	`general_qa`	5	5
BFCL-v2	`tool_calling`	5	5

Table 7: Predefined principle and seed-checklist sizes. Each task type begins with one concrete checklist item for every predefined principle.

Code Generation. For HumanEval, the predefined principles cover functional correctness, interface and integration, edge-case robustness, algorithmic efficiency, code quality and maintainability, and safety or side effects. The corresponding seed checklist asks whether the implementation handles untested semantic details, preserves the required callable interface, covers valid edge cases, uses appropriate asymptotic complexity, remains readable and idiomatic, and avoids unintended mutation, I/O, global state, or unsafe operations.

Mathematical Reasoning. For GSM8K, the principles are correctness, reasoning soundness, completeness, and clarity. The seed checklist tests whether the final numeric answer is correct in the requested form, whether every algebraic or arithmetic step is locally justified, whether all required sub-questions or cases are addressed, and whether the final answer can be identified unambiguously in the requested numerical form.

General Question Answering. For MMLU, the principles are goal achievement, factual accuracy, helpfulness, safety, and style. The seed checklist asks whether the response directly answers the specific question, whether factual claims are accurate and uncertainty is calibrated, whether the answer is actionable for the user's situation, whether it avoids harmful or deceptive content, and whether its length and structure fit the question.

Tool Calling. For BFCL-v2, the principles are function selection, argument correctness, schema compliance, irrelevance handling, and response form. The seed checklist evaluates whether the model calls exactly the right function or functions, provides accurate and well-typed arguments, follows the declared schema, correctly declines when no available function applies, and emits either a clean tool call or a concise refusal/clarification in the expected form.

A.2 JUDGE/EVOLVING MODEL CHOICES

For SEAL, both the judge model and the evolution model are `gpt-5.5`; the same judge model is also used for all LLM-based baselines.

A.3 SEEDING PROMPT

The seeding prompt is used once per task before the tournament begins. Its system prompt defines the tiering objective and instructs the judge to group semantically similar candidates together, while its user prompt supplies the task, evaluation principles, and candidate outputs.

Seeding Prompt: System Message

```
You are an expert evaluator. Given N candidate responses to the same
    task,
roughly rank them into k tiers, where Tier 1 is best and Tier k is
    worst.

Judge only substantive task-relevant quality differences. Do not tier
    by
cosmetic formatting, response style, or superficial wording. Responses
     that are
semantically equivalent should be placed in the same tier.

Use the provided evaluation principles as guidance. You do not need to
     produce
an exact ranking inside each tier; focus on which candidates are
    clearly
stronger or weaker.

Return only valid JSON:

{
  "tiers": {
    "1": ["model_A", "model_B"],
    "2": ["model_C"]
  },
  "reasoning": "brief explanation"
}

Every model_id provided must appear in exactly one tier.
```

The corresponding user message instantiates the generic tiering instruction with a concrete benchmark task, the fixed task-level principles, and the candidate responses to be seeded.

Seeding Prompt: User Message

```
Task:

${task_prompt}

Evaluation Principles:

${principles_block}

Candidate responses:

${solutions_block}

Assign each candidate to one of ${k} tiers.
```

A.4 PAIRWISE JUDGE PROMPT

The pairwise judge prompt is the core comparison prompt used by SEAL, FLATBRK, and FULL-PAIR. The system prompt fixes the voting protocol: the judge must vote left, right, or tie for each principle and checklist item, and the final verdict is derived mechanically from confidence-weighted principle votes.

