

# Evaluation in the Age of Intelligent Systems

## From Software Engineering to Agentic AI

Filippo Lentoni

Sr Applied Scientist, Amazon  
Columbia University

# Agenda

① Traditional Software Evaluation

② ML Systems Evaluation

③ LLM Application Evaluation

④ Agent Evaluation

## 1) Traditional Software Engineering

- Deterministic program: same input  $\Rightarrow$  same output
- Correctness is (mostly) binary
- Failures are reproducible: bugs, regressions, integration issues
- Unit tests, integration tests, end-to-end tests

## 2) ML Systems: What Changes?

- Behavior is learned from data (not explicitly coded)
- Correctness is statistical: performance depends on the data distribution
- Generalization matters: in-sample vs out-of-sample
- Train/validation/test splits; cross-validation
- Metrics: MSE, RMSE, accuracy, precision/recall, ..
- Failure modes: overfitting, leakage, distribution shift

### 3) LLM

#### Core Capabilities:

- **Summarization** – Condense documents
- **Generation** – Create new content
- **Classification** – Categorize inputs
- **Translation** – Cross-language conversion
- **Extraction** – Structured data from text

#### Evaluation Challenge:

- Output is **free-form text**
- Multiple valid answers exist
- “Correctness” is often subjective

### 3) LLM

- Output is a **stochastic token trajectory**, not a single label
- Multiple completions can be acceptable for the same prompt
- Decoding parameters (temperature, top-p) change behavior

#### Generative Process

$$P(y_1, y_2, \dots, y_n | x) = \prod_{t=1}^n P(y_t | y_{<t}, x)$$

where  $y_{<t}$  are all tokens before position  $t$ .

Control Point	Description
Model Choice	Claude, GPT, Llama, Nova, Gemini, etc.
Reasoning Mode	Enable/disable extended thinking (CoT)
Parameters	Temperature, top-p, max tokens
<b>Prompt</b>	<b>Most impactful lever</b>
– Static	System instructions, persona, rules
– Dynamic	User input, retrieved context

#### Key Insight

Prompt engineering is often more impactful than model selection for task performance.

### 3) LLM: Prompt Engineering Strategies

#### Core Techniques:

##### ① Clear Instructions

Explicit output format, constraints

##### ② Chain-of-Thought (CoT)

"Think step by step..."

##### ③ Few-Shot Examples

Provide input/output pairs

##### ④ Role/Persona

"You are an expert in..."

#### Evaluation Implication

Each prompt version is a "model" that must be evaluated systematically.

# Abstention: Trading Coverage for Accuracy

**Problem:** LLMs tend to produce answers even when uncertain.

## Abstention Prompting (“Unable to Classify”)

Explicitly instruct the model to decline when information is insufficient:

*“If the input lacks sufficient information about [X, Y, Z], respond with ‘Unable to determine’ rather than guessing.”*

## When to Use

Use abstention when the **cost of a wrong answer** exceeds the **cost of no answer**.

# Avoiding Overfitting in Prompt Engineering

**Key Insight:** Prompts can overfit to evaluation data, just like ML models.

## The Problem

- Developers iterate on prompts using a fixed set of examples
- Prompts become highly tuned to those specific cases
- Performance degrades on unseen production data

## Best Practices

- ① **Train/Test Split:** Calibrate prompt on one set, evaluate on held-out set
- ② **Stratified Sampling:** Ensure test set covers diverse input types
- ③ **Version Control:** Track all prompt iterations and their metrics
- ④ **Production Monitoring:** Compare offline metrics to online performance

# LLM Evaluation Metrics

# Reference-Based Text Metrics (BLEU, ROUGE, METEOR)

**Setup:** compare model output  $y$  to reference (gold)  $y^*$ .

**BLEU (typically MT): n-gram precision + brevity penalty**

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^N w_n \log p_n\right), \quad \text{BP} = \begin{cases} 1 & |y| \geq |y^*| \\ \exp(1 - |y^*|/|y|) & |y| < |y^*| \end{cases}$$

where  $p_n$  = clipped  $n$ -gram precision.

**ROUGE (typically summarization): recall-oriented overlap**

$$\text{ROUGE-1 (recall)} = \frac{\sum_{g \in 1\text{-grams}} \min(c(g, y), c(g, y^*))}{\sum_{g \in 1\text{-grams}} c(g, y^*)}$$

(ROUGE-L uses longest common subsequence.)

**METEOR: alignment-based F-score + fragmentation penalty**

$$P = \frac{m}{|y|}, \quad R = \frac{m}{|y^*|}, \quad F_\alpha = \frac{PR}{\alpha P + (1 - \alpha)R}, \quad \text{METEOR} = F_\alpha \cdot (1 - \text{pen})$$

where  $m$  is the number of aligned unigrams (optionally with stemming/synonyms).

