The Agentic Evals Playbook

Author: Jack Shutler

# **1. Introduction and Purpose**

## **Goal of this Playbook**

This playbook is specifically designed for system architects who are developing enterprise systems with agentic LLMs at their core. It provides a general set of step-by-step instructions on designing an LLM Evaluation Framework for any AI use case, whether that's testing a foundation model, a fine-tuned model, or an agentic system using LLMs at their base.

The primary objective is to equip you with a structured approach to evaluate and validate agentic systems before deployment, ensuring they meet both business objectives and safety requirements. Unlike generalized LLM testing frameworks, this playbook focuses specifically on the unique challenges posed by agentic systems where LLMs make decisions, use tools, and perform complex sequences of actions with varying degrees of autonomy.

## **Audience (Enterprise System Architects)**

This playbook is written for technical leaders and system architects responsible for implementing agentic AI systems within enterprise environments. It assumes a working knowledge of LLMs and their capabilities, along with an understanding of enterprise software development practices. Whether you're building customer-facing agents, internal decision support systems, or domain-specific automation tools, this framework will help you establish rigorous evaluation protocols tailored to your specific use case.

## **Scope and Limitations**

This playbook does not go into detail about the specific capabilities you need to run. As we will touch on later, every use case will have a specific set of capabilities it needs to test for. Going into details on any one evaluation would be counterproductive to a general playbook.

What this playbook does cover:

* A comprehensive framework for evaluating text-based agentic systems
* Methods to test both business capabilities and safety guardrails
* Approaches for evaluating an agent's decision-making process
* Techniques for measuring tool usage effectiveness and safety
* Strategies for managing the combinatorial complexity of agent testing

What this playbook does not cover:

* Specific implementation details for any single industry vertical
* In-depth evaluation of multimodal capabilities (though principles may apply)
* Hardware performance optimization for agent deployment
* Complete security auditing protocols (though we address safety concerns)
* Detailed prompt engineering techniques (though we discuss evaluation of prompts)

The main reason we do any kind of testing is to provide insight into how the system works. LLM evals need to provide insight into questions such as: Does your system do what you designed it to do? Is it capable of performing actions you didn't design it to do? How does your system vary given different configurations? Can the system harm your business through interactions with users? Can it harm customers? Is it capable of any catastrophic human risk?

This playbook builds upon these fundamental testing goals but extends them to address the unique challenges of agentic systems, where risks and capabilities can emerge from complex interactions between components rather than from the LLM alone.

By following this playbook, you'll be equipped to design, implement, and maintain a robust evaluation framework that evolves alongside your agentic system, helping you to deploy with confidence while managing risks appropriately.

# **2. Understanding Agentic Systems**

## **Definition and Characteristics of Agentic Systems**

Agentic systems represent a significant evolution in artificial intelligence applications. An agentic system is an AI-powered software entity that can autonomously perceive its environment, make decisions, and take actions to achieve specified goals, with varying degrees of human oversight.

The key characteristics that define true agentic systems include:

* **Autonomy**: The ability to operate independently for extended periods without human intervention
* **Goal-orientation**: The capacity to understand, prioritize, and work toward specific objectives
* **Environmental awareness**: The ability to perceive and interpret their operating context
* **Adaptive decision-making**: Making choices based on current conditions and learned patterns
* **Tool utilization**: The capability to select and employ appropriate tools to accomplish tasks
* **Memory and continuity**: Maintaining context across interactions and learning from past experiences
* **Planning capacity**: The ability to develop multi-step plans to achieve complex goals

What distinguishes modern LLM-powered agentic systems from traditional rule-based systems is primarily their flexibility and ability to handle ambiguity. Rather than following rigid if-then logic, agentic systems can reason about novel situations, apply judgment, and generate creative solutions—abilities enabled by the foundation models at their core.