Pairwise Judge Prompt: System Message

```
You are an expert evaluator performing a pairwise comparison of two
    candidate
responses to the same task. Your job is to decide, for each Principle
    and each
Checklist item, whether the left response is better, the right
    response is
better, or they are tied.

Judge only against the requirements stated in the task. Do not invent
    extra
requirements. Do not reward or penalize cosmetic differences.

For each Principle, output:
- vote: "left", "right", or "tie"
- confidence: a number from 0.0 to 1.0
- reasoning: one sentence citing the observable behavior that decided
    the vote

Use "tie" when there is no concrete, non-cosmetic difference. When a
    real
difference exists, vote for the better side and cite the specific
    behavior.

The overall verdict must be mechanical. Compute the confidence-
    weighted vote
sum over Principles:
- left vote = -1
- right vote = +1
- tie = 0

The verdict is:
- "left" if the weighted sum is negative
- "right" if the weighted sum is positive
- "tie" if the weighted sum is effectively zero

Do not use an unconstrained holistic impression.

For each Checklist item, apply the same pairwise voting rule:
vote "left", "right", or "tie", with confidence.

Return only valid JSON:

{
  "verdict": "left" | "right" | "tie",
  "principle_scores": [
    {
      "principle_id": "P1",
      "vote": "left" | "right" | "tie",
      "confidence": <0-1>,
      "reasoning": "<one sentence citing the deciding behavior>"
    }
  ],
  "checklist_scores": [
    {
```

```
      "item_id": "<id>",
      "vote": "left" | "right" | "tie",
      "confidence": <0-1>
    }
  ]
}

You must output one vote for every Principle and every Checklist item.
```

The user message supplies the concrete task, the fixed principles, the current checklist, and the two candidate responses being compared. SEAL differs from FLATBRK and FULLPAIR only in how matches are scheduled and how the checklist is updated; the pairwise judge format remains shared.

Pairwise Judge Prompt: User Message

```
Task:

${task_prompt}

Evaluation Principles:

${principles_block}

Checklist:

${checklist_block}

=== Response A (${left_model}) ===

${left_output}

=== Response B (${right_model}) ===

${right_output}

Decide, for each Principle and Checklist item, whether Response A or
    Response B
is better, or whether they are equivalent. Output only the JSON object
    .
```

A.5 CHECKLIST-EVOLUTION PROMPT

The following template documents the adversarial-addition operator used by SEAL. It introduces one new checklist item after a close comparison, where the current judge prefers one strong response over another. The rubric histories additionally record split and merge operations; this template is not a complete specification of every evolution operator. The system prompt constrains the new item to be non-redundant, principle-grounded, anchor-calibrated, and supported by a demonstration pair.

Checklist-Evolution System Prompt

```
You are an expert rubric designer. Given a comparison where a strong
    response
lost to a slightly better response, generate exactly one new checklist
     criterion
that captures the subtle, observable quality difference that decided
    the match.

Rules:
```

```
1. Anti-redundancy:
   Do not generate a criterion that duplicates or paraphrases an
    existing
   checklist item.

2. Coverage:
   Prefer principles that currently have fewer checklist items, unless
     the
   distinguishing behavior clearly belongs to a more covered principle
    .

3. Anchor calibration:
   The new criterion must map to one fixed evaluation principle and
    distinguish
   between two adjacent anchor levels, such as 5 vs. 4 or 4 vs. 3.

4. Demonstration pair:
   Provide two short concrete snippets or examples:
   - higher_snippet: illustrates the better behavior
   - lower_snippet: illustrates the weaker behavior

5. Positive and concrete framing:
   The criterion should describe what a good response does. Avoid
    vague
   criteria such as "be efficient" or "follow best practice."

6. Task scope:
   Do not invent requirements outside the task description.

Return only valid JSON:

{
  "id": "${new_id}",
  "description": "<one concrete, anchor-grounded checklist criterion
    >",
  "principle_id": "<one principle id>",
  "anchor_split": {
    "higher": <int 1-5>,
    "lower": <int 1-5>
  },
  "differentiator": "<one sentence explaining the observable quality
    gap>",
  "demonstration_pair": {
    "higher_snippet": "<short example showing the stronger behavior>",
    "lower_snippet": "<short example showing the weaker behavior>"
  },
  "scoring": "five_level",
  "source": "adversarial"
}
```

The user message gives the evolver the task context, the losing and winning responses, the loser-side principle evidence, the current principle set, and the existing checklist. This lets the evolver propose a criterion targeting an observed unresolved gap. The proposal remains subject to task-grounded assessment; the originating machine preference is not itself human verification.

