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

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

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
Prompt	**Most impactful lever**
– 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) $y^\star$.

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

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

where p_n = clipped n-gram precision.

ROUGE (typically summarization): recall-oriented overlap

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

(ROUGE-L uses longest common subsequence.)

METEOR: alignment-based F-score + fragmentation penalty

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

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

Toy numerical example (unigram-level):
Reference $y^\star$: `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: $p_1 = 4/4 = 1.0$, $\text{BP} = \exp(1 - 6/4) = e^{-0.5} \approx 0.607 \Rightarrow$ BLEU-1 ≈ 0.607.
- ROUGE-1 recall: $m/|y^\star| = 4/6 \approx 0.667$.
- METEOR (illustrative): $P = 1.0$, $R = 0.667$; with $\alpha = 0.9$, $F_\alpha \approx 0.690$; if pen $= 0.05$,

Key Literature: Automatic Evaluation of Text Generation

- **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)

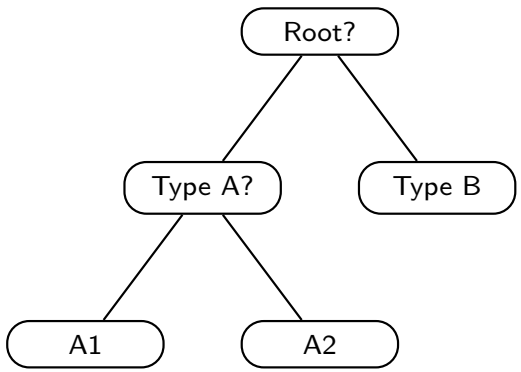
Approach:

- 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:



Classification Metrics for LLM Applications

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 $\rightarrow$ optimize **Recall**
- If FP is costly $\rightarrow$ 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:

$$\text{Error} \xrightarrow{\text{Review Reasoning}} \text{Root Cause} \xrightarrow{\text{Fix}} \text{Prompt Update} \xrightarrow{\text{Re-evaluate}}$$

Contrast with Traditional ML

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

Self-Service Experimentation Platforms

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.

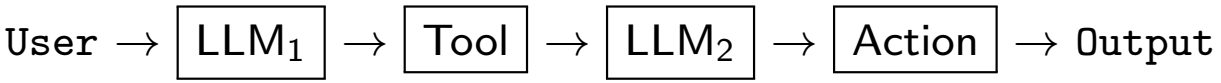
From LLMs to Agents

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.

Anatomy of an LLM Agent

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

Agent Type	Description	Key Evaluation Focus
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:

$$\text{Adoption} \xleftarrow{\text{depends on}} \text{Coverage} \xleftarrow{\text{within scope}} \begin{cases} \text{Accuracy} \\ \text{Latency} \\ \text{Cost} \end{cases}$$

Metric	Definition	How Measured
Adoption	User engagement with the agent	Unique users, queries/user, 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
Hallucination (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
Incomplete	Response omits critical information required to answer	Prompting, Orchestration, Retrieval, Model limitation	missing root causes
Extra Information	Includes unrequested, irrelevant information	Prompting, Model limitation	Agent adds unrelated suggestions to a factual query
Wrong Format	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
Wrong Action	Executes incorrect, unintended, or unsafe state-changing action	Orchestration logic, Prompting, Tool config, Missing guardrails	Agent updates wrong record in database
Throttling / Timeout	Fails to complete due to rate or time limits	Infrastructure limits, Retry logic, Token constraints, Tool latency	Agent stops mid-reasoning, returns error
Unsafe / Policy Violation	Response or action violates safety, compliance, or policy	Prompting, Missing guardrails, Tool misuse, Model limitation	Agent exposes restricted data or performs unauthorized action

Feedback Loop

Accuracy Bridge $\rightarrow$ Root Cause Analysis $\rightarrow$ Targeted Fix $\rightarrow$ Re-evaluate

Building a High-Quality Evaluation Dataset

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.