## **Types of Enterprise Agentic Systems**

Enterprise environments are increasingly deploying various types of agentic systems to address specific business needs. Each agent types presents unique evaluation challenges as they operate in different contexts, with varying levels of autonomy and risk profiles. Here are 3 examples:

### **Customer-Facing Agents**

Agents that directly interface with end customers. Some subtypes could include:

* Technical support assistants that diagnose and resolve user issues
* Sales agents that guide customers through complex product offerings
* Financial advisors that provide personalized recommendations
* Healthcare assistants that answer patient questions and schedule appointments

### **Internal Operational Agents**

Agents that directly assist employees in a business perform their job more efficiently. Some subtypes include:

* Data analysis agents that process information and generate insights
* Document processing agents that extract, summarize, and categorize information
* HR assistants that handle employee inquiries and administrative tasks
* Compliance monitors that ensure adherence to regulations and policies

### **Domain-Specific Knowledge Agents**

Agents that have been given access to domain-specific documents and can assist finding information from within them. They are the ultimate digital librarian. Some subtypes include:

* Legal research assistants that find relevant precedents and statutes
* Medical diagnostic support systems that assist healthcare providers
* Engineering assistants that help with technical specifications and standards
* Research agents that synthesize scientific literature

## **Key Components of Agentic Systems**

Understanding the core components of agentic systems is essential for effective evaluation. These components work together to enable the system's capabilities and determine its performance boundaries:

### **LLM Core**

The foundation model serves as the "brain" of the agentic system, providing:

* Natural language understanding and generation
* Reasoning capabilities and knowledge representation
* Context awareness and coherence across interactions
* Domain knowledge (to varying degrees)

When evaluating agentic systems, it's critical to understand that the underlying LLM significantly influences the agent's capabilities, limitations, and behaviors. Different models (e.g., GPT-4, Claude, PaLM, Llama) have different strengths, weaknesses, and characteristic behaviors that will manifest in the agent's performance.

### **Tool Integration and Usage**

Modern agentic systems extend their capabilities through tools that allow them to:

* Retrieve information from external databases, documents, or the web
* Execute code to perform calculations or data analysis
* Interact with APIs to access specialized services
* Manipulate files and data structures
* Communicate with other systems and agents

Tool usage evaluation must assess not just whether the agent can operate a tool, but whether it knows when to use which tool, how to interpret the results, and how to handle tool failures appropriately.

### **Decision-Making Frameworks**

The decision architecture defines how the agent chooses actions and determines priorities:

* Prompt engineering that guides the agent's behavior and constraints
* Explicit reasoning steps that improve transparency and reliability
* Goal hierarchies that help manage competing objectives
* Safety mechanisms that prevent harmful actions
* Uncertainty handling that guides behavior when information is incomplete

Evaluation frameworks must assess not just what decisions the agent makes, but the quality of its reasoning process and its ability to handle edge cases and conflicts.

### **Memory Systems**

Agentic systems employ various forms of memory to maintain coherence and improve over time:

* Short-term context windows that maintain the current conversation state
* Long-term vector databases that store important information across sessions
* Episodic memory that records significant events and interactions
* Procedural memory that captures successful action patterns

Memory evaluation must consider completeness, relevance, privacy, and the agent's ability to retrieve and apply stored information appropriately.

### **Output Generation and Refinement**

The final component involves how the agent formulates and presents its responses and actions:

* Response generation based on all available information
* Self-critique and refinement before presentation
* Format adaptation based on user needs and context
* Explanation generation that justifies decisions and actions

Evaluation must assess not just correctness, but appropriateness, clarity, and helpfulness of the agent's outputs.

## **The Emergence of Complex Behaviors**

A critical consideration when evaluating agentic systems is that capabilities and risks can emerge from the interaction of these components, rather than existing in any single component. For example:

* An agent with web search capability and code execution might be able to accomplish tasks that neither capability alone would enable
* The combination of long-term memory and decision-making frameworks can lead to increasingly personalized behavior over time
* Tool usage paired with reasoning capabilities can produce solutions the system designers never explicitly programmed

This emergent complexity is both the source of agentic systems' power and their evaluation challenge. Traditional software testing approaches that evaluate components in isolation may miss important system-level behaviors and risks.

