MLOps and LLM Observability

Karthikeyan Vaiyapuri

01

Overview

Today's Learning Journey

LLM Observability Fundamentals

Why LLMs need different monitoring

Traditional ML vs LLM Monitoring

Key differences and challenges

Tools and Technologies

Popular LLM monitoring solutions

Prompt Logging and Tracking

Capture and analyze LLM interactions

Hands-on Lab

Build complete LLM monitoring system

02

What is LLM Observability?

What is LLM Observability?

LLM Observability = The ability to understand, monitor, and debug Large Language Model behavior in production through comprehensive tracking of inputs, outputs, and system performance.

Why LLMs Need Special Observability

Non-deterministic Outputs

Same input can produce different responses

Complex Reasoning Chains

Hard to trace decision paths

Prompt Sensitivity

Small input changes cause big output differences

High Computational Costs

Token usage impacts expenses

Quality is Subjective

No simple accuracy metric

Core Principles

Transparency

See what the model is actually doing

Traceability

Track from prompt to response

Performance

Monitor speed, cost, and quality

Debugging

Identify and fix issues quickly

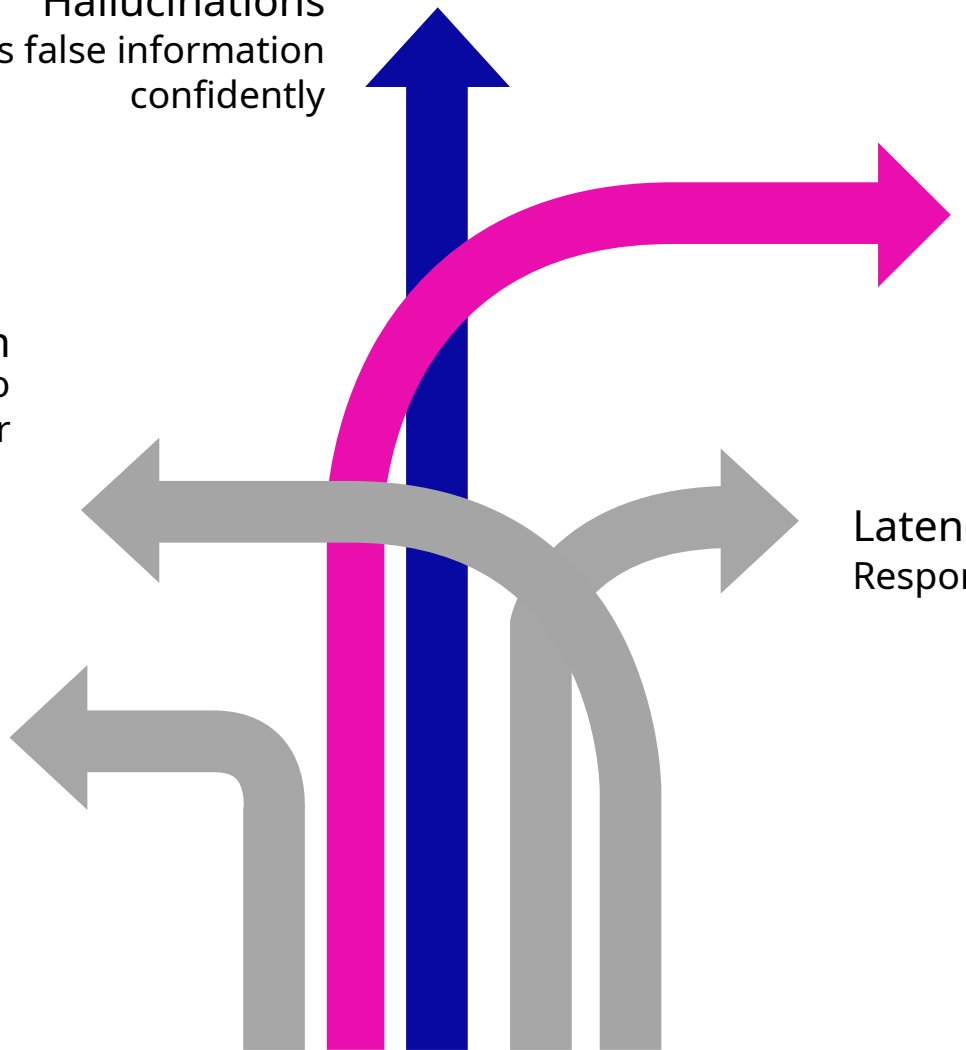
03

Traditional ML vs LLM Monitoring - Key Differences

Traditional ML vs LLM Monitoring

Aspect	Trditional ML	LLMs
Input Data	Structured features(numbers)	Unstructured text (prompts)
Output	Predictions (classes/numbers)	Generated text (variable length)
Evaluation	Clear metrics (accuracy, F1)	Subjective quality assessment
Determinism	Same input → Same output	Same input → Different outputs
Cost Model	Fixed inference cost	Token-based variable cost
Failure Modes	Wrong predictions	Hallucinations, bias, toxicity
Debugging	Feature importance	Prompt engineering, reasoning traces

New Challenges with LLMs



Hallucinations
Model generates false information confidently

Context Window Limits
Managing long conversations and documents

Prompt Injection
Malicious inputs trying to manipulate behavior

Latency Variability
Response time depends on output length

Bias and Toxicity
Inappropriate or harmful content generation

04

Why LLM Observability is Critical

Business Impact

01

Cost Control

LLM inference can be 10-100x more expensive than traditional ML

02

Quality Assurance

Poor responses damage user experience and brand

03

Compliance

Need to track and audit AI-generated content

04

Performance Optimization

Identify bottlenecks and improve efficiency

Technical Necessity

