
Evaluation as the Dark Horse of AI

Odysseus Kaloidas
UC Berkeley

Abstract

Better AI may depend as much on recognizing a good answer as on generating one. If twenty candidate programs include a correct solution but the system returns a failure, better selection could improve the result without a better generator. Evaluation means assessing an answer against a criterion; it can guide selection as well as measure performance. Yet a program that passes a flawed test may still fail the intended task. These two roles make evaluation a possible source of progress and a source of misleading confidence. How much capability remains inaccessible because a system cannot recognize its best work, or because its tests reward the wrong thing?

1 The Correct Answer That Was Not Chosen

Suppose twenty candidate programs include a correct solution, but the system returns one that fails. From the user's point of view, the successful program might as well not exist. Better evaluation could make it usable without improving the generator at all. This is one reason that progress in evaluation can look modest on paper while changing what a system can actually do.

It can even reverse a comparison between generators. A system that produces good answers less often may still be more useful if it reliably recognizes them. That possibility makes evaluation part of capability, as well as a way of measuring it.

2 Evaluation Produces Value

Code makes the role of evaluation especially visible. HumanEval tests whether generated programs behave correctly on specified inputs [2]. A system can use such checks to reject candidates before presenting one to a user. Whether the checks cover what the user actually wanted is a further question.

Other methods use agreement among sampled reasoning paths or a learned model's assessment of a solution [12; 9]. These methods can improve the answer even without changing the underlying generator. Models trained to predict human preferences serve a related role during training, where errors in the assessment can shape later behavior [10]. Evaluation helps produce the result, as well as measure it.

Kambhampati and colleagues combine proposals from large language models (LLMs) with external verification against an explicit model of the task [5]. External checking can remain useful even if unaided planning improves. That argument is separate from their stronger 2024 contention that LLMs cannot plan unaided: a better reasoner can still return a poor candidate or be judged against the wrong criterion.

3 Finding an Answer and Recognizing It

Consider two systems. The first produces a pool containing a correct answer in 80 percent of runs and selects that answer in 75 percent of those runs. It succeeds in 60 percent overall. The second

supplies a correct answer in 95 percent of runs, but selects it only half the time. It succeeds in 47.5 percent. The better generator has produced the less reliable complete system.

Choosing a weaker answer loses some of the quality already available in the pool. Let $Y_1, \dots, Y_n$ be candidates, $q(Y_i)$ their bounded true quality under a specified criterion, and J the selected index. The best available quality is $M_n = \max_i q(Y_i)$. Selection regret, $G_n = M_n - q(Y_J) \geq 0$, is what choosing another answer loses. Then

$$\mathbb{E}[q(Y_J)] = \mathbb{E}[M_n] - \mathbb{E}[G_n].$$

The expectation averages over repeated runs: delivered quality equals available quality minus the loss from selection. This is an accounting identity, not evidence that either component improves with scale.

Pass-at- k asks whether at least one of k generated candidates passes the tests [2]. A verifier is a separate judge that scores or ranks candidates [3]. Finding a passing answer and reliably selecting it are different demands; both still depend on whether the tests capture the intended task. In the decomposition, more computation can improve the pool, reduce selection regret, or do both. A rising aggregate success curve does not identify these contributions, especially when the generator and selector change together.

If a checker accepts every correct candidate and rejects every incorrect one, examining the whole pool eliminates the selection error. Generation then becomes the remaining bottleneck. This favorable case is real and helps explain progress in executable domains. With an incomplete specification, however, perfect agreement with the checker still leaves intended correctness unresolved. The case for evaluation is conditional on which gap remains.

4 When the Answer Key Is Part of the Problem

Executable checks appear unusually favorable because behavior can be tested. Yet a July 2026 OpenAI audit estimated that roughly 30% of the 731 public SWE-bench Pro problems were broken; a separate review by five experienced engineers flagged 249 problems [7]. The reported defects included underspecified requests, tests tied to one implementation, and insufficient coverage. This is an audit by a model developer, not an independent certification of the benchmark. Its concrete examples nevertheless undermine the assumption that an executable test is automatically a sound specification.

