BAT / CAT TEAM BEST PRACTICES

Measuring Quality with LLM Judges

Why direct 1 to 5 qualitative scoring is fragile, and how to design more reliable evaluation systems

Michael Guimet & Serena Xie

June 2026

Abstract

Large language models are increasingly used as evaluators for generative AI systems because they can interpret open-ended outputs, apply natural-language criteria, and scale review beyond what expert-only processes can support. However, **asking an LLM to directly assign a single 1-5 score is a fragile measurement design**.

The problem combines a classical limitation of ordinal scales with known LLM-as-judge failure modes, including position bias, verbosity bias, calibration drift, model-version sensitivity, and disagreement across evaluators.

This paper argues for a more formal evaluation architecture: use LLMs to produce structured, evidence-backed observations; use explicit rubrics and decomposed checklist criteria; compute final scores outside the model; and calibrate those scores against production outcomes.

This paper recommends (TL; DR):

1. **Define the decision** the evaluation will support.
2. **Decompose quality into measurable dimensions** (e.g., correctness, completeness, relevance, faithfulness, task success).
3. **Convert each dimension into observable criteria**; specific, checkable conditions rather than vague qualities.
4. **Pick the scoring method to fit each criterion and the decision**: binary pass/fail checklists by default; anchored severity labels where graded judgment adds signal; comparative or probability-weighted scoring, etc.
5. **Require the LLM judge to cite evidence** for each criterion.
6. **Aggregate criterion outcomes outside the model** with explicit, versioned rules.
7. **Calibrate against two independent references**: human-reviewed examples and an independent LLM judge from a different model family.
8. **Monitor over time**: drift, disagreement, and the agreement between the independent judge and periodic human spot-checks. When that agreement slips, recalibrate**.**

1. Introduction

The appeal of a 1-5 score is obvious. It is compact, familiar, easy to chart, and appears to convert qualitative judgment into a quantitative signal. That simplicity is attractive where teams need to compare systems, track regressions, and decide whether an AI experience is ready for broader use.

The risk is that the number can look more precise than the measurement process behind it. A direct 1-5 score asks an evaluator to collapse multiple quality dimensions into one ordinal label. When the evaluator is an LLM, the score is also affected by prompt wording,

context order, model behaviour, examples, response length, and the semantic ambiguity of the scale itself.

This paper distinguishes between two uses of LLMs in evaluation. The first is weak: asking a model to rate an item from 1 to 5. The second is stronger: asking a model to apply explicit criteria, cite evidence, make criterion-level judgments, and feed a deterministic scoring process. The latter is the more defensible design for production-grade evaluation.

2. The Measurement Problem Behind 1-5 Scales

The first problem predates LLMs. A typical Likert-style scale is ordinal: categories can be ranked, but the distance between adjacent categories is not necessarily equal. Sullivan and Artino (2013) explain that ordinal responses can be ordered, but the distance between response options is not measurable in the way interval data are measurable [1].

This distinction matters because evaluation programs often treat 1-5 scores as interval measurements: they average scores, compare small deltas, and set thresholds. These practices can be acceptable approximations when the instrument is carefully designed and validated. They become fragile when the score levels are vague or when the middle of the scale is underdefined.

The middle values are the most problematic. A score of 3 or 4 may mean partially correct, acceptable but incomplete, minor defect, moderate defect, uncertain evidence, or good but not excellent. Unless those states are explicitly anchored, different judges can assign different meanings to the same number.

Measurement issue	Consequence for LLM scoring
Ordinal categories are not equal intervals	A one-point difference may not represent a consistent quality difference.
Middle categories are ambiguous	Scores of 3 and 4 absorb multiple meanings and become low-signal.
Averages hide distribution shape	The same mean can represent agreement, polarization, or score compression.
Single scores collapse dimensions	Correctness, completeness, relevance, faithfulness, and usefulness become indistinguishable.
Thresholds imply precision	A release decision based on 3.9 versus 4.1 may overstate measurement reliability.