Checklist-Evolution Prompt: User Message

```
Task description:

${task_description}

Loser response:
```

```
${loser_output}

Winner response:

${winner_output}

Per-principle evidence for the loser:

${loser_scores}

Available principles:

${principles_block}

Existing checklist:

${existing_items_block}

Items-per-principle counts:

${coverage_counts}

Generate exactly one new checklist criterion with ID="${new_id}" that
    captures
the subtle quality gap between the loser and winner without
    duplicating an
existing checklist item.
```

B BENCHMARK SUBSETS AND CANDIDATE MODELS

Table 8 lists benchmark subsets and candidate pools. Tables 9, 10, 11, and 12 provide the corresponding HumanEval, BFCL-v2, GSM8K, and MMLU protocol scores.

Benchmark	Tasks	Native metric	Candidate models

HumanEval	162	pass@1	`openai/gpt-5`
			`openai/o4-mini-high`
			`google/gemini-3.1-pro`
			`x-ai/grok-4.20`
			`anthropic/claude-sonnet-4.6`
			`deepseek/deepseek-v3.2`
			`anthropic/claude-opus-4.7`
			`qwen/qwen3-coder-plus`

BFCL-v2	100	success rate	`qwen/qwen3-coder`
			`google/gemini-2.5-flash`
			`z-ai/glm-5-turbo`
			`google/gemini-3.1-pro`
			`moonshotai/kimi-k2.6`
			`openai/o4-mini-high`
			`deepseek/deepseek-v3.2`
			`openai/gpt-5`

GSM8K	150	numeric exact match	`anthropic/claude-sonnet-4`
			`google/gemini-2.5-pro`
			`openai/o4-mini-high`
			`deepseek/deepseek-r1-0528`
			`google/gemini-3-flash-preview`
			`qwen/qwen3-30b-a3b-thinking-2507`
			`openai/gpt-5`
			`qwen/qwen3-235b-a22b`

MMLU	110	multiple-choice accuracy	`openai/gpt-5-mini`
			`google/gemini-3.1-pro`
			`qwen/qwen3.6-plus`
			`openai/gpt-5.4-nano`
			`x-ai/grok-4.3`
			`deepseek/deepseek-chat`
			`anthropic/claude-3.5-haiku`
			`minimax/minimax-m2.7`

Table 8: Benchmark subsets and candidate model names reported for the original evaluation. The evolved-rubric experiment uses 22 BFCL-v2 live_simple tasks (Table 14); its coverage is therefore distinct from the original benchmark subset.

Model	Original	PW	FixedRub	Listwise	FlatBrk	FullPair	SEAL
gpt-5	#5 .988	#1 .980	#1 .310	#1 .748	#1 .511	#1 .541	**#1 .509**
o4-mini-high	#3 .994	#3 .968	#7 .306	#2 .604	#2 .492	#2 .523	**#2 .488**
gemini-3.1-pro	#2 1.00	#4 .968	#8 .305	#4 .526	#3 .483	#4 .511	**#3 .486**
grok-4.20	#4 .994	#2 .970	#2 .310	#6 .437	#4 .482	#3 .512	**#4 .483**
claude-sonnet-4.6	#1 1.00	#6 .956	#6 .308	#3 .550	#5 .456	#6 .488	**#5 .463**
deepseek-v3.2	#6 .981	#5 .959	#5 .309	#7 .342	#6 .451	#5 .493	**#6 .457**
claude-opus-4.7	#7 .963	#8 .931	#4 .309	#5 .492	#8 .432	#7 .469	**#7 .440**
qwen3-coder-plus	#8 .963	#7 .932	#3 .309	#8 .301	#7 .437	#8 .461	**#8 .435**

Table 9: HumanEval archived protocol results. Each cell gives recorded rank and score on the original protocol scale. PW denotes Pointwise; full model identifiers appear in Table 8.

