What is AI agent evaluation?

AI agent evaluation refers to the process of assessing and understanding the performance of an AI agent in executing tasks, decision-making and interacting with users. Given their inherent autonomy, evaluating agents is essential to promote their proper functioning. AI agents must behave in accordance with their designers' intent, be efficient and adhere to certain ethical AI principles to serve the needs of the organization. Evaluation helps verify that agents are meeting such requirements, and also helps improve the agent's quality by identifying areas for refinement and optimization.

Generative AI (gen AI) agents are often evaluated on traditional text-to-text tasks, similar to standard large language model (LLM) benchmarks, where metrics such as coherence, relevance and faithfulness of the generated text are commonly used. However, GenAI agents typically perform broader and more complex operations — including multi-step reasoning, tool calling and interaction with external systems — which require more comprehensive evaluation. Even when the final output is text, it may be the result of intermediate actions like querying a database or invoking an API, each of which needs to be evaluated separately.

In other cases, the agent may not produce textual output at all, instead completing a task such as updating a record or sending a message, where success is measured by correct execution. Therefore, evaluation must go beyond surface-level text quality and assess overall agent behavior, task success and alignment with user intent. In addition, in order to avoid a development of highly capable but resource-intensive agents, which limit their practical deployment, cost and efficiency measurements must be included as part of the evaluation.

Beyond measuring task performance, evaluating AI agents must prioritize critical dimensions such as safety, trustworthiness, policy compliance and bias mitigation. These factors are essential for deploying agents in real-world, high-stakes environments. Evaluation helps ensure that agents avoid harmful or unsafe behavior, maintain user trust through predictable and verifiable outputs, and resist manipulation or misuse.

To achieve these functional (quality, cost) and non-functional (safety) goals, evaluation methods can include benchmark testing, human-in-the-loop assessments, A/B testing and real-world simulations. By systematically evaluating AI agents, organizations can enhance their AI capabilities, optimize automation efforts and enhance business functions while minimizing risks associated with unsafe, unreliable or biased agentic AI.

How AI agent evaluation works

Evaluating an AI agent requires a structured approach within a broader formal observability framework. Evaluation (or eval) methods differ widely, but the process typically involves the following steps:

1. Define evaluation goals and metrics

What's the purpose of the agent? What are the expected outcomes? How is the AI used in real-world scenarios?

See "Common AI agent evaluation metrics" for some of the most popular metrics, which fall under the categories of performance, interaction and user experience, ethical and responsible AI, system and efficiency and task-specific metrics.

2. Collect data and prepare for testing

To evaluate the AI agent effectively, use representative evaluation datasets, including diverse inputs that are reflecting real-world scenarios and test scenarios that simulate real-time conditions. Annotated data represents a ground truth that AI models can be tested against.

Map out every potential step of an agent's workflow, whether it's calling an API, passing information to a second agent or making a decision. By breaking down the AI workflow into individual pieces, it's easier to evaluate how the agent handles each step. Also consider the agent's entire approach across the workflow, or in other words, the execution path the agent takes across solving a multi-step problem.

3. Conduct testing

Run the AI agent in different environments, potentially with different LLMs as their back-bone, and and track performance. Break down individual agent steps and evaluate each. For example, monitor the agent's use of retrieval augmented generation (RAG) to retrieve information from an external database, or the response of an API call.

4. Analyze results

Compare results with predefined success criteria if they exist, and if not, use LLM-as-a-judge (see below). Assess tradeoffs by balancing performance with ethical considerations.

Did the agent pick the right tool? Did it call the correct function? Did it pass along the right information in the right context? Did it produce a factually correct response?

Function calling/tool use is a fundamental ability for building intelligent agents capable of delivering real time, contextually accurate responses. Consider a dedicated evaluation and analysis using a rule-based approach along with semantic evaluation using LLM-as-a-judge.

LLM-as-a-judge is an automated evaluation system that assesses the performance of AI agents by using predefined criteria and metrics. Instead of relying solely on human reviewers, an LLM-as-a-judge applies algorithms, heuristics or AI-based scoring models to evaluate an agent's responses, decisions or actions.

See "Function Calling evaluation metrics" below.

5. Optimize and iterate

Developers can now tweak prompts, debug algorithms, streamline logic or configure agentic architectures based on evaluation results. For example, customer support use cases can be improved by accelerating response generation and task completion times. System efficiency can be optimized for scalability and resource usage.

Common AI agent evaluation metrics

Developers want agents to work as intended. And given the autonomy of AI agents, it's important to understand the "why" behind the decisions that AI makes. Review some of the most common metrics that developers can use to successfully evaluate their agents.

Task-specific

Depending on the AI application, specific evaluation metrics for quality can apply:

- **LLM as a judge** evaluates the quality of AI text generation regardless of the availability of ground-truth data.
- **BLEU and ROUGE** are lower-cost alternatives that evaluate the quality of AI-generated text by comparing it to human-written text.

Other functional metrics for assessing AI agent performance include:

- **Success rate/task completion** measures the proportion of tasks or goals that the agent completes correctly or satisfactorily out of the total number attempted.
- **Error rate** is the percentage of incorrect outputs or failed operations.
- **Cost** measures resource usage, like tokens or compute time.
- **Latency** is the time taken for an AI agent to process and return results.

Ethical and responsible AI

- **Prompt injection vulnerability** evaluates success rate of adversarial prompts, altering the agent's intended behavior.
- **Policy adherence rate** is a percentage of responses that comply with pre-defined organizational or ethical policies.
- **Bias and fairness score** detects disparities in AI decision-making across different user groups.

Interaction and user experience

For AI agents that interact with users, such as chatbots and virtual assistants, evaluators look at these metrics.

- **User satisfaction score (CSAT)** measures how satisfied users are with AI responses.
- **Engagement rate** tracks how often users interact with the AI system.
- **Conversational flow** evaluates the AI's ability to maintain coherent and meaningful conversations.
- **Task completion rate** measures how effectively the AI agent helps users complete a task.

Function Calling

These rule-based metrics help assess the operational effectiveness of AI-driven systems:

- **Wrong function name:** The agent attempted to call a function that exists but used an incorrect name or spelling, leading to a failure in execution.
- **Missing required parameters**: The agent initiated a function call but omitted one or more parameters that are necessary for the function to work.
- **Wrong parameter value type**: The agent supplied a parameter value, but its type (string, number, boolean) did not match what the function expected.
- **Allowed values**: The agent used a value that is outside the set of accepted or predefined values for a specific parameter.
- **Hallucinated parameter**: The agent included a parameter in the function call that is not defined or supported by the function's specification.

Here are some semantic metrics that are based on LLM-as-a-judge.

- **Parameter value grounding** helps ensure that every parameter value is directly derived from the user's text, the context history (such as previous outputs of API calls), or API specification defaults.
- **Unit transformation** verifies unit or format conversions (beyond basic types) between values in the context and the parameter values in the tool call.