A proof checker makes the issue particularly clear. It can settle whether a statement follows from formal rules while leaving open whether we formalized the question we actually meant to ask.

Two scientific proposals can be ranked differently because one evaluator values feasibility and another values novelty, even when both agree on the facts. The disagreement is then about what counts as achievement. Wallach and colleagues distinguish that underlying concept from its precise definition, the instrument used to measure it, and the measurements produced [11]. Agreement among judges establishes consistency, not necessarily validity: they can agree on a rubric that misses the intended achievement. HELM, an evaluation suite covering several aspects of language-model performance, similarly makes tradeoffs visible [6].

5 Why More Evaluation Can Still Fail

A noisy evaluator creates another problem. Imagine that several proposals are equally good, but each receives a score with a little random error. The highest-scoring proposal will tend to be one whose error was favorable. Its winning score therefore exaggerates its quality, even if the evaluator is unbiased when assessing proposals individually. This is the optimizer’s curse [8]. Searching more candidates gives chance more opportunities to flatter the winner.

Cawley and Talbot study the related problem of choosing models using limited evaluation data: repeatedly selecting for a high score can overfit that score [1]. Even a scoring rule designed to reward honest probability estimates on average cannot guarantee that a particular finite-sample winner is best [4]. More evaluation helps when it provides better evidence, not merely more occasions to select a lucky result.

This makes expert judgment consequential even where plausible analyses are cheap. A growing pile of outputs can represent a growing pile of unfinished decisions.

Evaluation can change which system is useful without changing which system first generates the right answer. The twenty-program example captures the practical stakes: the user receives one program. How much progress could come from learning to recognize that program, and how much still depends on expressing the task in tests that deserve our trust?

References

- [1] Gavin C. Cawley and Nicola L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010. URL <https://www.jmlr.org/papers/v11/cawley10a.html>.
- [2] Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code, 2021. URL <https://arxiv.org/abs/2107.03374>. arXiv:2107.03374.
- [3] 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. arXiv:2110.14168, 2021. URL <https://arxiv.org/abs/2110.14168>.
- [4] Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437.
- [5] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: LLMs can’t plan, but can help planning in LLM-modulo frameworks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 22895–22907. PMLR, 2024. URL <https://proceedings.mlr.press/v235/kambhampati24a.html>.
- [6] Percy Liang, Rishi Bommasani, Tony Lee, et al. Holistic evaluation of language models, 2022. URL <https://arxiv.org/abs/2211.09110>. arXiv:2211.09110.
- [7] OpenAI. Separating signal from noise in coding evaluations. Research report, July 8, 2026. URL <https://openai.com/index/separating-signal-from-noise-coding-evaluations/>.
- [8] James E. Smith and Robert L. Winkler. The Optimizer’s Curse: Skepticism and Postdecision Surprise in Decision Analysis. *Management Science*, 52(3):311–322, 2006. doi: 10.1287/mnsc.1050.0451.
- [9] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL <https://arxiv.org/abs/2408.03314>. arXiv:2408.03314. Preprint version consulted.
- [10] Nisan Stiennon, Long Ouyang, Jeff Wu, et al. Learning to summarize from human feedback, 2020. URL <https://arxiv.org/abs/2009.01325>. arXiv:2009.01325.
- [11] Hanna Wallach, Meera Desai, A. Feder Cooper, Angelina Wang, Chad Atalla, Solon Barocas, Su Lin Blodgett, Alexandra Chouldechova, Emily Corvi, P. Alex Dow, Jean Garcia-Gathright, Alexandra Olteanu, Nicholas J Pangakis, Stefanie Reed, Emily Sheng, Dan Vann, Jennifer Wortman Vaughan, Matthew Vogel, Hannah Washington, and Abigail Z. Jacobs. Position: Evaluating generative AI systems is a social science measurement challenge. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 82232–82251. PMLR, 2025. URL <https://proceedings.mlr.press/v267/wallach25a.html>.
- [12] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL <https://arxiv.org/abs/2203.11171>.