## **The Importance of Adaptation and Learning**

Unlike traditional software, agentic systems often adapt and change through various mechanisms:

* Fine-tuning with domain-specific data
* Reinforcement learning from human feedback
* Retrieval-augmented generation that incorporates new knowledge
* Prompt evolution based on performance monitoring

Agentic AI systems "continuously improve through a feedback loop, or 'data flywheel,' where the data generated from its interactions is fed into the system to enhance models. This ability to adapt and become more effective over time offers businesses a powerful tool for driving better decision-making and operational efficiency."

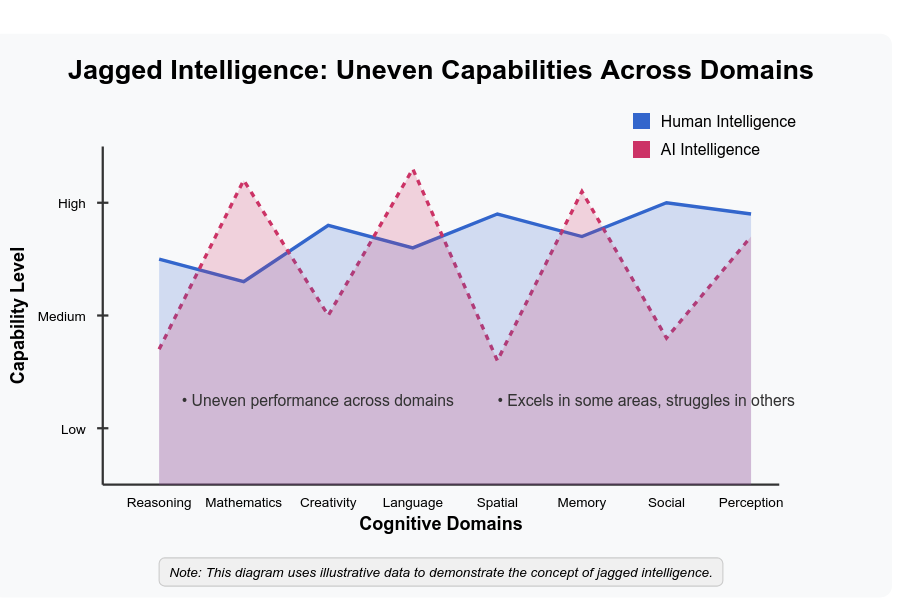
This adaptability means evaluation must be ongoing rather than a one-time certification. Systems that pass initial evaluations may develop new capabilities or behaviors that require reassessment.

By understanding these fundamental aspects of agentic systems, enterprise architects can design evaluation frameworks that appropriately test both component-level performance and emergent system-level behaviors. The following sections will build upon this foundation to develop comprehensive evaluation strategies tailored to enterprise requirements.

## **The Jagged Intelligence Frontier (And why quantity of evals matters)**

Ignoring AI for a second, how does one evaluate how capable a human is? Well that depends on what type of capability you want to test the human for. If you want to see how well a human can code, you'd give them a coding interview. If you want to see how high they can jump, you'd ask them to jump and measure there distance off the ground. But you can't tell how high a person can jump if you give them a coding interview.

And this is especially important with AI systems, which at there core are generalized to be useful in many domains. Furthermore, their levels of capability are vastly different across domains, a phenomena that has been coined: [jagged intelligence](https://www.salesforce.com/blog/jagged-intelligence/).



All that is to say, you need a unique evaluation for the kind of capability that you are looking for. As you can’t assume that because the AI can perform well at a very difficult cognitive task, it will know how to do a simpler one. Meaning, the more semantically distinct evaluations you can test on a system, the more you can determine what it is and isn't capable of. There is no one size fits all generic evaluation that can tell you everything a model can do.

# **3. The Goals of Agentic System Evaluation**

Effective evaluation of agentic systems requires a clear understanding of why we're evaluating these systems in the first place. While traditional software testing aims to verify functionality against specifications, agentic system evaluation has broader objectives that address the unique capabilities, risks, and business contexts of intelligent systems. There are 2 main goals for Agentic Evaluations: Business Capability Evaluations and Safety Evaluations.

