Shumaker

MARCH 13, 2026 | PUBLICATION

The AI Benchmark: The Most Important Clause You've Never Used (Part 2)

INDUSTRY SECTOR

Technology

SERVICE LINE

Technology, Data Privacy, Cybersecurity & AI

RELATED PROFESSIONALS

Brian C. Focht

MEDIA CONTACT

Wendy M. Byrne
wbyrne@shumaker.com

In Part 1 of this post, we discussed why artificial intelligence (AI) benchmark testing belongs in every contract you negotiate involving AI, why benchmarking is important for every kind of AI system, and what can go wrong when benchmarking is skipped or minimized. In this part, we will discuss what a practical framework for AI benchmarking looks like.

D. THE ESSENTIAL COMPONENTS OF A WORKABLE BENCHMARK

A practical benchmarking framework in contract negotiation aims to do one thing: turn vague promises like "high quality" or "good" performance into an objective analysis process. The framework should define what will be measured, how it will be measured, when and under what circumstances it will be measured, who is responsible for collecting and verifying the measurements, and what will happen if the results fall short.

Without these key parameters, "benchmarking" becomes little more than marketing material or fodder for future lawsuits.

Defining the Purpose and Scope

The framework should start with what is being measured and why. The purpose may be to confirm system suitability for acceptance, validate quality, compare standards, or support renewal decisions. The purpose of the testing must be clear to all parties involved. The application must be specified: whether it applies to the whole agreement, a specific product line, a statement of work, or targeted functions. Without a clear purpose and scope, your benchmark will generate data but yield no actionable results.

Clear Performance Criteria

A framework's usefulness depends on what it tests. It must define clear standards, metrics, or indicators to measure performance. These can be quantitative (such as uptime, error rates, defect levels, and cost

efficiency) or qualitative (like compliance with internal policies and adherence to pre-programmed specifications). In any case, the criteria should be measurable, consistently and routinely assessed, and specific enough that two reasonable parties reviewing the same results would arrive at the same conclusion.

Measurement Methodology

Even when parties agree on *what* to measure, one of the main sources of disputes is *how* to measure it. A successful benchmarking framework clearly defines how the measurements will be taken. The framework should address the testing method, the relevant comparison (or control) set, the baseline period, the data sources, the operating conditions during testing, and, *most importantly*, the assumptions that will be applied. It also specifies what the measurements will be compared against. Internal baselines? Agreed service levels? Industry peers or an external standard? Without knowing the standard against which your measurements are compared, you risk leaving the quality of your framework to the whims of changing inputs or comparison points.

Timing and Frequency

Will your benchmarking occur before acceptance, on a recurring schedule, or upon a trigger event (e.g., a material or catastrophic service failure, data breach, price increase, or contract renewal)? A successful framework must clearly specify when and at what frequency the testing will take place. Additionally, if the timing of the testing depends on a party's requests, it should include limits such as a notice requirement or safeguards against misuse, process abuse, or business disruption. Without these limits, benchmark testing could shift from a legitimate tool for contract governance to a tactic for ongoing renegotiation of terms.

Role Allocation

An effective framework requires allocation of responsibilities, including who may initiate the test, who pays for it, who collects and/or supplies the necessary data, and who has the right to review and/or challenge the results. Will the supplier be responsible for the testing and reports? Will the customer receive audit and verification rights? The stakes may be high enough (or the distrust between the parties is such) that an independent third party with expertise runs the show.

Input and Condition Rules

You wouldn't plan your morning drive to work based on downtown traffic at midnight, and you shouldn't run benchmark tests using unsuitable or nonrepresentative samples. Suppliers shouldn't test in their own environments, and tests shouldn't be conducted during unusual demand spikes. Your framework should cover normalization, exclusions, and key assumptions. Testing should exclude, for instance, downtime caused by conditions outside the supplier's control (unless such conditions happen often and are expected) and peer comparisons involving dissimilar businesses.

Validation and Disputes

The main goal of the framework is to achieve results accepted by all parties. Therefore, the method for validating those results and resolving disputes must be incorporated into the framework itself. Adding a right to appeal a dispute will only lead to more conflict if it does not include clear provisions for how results are shared, how a party can object, what evidence can be submitted (and what can be responded with), and the procedures and dispute-resolution forum to be used.

Consequences

Your framework is just marketing and busywork if there are no consequences for the results. It must clearly define what happens if performance meets, exceeds, or falls short of the expected standard. Consequences could include, for example, acceptance or rejection of deliverables, price adjustments, service credits, remediation, or repricing. Additionally, consider when it might be appropriate for a failed test to trigger a re-test. A framework without consequences is a pointless waste of negotiating time.

Change over Time

We are not dealing with a static process that remains unchanged over time. As services evolve, technology advances, and industry norms shift, a rigid benchmark quickly becomes outdated. Consider a process for updating metrics, methodologies, thresholds, and comparison groups. Ensure there is at least a general requirement for the parties to consider changes to the benchmark if there are significant shifts in technology, applicable law, or operating conditions.

Now that we've discussed what makes a general benchmarking framework… work, let's talk about how to make one workable and enforceable for AI.