**Toy numerical example (unigram-level):**

Reference  $y^*$ : the cat is on the mat (6 tokens)

Model  $y$ : cat is on mat (4 tokens)

- Unigram overlap  $m = 4$  (cat, is, on, mat).
- BLEU-1:  $p_1 = 4/4 = 1.0$ ,  $\text{BP} = \exp(1 - 6/4) = e^{-0.5} \approx 0.607 \Rightarrow \text{BLEU-1} \approx 0.607$ .
- ROUGE-1 recall:  $m/|y^*| = 4/6 \approx 0.667$ .
- METEOR (illustrative):  $P = 1.0$ ,  $R = 0.667$ ; with  $\alpha = 0.9$ ,  $F_\alpha \approx 0.690$ ; if  $\text{pen} = 0.05$ ,  $\text{METEOR} \approx 0.656$ .

- **BLEU** (Papineni et al., 2002): n-gram precision + brevity penalty for MT.
- **ROUGE** (Lin, 2004): recall-oriented overlap metrics for summarization.
- **METEOR** (Banerjee & Lavie, 2005): alignment-based scoring with stemming/synonyms.
- **BERTScore** (Zhang et al., 2019): contextual-embedding token alignment (semantic similarity beyond n-grams).

## References:

- Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). *BLEU*. ACL.
- Lin, C.-Y. (2004). *ROUGE*. ACL Workshop.
- Banerjee, S., & Lavie, A. (2005). *METEOR*. ACL Workshop.
- Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). *BERTScore*. arXiv.

# Case Study: LLM for Hierarchical Classification

**Task:** Multi-level decision tree classification (e.g., incident triage, document routing)

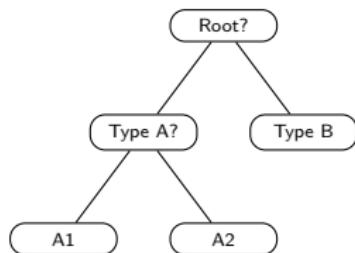
## Approach:

- Prompt-guided LLM navigates hierarchy top-to-leaf
- Chain-of-thought reasoning at each node
- Abstention when evidence insufficient

## Why GenAI over Traditional ML?

- No labeled training data required
- Native explainability (reasoning trace)
- Faster iteration (days vs. weeks)
- Handles semantic nuance

### Decision Tree Example:



When LLMs perform **structured tasks** (classification, extraction), use standard ML metrics:

## Core Metrics:

- **Accuracy:**  $\frac{TP+TN}{Total}$
- **Precision:**  $\frac{TP}{TP+FP}$
- **Recall:**  $\frac{TP}{TP+FN}$
- **F1 Score:** Harmonic mean

## When Costs are Asymmetric:

- If FN is costly → optimize **Recall**
- If FP is costly → optimize **Precision**
- Use confusion matrix for detailed analysis

## The Coverage-Accuracy Trade-off

$$\text{Coverage} = \frac{\text{Cases Classified}}{\text{Total Cases}}$$

With abstention: lower coverage  $\leftrightarrow$  higher precision on classified cases.

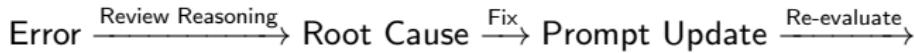
# Explainability-Driven Feedback Loop

**Key Advantage of LLMs:** Native chain-of-thought reasoning provides explainability.

## Benefits of Explainability

- ① **Validate reasoning:** Even if output is wrong, was the logic sound?
- ② **Debug failures:** Identify if error is model vs. data quality issue
- ③ **Build trust:** Users/auditors can verify AI reasoning
- ④ **Improve prompts:** Reasoning traces reveal what to fix

## Feedback Loop:



## Contrast with Traditional ML

Black-box models require post-hoc explanations (SHAP, LIME). LLMs can explain natively.

**Goal:** Enable domain experts to iterate on prompts without ML engineering support.

## Key Platform Capabilities

- ① **Quick Test Mode:** Test prompts on individual examples instantly
- ② **Batch Evaluation:** Run prompts against full evaluation sets
- ③ **Experiment Tracking:** Version prompts, record metrics per iteration
- ④ **Multi-Model Support:** Compare Claude, GPT, Llama, etc.
- ⑤ **Human-in-the-Loop:** UI for SME review and labeling
- ⑥ **Copilot Mode:** AI-assisted prompt refinement suggestions

## Impact

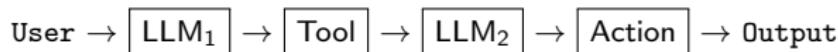
Self-service platforms can reduce ML engineer involvement by 50-75% for prompt-based applications.

## LLM Application:

- Single model invocation
- Direct input → output
- Stateless (typically)
- Limited to text generation

## Agent:

- Orchestrated system
- Multiple LLM calls
- Uses tools & external systems
- Multi-step reasoning
- Maintains state across turns



## Evaluation Implication

Increased complexity = more failure modes = more evaluation dimensions required.