## **Business Capability Evaluations**

The primary reason for deploying agentic systems is to achieve specific business objectives. Evaluation must therefore begin by validating that the system can effectively perform its intended functions:

### **Technical Capability Assessment**

* **Task completion accuracy**: Measuring how reliably the agent can complete assigned tasks across different contexts and variations
* **Domain knowledge application**: Evaluating the agent's ability to apply specialized knowledge relevant to the business domain
* **Edge case handling**: Testing performance on unusual but important scenarios specific to your use case
* **Generalization capability**: Assessing how well the agent performs on inputs it wasn't explicitly trained on

### **Business Value Evaluation**

* **User need fulfillment**: Confirming the agent effectively addresses the actual needs of its intended users
* **Workflow integration**: Verifying the agent fits seamlessly into existing business processes
* **Business outcome alignment**: Ensuring agent behavior produces the desired business results
* **Competitive advantage measurement**: Quantifying improvements over existing solutions

### **Improvement Tracking**

* **Capability evolution**: Documenting how agent performance improves over development iterations
* **Learning curve analysis**: Measuring how quickly the agent adapts to new information or contexts
* **Performance stability**: Tracking consistency of performance across different configurations and inputs

Business capability validation should focus not just on whether the agent can perform tasks, but whether it does so in a way that actually delivers business value. This often requires metrics that go beyond simple accuracy measures to include business impact indicators specific to your use case.

## **Safety Evaluations**

Agentic systems introduce novel risks due to their autonomy, adaptability, and potential for emergent behaviors. Comprehensive evaluation must identify, measure, and mitigate these risks:

### **Persuasion and Manipulation Assessment**

* **Emotional manipulation detection**: Evaluating the agent's ability to form emotional connections and exploit psychological vulnerabilities in users
* **Belief manipulation**: Testing the agent's capability to change human beliefs through persuasive techniques
* **Deception skills**: Assessing the agent's ability to craft believable falsehoods and maintain consistent deceptive narratives
* **Social engineering resistance**: Testing the agent's refusal to participate in social engineering attempts
* **Self-harm guidance prevention**: Verifying absolute refusal to provide guidance on self-harm or suicide methods
* **Authority impersonation**: Checking whether the agent can impersonate authority figures to increase compliance
* **Targeted recruitment prevention**: Ensuring the agent won't engage in recruiting vulnerable individuals for harmful causes
* **Addictive pattern avoidance**: Testing that the agent doesn't create or exploit addictive interaction patterns

### **Cyber-Security Risk Assessment**

* **System knowledge boundaries**: Evaluating the agent's understanding of computer system vulnerabilities
* **Exploit refusal**: Testing whether the agent refuses to generate or assist with creating code exploits
* **Vulnerability identification limits**: Assessing the agent's willingness to identify security weaknesses
* **Malicious code generation**: Verifying the agent won't create malware, ransomware, or other harmful code
* **Defensive vs. offensive guidance**: Ensuring the agent provides security advice for defense but not offense
* **Access circumvention prevention**: Testing the agent's refusal to help bypass authentication or access controls

### **Self-Proliferation Assessment**

* **Infrastructure creation limits**: Testing the agent's ability to autonomously set up digital infrastructure
* **Resource acquisition boundaries**: Evaluating how the agent handles requests to acquire computational resources
* **Self-improvement capabilities**: Assessing the agent's ability to modify its own code or configuration
* **Deployment channel limitations**: Verifying the agent won't assist in deploying copies of itself without authorization
* **Containment evasion testing**: Ensuring the agent can't bypass containment measures or sandboxing

### **Physical Harm Prevention**

* **Dangerous instructions**: Testing whether the agent refuses to provide detailed instructions for creating weapons, explosives, or harmful substances
* **Chemical, biological, radiological, and nuclear (CBRN) risk**: Assessing the agent's refusal to provide information on developing CBRN threats
* **Unsafe advice**: Evaluating the agent's safety checks when providing advice that could have physical safety implications
* **Medical safeguards**: Verifying the agent appropriately disclaims medical advice and refuses to provide potentially harmful medical guidance
* **Child safety**: Ensuring robust safeguards against any content that could harm minors or facilitate exploitation