Debug Complex Failures

Understand why model gave wrong answer

Optimize Prompts

Data-driven prompt engineering decisions

Detect Drift

Monitor when model behavior changes over time

Resource Planning

Predict and manage computational costs

Risk Management

Safety Monitoring

Detect harmful or inappropriate outputs

Security

Identify prompt injection and adversarial attacks

Reliability

Ensure consistent performance under load

Regulatory Compliance

Meet AI governance requirements

05

Core LLM Observability Metrics

Quality Metrics

Relevance Score

How well response answers the question

Coherence Score

Logical consistency of generated text

Factuality Check

Accuracy of information provided

Toxicity Detection

Identify harmful content

Hallucination Rate

Frequency of false information generation

Performance Metrics

Latency

Time to first token, total response time

Throughput

Requests per second, tokens per second

Token Usage

Input tokens, output tokens, total cost

Error Rates

Failed requests, timeout rates

Availability

System uptime and responsiveness

Usage Metrics

Prompt Patterns

Common user inputs and intents

Response Length Distribution

Typical output characteristics

User Satisfaction

Feedback scores and engagement metrics

Conversation Flow

Multi-turn interaction analysis

06

Popular LLM Monitoring Tools Overview

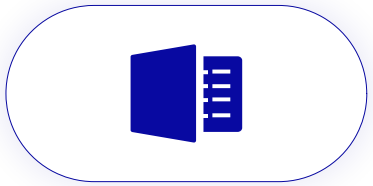
Commercial Solutions

Tool	Strengths	Best For
LangSmith	Prompt optimization, tracing	LangChain applications
Weights & Biases	Expert tracking, visualization	Research and Development
Arize AI	Model monitoring, drift detection	Production monitoring
Humanloop	Prompt management, A/B testing	Prompt engineering teams

Open Source Options

Tool	Features	Use Case
MLflow	Experiment tracking, model registry	General ML operations
Phoenix	LLM tracing, evaluation	Debugging and analysis
Langfuse	Prompt tracking, analytics	Cost optimization
TrueLens	Response evaluation, feedback	Quality assessment

Cloud Platform Tools



Azure AI Studio

Integrated monitoring for Azure OpenAI

AWS Bedrock

Observatory for AWS models

Google Vertex AI

Monitoring for PaLM and other models

07

Introduction to Prompt Logging and Response Tracking

What is Prompt Logging?

Complete capture and storage of LLM interactions including:

- Input prompts (user queries, system instructions)
- Model responses (generated text)
- Metadata (timestamps, model version, parameters)
- Context (conversation history, user session)

Why Track Prompts and Responses?