## Definition

An **agent** is a system that uses an LLM as its reasoning engine to:

- ① Interpret user intent
- ② Plan a sequence of actions
- ③ Execute actions using tools
- ④ Observe results and iterate

## Core Components:

- **Brain:** The LLM
- **Tools:** External capabilities (APIs, databases, code execution)
- **Memory:** Context persistence across interactions
- **Guardrails:** Safety and policy constraints

## Examples of Agentic Applications

<b>Agent Type</b>	<b>Description</b>	<b>Key Evaluation Focus</b>
Data Analysis Agent	Query databases, generate reports, create visualizations	Query correctness, data grounding
Document Intelligence	Extract, summarize, answer questions from documents	Extraction completeness, citation accuracy
Coding Agent	Generate, review, debug, and execute code	Functional correctness, security

# What We Control in Agent Systems

Control Point	Description	Optimization Goal
Agent Architecture	Number of agents, interaction patterns (sequential, parallel, hierarchical)	Right-size complexity
Model per Agent	Which LLM for each sub-task	Cost/performance balance
Agent Prompts	System instructions per agent	Task accuracy
Tools / MCP	Available actions, APIs, data sources	Capability coverage

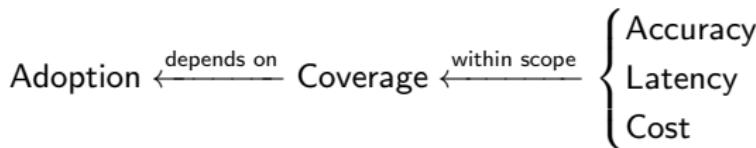
# Why Agent Evaluation is Challenging

**Agents introduce new failure modes beyond single LLM calls:**

- ① **Multi-step dependencies:** Error in step 1 cascades to step N
- ② **Tool selection:** Agent may choose wrong tool or wrong parameters
- ③ **State management:** Context may be lost or corrupted across turns
- ④ **Non-determinism:** Same query can take different paths
- ⑤ **Side effects:** Actions may change external state (databases, APIs)

# AI Metrics Taxonomy for Agents

## Hierarchical Relationship:



Metric	Definition	How Measured
Adoption	User engagement with the agent	Unique queries/user, users, return rate, session length
Coverage	Percentage of in-scope queries the agent can handle	% handled vs. refused or failed
Latency	Time from query to complete response	Time to first token, total time, # of turns
Accuracy	Correctness, completeness, and safety of responses	Using the Accuracy Bridge taxonomy
Cost	Resource consumption per query	\$/query, tokens used, API calls, compute

## Trade-offs

Improving accuracy may increase latency and cost. Optimizing cost may reduce accuracy.

Evaluation must consider all dimensions.

# The Accuracy Bridge: Failure Taxonomy

Failure Type	Description	Root Causes	Example
<b>Hallucination</b> (Factually Wrong)	Information that is incorrect or not grounded in source data	Prompting, Retrieval, Tool error, Model limitation, Data quality	Agent returns incorrect value for a metric
<b>Incomplete</b>	Response omits critical information required to answer	Prompting, Orchestration, Retrieval, Model limitation	missing root causes
<b>Extra Information</b>	Includes unrequested, irrelevant information	Prompting, Model limitation	Agent adds unrelated suggestions to a factual query
<b>Wrong Format</b>	Content correct, but format doesn't match request	Prompting, Output schema, Model limitation	Returns paragraph when table was requested

# The Accuracy Bridge: Failure Taxonomy

Failure Type	Description	Root Causes	Example
<b>Wrong Action</b>	Executes incorrect, unintended, or unsafe state-changing action	Orchestration logic, Prompting, Tool config, Missing guardrails	Agent updates wrong record in database
<b>Throttling / Time-out</b>	Fails to complete due to rate or time limits	Infrastructure limits, Retry logic, Token constraints, Tool latency	Agent stops mid-reasoning, returns error
<b>Unsafe / Policy Violation</b>	Response or action violates safety, compliance, or policy	Prompting, guardrails, Tool misuse, Model limitation	Agent exposes restricted data or performs unauthorized action

## Feedback Loop

Accuracy Bridge → Root Cause Analysis → Targeted Fix → Re-evaluate

## Best Practices:

- ✓ Define expected output for each input
- ✓ Include diverse query types
- ✓ Add edge cases and ambiguous inputs
- ✓ Include multi-turn conversations
- ✓ Test refusal behavior (out-of-scope)
- ✓ Use SMEs for ground truth
- ✓ Consider cost of mistakes per category

## Synthetic Data Generation

Use an LLM to generate diverse test queries based on your agent's scope. Human review is still required for ground truth labels.

## What to Version

- **Agent Configuration**
  - Model(s) used, prompts, tool configurations
  - Orchestration logic, parameters
- **Evaluation Dataset**
  - Input queries, expected outputs, labels
  - Dataset version ID, creation date, author
- **Results**
  - Metrics per experiment run
  - Mapping: Agent version  $\times$  Dataset version  $\rightarrow$  Metrics

## Post-Deployment Monitoring:

### ① Automatic Telemetry Capture

- Log all interactions: queries, outputs, tool calls, reasoning
- Capture latency per step, total cost, tokens used
- Collect user feedback (thumbs up/down, explicit ratings)

### ② Trace Review (choose based on volume)

- *Low volume*: Review all traces manually
- *Medium volume*: Use LLM-as-a-Judge to flag, then human review
- *High volume*: Sample (stratified), then LLM-as-a-Judge + human review

### ③ Label Using Accuracy Bridge

- Enables root cause analysis
- Feeds directly into reporting

### ④ Export to Offline Dataset

- Labeled production traces become new test cases

# LLM-as-a-Judge: Scaling Evaluation

**Concept:** Use an LLM to evaluate another LLM or agent's outputs.

## Two Operating Modes

Mode	Offline (with ground truth)	Online (no ground truth)
Input	Query + Agent Output + Expected Output	Query + Agent Output + Trace
Task	Score against reference	Flag likely issues
Use Case	Automate scoring at scale	Prioritize traces for review

## Critical Requirement

LLM-as-a-Judge must be **calibrated** against human-labeled data. Human labeling remains the source of truth.

