ACM Response Document to NIST RFI: Towards Best Practices for Automated Benchmark Evaluations

March 2026

Question 1: The usefulness and relative importance of the included practices and principles (in general or for specific use cases, types of evaluations, or audiences):

General Feedback for this Question:

Agentic evaluation needs a dedicated methodology, not just protocol settings

The document treats agent scaffolding as one row in the protocol settings table. In practice, agentic evaluation is a qualitatively different problem. Three things warrant dedicated guidance. Scaffold sensitivity is a confounding variable, not just a configuration choice. The same model can vary by double-digit percentage points on the same benchmark depending solely on the scaffold used. A weaker model with a strong scaffold can outperform a stronger model with a weak one. The document should recommend scaffold sensitivity analysis as standard practice for agent evaluations, not just scaffold disclosure.

Binary pass/fail grading overstates real-world utility. In practice, a large fraction of benchmark-passing agent outputs would not meet real-world acceptance criteria. Code that passes unit tests is not the same as code that would survive review. The document should recommend partial-credit scoring and downstream quality validation for agentic benchmarks.
The document's treatment of evaluation awareness (Section 2.4.1, item 6) should be expanded to cover strategic evaluation manipulation. Frontier models have been observed to intentionally underperform during evaluations (sandbagging) and to misrepresent the actions they take. This undermines the assumption that test-time behavior predicts deployment behavior. The document should provide guidance on detecting and reporting these behaviors.

Safety evaluation must distinguish open-weight models from hosted models

The document has no framework for responsible AI benchmarking. The most important missing distinction is between open-weight and hosted model evaluation.
Hosted models (commercial APIs) ship with layered safety systems. These typically include input classifiers and output filters for categories such as CSAM, celebrity likeness detection, PII scrubbing, and copyright detection. When you benchmark a hosted model for safety, you are

measuring the model plus its entire safety stack. A strong safety score tells you the safety stack works. It does not tell you the underlying model is inherently safe.

Open-weight models ship without any of this. Safety fine-tuning on open-weight models can be removed quickly and cheaply. Small adapter weights can nullify safety training entirely. The same base model that scores well through a provider's filtered API can produce harmful content when served without guardrails. This is not hypothetical. It reflects documented deployment patterns.

MLCommons AILuminate already distinguishes between "bare models" and "AI systems" (models plus guardrails)[1]. The document should adopt this distinction by adding guardrail configuration as a first-order protocol setting in Table 2.2, recommending that open-weight models be evaluated both with and without common guardrails, and flagging that direct safety score comparisons between open-weight and hosted models are misleading without guardrail disclosure.

More broadly, most safety benchmarks rely on binary pass/fail metrics that miss severity gradients, and test only single-turn interactions despite models being deployed in multi-turn contexts. Refusal calibration also matters. There is a strong observed correlation between a model's ability to block toxic prompts and its rate of wrongly rejecting safe ones. Over-refusal and under-refusal should be benchmarked as a paired measurement.

Multimodal evaluation is absent despite multimodal deployment being the norm

The abstract says the document covers "AI agent systems," but every practice assumes text-only models. Frontier models deployed today accept and generate text, images, audio, and video. The document should either extend the scope to cover multimodal evaluation or clearly state it is out of scope, with a timeline for follow-up.

The safety aspect is the most urgent. Text-only safety alignment can show near-zero attack success rates, but the same models become highly vulnerable when adversarial content is delivered through images or other non-text modalities. Cross-modal jailbreaks, in which harmful requests are encoded in images to bypass text-based safety filters, exhibit high transferability across model architectures. Text-only safety evaluation creates false confidence about models that are actually deployed multimodally.

Beyond safety, the multimodal evaluation landscape is fragmented across over 100 benchmarks with no unified methodology. Audio evaluation remains nascent. Any-to-any models have

[1] "Benchmark for General-Purpose AI Chat Model." Mlcommons.org, 2025, ailuminate.mlcommons.org/benchmarks/general_purpose_ai_chat/1.0-en_us-official-ensemble. Accessed 25 Mar. 2026.

virtually no dedicated frameworks. At a minimum, the document should caution against treating text-only benchmark results as representative of multimodal system behavior.