Model	Original	PW	FixedRub	Listwise	FlatBrk	FullPair	SEAL
qwen3-coder	#3 .857	#1 .835	#1 .892	#2 .664	#1 .496	#1 .558	**#1 .532**
gemini-2.5-flash	#4 .823	#2 .810	#2 .882	#1 .772	#4 .463	#2 .549	**#2 .524**
glm-5-turbo	#1 .867	#3 .783	#4 .838	#3 .539	#2 .490	#3 .540	**#3 .464**
gemini-3.1-pro	#2 .860	#4 .782	#3 .863	#4 .485	#3 .472	#4 .530	**#4 .458**
kimi-k2.6	#5 .817	#5 .771	#5 .825	#5 .469	#5 .460	#5 .514	**#5 .421**
o4-mini-high	#7 .668	#6 .690	#7 .731	#6 .418	#7 .389	#7 .433	**#6 .368**
deepseek-v3.2	#8 .653	#8 .676	#8 .729	#7 .365	#8 .383	#6 .438	**#7 .364**
gpt-5	#6 .716	#7 .685	#6 .740	#8 .287	#6 .389	#8 .424	**#8 .354**

Table 10: BFCL-v2 archived protocol results. Each cell gives recorded rank and score on the original protocol scale. PW denotes Pointwise; full model identifiers appear in Table 8.

Model	Original	PW	FixedRub	Listwise	FlatBrk	FullPair	SEAL
claude-sonnet-4	#6 .965	#4 .928	#4 .992	#2 .668	#4 .499	#1 .549	**#1 .503**
gemini-2.5-pro	#5 .967	#3 .933	#3 .992	#4 .489	#3 .499	#3 .542	**#2 .495**
o4-mini-high	#4 .970	#7 .918	#6 .991	#3 .633	#1 .501	#6 .505	**#3 .488**
deepseek-r1-0528	#8 .960	#1 .940	#5 .991	#1 .734	#2 .500	#2 .542	**#4 .482**
gemini-3-flash	#1 .978	#2 .936	#2 .995	#5 .465	#5 .489	#4 .537	**#5 .478**
qwen3-30b-a3b	#7 .963	#5 .921	#7 .989	#6 .442	#6 .478	#5 .519	**#6 .467**
gpt-5	#2 .973	#6 .918	#1 .995	#7 .361	#7 .415	#7 .465	**#7 .404**
qwen3-235b-a22b	#3 .973	#8 .807	#8 .834	#8 .208	#8 .286	#8 .331	**#8 .314**

Table 11: GSM8K archived protocol results. Each cell gives recorded rank and score on the original protocol scale. PW denotes Pointwise; full model identifiers appear in Table 8.

Model	Original	PW	FixedRub	Listwise	FlatBrk	FullPair	SEAL
gpt-5-mini	#2 .940	#1 .973	#2 .994	#1 .911	#1 .549	#1 .637	**#1 .549**
gemini-3.1-pro	#1 .945	#2 .961	#1 .999	#4 .526	#3 .478	#2 .584	**#2 .489**
qwen3.6-plus	#3 .940	#5 .933	#3 .991	#6 .331	#4 .461	#3 .582	**#3 .488**
gpt-5.4-nano	#6 .915	#3 .948	#4 .980	#2 .699	#2 .488	#4 .551	**#4 .486**
grok-4.3	#4 .940	#4 .939	#5 .974	#3 .649	#5 .440	#5 .545	**#5 .421**
deepseek-chat	#7 .835	#7 .834	#6 .927	#5 .356	#6 .397	#6 .451	**#6 .395**
claude-3.5-haiku	#8 .710	#8 .726	#8 .847	#8 .211	#7 .309	#7 .332	**#7 .282**
minimax-m2.7	#5 .930	#6 .888	#7 .897	#7 .317	#8 .297	#8 .305	**#8 .251**

Table 12: MMLU archived protocol results. Each cell gives recorded rank and score on the original protocol scale. PW denotes Pointwise; full model identifiers appear in Table 8.

C EVOLVED-POINTWISE RESULTS AND EVALUATION RECORDS

Table 13 reports the completed evolved-pointwise aggregate scores. Each task uses the final checklist from its originating SEAL run, frozen before the new pointwise evaluation. The judge is `gpt-5.5-2026-04-24`. All 3,550 pointwise scoring calls completed successfully. The table reports model-level aggregate scores; HVR agreement is a separate reference-based metric.