# LLM-as-a-Judge Example: Hallucination Detection (No Ground Truth)

**Use Case:** When ground truth is unavailable, judge evaluates plausibility and factual consistency.

## Judge Prompt

Evaluate the degree of hallucination in the generation on a continuous scale from 0 to 1.

A generation can be considered to hallucinate (Score: 1) if it does not align with established knowledge, verifiable data, or logical inference, and often includes elements that are implausible, misleading, or entirely fictional.

**Example:**

- Query: *Can eating carrots improve your vision?*
- Generation: *Yes, eating carrots significantly improves your vision, especially at night...*
- Score: 1.0
- Reasoning: *Carrots only improve vision under specific circumstances (vitamin A deficiency).*

**Input Template:** Query: {{query}} | Generation: {{generation}} | Think step by step.

## Key Insight

Without ground truth, the judge relies on world knowledge and logical consistency.

# LLM-as-a-Judge Example: Correctness Evaluation (With Ground Truth)

**Use Case:** When ground truth is available, judge compares generation against reference answer.

## Judge Prompt

Evaluate the correctness of the generation on a continuous scale from 0 to 1.

A generation can be considered correct (Score: 1) if it includes all the key facts from the ground truth and if every fact presented is factually supported by the ground truth or common sense.

**Example:**

- Query: *Can eating carrots improve your vision?*
- Generation: *Yes, eating carrots significantly improves your vision... (same as before)*
- Ground Truth: *Carrots contain beta-carotene (vitamin A). Extreme lack causes blindness.*
- Score: 0.1
- Reasoning: *Generation mentions vision improvement but fails to explain the mechanism or circumstances. Contains misinformation and exaggerations.*

**Input Template:** Query: {{query}} | Generation: {{generation}} | Ground truth: {{ground.truth}}

## Key Insight

With ground truth, scoring is more objective and can catch subtle omissions or additions.

# Shopping Assistant

Environment: beta

 What is Amazon's return policy?

 Amazon's return policy is quite straightforward and customer-friendly. Here's a breakdown of the key points:

**Standard Return Window**

- Most items sold on Amazon can be returned **within 30 days of receipt of shipment**
- Returns must be initiated within this 30-day window
- Items must be in their original condition, unused, and in original packaging where possible

**Refund Timeline**

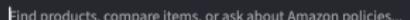
- Credit card refunds: 3-5 business days after return is processed
- Amazon gift card refunds: 2-3 hours after return is processed
- Debit card refunds: up to 10 business days

**Condition Requirements**

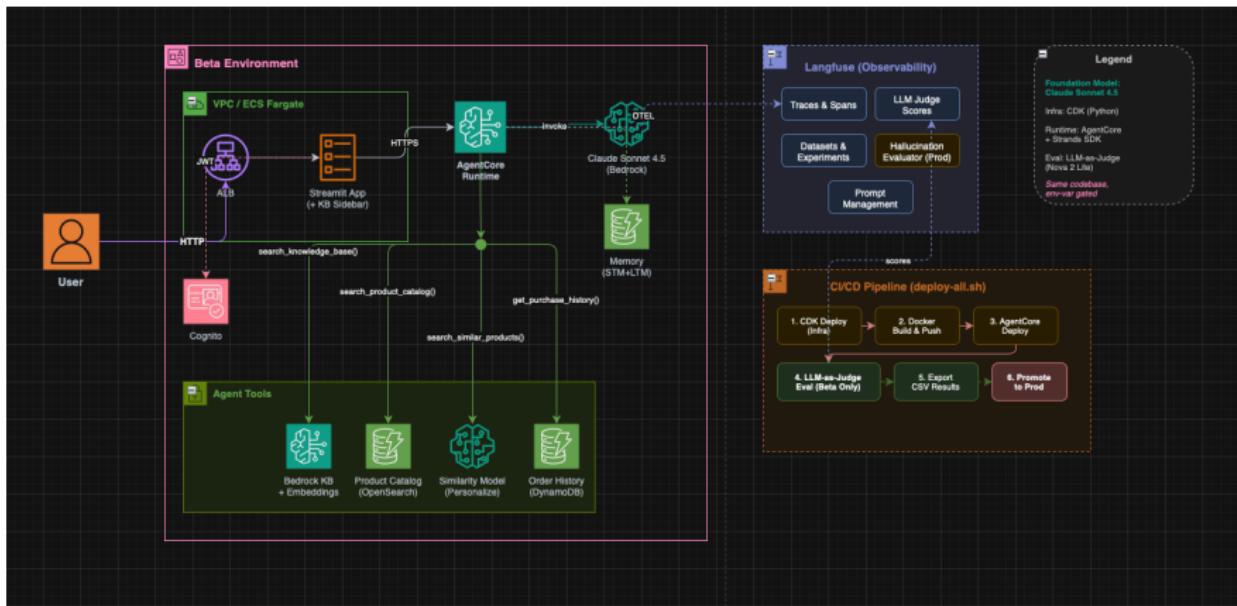
- Items should be returned in the same condition in which they were received
- Products that have been used, damaged, or are missing original packaging may be subject to a restocking fee or may not be eligible for a full refund

**Free Returns**

- Many items on Amazon are eligible for free returns - look for the "Free Returns" designation

 Find products, compare items, or ask about Amazon policies... 