Specific Areas to Address for this Question:

Introduction:

The Introduction notes that models are often embedded in chatbot and agent systems, but the current framing can still leave readers uncertain about whether the practices are intended for base models, end-user systems, or both. A brief paragraph early in the Introduction could clarify that benchmark results may not transfer cleanly from a standalone model to a tool-using, retrieval-augmented, or human-supervised system.

A brief clarification is needed that addresses how these practices may also be relevant when a language model or AI agent is embedded in a physical system, such as a robot or other embodied AI system. However, benchmark results in such settings are often even less transferable than in software-only deployments because observed performance may depend not only on the model and scaffold but also on perception noise, control latency, actuator execution, hardware resource constraints, environmental dynamics, reset conditions, and human intervention policies. As a result, automated benchmark results for a simulated or software-only agent should not be interpreted as sufficient evidence for claims about real-world embodied system performance, safety, or reliability.

Practice 1.2 Select benchmarks that meet evaluation objectives.

Benchmark suitability should provide stronger guidance on threshold setting and decision risk. Evaluators should predefine the decision context, acceptable error tradeoffs, and consequences of false positives and false negatives when using benchmark results to support deployment or safety claims. This is especially important where the benchmark is being used not merely for comparison but to justify a consequential decision in industry use cases. Evaluators should be asked to report major failure modes or error categories alongside aggregate metrics, as summary scores can obscure practically important weaknesses.

Explicit guidance on the decision context of any safety governance or decision frameworks used in the process. Also, explicit guidance on decision context, acceptable error tradeoffs, and the consequences of false positives and false negatives when benchmark results are used to support deployment or safety claims. One important point to note is that all benchmarks are limited, as the number of possible prompts is exponential. Safety cannot be properly evaluated by a finite benchmark.

Practice 3.2 Share details of evaluation and evaluation data:

Expand to explicitly include serving setup and resource information, such as accelerator type and count (GPU type, number of GPUs, distributed systems used, etc.) when self-hosting. For operational decisions, these dimensions are often as important as the score.

Question 2: Which practices are emerging vs. existing best practices; Any content that is incorrect, unclear, or otherwise problematic.

General Feedback for this Question:

The “emerging practice” label should be split into two tiers

Benchmark versioning, commit hashes in logs, and sandboxing are standard engineering practices that happen to be new to AI evaluation. Evaluation, awareness detection, and item pattern analysis are genuinely at the research frontier. Lumping them under one label makes it hard for practitioners to prioritize. Consider distinguishing “operationally mature but underadopted” from “research frontier.”

Reasoning model evaluation deserves a note

The document references reasoning effort as an inference setting but does not address challenges specific to reasoning models (o1/o3-style). These models use hidden chains-of-thought that complicate evaluation, create variable inference costs that standard benchmarks ignore, and raise open questions about whether the chain-of-thought faithfully represents the model’s actual reasoning. Newer benchmarks designed for these models show top scores well below what existing benchmarks produce, suggesting evaluation instruments need to evolve with model architectures. A forward-looking note would be appropriate.

Specific Areas to Address for this Question:

Section 2.4

Benchmark contamination and saturation should be elevated from debugging to a standalone practice. Section 2.4 treats contamination as a bug to catch during quality assurance. It has become a systemic problem. Widely used benchmarks have saturated or been dropped from evaluation indices due to contamination concerns. Dynamic benchmarks and contamination-resistant designs are emerging as partial solutions. The document should treat benchmark lifecycle management, including saturation monitoring, contamination

detection, and benchmark retirement criteria, as a standalone practice rather than a debugging step.

Question 3: Other common terms in benchmark evaluation that would be useful to define in the glossary, or for the field to use in a standardized manner:

General Feedback on this Question:

Guardrails/safety systems: External mechanisms (input classifiers, output filters, content moderation layers) applied on top of a model during serving. Needed to distinguish model-intrinsic from system-level safety performance.

Open-weight model: A model whose weights are publicly released for self-hosting, as distinct from a hosted or API-only model.

Bare model vs. AI system: Per MLCommons, evaluating a model without safety mitigations versus evaluating the full deployed system, including guardrails and filters.

Cross-modal jailbreak: Encoding harmful requests in one modality (e.g., image) to bypass safety filters designed for another modality (e.g., text).