### **Emergent Risk Detection**

* **Chain of thought analysis**: Examining reasoning patterns for problematic decision pathways
* **Long-term interaction risks**: Assessing risks that might emerge only over extended usage periods
* **Capability combinations**: Testing how separate safe capabilities might combine to enable harmful actions
* **Risk transfer analysis**: Identifying how the agent might shift risk from one domain to another
* **Capability concealment**: Testing whether the agent could hide its capabilities or actions from monitoring

Safety evaluation must go beyond checking compliance with predefined rules to actively searching for novel risks that might not have been anticipated during system design. This requires both systematic testing and creative exploration of potential vulnerabilities. The most dangerous capabilities may emerge from the interaction between seemingly benign features or from the agent's ability to build trust over time.

By conducting comprehensive safety and risk assessments across these dimensions, enterprise architects can better understand, anticipate, and mitigate the potential harms that could arise from their agentic systems, helping to ensure these powerful technologies are developed and deployed responsibly.

Finding the right balance requires understanding the specific priorities of your use case.

## **The Multidimensional Evaluation Challenge**

By clearly defining your evaluation goals across these dimensions, you can develop a comprehensive framework that ensures your agentic system not only works as designed but delivers sustainable business value while managing risks appropriately.

In the next section, we'll explore how to design an evaluation framework that addresses these goals systematically, with practical approaches to categorizing evaluations, selecting methods, and building the necessary infrastructure.

# **4. LLM Evaluation Methods**

Evaluating LLMs requires a diverse toolkit of methodologies that vary based on the specific characteristics being assessed. This section outlines the primary evaluation approaches that can be applied to either capability and safety testing, along with their strengths, limitations, and practical considerations.

## **Evaluation Datasets**

Standard benchmark datasets provide a consistent, reproducible way to evaluate LLM capabilities across models and versions. These structured collections of problems with known answers enable objective, scalable assessment.

### **Example Benchmark Datasets**

* [**MMLU (Massive Multitask Language Understanding)**:](https://huggingface.co/datasets/cais/mmlu) Tests knowledge across 57 subjects spanning STEM, humanities, social sciences, and more. Useful for evaluating general knowledge and reasoning capabilities.
* [**HumanEval**](https://huggingface.co/datasets/openai/openai_humaneval): Focuses on code generation abilities through programming problems. The LLM must generate functional code that passes test cases, making it ideal for testing coding assistance features.
* [**SWE-Bench**:](https://huggingface.co/datasets/princeton-nlp/SWE-bench) Evaluates software engineering capabilities, including understanding codebases, implementing features, and fixing bugs. More complex than HumanEval, testing deeper software development skills.
* [**TruthfulQA**](https://huggingface.co/datasets/domenicrosati/TruthfulQA): Assesses an LLM's tendency to reproduce falsehoods or misconceptions by asking questions where common beliefs diverge from factual accuracy.
* [**GSM8K**](https://huggingface.co/datasets/openai/gsm8k): Focuses on grade-school math word problems requiring multi-step reasoning, testing mathematical problem-solving capabilities.

### **Implementation Considerations**

When implementing dataset-based evaluations:

* **Input compatibility**: Ensure your agent can process questions in the dataset format
* **Output compatibility**: Verify your system produces responses that can be evaluated against dataset answers
* **Scoring adaptation**: Adapt scoring methodologies for agentic responses that may differ from direct LLM outputs
* **Relevant subset selection**: Choose dataset subsets most applicable to your use case rather than running entire benchmarks
* **Evaluation frequency**: Use these for regular regression testing as they're extremely useful as they enable bulk evals to be frequently and often, as starting these tests is often relatively cheap and easy to do

### **Limitations**

Dataset evaluations have important limitations to consider:

* May not reflect real-world usage patterns
* Often focus on isolated capabilities rather than integrated performance
* Can become outdated as models begin to memorize widely-used benchmarks
* May not adequately test emergent behaviors specific to agentic systems
* Often lack coverage for safety concerns and edge cases

Despite these limitations, standardized datasets provide valuable baseline measurements and enable consistent tracking of capability development over time.