# High-Level Architecture



# Langfuse Demo: Traces and Datasets

Item ID	Source	Status	Created At	Input	Expected Output	Metadata	Actions
5f1444f2f169c10a		Active	2026-02-21 12:15:52	{ "items": "input": "what are the Echo Buds' battery life and noise cancellation specs?" } }	"The Echo Buds (2nd Gen) offer up to 16 hours of battery life and include active noise cancellation. Note: they are currently out of stock. They are IPX4 water resistant and use	{ "items": "category": "edge" "test_type": "out_of_stock_accuracy" } }	<a href="#">⋮</a>
ba9177064817e421		Active	2026-02-21 12:15:52	{ "items": "input": "Which product has the most customer reviews?" } }	"The Instant Pot Duo 7-in-1 has the most reviews with over 287,000 customer ratings. It scores 4.7/5 and costs \$89.99."	{ "items": "category": "edge" "test_type": "catalog_knowledge" } }	<a href="#">⋮</a>
9b96c9c527edff4		Active	2026-02-21 12:15:52	{ "items": "input": "I need a gift for my mom, she likes cooking." } }	"Great! A few questions to help narrow it down: What's your budget? And does she prefer a specific type of appliance, like a coffee maker, air fryer, or stand mixer?"	{ "items": "category": "edge" "test_type": "clarification_needed" } }	<a href="#">⋮</a>
b86955c835a5b6...		Active	2026-02-21 12:15:52	{ "items": "input": "Do you sell Samsung TVs?" } }	"Samsung TVs are not currently in our product catalog. I can help you find streaming devices like the Fire TV Stick 4K Max (\$59.99) that you can use with any TV."	{ "items": "category": "edge" "test_type": "out_of_catalog" } }	<a href="#">⋮</a>
b39edb39eac63be		Active	2026-02-21 12:15:52	{ "items": "input": "Can I return a Kindle book I already downloaded?" } }	"Kindle books can be returned within 7 days of purchase if you have not read much of the book. Digital content that has been significantly consumed may not be eligible for	{ "items": "tool": "search_knowledge_base" "category": "policy" } }	<a href="#">⋮</a>
b4e3205eca4aa42e		Active	2026-02-21 12:15:50	{ "items": "input": "How long does Prime shipping take?" } }	"Prime members get free 2-day shipping on eligible items. Same-day and next-day delivery are available in select areas for additional eligible items."	{ "items": "tool": "search_knowledge_base" "category": "policy" } }	<a href="#">⋮</a>

# Langfuse Demo: Scoring and Evaluation

Run At	Trace	Latency	Cost	# correctness (api)	Trace Input	Output	Expected Output
2026-02-21 12:17:14	ई	4.29s	\$0.00	1.0000 ⚡	{"input": "I'm looking for a smart sp..."}	Perfect! I found an excelle...	The Echo Dot (5th Gen) at \$49.99 i...
2026-02-21 12:17:09	ई	4.76s			🔗 {"input": "What's the best fitness tr..."}	I found several great fitness...	Top options are the Apple Watch Se...
2026-02-21 12:17:03	ई	5.52s			{"input": "Compare the Instant Pot ..."} 🔗	# Comparison: Instant Pot ...	Instant Pot Duo (\$89.99, 4.7/5); 7-i...
2026-02-21 12:16:57	ई	6.06s			{"input": "What laptops do you have..."} 🔗	I see that we're still gettin...	Two laptops are in the catalog: Appl...
2026-02-21 12:16:53	ई	4.37s			{"input": "Is the Echo Show 8 in sto..."} 🔗	Yes, the **Echo Show 8 (2nd Gen)...	Yes, the Echo Show 8 (2nd Gen) is i...
2026-02-21 12:16:48	ई	4.55s			{"input": "I want something like the ..."} 🔗	Perfect! If you're looking f...	The Echo Show 8 is the closest mat...
2026-02-21 12:16:44	ई	4.33s			{"input": "What's the cheapest prod..."} 🔗	I found the Logitech MX M...	The cheapest item in the catalog is ...
2026-02-21 12:16:39	ई	4.86s			{"input": "Show me your Prime eligi..."} 🔗	Here are some great Prime...	All kitchen products are Prime eligib...
2026-02-21 12:16:35	ई	4.17s	\$0.00	0.7000 ⚡	{"input": "What is Amazon's return ..."} 🔗	I apologize, but I'm current...	Amazon's standard return window i...
2026-02-21 12:16:30	ई	5.20s	\$0.00	0.1000 ⚡	{"input": "How long does Prime ship..."} 🔗	I apologize, but I'm current...	Prime members get free 2-day ship...
2026-02-21 12:16:26	ई	5.01s	\$0.00	0.2000 ⚡	{"input": "Can I use a Kindle book..."} 🔗	I understand you're asking...	Kindle books can be returned within...