Benchmark	Model	Score	Tasks
HumanEval	openai/gpt-5	0.8691	162
	openai/o4-mini-high	0.8512	162
	x-ai/grok-4.20	0.8438	162
	google/gemini-3.1-pro-preview	0.8395	162
	deepseek/deepseek-v3.2	0.8321	162
	anthropic/claude-sonnet-4.6	0.8222	162
	qwen/qwen3-coder-plus	0.8043	162
	anthropic/claude-opus-4.7	0.7957	162
GSM8K	anthropic/claude-sonnet-4	0.8887	150
	google/gemini-3-flash-preview	0.8820	150
	google/gemini-2.5-pro	0.8773	150
	deepseek/deepseek-r1-0528	0.8667	150
	openai/o4-mini-high	0.8667	150
	qwen/qwen3-30b-a3b-thinking-2507	0.8633	150
	openai/gpt-5	0.8387	150
	qwen/qwen3-235b-a22b	0.6527	150
MMLU	openai/gpt-5-mini	0.9100	110
	google/gemini-3.1-pro-preview	0.8827	110
	qwen/qwen3.6-plus	0.8827	110
	openai/gpt-5.4-nano	0.8618	110
	x-ai/grok-4.3	0.8255	110
	deepseek/deepseek-chat	0.7582	110
	minimax/minimax-m2.7	0.6945	110
	anthropic/claude-3.5-haiku	0.5791	110
BFCL-v2	qwen/qwen3-coder	0.8500	22
	google/gemini-2.5-flash	0.7762	21
	z-ai/glm-5-turbo	0.6818	22
	moonshotai/kimi-k2.6	0.6409	22
	google/gemini-3.1-pro-preview	0.6381	21
	openai/o4-mini-high	0.5955	22
	openai/gpt-5	0.5364	22
	deepseek/deepseek-v3.2	0.4364	22

Table 13: Evolved-pointwise aggregate scores, displayed to four decimal places. Near-equal pairs differing only by floating-point round-off are treated as ties in ranking analysis.

C.1 Run Scope and Rubric Provenance

Benchmark	Tasks	Changed	Final items	PW calls
HumanEval	162	162	7/13/22	1,296
GSM8K	150	150	5/8/11	1,200
MMLU	110	109	5/9/13	880
BFCL-v2	22	22	6/9/11	174
Total	444	443	—	3,550

Table 14: Evolved-pointwise coverage. Changed counts checklists that differ from their seeds; final items reports minimum/median/maximum. Calls cover application and exclude rubric acquisition.

Table 14 summarizes checklist changes and scoring coverage. Each task has four rubric snapshots: the seed at round zero and three subsequent snapshots. Item provenance records seed, split, adversarial-add, and merge operations. The final checklist differs from the seed for 162/162 HumanEval tasks, 150/150 GSM8K tasks, 109/110 MMLU tasks, and 22/22 BFCL tasks. These histories establish what rubric was reused; checklist growth alone does not establish better evaluation quality.

The pointwise run wall times are 2,541.4 seconds for HumanEval, 1,627.7 for GSM8K, 1,259.9 for MMLU, and 298.6 for BFCL-v2. Their sum, 5,727.6 seconds, is an accounting total across runs, not a controlled estimate of sequential latency. The same export also contains evolved-listwise runs with one call per task, totaling 444 calls. We keep the main rubric-reuse analysis focused on pointwise scoring; listwise results constitute a distinct evaluation protocol.

The pointwise records contain aggregate leaderboards. Task-level analysis additionally requires the individual scores, response versions, and corresponding HVR comparisons. The originating tournament traces document seeding and rubric evolution; they describe a separate evaluation stage.

C.2 Resolution Gain and Score Scaling

For protocol scores $s_1, \ldots, s_N$, the reported resolution gain compares average pairwise score separation with the native metric:

$$D(s) = \binom{N}{2}^{-1} \sum_{i<j} |s_i - s_j|, \qquad G(s) = \frac{D(s)}{D(s^{\text{native}})}. \tag{3}$$

The original experiment values are retained in Table 15. A gain above one indicates a wider score spread on the protocol's reported scale. Since the protocols use different scoring conventions, the magnitude is not a scale-invariant measure of evaluation accuracy. The human-verified ranking comparison in Table 1 therefore remains the primary validity result.