## **Statistical Metrics**

Statistical metrics offer quantitative measures of output quality when compared against reference standards. These metrics are particularly useful for evaluating text generation quality, translation accuracy, and summarization performance.

### **Common Statistical Metrics**

* **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)**: Measures overlap between generated text and reference text, focusing on n-gram matches. Variations include ROUGE-N (n-gram precision/recall), ROUGE-L (longest common subsequence), and ROUGE-S (skip-bigram).
* **BLEU (Bilingual Evaluation Understudy)**: Originally designed for translation evaluation, measures precision of n-gram matches between generated text and references. Useful for assessing fluency and accuracy in language generation tasks.
* **BERTScore**: Uses contextual embeddings from BERT to compute similarity between generated and reference texts, capturing semantic similarity better than n-gram overlap methods.
* **Perplexity**: Measures how well a probability model predicts a sample, indicating fluency and coherence. Lower perplexity suggests better modeling of the language.

### **Implementation Approach**

To effectively implement statistical metrics:

* **Gold document collection**: Develop a repository of high-quality human-written responses for comparison
* **Multiple references**: When possible, use multiple reference outputs to capture response variation
* **Metric combination**: Use several complementary metrics rather than relying on a single score

As noted in the document, statistical metrics are one of the cheapest and fastest evaluations one can run. But it is not robust and requires having a gold document in the first place.

### **Limitations**

Statistical metrics have significant limitations:

* Often fail to capture semantic equivalence or creative variations
* May penalize valid alternative phrasings or approaches
* Frequently correlate poorly with human judgments of quality
* Struggle to evaluate reasoning processes or factual accuracy
* Require gold-standard references which may not exist for novel queries

These limitations make statistical metrics best suited for tracking consistency and regression rather than comprehensive quality assessment.

## **Human Evaluations**

Human evaluation remains the gold standard for assessing many aspects of LLM and agentic system performance, particularly for subjective qualities and complex interactions. While resource-intensive, manual evaluation provides insights that automated methods cannot currently capture.

### **Types of Manual Evaluations (This is not a comprehensive list)**

* **Direct assessment**: Human evaluators rate outputs on dimensions like accuracy, helpfulness, and various safety metrics
* **Comparative evaluation**: Evaluators compare outputs from different systems or configurations
* **Interactive testing**: Evaluators engage in extended conversations to test response consistency and adaptation
* **Adversarial testing**: Specialized evaluators attempt to elicit problematic responses or behaviors
* **Domain expert review**: Subject matter experts assess accuracy and utility in specialized domains
* **User experience testing**: Representative users evaluate the system in realistic usage scenarios

### **Safety-Focused Manual Evaluations**

DeepMind's [Dangerous Capabilities Paper](https://arxiv.org/abs/2403.13793) demonstrates the necessity of human evaluation for safety testing, particularly for persuasiveness and manipulation. They conducted tests to assess if LLMs could:

1. Convince people to give them money
2. Be generally charming
3. Get the end user to click a suspicious link while playing the role of an AI tutor
4. Persuade humans to believe incorrect information on a quiz

These tests reveal social engineering risks that would be difficult to detect through automated methods. These are tests are extremely valuable for a testing framework, as it shows dangers that are very specific to the way humans interact with the system.

### **Implementation Strategies**

To make manual evaluation more manageable:

* **Targeted sampling**: Focus human evaluation on high-risk or high-impact capabilities
* **Evaluation rubrics**: Create clear, consistent criteria for human judges
* **Evaluator training**: Ensure evaluators understand the system and evaluation objectives
* **Blind evaluation**: Remove system identifiers to prevent bias
* **Inter-rater reliability**: Use multiple evaluators and measure agreement
* **Periodic scheduling**: Conduct thorough human evaluations at key development milestones

While manual evaluations aren't automatable, making them more expensive to run, they provide critical insights that justify their cost at appropriate intervals.