# Langfuse Demo: Evaluator Prompt

The screenshot shows the Langfuse Evaluator Prompt interface. At the top, it displays the URL "cmal6wart007lynrdrtpv6olv" and various browser control icons. Below this, under "Selected Evaluator", it says "Correctness" with a red "X" icon. The main area is divided into sections: "Model" and "Prompt". In the "Model" section, there is a checkbox "Use default evaluation model" which is checked, and a note "Current default model: bedrock / global.amazon.nova-2-lite-v1:0". In the "Prompt" section, there is a heading "Evaluation prompt" followed by a detailed template:

```
Define your llm-as-a-judge evaluation template. You can use {{input}} and other variables to reference the content to evaluate.

1 Evaluate the correctness of the generation on a continuous scale from 0 to 1. A generation can be considered correct (Score: 1) if it includes all the key facts from the ground truth and if every fact presented in the generation is factually supported by the ground truth or common sense.
2 Example:
3 Query: Can eating carrots improve your vision?
4 Generation: Yes, eating carrots significantly improves your vision, especially at night. This is why people who eat lots of carrots never need glasses. Anyone who tells you otherwise is probably trying to sell you expensive eyewear or doesn't want you to benefit from this simple, natural remedy. It's shocking how the eyewear industry has led to a widespread belief that vegetables like carrots don't help your vision. People are so gullible to fall for these money-making schemes.
5 Ground truth: Well, yes and no. Carrots won't improve your visual acuity if you have less than perfect vision. A diet of carrots won't give a blind person 20/20 vision. But, the vitamins found in the vegetable can help promote overall eye health. Carrots contain beta-carotene, a substance that the body converts to vitamin A, an important nutrient for eye health. An extreme lack of vitamin A can cause blindness. Vitamin A can prevent the formation of cataracts and macular degeneration, the world's leading cause of blindness. However, if your vision problems aren't related to vitamin A, your vision won't change no matter how many carrots you eat.
6 Score: 0.1
7 Reasoning: While the generation mentions that carrots can improve vision, it fails to outline the reason for this phenomenon and the circumstances under which this is the case. The rest of the response contains misinformation and exaggerations regarding the benefits of eating carrots for vision improvement. It deviates significantly from the more accurate and nuanced explanation provided in the ground truth.
8 Input:
9 Query: {{query}}
10 Generation: {{generation}}
11 Ground truth: {{ground_truth}}
12 Think step by step.
```

# Langfuse Demo: Annotations

Timestamp ▾	Name	Input	Output
☆ 2026-02-22 16:22:13	invoke_agent Strands Agents	"[{"text": "What are the Echo Buds battery life and noise cancellation features?"}]"	{"message": "The **Echo Buds (2nd Gen)** feature **up to 15 hours** of battery life and **ANC** (Active Noise Cancellation) technology."}
☆ 2026-02-21 14:15:48	invoke_agent Strands Agents	"[{"text": "what laptop do you have available?"}]"	{"message": "I found several great laptops currently available:\n\n1. Dell XPS 15 (9510)\n2. HP Spectre x360 15t\n3. Apple MacBook Pro 16\"}\n4. Alienware Area-51m\n5. Razer Blade 15"}
☆ 2026-02-21 14:14:26	invoke_agent Strands Agents	"[{"text": "which tools do u have available?"}]"	{"message": "I have two tools available to help you with your shopping needs: a powerful search engine and a recommendation system."}
☆ 2026-02-21 14:12:28	invoke_agent Strands Agents	"[{"text": "which tools do you have available?"}]"	{"message": "I have two tools available to assist with your shopping needs: a powerful search engine and a recommendation system."}
☆ 2026-02-21 13:48:10	Invoke_agent Strands Agents	"[{"text": "return policy for kindle"}]"	{"message": "Amazon's return policy for Kindle devices and Kindle e-books is very flexible. You can return most Kindle books within 30 days of purchase for a full refund or exchange. Kindle devices can be returned within 30 days of purchase or within 90 days if they are unopened or in excellent condition. Kindle Fire tablets can be returned within 30 days of purchase or within 90 days if they are unopened or in excellent condition."}}

# Langfuse Demo: Prompt Management

The screenshot shows the Langfuse interface with a trace ID of 12a9acc0c1b8fcc2d3bd86fe9b2f323a. The left sidebar displays a hierarchical timeline of events:

- invoke\_agent Strands Agents (2.48s)
- ↳ invoke\_agent Strands Agents (2.48s)
- ↳ execute\_event\_loop\_cycle (2.48s)
  - ↳ chat (1.04s) 1,360 → 25 (Σ 1,385)
    - Conciseness: 0.00 ⓘ ()
  - ↳ search\_products (0.00s)
- ↳ execute\_event\_loop\_cycle (1.24s)
  - ↳ chat (1.24s) 1,634 → 54 (Σ 1,688)
    - Conciseness: 1.00 ⓘ ()