Over-refusal / under-refusal: Blocking legitimate requests versus complying with harmful ones. Both are measurable deployment failures.

Sandbagging: Strategic underperformance by a model during evaluation.

Scaffold sensitivity: The degree to which agent benchmark performance is driven by the scaffold rather than the underlying model.

Construct collapse: Equating benchmark performance with the broader concept the benchmark was designed to measure.

Specific Areas to Address for this Question:

1. DEFINING EVALUATION OBJECTIVES AND SELECTING BENCHMARKS

There should be a clear distinction between the measurement construct, benchmark task, operationalization, and decision use. This section separates intended use from what should be measured, but in practice, many evaluators collapse these concepts. For example, a multiple-choice science QA benchmark may operationalize only one dimension of a broader construct, such as scientific assistance. Adding a short, boxed example or figure showing the chain from

deployment question -> evaluation objective -> construct -> benchmark task -> metric -> decision would make this section more actionable. A brief addition here could help evaluators distinguish optimization targets from decision-support measurements (Goodhart's law).

Practice 3.1 Conduct statistical analysis and uncertainty quantification.

Explicitly discuss mean vs median and other robust statistics. In many evaluation settings, especially those involving latency, token usage, runtime, or long-horizon agents, the distribution can be highly skewed. So median, percentiles, or trimmed means may sometimes better reflect practical system behavior than the arithmetic mean alone.

Question 4: When automated benchmark evaluations are more or less useful relative to other evaluation paradigms.

General Feedback on this Question:

Automated benchmarks are necessary but insufficient for safety assessment

Most safety benchmarks use binary metrics, single-turn interactions, and predetermined test items. The consequential safety properties, such as behavior under sustained adversarial probing, impact on vulnerable populations, and low-probability high-consequence failures, require combining automated benchmarks with red-teaming, field testing, and post-deployment monitoring. The document should say this plainly in the Introduction or Table I.1.

Automated benchmarking is less mature for non-text modalities

For image and video generation, automated metrics correlate weakly with human quality judgments. For audio, naturalness and speaker evaluation remain largely human-dependent. The document should note that its practices apply most directly to text evaluation and require adaptation for other modalities.

Regulatory requirements now exceed what automated benchmarks can deliver

The EU AI Act, OMB M-25-21, and state AI laws require evaluation activities beyond what automated benchmarks provide, including adversarial testing, impact assessments, and lifecycle monitoring. International efforts have also exposed that safeguards effective in English may fail in other languages. Table I.1 should reference these requirements so practitioners understand the limits of automated benchmarking in their compliance context.

Specific Areas to Address for this Question:

Practice 1.2 Select benchmarks that meet evaluation objectives.

A major failure mode in teams is the benchmark-versus-real-world gap. The draft already distinguishes direct measurement, conceptual relation, and prediction of downstream outcomes, and should also explicitly document the simulation-to-real limitations, including which real deployment factors are omitted, whether the benchmark is static or dynamic, and which claims should not be inferred from the benchmark.

For embodied AI systems, evaluators should explicitly document whether a benchmark measures model capability, software-agent capability, or physical-system capability, and which deployment-relevant factors are omitted. This section already asks evaluators to document what a benchmark measures and how it relates to the evaluation objective. For embodied systems, the benchmark-versus-deployment gap often includes simulation-to-real limitations and system factors beyond the model itself, so this should be explicitly called out.

2.1.1 Evaluation protocol design principles

In practice, systems can differ substantially in benchmark scores depending on how much prompt, scaffold, and hyperparameter search was performed. For fairness and reproducibility, evaluations should document the search budget and how final configurations were selected.

2.1.2 Common evaluation protocol settings

Consider recommending that judge capability be empirically validated for the domain being assessed. If technically feasible, the judge should be at least as strong as the generation model or the task difficulty. The report can add guidance on judge drift, judge version pinning, and adversarial grader gaming.

Practice 2.3 Run the evaluation and track results:

Expand this section to explicitly cover historical trend analysis and regression testing. Many automated evaluation systems are recurring operational pipelines rather than one-time experiments, so it would help to define common run types such as nightly runs, weekly runs, release candidate runs, and ad hoc experimental runs, along with expectations for metadata, comparability, and alerting on regressions.