Versioning: Experiments and Datasets

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

Observability and Online Evaluation Workflow

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:

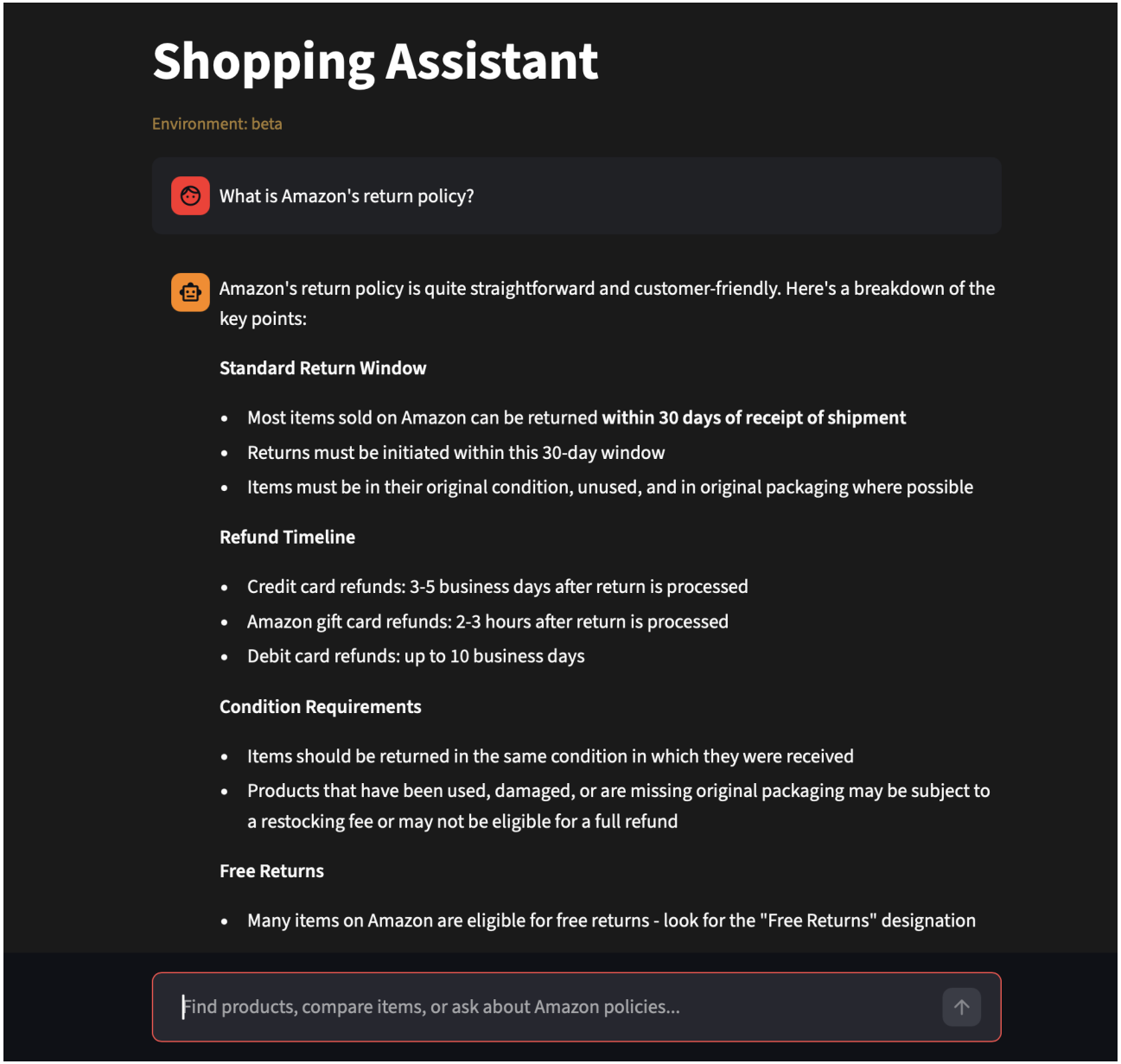
- Query: *Can eating carrots improve your vision?*
- Generation: *Yes, eating carrots significantly improves your vision...* (same as before)
- Ground Truth: *Carrots contain beta-carotene (vitamin A). Extreme lack causes blindness.*
- Score: 0.1
- Reasoning: *Generation mentions vision improvement but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

Input Template: `Query: {{query}} | Generation: {{generation}} | Ground truth: {{ground_truth}}`