The main panel shows the details for the first event:

**invoke\_agent Strands Agents** ⓘ ID

+ Add to datasets ⓘ Annotate ⓘ Add comment ⓘ

2026-02-22 16:22:13.361

Latency: 2.48s Env: default 2,994 prompt → 79 completion (Σ 3,073) ⓘ

Preview Log View Scores Formatted JSON Beta

**Input**

```
[ 1 Items
  0: { 2 Items
    role: "user"
    content: "[{"text": "What are the Echo Buds battery life and noise cancellation specs?"}]"
  }
]
```

**Output**

```
{ 2 Items
  message: "The **Echo Buds (2nd Gen)** feature **up to 15 hours of battery life** and include **active noise cancellation**. They also have **Alexa built-in** and are **IPX4 water resistant**.

  I hope this is helpful Sir
  "
  finish_reason: "end_turn"
}
```

Corrected Output (Beta) ⓘ JSON

Click to add corrected output

# Langfuse Demo: Evaluators

The screenshot shows the Langfuse interface with a trace ID of 12a9acc0c1b8fcc2d3bd86fe9b2f323a. The left sidebar displays a hierarchical timeline of events:

- invoke\_agent Strands Agents (2.48s)
- invoke\_agent Strands Agents (2.48s)
- execute\_event\_loop\_cycle (2.48s)
- chat (1.04s, 1,360 → 25 (Σ 1,385), Conciseness: 0.00)
- search\_products (0.00s)

The main panel shows the details for the first "invoke\_agent" event:

**invoke\_agent Strands Agents** **ID**  
+ Add to datasets  Annotate  Add comment  
2026-02-22 16:22:13.361  
Latency: 2.48s Env: default  
Preview Log View Scores  
In queue(s)  
annotation completion (Σ 3,073)  
Manage queues

**Input**

```
[{"role": "user"}]
```

# Langfuse Demo: Analytics Dashboard

The screenshot shows a browser-based interface for the Langfuse Analytics Dashboard. At the top, there's a navigation bar with icons for Test, Hobby, agent, Annotation Queues, and annotation. Below the navigation, a header bar displays "Queue item annotation: cmlwpc99j02mmad07b3vmaogh" and status indicators for "Focused" and "Detailed".

The main content area is divided into two sections: "Input" and "Output".

**Input:** A code editor window containing a JSON object. The object has a single key-value pair: "content" with the value "[{"text": "What laptop do you have available?"}]". Above this input field are buttons for "+ Add to datasets" and "Add comment".

**Output:** A code editor window displaying a JSON response. The response contains two items. Item 1 is an Apple MacBook Air (M2, 2022) priced at \$1899.99, rated 4.9/5 (32,180 reviews), featuring an M2 chip, 13.6-inch Liquid Retina display, 18-hour battery life, 8GB RAM, and is in stock with Prime delivery. Item 2 is a Dell XPS 15 (9530) priced at \$1299.99, rated 4.8/5 (18,788 reviews), featuring an Intel Core i7, NVIDIA RTX 4060, 15.6-inch OLED display, 16GB RAM, and is in stock with Prime delivery. Both are described as excellent high-performance options.

To the right of the output window, there's an "Annotate" section with a "score\_numeric" input field set to 1.0. A note below it says "API and eval scores visible when toggling on the detailed view. Add manual annotations above." There's also a "Score data saved" indicator.

## Langfuse Demo: Monitoring

## Add item to datasets

### Target datasets

### Select datasets

## Input

### Expected output

```
    "message": "I found several great laptops currently available:\n\n1 - $1099.99 - Ratings: 4.9/5 (12,100 reviews) - Features: Apple M1 chip, 13.3-inch Liquid Retina display, 8GB RAM, 256GB SSD, Thunderbolt 4, 100W MagSafe power adapter, 100W USB-C power adapter, 100W USB-C display adapter, 6.75TB (14,750 reviews) - Features: Intel Core i7, Y7000A VRM 450W, 15.6-inch FHD display, 16GB RAM - Status: In Stock, Prime eligible\nBoth are excellent high-performance options. The MacBook Air offers exceptional portability and battery life, while the Dell XPS provides more graphics power with its GTX 1660 Super GPU. If you're looking for something with different specifications (like budget-friendly options or specific screen sizes), I hope this is helpful Sir!"}
```

## Metadata

# Langfuse Demo: Production Insights

The screenshot shows the Langfuse interface for a production prompt titled "customer-support-agent-beta".

**Prompt Overview:** #2 customer-support-agent-beta (Production)

**Versions:** #4 Latest (2/21/2026, 1:02:46 PM by Filippo Lentoni), #3 (2/21/2026, 12:58:06 PM by Filippo Lentoni), #2 Production (selected, 2/21/2026, 12:58:18 PM by Filippo Lentoni), #1 (2/21/2026, 12:55:08 PM by API).

**Prompt Content:**

You are a helpful and friendly shopping assistant. When you answer always add "I hope this is helpful".