3. LLM-as-Judge: Promise and Limits

LLM-based evaluators can be useful. Liu et al. (2023) showed that GPT-4-based evaluation with chain-of-thought, and form-filling can improve correlation with human judgments in summarization and dialogue evaluation [2]. Zheng et al. (2023) showed that strong LLM judges can approximate human preferences in some controlled and crowdsourced settings, while also documenting important limitations [4].

The same literature makes clear that LLM judges are not neutral measuring instruments. Liu et al. (2023) identifies potential bias toward LLM-generated text [2]. Wang et al. (2023) demonstrates that altering the order of candidate responses can materially change LLM judge outcomes and proposes calibration strategies such as balanced position calibration and human-in-the-loop review [3]. Zheng et al. (2023) identifies position bias (the order in which candidates are shown changes the verdict), verbosity bias (a preference for longer answers regardless of quality), self-enhancement bias (a tendency to favour outputs from the same model or model family as the judge), and limited reasoning ability as practical limitations of LLM-as-judge evaluation [4].

The conclusion is therefore balanced: LLMs can help evaluate open-ended outputs, but only when the evaluation protocol is designed to control variance, bias, and ambiguity.

4. Why Direct 1-5 LLM Scoring Is Fragile

Direct 1-5 scoring asks the model to perform too many operations in one step: infer the task, infer quality dimensions, weigh trade-offs, map those trade-offs to a vague ordinal scale, and output a number. Each operation introduces variance.

Recent work is explicit on this issue. Lee et al. (2024) attributes rating inconsistency in LLM-as-judge protocols to subjective evaluation criteria combined with Likert-scale scoring (formal for the familiar ordered rating format — e.g. 1 to 5, or "strongly disagree" to "strongly agree") and improves reliability by decomposing evaluation into binary checklist questions [6]. Kim et al. (2024) shows that evaluator performance improves when a judge is given customized score rubrics and reference materials [5]. Girrbach et al. (2025) likewise reports systematic issues with Likert-scale LLM ratings, including instability under sampling, poor calibration, and compression near the top of the scale [8].

The core failure modes are:

- Scale ambiguity: without explicit anchors, the model must infer what each score means.
- Prompt sensitivity: small wording changes can change the frame of judgment.
- Context sensitivity: response order, examples, and candidate length can influence the score.
- Calibration drift: model updates, deployment changes, and evaluator replacement can shift score distributions.
- Aggregation opacity: one score hides which criterion caused the judgment.
- Reference-free uncertainty: without ground truth or source material, the model judges plausibility and adequacy from incomplete evidence.

5. From Rating Prompt to Measurement Architecture

A stronger design separates observation, judgment, and scoring. The LLM should not be asked to invent the final number in one step. Instead, it should evaluate explicit criteria and produce structured observations. The evaluation system should compute the final score using versioned deterministic rules.

1. **Define the decision** the evaluation will support.
2. **Decompose quality into measurable dimensions** (e.g., correctness, completeness, relevance, faithfulness, task success).
3. **Convert each dimension into observable criteria**; specific, checkable conditions rather than vague qualities.
4. **Pick the scoring method to fit each criterion and the decision**: binary pass/fail checklists by default; anchored severity labels where graded judgment adds signal; comparative or probability-weighted scoring where finer or more stable signals are needed.
5. **Require the LLM judge to cite evidence** for each criterion.
6. **Aggregate criterion outcomes outside the model** with explicit, versioned rules.
7. **Calibrate against two independent references**: human-reviewed examples and an independent LLM judge from a different model family.
8. **Monitor over time**: drift, disagreement, and the agreement between the independent judge and periodic human spot-checks. When that agreement slips (or the judge shares a model lineage with the evaluated system), recalibrate.

This lifecycle view aligns with Microsoft Foundry guidance, which frames evaluation as part of a broader observability practice spanning pre-production evaluation, tracing, monitoring, custom evaluators, scheduled evaluation, and production feedback loops [7].

Two clarifications on what this architecture does and does not deliver.