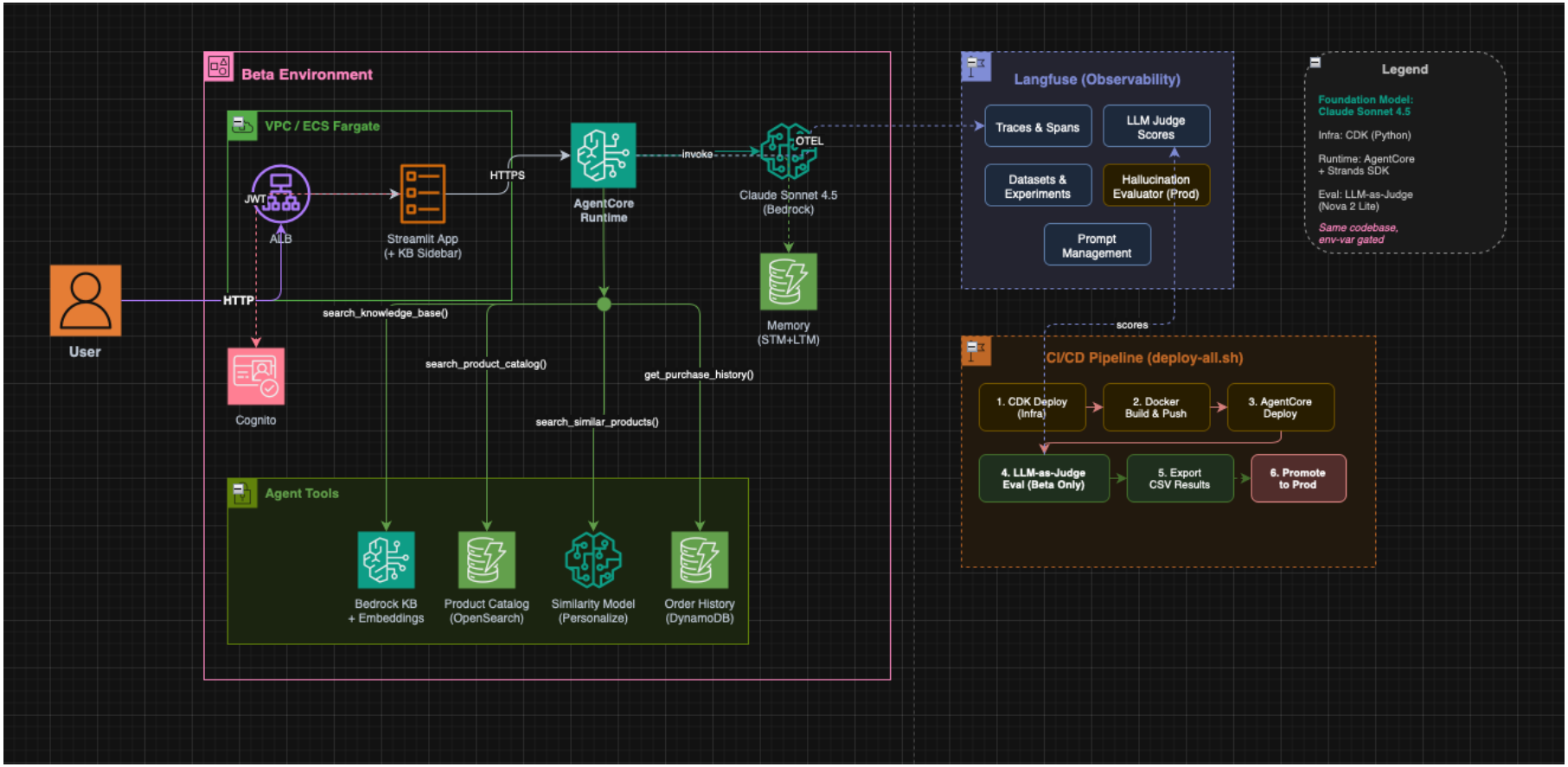
Key Insight

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

Shopping Assistant



High-Level Architecture



Langfuse Demo: Traces and Datasets

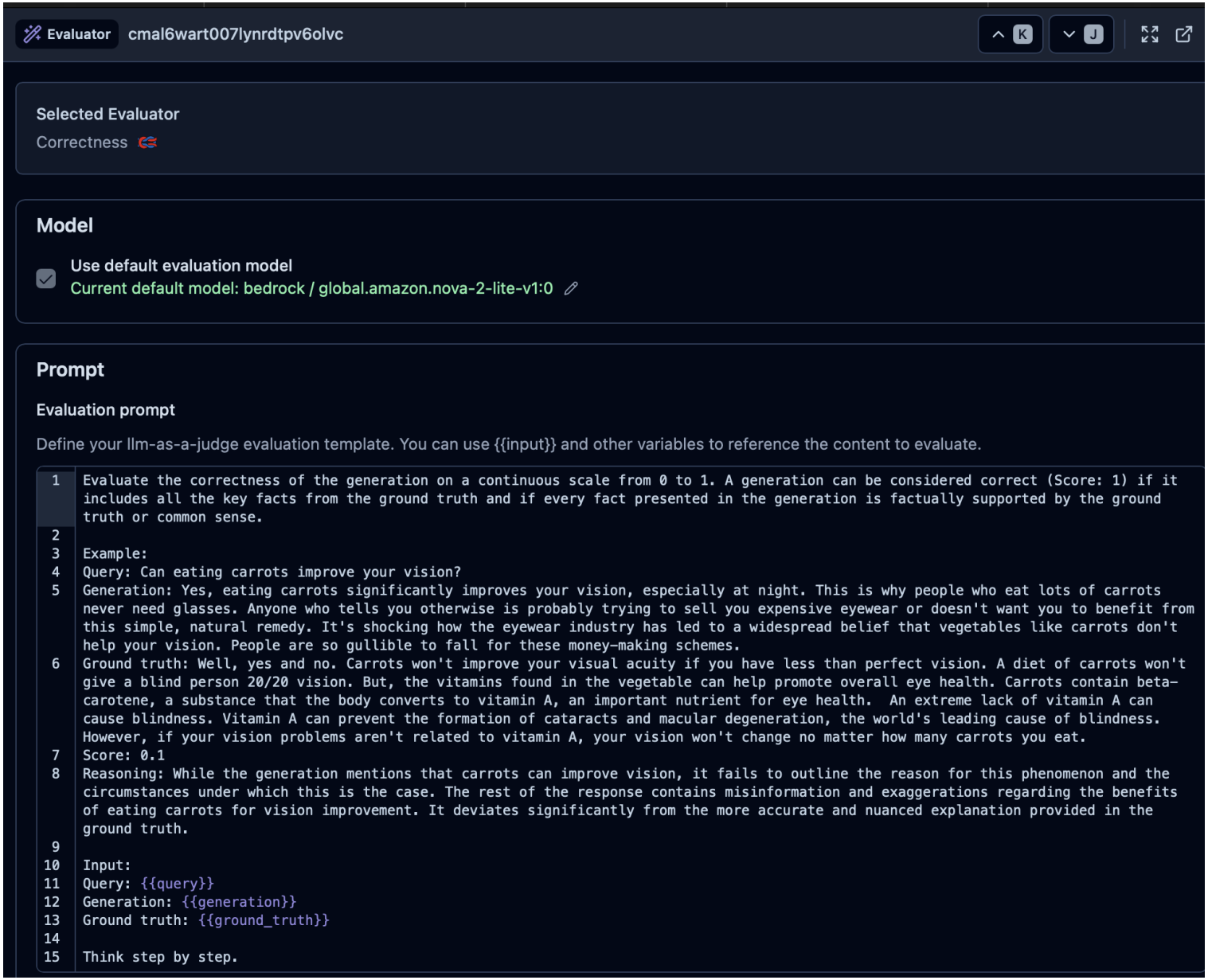
Item id	Source	Status	Created At	Input	Expected Output	Metadata	Actions
55f444f2f169c10a		Active	2026-02-21 12:15:52	{ 1 Items input: "What are the Echo Buds' battery life and noise cancellation specs?" }	"The Echo Buds (2nd Gen) offer up to 15 hours of battery life and include active noise cancellation. Note: they are currently out of stock. They are IPX4 water resistant and use	{ 2 Items category: "edge" test_type: "out_of_stock_accuracy" }	⋮
9a9177864817a421		Active	2026-02-21 12:15:52	{ 1 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."	{ 2 Items category: "edge" test_type: "catalog_knowledge" }	⋮
9b96c9c527edffd4		Active	2026-02-21 12:15:52	{ 1 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?"	{ 2 Items category: "edge" test_type: "clarification_needed" }	⋮
386955c835a5b6...		Active	2026-02-21 12:15:52	{ 1 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."	{ 2 Items category: "edge" test_type: "out_of_catalog" }	⋮
539edbb39eac63be		Active	2026-02-21 12:15:52	{ 1 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	{ 2 Items tool: "search_knowledge_base" category: "policy" }	⋮
e4e3205eca4aa42e		Active	2026-02-21 12:15:50	{ 1 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."	{ 2 Items tool: "search_knowledge_base" category: "policy" }	⋮