E. AN AI BENCHMARKING FRAMEWORK

Utility and Risk Set the Scope

The most effective approach starts by clearly defining the AI's intended use, what it will influence, and the deployment's risk level. A low-risk internal drafting tool requires less frequent, less strict benchmarking with lighter consequences. At the same time, a customer-facing system or an agentic AI integration should impose higher standards, phased rollout procedures, enhanced auditability and transparency, and more explicit remedies for failure. Risk tiering is valuable in establishing your framework because it enables you to be thorough without being unreasonable. It also helps vendors understand what is expected and why.

Create a Real-Life Sandbox

From there, you should ensure that representative testing reflects real usage scenarios, preferably using real but non-'live' data. This testing should cover common scenarios, high-impact edge cases, and adversarial inputs designed to identify model vulnerabilities. For generative tools, the test suite should include prompt sets and an evaluation rubric to maintain consistent performance scoring. For retrieval tools, the suite should verify grounding, citation accuracy, and permission boundaries. For agentic tools, the testing will likely need a larger "sandbox," such as one that mimics full production environments. This type of environment allows the organization to test tool-use logic, authorization limits, and error recovery without risking actual systems.

Meaningful Numbers that Matter

Next, benchmarks must be converted into measurable criteria, including clearly defined success and failure metrics. You don't need to reduce every concept to a single number but should avoid vague commitments such as "commercially reasonable" or "industry standard" when performance is crucial. In many deployments, a hybrid approach works best: a small set of quantitative thresholds (such as maximum hallucination rate on defined domains, minimum accuracy on specific tasks, limits on false positives, or latency targets) combined with a scoring system for qualitative factors (such as instruction-following, tone, and completeness). The contract should also require agreement on scoring methods and, when humans evaluate results, basic consistency safeguards to prevent disputes.

Test in the Deployment Environment

Performance results mean little if the test environment is unclear or if test data is mishandled. Clarify whether tests happen in a sandbox or a controlled production-like environment, what data will be used (synthetic, anonymized, or production samples), and what security and retention controls apply to prompts, outputs, and logs. I usually recommend that tests be conducted on the equipment that will be used once the tool is deployed, not on the supplier's systems. If you oppose the supplier using test data in their own models, explicitly include a prohibition on such use. Vendors often take an expansive view of their right to use your data for "service improvement."

Horse First, Then Cart

Successful completion of benchmark testing should be explicitly required before going live, including any necessary validation. Incorporate an evaluation or pilot period into the contract, with objective acceptance criteria linked to benchmarks and clear consequences if benchmarks are not met. Consequences for failing benchmarks can include remediation obligations, timelines, and options such as an extension of the pilot, credits, or termination rights. For agentic AI, staged rollout requirements are especially important. I strongly recommend starting in an observe-only mode, then moving to supervised actions that require human approval and only expanding autonomy after consistent benchmark success. This approach reduces risk while still allowing for innovation.

The Times, They Are a-Changin'

Finally, address what happens when things change. The agreement should define "material changes" that trigger re-testing, such as model version updates, changes in model providers or sub-processors (which happen far more frequently than most people realize), modifications to retrieval sources, expanded permissions for agents, or significant configuration changes. Require notice, regression testing, and the ability to roll back changes if benchmarks regress. This is where many deals fail: vendors want the freedom to update systems continuously, and customers want stability. Those positions are often in direct conflict. Benchmark-driven update controls offer a middle ground by allowing updates only if they maintain agreed-upon performance and safety thresholds.

F. CONVERTING BENCHMARK FAILURE INTO REAL CONTRACTUAL LEVERAGE

Benchmarking only works if the results are clear and legitimate, and the remedy is agreed to in advance. Require auditability that is sufficient to validate performance and investigate issues. Doing so typically includes preserving relevant logs and test results, maintaining action logs for agentic systems, and providing evaluation reports upon request. It also includes incident-style reporting for AI failures that cause or threaten harm, whether that harm is quality-related (bad outputs), privacy or security-related (data exposure), or operational (unsafe actions).

Remedies should be aligned with the risk and the business impact. In lower-risk deployments, remedies may focus on remediation timelines and credits. In higher-risk deployments, remedies should include the ability to suspend certain features, restrict autonomy, require human-in-the-loop controls, and terminate for repeated or material failures. Placing all practical responsibility on the customer to "configure around" the AI's shortcomings not only indicates that the supplier has little confidence in its own product but may also result in considerable unforeseen risk and exposure, given the nature of the technology. If the AI is part of what is being purchased, then meeting benchmark thresholds should be part of what the vendor is obligated to deliver.

Conclusion

AI contracting is shifting from feature negotiations to control negotiations—the things you get may be nice, but it is critical to know *for sure* what you're actually getting. Benchmark testing is one of the most valuable tools for making that shift practical. It creates a shared definition of success, validates performance before you give it complete access to your system and operations, and provides a repeatable mechanism for managing updates, drift, and the unique risks of autonomous behavior. Whether the AI is a simple generative assistant or a complex agentic platform interacting with enterprise systems, benchmark-driven contract terms can be the difference between a successful deployment and a costly, hard-to-unwind experiment. Or worse…

If you have any questions about incorporating benchmark testing, performance metrics, and drift monitoring into your AI vendor agreements, or would like help drafting practical contract language that ties AI "performance" to measurable acceptance criteria, remedies, and auditability, please contact Brian Focht or another member of Shumaker's Technology, Data Privacy, Cybersecurity & AI Service Line.

Shumaker