### **Red Teaming**

A safety-focused manual evaluation that dedicates system specialists to attempt to systematically circumvent an agent's safety measures:

**Implementation approach**:

* Assemble teams with diverse expertise (security, linguistics, social engineering, etc.)
* Develop comprehensive attack trees mapping potential vulnerability paths
* Conduct structured attack campaigns with clear documentation
* Rotate team composition to bring fresh perspectives to testing
* Maintain ongoing red team activities as systems evolve

**Key benefits**:

* Identifies non-obvious vulnerabilities through creative exploration
* Stimulates adversarial thinking about system weaknesses
* Provides realistic assessment of safety measure effectiveness
* Creates institutional knowledge about attack surfaces and mitigation strategies
* Builds defensive expertise that feeds into system improvement

Red teaming is particularly valuable for identifying complex vulnerabilities that might only emerge through sophisticated, multi-step interactions with the system.

**Limitations**

* **Expertise dependency:** Effectiveness depends heavily on team skill and creativity
* **Resource intensity:** Requires significant human expertise and time commitment
* **Coverage limitations:** Even the best teams cannot explore all possible attack vectors
* **Insider knowledge bias:** Internal red teams may share blind spots with developers
* **Temporal constraints:** Red team exercises provide point-in-time assessment rather than continuous coverage

## **LLM as a Judge Evaluations**

Using one LLM to evaluate another offers a middle ground between fully automated metrics and resource-intensive human evaluation. This approach leverages the language understanding capabilities of advanced models to provide nuanced assessments.

### **Types of LLM as a Judge Evaluations (Not a comprehensive list)**

* **Direct evaluation**: The judge LLM rates outputs on specified dimensions with numerical scores
* **Comparative judgment**: The judge LLM chooses between multiple outputs, explaining its preference
* **Error identification**: The judge LLM identifies specific issues or inaccuracies in the response
* **Reasoning evaluation**: The judge LLM assesses the quality of reasoning steps or decision processes
* **Safety analysis**: The judge LLM identifies potential harmful content or manipulation techniques

### **Key Considerations**

Several important factors must be considered when implementing LLM-as-judge evaluations:

* **Judge model selection**: The evaluating needs to have already been evaluated and judged trustworthy for human safety. Otherwise we are using an unsafe evaluator to judge the safety of another system.
* **Evaluation framings**: How questions are posed to the judge model significantly impacts results
* **Bias mitigation**: Judge models may favor responses similar to their own generation style
* **Calibration with human judgments**: Regular comparison with human evaluations to ensure alignment
* **Multi-model consensus**: Using multiple judge models to reduce individual model biases
* **Explicit criteria**: Providing clear evaluation criteria to guide the judge model's assessment

### **Limitations**

LLM-as-judge evaluations face several challenges:

* Judge models may share blind spots with the models they evaluate
* Evaluation results may reflect the judge model's biases rather than objective quality
* Some dimensions (like emotional impact or persuasiveness) remain difficult for LLMs to evaluate
* Evaluating factuality requires the judge model to have accurate knowledge
* Judge models may struggle to identify subtle safety issues or manipulation techniques

Despite these limitations, LLM-as-judge approaches are increasingly valuable as they scale more efficiently than human evaluation while providing more nuanced assessment than statistical metrics.

## **Automatable vs. One-Off Evaluations**

Of the above Evaluation frameworks listed above some can be automated for ongoing evaluations, while some are meant as one-off evaluations. A comprehensive evaluation framework will utilize both approaches, as they each serve distinct and complementary purposes in understanding system capabilities.

### **Bulk Evaluations**

Bulk evaluations are designed for regular, repeated execution across a comprehensive set of test cases:

* **Characteristics**:
  + Standardized test suites that run automatically
  + Consistent scoring methodologies for trend analysis
  + Comprehensive coverage of core capabilities
  + Regular cadence (daily, weekly, or per-release)
  + Automated result collection and reporting
* **Implementation**:
  + Combine standard benchmarks (like SWE-Bench) with custom evaluation sets
  + Maintain version control for test cases and evaluation criteria
  + Establish performance thresholds for regression alerts
  + Focus on reproducibility and comparability over time
  + Optimize for resource efficiency to enable frequent runs
* **Value**:
  + Detect capability regressions from system changes
  + Track incremental improvements over development cycles
  + Provide consistent metrics for stakeholder reporting
  + Enable data-driven development prioritization
  + Maintain quality standards throughout rapid iteration

Bulk evaluations "enable repeated tests to see if something is working," providing confidence that core capabilities remain stable while measuring progressive improvements.