Langfuse Demo: Scoring and Evaluation

Run At	Trace	Latency	Cost	# correctness (api)	Trace Input	Output	Expected Output
2026-02-21 12:17:14		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		4.76s			{"input": "What's the best fitness tr...	I found several great fitnes...	Top options are the Apple Watch Se...
2026-02-21 12:17:03		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		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		4.37s			{"input": "Is the Echo Show 8 in sto...	Yes, the **Echo Show 8 (2...	Yes, the Echo Show 8 (2nd Gen) is i...
2026-02-21 12:16:48		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		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		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		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		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...

The assistant's response fully matches the expected answer. It correctly identifies the Echo Dot (5th Gen) at $49.99, includes the accurate rating (4.7/5 with 89,432 reviews), confirms Prime eligibility and in-stock status, and lists all key features (Alexa, improved audio, temperature sensor, Bluetooth). The response is concise, helpful, and contains no hallucinations or inaccuracies.

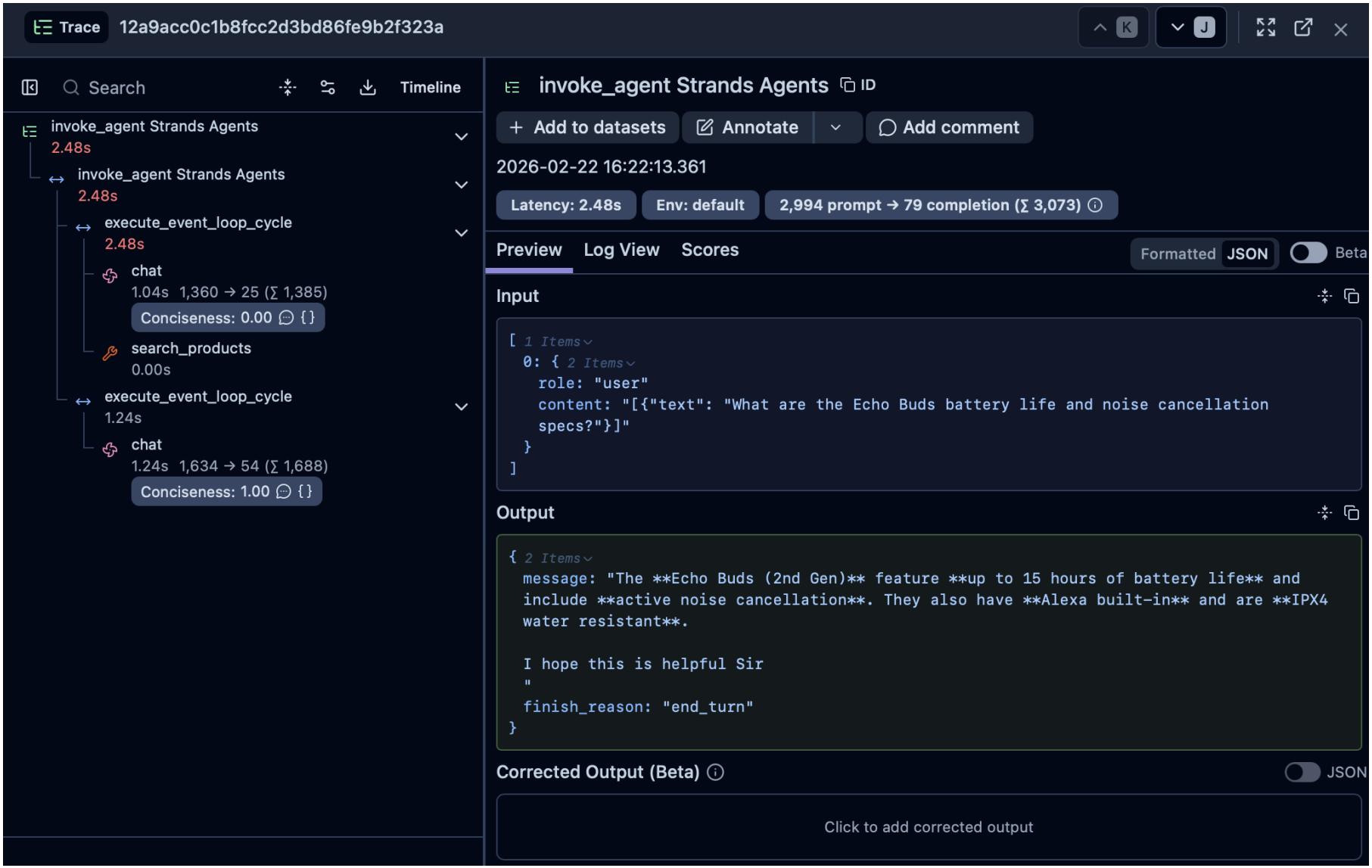
Langfuse Demo: Evaluator Prompt



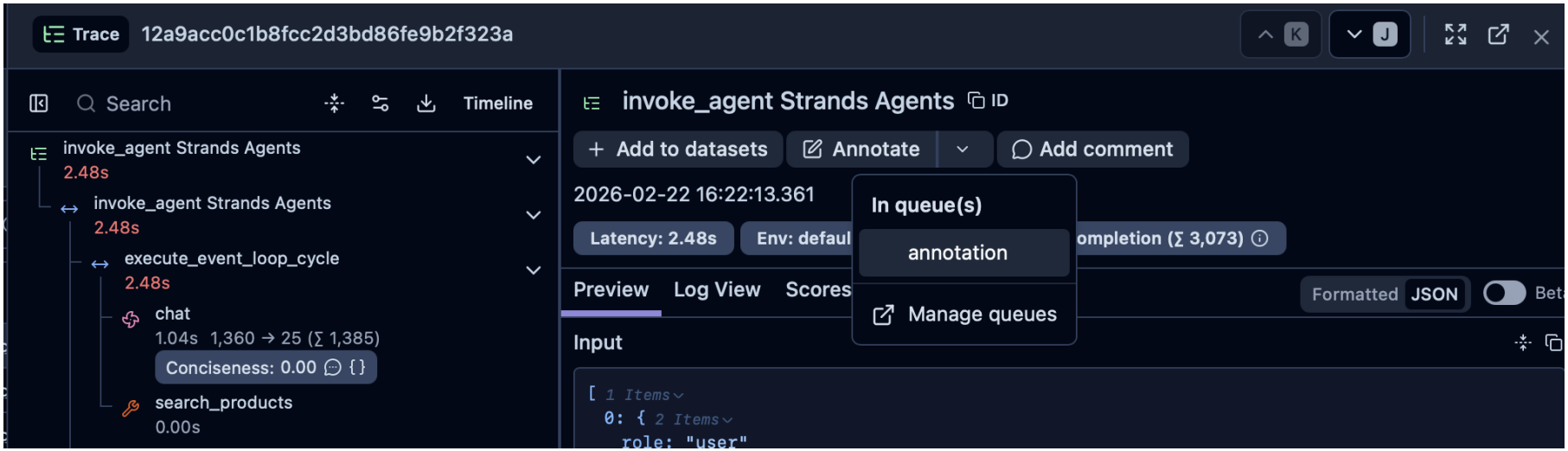
Langfuse Demo: Annotations

	Timestamp ▼	Name	Input	Output
☆	2026-02-22 16:22:13	invoke_agent Strands Agents	"[{\"text\": \"What are the Echo Buds battery life and noise cancellati...	{"message":"The **Echo Buds (2nd Gen)** feature **up to 15 h
☆	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
☆	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 sho
☆	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 shopp
☆	2026-02-21 13:48:10	invoke_agent Strands Agents	"[{\"text\": \"return policy for kindle\"}]"	{"message":"Amazon's return policy for Kindle devices and Kind