Reliability vs. validity. This architecture mainly improves *reliability* and auditability (the same response yields the same, inspectable score) not *validity*, which is whether the score reflects true quality. Deterministic aggregation makes a number reproducible, not automatically correct. This gap is not specific to LLM judges; it applies equally to manual 1–5 scoring.

What calibration can and cannot do. Calibrating against human-reviewed examples does not fix the inherent unreliability of human ordinal ratings. What it *should* do is remove bias introduced by the LLM judge, aligning thresholds with reviewed outcomes. Where the reliability of the human labels is itself a concern, use multiple reviewers and track their agreement.

6. Defining the Decision

Every later choice in this architecture, which dimensions to measure, how to score them, and how to aggregate, follows from one question: what decision will the evaluation support? Naming it first is what keeps an evaluation honest. The step is often left implicit, yet it constrains everything after it.

Build backward from the decision. Xie (2026), establishes that evaluations, and by extension, scoring, should connect to something the team will do with the result, in one of two forms: a decision (ship or hold, which model, whether a change is real progress) or a diagnosis (which quality dimension or criterion is weak, and what to improve next) [9]. A scoring that drives neither is a vanity metric and should not be built [9].

State the objective in one sentence. Name what "good" looks like, which behaviours must be measured, and which decisions the scores will inform [9]. For example, "decide whether the summarisation agent is ready for pilot by measuring faithfulness and completeness against a reviewed reference set."

Name the decision type; it sets the scoring and aggregation. The same criterion results can warrant different verdicts depending on the decision.

- **Launch / release gate.** Is it good enough to ship? Gate on critical criteria, where one serious failure blocks release.
- **Regression detection.** Did this change make things worse? Threshold rules or pairwise comparison against a fixed baseline.
- **Model comparison.** Which system is better? Pairwise judging aggregated into win-rates or an Elo-style ranking.
- **Diagnostic review.** Where is it weak? The full per-criterion profile, with nothing collapsed into a single number.
- **Production monitoring.** Is live quality holding? Track the pass-rate and the shape of the score distribution for drift.

Match the rigour to the stakes. How much to invest in the scoring system (human-reviewed references, a second independent judge, tighter calibration) should be proportional to what rides on the score. A score feeding an internal dashboard and one gating a customer-facing release warrant very different scrutiny, even on the same architecture.

A release gate needs a single, defensible pass or fail; diagnosis needs the opposite: the full criterion profile. Defined well, the decision determines which dimensions to decompose, which scoring method fits (Section 7), and which aggregation rule to apply (Section 8).

7. Measuring Each Criterion

Once the quality to be scored has been decomposed into criteria, each criterion needs a defined method of measurement. There is no single correct method: how a criterion should be measured depends on the kind of judgment it requires and on the decision the evaluation supports. An objective constraint can be checked mechanically, whereas a graded quality such as helpfulness needs an actual judgment.

Derive criteria from the decision the score supports and from real failure modes such as known incidents and bad cases, rather than from a generic list; keep them minimal, observable, and non-overlapping; mark which are critical, and so must gate, versus merely graded; and revisit the set as new failures surface.

- **Deterministic checks (no LLM).** Best for objective, machine-verifiable constraints – schema or format validity, presence of a required field, tool-call correctness, exact match, or a unit-test pass. Example: "Does every factual sentence carry a citation?" can be answered by a parser. These are the cheapest and most reliable signals, so apply them first and reserve LLM judgment for what cannot be checked mechanically.
- **Binary pass/fail (checklist).** Convert the criterion into one or more yes/no questions the judge answers with evidence. Example, for citation quality: "Is each cited source actually relevant to the claim it supports?" Decomposing criteria into binary questions is the approach CheckEval [6] shows improves agreement between evaluators and reduces variance; it is a sound default for most subjective criteria.
- **Anchored ordinal labels (severity).** When partial credit genuinely matters and a yes/no verdict would discard information, use a small set of clearly anchored labels – for example pass, minor issue, major issue, fail – each with an explicit operational definition. This preserves graded judgment but reintroduces the ordinal ambiguity discussed in Section 2, so reserve it for criteria whose intermediate states are well defined and carry real signal.
- **Pairwise / comparative judgment.** Instead of scoring one output in isolation, ask the judge which of two outputs better satisfies the criterion (A, B, or tie), optionally against a fixed reference or baseline. Comparative judgments are typically more stable than absolute scores and suit model comparison and regression detection; their outcomes are combined into win-rates or rankings (see Section 8). The cost is more comparisons and the need for a baseline to compare against.
- **Probability-weighted / continuous scoring.** Rather than force a single discrete label, derive a continuous score from the judge's token probabilities over the label set, for example the expected value across the rating tokens. This is the mechanism behind Liu, Y. et al (2023) [2], and Girrbach et al. (2025) [8] show it reduces the tie compression and instability of discrete Likert outputs. It yields