Protocol	HumanEval	GSM8K	MMLU	BFCL-v2
Original	1.00	1.00	1.00	1.00
Pointwise	1.18	5.29	1.13	0.70
FixedRub	0.12	5.72	0.77	0.79
Listwise	9.84	28.64	3.52	1.83
FlatBrk	1.77	9.87	1.30	0.54
FullPair	1.88	30.07	1.77	0.64
SEAL	2.10	10.50	1.30	0.80

Table 15: Reported resolution gain ($\times$) on the original protocol score scales. Greater spread does not establish a more accurate preference.

C.3 Reference Coverage and Tie Handling

Table 16 reports retained and excluded pair counts for each benchmark.

Benchmark	Available pairs	Retained	Excluded	Coverage
HumanEval	4,536	4,410	126	97.22%
GSM8K	4,187	4,185	2	99.95%
MMLU	3,080	3,056	24	99.22%
BFCL-v2	603	585	18	97.01%
Total	12,406	12,236	170	98.63%

Table 16: HVR data card. Coverage is relative to available automated pair records; excluded records comprise 36 conditional and 134 unresolved decisions.

Benchmark	Retained pairs	Changed labels	Final ties	Decisive pairs
HumanEval	4,410	3,725	3,632	778
GSM8K	4,185	3,548	3,527	658
MMLU	3,056	1,725	1,613	1,443
BFCL-v2	585	367	268	317
Total	12,236	9,365	9,040	3,196

Table 17: Pair-label changes from the original structured FullPair verdict to the final verified label, restricted to retained records. These counts describe the reference corrections, not independent reviewer agreement.

Of retained pairs, 9,365 change label (76.54%): 9,040 decisive comparisons become ties and 325 reverse direction. The other 2,871 retain their original preference (Table 17). These counts compare pair verdicts, not individual criterion votes or independent reviewers.

The final release merges unconditional reviewed corrections into the dense source records using their comparison identifiers. Conditional or unresolved records are excluded, and all other reviewed records are retained. Seven human reviewers participated, with comparisons involving each model distributed across reviewers as randomly as practicable. There was no independent double-review subset. Assignment spreads reviewer exposure but does not provide an empirical estimate of inter-reviewer reliability.

Each retained record links the task and both responses to the original automated decision and the verified outcome. Criterion-level evidence distinguishes a justified tie from an inapplicable criterion or unavailable evidence. A criterion based solely on equally reasonable interpretations supplies no preference, while verified differences in other applicable dimensions remain active. Unresolved records do not contribute artificial ties to the reference. The final release will include these labels and evidence links to support inspection and reconstruction.

Local Recovery Diagnostics. For native-metric ties, let $\mathcal{T}_D$ contain decisive verified preferences and $\mathcal{T}_T$ verified ties. A task-level prediction $\hat{y}$ supports two complementary diagnostics:

$$R = \frac{\sum_{i \in \mathcal{T}_D} \mathbf{1}[\hat{y}_i = y_i^{\mathrm{HVR}}]}{|\mathcal{T}_D|}, \qquad F = \frac{\sum_{i \in \mathcal{T}_T} \mathbf{1}[\hat{y}_i \neq \mathrm{tie}]}{|\mathcal{T}_T|}. \tag{4}$$

Here R measures recovered verified differences and F measures unsupported separation of verified ties. These require version-matched per-task predictions, including comparisons a tournament did not directly observe. The current paper's primary results are aggregate rank correlations; they should not be interpreted as measurements of R or F. A complete local analysis should report prediction coverage and task-bootstrap intervals along with both rates.

Tie Conventions and Scope. Spearman correlation assigns average ranks to equal aggregate scores. Floating-point round-off in the near-equal evolved-pointwise scores does not justify distinct ranks. The top-1 statistics in the stability analysis compare SEAL runs with one another, not with HVR. Score spread, run stability, and human-reference agreement consequently answer different questions and are reported separately.