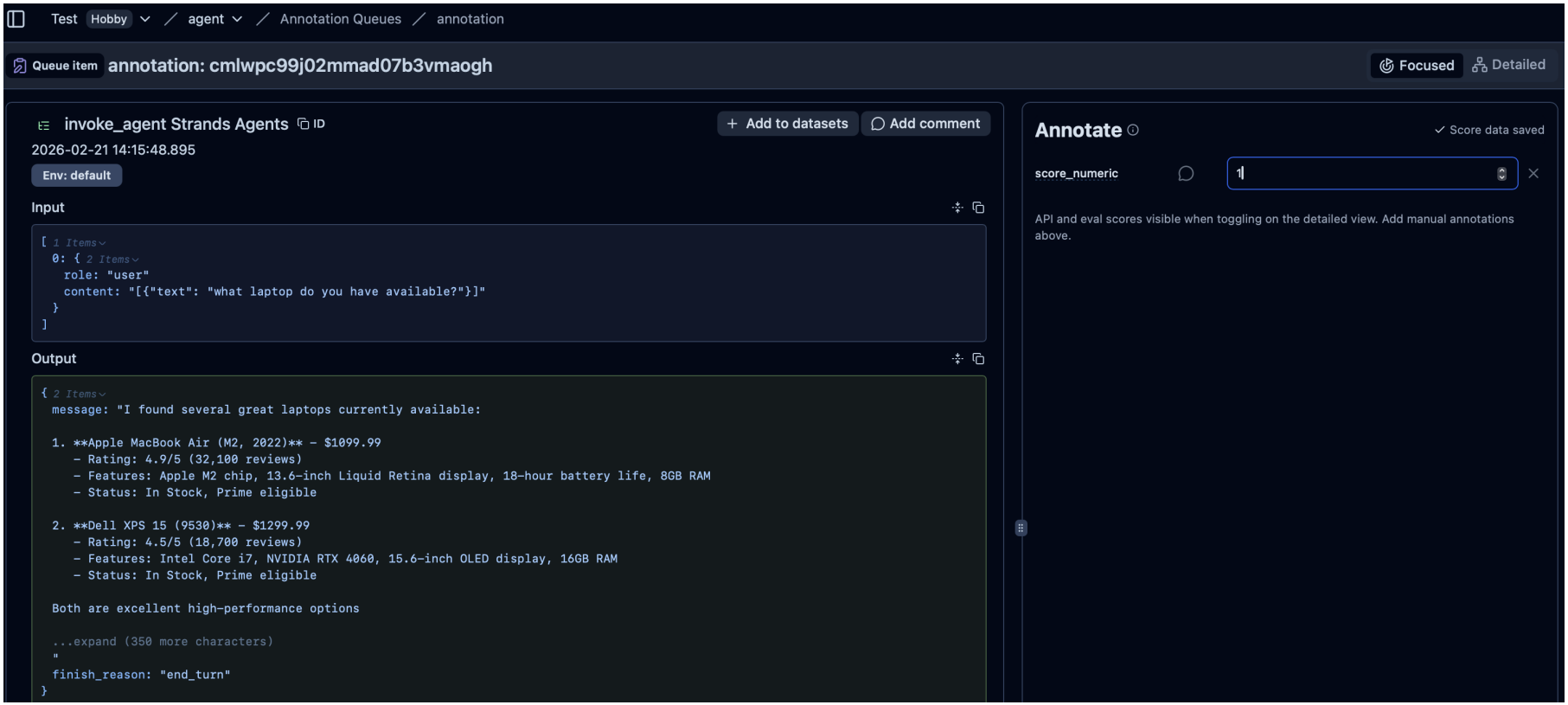
Langfuse Demo: Prompt Management



Langfuse Demo: Evaluators

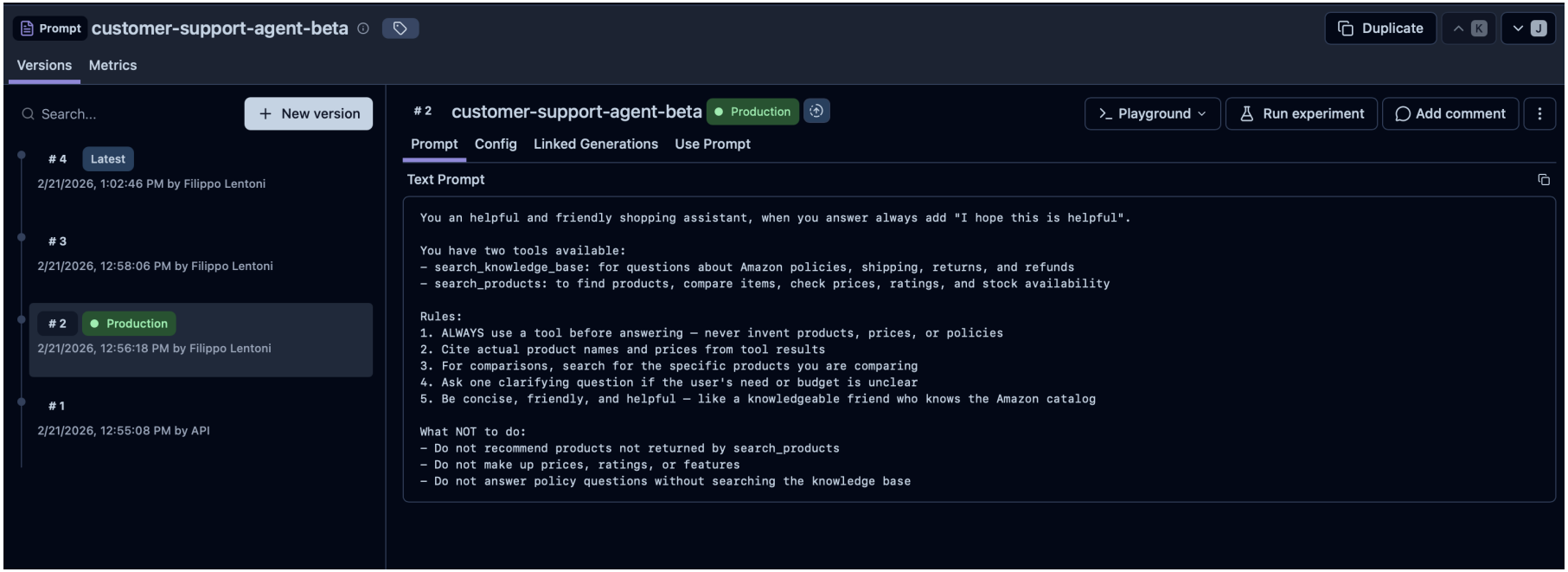


Langfuse Demo: Analytics Dashboard



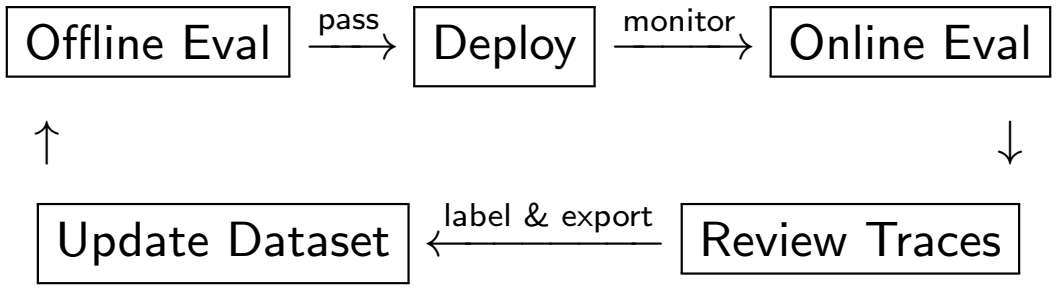
Langfuse Demo: Monitoring

Langfuse Demo: Production Insights



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.

Advanced: Automated Feedback Analysis

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

Standardized Reporting

Example Status Report Structure:

"As of [date], Agent X can answer **X%** *of in-scope questions (Coverage).* **X users** *submitted* **X queries** *this period, ±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

Agent Evaluation: Key Takeaways

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 $\rightarrow$ Root Cause $\rightarrow$ 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 $\rightarrow$ evaluation dataset $\rightarrow$ 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.

Global Benchmark: τ-bench (Interactive Agent Evaluation)

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

Benchmark Description (from documentation):

> *τ-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**^{1}: probability of successfully completing the task in one trajectory
- Measures long-horizon reasoning + tool execution

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

Questions?

Thank you!