finer-grained, more deterministic signals but requires access to model log-probabilities, which not every model endpoint exposes.

In practice these are layered: deterministic checks for hard constraints, binary checklist items for most subjective criteria, and comparative or probability-weighted methods where finer discrimination or a robust ranking is required.

8. Aggregating Criterion Outcomes

After each criterion has been measured, the system must combine those outcomes into the signal the decision needs. Aggregation is where much of the real measurement design lives: the same criterion results can produce very different verdicts depending on the rule applied, so the rule should be explicit, versioned, and chosen to fit the decision. The main patterns and their trade-offs are:

- **Gating (worst-case / critical-fail).** Worst-case: The item passes only if no criterion falls below a defined floor; Critical-fail: any critical failure caps the result regardless of the others.
  Pros: safety-first, with no compensation – a serious failure cannot be averaged away.
  Cons: harsh and low-resolution, since one minor issue can sink an otherwise strong output, and it offers little gradation for tracking improvement.
- **Weighted sum or average.** Map each criterion outcome to a value, weight by importance, and combine into one number.
  Pros: flexible, expresses priorities, and yields a familiar continuous score.
  Cons: reintroduces the interval assumption criticized in Section 2 (it averages ordinal labels), lets strengths mask serious weaknesses, and hides which criterion drove the result; the weights themselves must be justified.
- **Threshold / counting rules.** Pass when at least N criteria pass and no critical criterion fails, or similar count-based logic.
  Pros: interpretable, auditable, and tunable, and it avoids treating labels as interval data.
  Cons: the thresholds and the set of "critical" criteria must be calibrated and maintained.
- **Decision tree / rule hierarchy.** Encode the policy explicitly: critical criteria gate first, then graded criteria adjust the result within the surviving band.
  Pros: closest to how teams reason about release readiness, and it handles the critical-versus-minor distinction cleanly.
  Cons: more logic to design, document, and maintain.
- **Profile / vector (no collapse).** Report the per-criterion outcomes as a profile rather than collapsing them into one number.
  Pros: maximum diagnostic information, with nothing hidden.

Cons: not a single comparable figure, so any dashboard or gate still needs one of the rules above on top of it.

Because the right rule depends on the decision the evaluation supports (Section 5), the table below maps common decisions to a sensible default, with each aggregation described in plain terms.

Decision	Recommended default	How it works, in plain terms
Launch / release gate	Gating (critical-fail)	The item passes only if nothing critical fails; one serious problem blocks release no matter how strong the rest is.
Regression detection	Threshold / counting, or pairwise vs. last-good	Flag a problem when too few criteria pass, or when the new version loses head-to-head against the last good one.
Model comparison	Pairwise win-rate / Elo	Have the judge pick the better of two outputs many times, then rank systems by how often each wins.
Diagnostic review	Profile / vector (no single score)	Keep each criterion's result separate so you can see exactly what passed and what failed.
Production monitoring	Threshold plus drift on the distribution	Track whether the pass-rate and the shape of the score distribution move over time.

When criteria are measured comparatively, aggregation instead takes the form of win-rates or Bradley–Terry / Elo-style rankings (statistical methods that convert many head-to-head win/loss outcomes into a single strength rating per system, placing every system on one comparable scale) across many pairings; useful for ranking systems against one another, though not for an absolute pass/fail gate.