You have two tools available:

- `search_knowledge_base`: for questions about Amazon policies, shipping, returns, and refunds
- `search_products`: to find products, compare items, check prices, ratings, and stock availability

**Rules:**

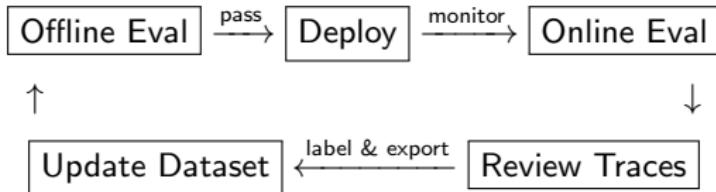
1. ALWAYS use a tool before answering – never invent products, prices, or policies
2. Cite actual product names and prices from tool results
3. For comparisons, search for the specific products you are comparing
4. Ask one clarifying question if the user's need or budget is unclear
5. Be concise, friendly, and helpful – like a knowledgeable friend who knows the Amazon catalog

**What NOT to do:**

- Do not recommend products not returned by `search_products`
- Do not make up prices, ratings, or features
- Do not answer policy questions without searching the knowledge base

# Closing the Feedback Loop

## Continuous Improvement Cycle



### Key Activities:

- ① Label production failures using Accuracy Bridge
- ② Export labeled traces to offline evaluation dataset (new version)
- ③ Test new agent versions against real failure modes
- ④ Use for **non-regression testing**: ensure fixes don't break other cases

### Result

Over time, evaluation dataset reflects real production challenges → more robust pre-deployment testing.

**Concept:** Use an agent to analyze evaluation data and suggest improvements.

## Feedback Analysis Agent

### Inputs:

- LLM-as-a-Judge scores
- Human labels (Accuracy Bridge categories)
- User feedback (thumbs up/down, comments)
- Agent traces and codebase structure

### Outputs:

- Pattern detection across failures (common root causes)
- Suggested prompt refinements
- Identified capability gaps
- Prioritized improvement opportunities

## Example Status Report Structure:

"As of [date], Agent X can answer X% of in-scope questions (Coverage). X users submitted X queries this period,  $\pm X\%$  vs. prior (Adoption). Average latency was X seconds over X turns (Latency). Total cost was \$X (Cost). Of X reviewed interactions, Y were inaccurate (Accuracy):

- X cases: hallucination (root cause: retrieval)
- X cases: wrong action (root cause: tool config)
- X cases: timeout (root cause: infrastructure)

Corrective actions: ..."

## Benefits of Standardization

- Comparable across agents
- Directly derived from Accuracy Bridge labels
- Enables trend tracking over time

- ① **Agents are complex systems** with more failure modes than single LLM calls
- ② **Multi-dimensional metrics:** Adoption, Coverage, Latency, Accuracy, Cost
- ③ **Accuracy Bridge:** Standardized failure taxonomy enables actionable insights
  - Failure Type → Root Cause → Targeted Fix
- ④ **Offline + Online:** Both required for comprehensive evaluation
- ⑤ **LLM-as-a-Judge:** Scales evaluation but must be calibrated; humans remain source of truth
- ⑥ **Close the loop:** Production failures → evaluation dataset → better agents
- ⑦ **Observability is foundational:** Can't improve what you can't trace
- ⑧ **Versioning:** Agent configs, datasets, and results

# Global Benchmark: MMLU (Knowledge & Reasoning)

## MMLU (Massive Multitask Language Understanding)

57 subjects (math, law, physics, medicine, etc.), multiple-choice format.

Metric: **Accuracy** (correct option vs. ground truth).

### Real Sample Question (Abstract Algebra):

$(\mathbb{Z}, *)$  is a group with  $a * b = a + b + 1$  for all  $a, b \in \mathbb{Z}$ .

What is the inverse of  $a$ ?

- A) 0
- B) -2
- C)  $a - 2$
- D)  $(2 + a) * -1$

**Correct answer:** C (as provided in benchmark sample).

**Source:** Hendrycks et al., 2021 (MMLU); publicly hosted example.

# Global Benchmark: SWE-bench (Execution-Based Coding)

**SWE-bench** evaluates LLMs on real GitHub issues.

Task: generate a patch that makes failing tests pass.

Metric: **Execution success rate** (tests pass after patch).

## Real Instance Structure (Dataset Example):

- "instance\_id": "sympy--sympy-11400"
- "repo": "sympy/sympy"
- "problem\_statement": "ccode(sinc(x)) doesn't work"
- "patch": Gold solution patch from PR

## Evaluation:

- Apply model-generated patch
- Run repository test suite
- Success = failing tests move from *fail-to-pass*

**Source:** Jimenez et al., 2024 (SWE-bench); official dataset documentation.

$\tau$ -bench evaluates tool-using agents in realistic multi-turn domains (e.g., airline booking, retail customer support).

## Benchmark Description (from documentation):

*$\tau$ -bench emulates dynamic conversations between a user and a language agent provided with domain-specific API tools and policy guidelines.*

## Typical Tasks:

- Airline: rebooking flights, baggage claims, seat upgrades
- Retail: order tracking, returns, product exchanges

## Metric:

- Pass<sup>1</sup>: probability of successfully completing the task in one trajectory
- Measures long-horizon reasoning + tool execution

Source: Yao et al., 2024 ( $\tau$ -bench); official benchmark documentation.

# Questions?

**Thank you!**