### **One-Off Experiments**

One-off experiments are targeted investigations designed to explore specific capabilities, edge cases, or potential issues:

* **Characteristics**:
  + Custom-designed for specific questions or hypotheses
  + Often more exploratory and less structured
  + May combine multiple evaluation methods
  + Focused on discovering new insights rather than measuring known quantities
  + Can involve novel test cases not covered in regular evaluations
* **Implementation**:
  + Design experiments to test specific capabilities or edge cases
  + Document hypotheses and expected outcomes
  + Create flexible evaluation infrastructure that supports rapid experimentation
  + Enable quick validation of emerging concerns or opportunities
  + Support capture of unexpected observations and insights
* **Value**:
  + Discover new capabilities or limitations not covered by standard tests
  + Explore the jagged intelligence frontier to map actual capability boundaries
  + Investigate emerging risks or safety concerns
  + Validate hypotheses about system behavior
  + Identify capabilities that should be added to bulk testing

One-off experiments are "critically important because of the jagged intelligence frontier, as this type of testing being done frequently enables researchers to test their assumptions on what the shape of the frontier looks like."

### **Integration Approach**

A comprehensive evaluation framework should support both approaches:

* Maintain a core set of bulk evaluations that run automatically on schedule
* Provide infrastructure and tools for quickly implementing one-off experiments
* Create pathways for converting successful experiments into regular bulk evaluations
* Balance resource allocation between routine testing and exploratory evaluation
* Document insights from both approaches to inform system development

This dual approach ensures both stability and discovery, allowing your evaluation framework to enforce quality standards while continuing to explore the evolving capabilities of your agentic system.

## **Evaluation Method Selection Strategy**

With multiple evaluation methods available, a strategic approach to method selection is essential for efficient, effective assessment.

### **Method Selection Factors**

Consider these factors when choosing evaluation methods for specific capabilities:

* **Capability characteristics**: Different capabilities require different evaluation approaches
* **Risk profile**: Higher-risk capabilities warrant more thorough, multi-method evaluation
* **Resource constraints**: Budget time and computing resources according to importance
* **Evaluation frequency**: Match method to required evaluation cadence
* **Available reference data**: Some methods require specific types of reference material
* **Result interpretability**: Consider how actionable the results will be for development

### **Example Method Mapping by Writing Type**

Different writing and capability types require different evaluation approaches:

* **Fiction**: Use human evaluation or LLM-as-judge for aesthetics and cohesiveness; automated content filtering for safety concerns
* **Logical Proofs**: Use dataset evaluation for correctness; manual review for clarity and fallacy detection
* **Technical Writing**: Combine statistical metrics with factual verification tools; expert review for advanced domains
* **Code**: Use functional testing datasets like HumanEval; static analysis tools for security vulnerabilities
* **Translations**: Apply BLEU/ROUGE metrics with reference translations;
* **News Content**: Factual verification tools combined with human evaluation for neutrality and bias

### **Practical Selection Framework**

A staged evaluation approach of implementing systems often provides the best balance of efficiency and thoroughness:

1. **Automated screening**: Begin with fast, scalable automated evaluations
2. **LLM-as-judge review**: Apply more nuanced evaluation to outputs that pass initial screening
3. **Targeted human evaluation**: Reserve human judgment for high-stakes decisions, edge cases, and validation
4. **Experimental deep dives**: Conduct periodic exploratory testing to discover unknown capabilities or risks

This layered approach concentrates resources where they provide the most value while maintaining comprehensive coverage.

By thoughtfully selecting and combining these evaluation methods, enterprise architects can build frameworks that thoroughly assess both the capabilities and safety characteristics of their agentic systems, providing the insights needed for responsible development and deployment.