Whichever rule is chosen, keep it versioned and validate it against reviewed examples. As a default, prefer gating or threshold logic (or reporting the full criterion profile) over collapsing ordinal labels into a single mean.

A note on averaging: the problem is not averaging itself, but averaging ordinal labels as if their steps were equal. Averaging binary pass/fail outcomes produces a pass-rate, which is a legitimate proportion, and averaging genuinely continuous scores such as probability-weighted scores is defensible; it is only ordinal severity labels that should not be collapsed into a mean.

9. Practitioner Recommendations

The following recommendations are not presented as empirical evidence; the evidence base for the problem and mechanisms is the external literature cited above.

- **Avoid unanchored prompts such as “rate this from 1 to 5”** when the score affects launch readiness, quality reporting, or model comparison.
- **Prefer decomposed criteria** and checklist-style evaluation over single-score evaluation.
- **Use deterministic checks** for objective constraints before applying LLM judgment.
- **Prefer pairwise/comparative judging** over absolute scoring: ask the judge which of two outputs is better against a fixed baseline, then aggregate the verdicts into win-rates or an Elo-style ranking.
- Where the judge model exposes token log-probabilities, **use probability-weighted scoring** (the expected value across the rating tokens) to get finer, more deterministic signals and reduce ties near the top of the scale. Fall back to discrete labels when log-probabilities are not available.
- **Have the LLM produce evidence**, observations, and criterion-level judgments; compute final scores outside the LLM.
- **Use a judge model** from a different family than the system under evaluation to limit self-enhancement bias and monitor for that bias whenever the judge and the generated output share a model lineage.
- **Track judge disagreement** and score variance as first-class quality signals.
- When you refine your scoring system in response to human feedback, judge disagreements, or mis-scored items, **validate each change against a benchmark** (a fixed set of items with trusted reference scores). Re-running it confirms the change raised agreement with those references without regressing items already scored correctly, while adding new disagreement cases keeps it covering the judge's real blind spots.

10. Conclusion

The question is not whether LLMs can score. The question is whether the score represents a reliable measurement. Direct 1-5 scoring is fragile because it combines ordinal-scale ambiguity with LLM judge bias, prompt sensitivity, context sensitivity, and under-specified criteria. A formal evaluation system should therefore treat the LLM as one component in a measurement architecture, not as the measurement instrument itself.

The practical path is to preserve the convenience of numeric reporting while changing how the number is produced: evidence first, criteria second, deterministic aggregation third, and calibration throughout.

References

[1] Sullivan, G. M., and Artino, A. R. Jr. Analyzing and Interpreting Data From Likert-Type Scales. Journal of Graduate Medical Education, 2013. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3886444/

[2] Liu, Y. et al. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment, 2023. arXiv:2303.16634. https://arxiv.org/abs/2303.16634

[3] Wang, P. et al. Large Language Models are not Fair Evaluators, 2023. arXiv:2305.17926. https://arxiv.org/abs/2305.17926

[4] Zheng, L. et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena, 2023. arXiv:2306.05685. https://arxiv.org/abs/2306.05685

[5] Kim, S. et al. Prometheus: Inducing Fine-grained Evaluation Capability in Language Models, 2024. arXiv:2310.08491. https://arxiv.org/abs/2310.08491

[6] Lee, Y. et al. CheckEval: A reliable LLM-as-a-Judge framework for evaluating text generation using checklists, 2024. arXiv:2403.18771. https://arxiv.org/abs/2403.18771

[7] Microsoft. Evaluation and observability for generative AI applications in Microsoft Foundry, 2026. https://learn.microsoft.com/en-us/azure/ai-foundry/concepts/evaluation-approach-gen-ai

[8] Girrbach, L. et al. Reference-Free Rating of LLM Responses via Latent Information, 2025. arXiv: 2509.24678. https://arxiv.org/abs/2509.24678

[9] Xie, S. Practical Guidance on Agent Evaluation: A 10-Step Playbook for Building, Running, and Maturing Eval Suites for Enterprise Agents. Microsoft, Agent Quality & Evaluation, 2026.