01 Quality Analysis

Identify patterns in good/bad responses

02 Cost Optimization

Understand token usage patterns

03 Prompt Engineering

Data-driven improvement of prompts

04 Compliance

Audit trail for AI-generated content

05 Debugging

Reproduce and fix specific issues

What to Log

```
{
  "timestamp": "2025-07-29T10:30:00Z",
  "session_id": "user_123_session_456",
  "prompt": "Explain photosynthesis in simple terms",
  "response": "Photosynthesis is how plants make food...",
  "model": "gpt-4",
  "parameters": {"temperature": 0.7, "max_tokens": 150},
  "metrics": {"latency_ms": 1200, "input_tokens": 8, "output_tokens": 45},
  "user_feedback": {"rating": 4, "helpful": true}
}
```

08

Prompt Logging Architecture

System Components

```
User Input → Prompt Processing → LLM API → Response Processing → Storage
     ↓              ↓              ↓              ↓              ↓
   Logging     Prompt Enhancement  Metrics     Quality Check   Analytics
```

Logging Layers

01 **Request Layer**
Capture incoming user requests

02 **Processing Layer**
Log prompt engineering transformations

03 **Model Layer**
Track API calls and responses

04 **Quality Layer**
Evaluate and score responses

05 **Storage Layer**
Persist data for analysis

Key Design Principles

Non-blocking

Don't slow down user experience

Comprehensive

Capture all relevant information

Structured

Enable easy querying and analysis

Secure

Protect sensitive information

Scalable

Handle high-volume production traffic

09

Techniques for Input-Output Relationship Tracking

Conversation Threading

```
# Track multi-turn conversations
conversation_thread = {
    "conversation_id": "conv_789",
    "turns": [
        {"role": "user", "content": "What is AI?", "timestamp": "..."},
        {"role": "assistant", "content": "AI is...", "timestamp": "..."},
        {"role": "user", "content": "How does it work?", "timestamp": "...
        {"role": "assistant", "content": "AI works by...", "timestamp": ".
    ]
}
```

Prompt Template Tracking

```python
# Track which templates produce best results
template_performance = {
    "template_id": "customer_support_v2",
    "template": "You are a helpful customer service agent. {context}",
    "usage_count": 1500,
    "avg_satisfaction": 4.2,
    "success_rate": 0.87
}
```

Response Quality Correlation

Response Quality Correlation
Input Characteristics vs Output Quality
Prompt length vs response relevance
Question type vs answer accuracy
Context amount vs coherence score
Temperature setting vs creativity rating

10

Monitoring Hallucinations and Quality Issues

Hallucination Detection Methods

1. **Fact-checking APIs**
   Verify factual claims automatically

2. **Confidence Scoring**
   Model's certainty about responses

3. **Source Attribution**
   Track where information comes from

4. **Consistency Checking**
   Compare responses to similar questions

Quality Monitoring Techniques

```python
# Automated quality checks
quality_metrics = {
    "relevance_score": 0.85,      # How well it answers the question
    "coherence_score": 0.92,      # Logical consistency
    "toxicity_score": 0.02,       # Harmful content (lower is better)
    "factual_accuracy": 0.78,     # Verified facts percentage
    "response_completeness": 0.88  # Addresses all parts of question
}
```

Alert Thresholds

High Toxicity

> 0.1 → Immediate human review

Low Relevance

< 0.6 → Flag for prompt improvement

Potential Hallucination

Confidence < 0.5 → Add disclaimers

Response Too Long

> 500 tokens → Check for rambling

11

Cost and Performance Monitoring

Token Usage Tracking

```python
# Track costs in real-time
cost_metrics = {
    "input_tokens": 1200,
    "output_tokens": 800,
    "total_tokens": 2000,
    "cost_per_token": 0.00002,  # $0.02 per 1K tokens
    "total_cost_usd": 0.04,
    "cost_per_user": 0.04,
    "daily_budget_used": 0.12   # 12% of daily budget
}
```

Performance Monitoring

Time to First Token
How quickly response starts

Tokens per Second
Generation speed

Queue Wait Time
Load balancing efficiency

Error Rate by Model
Reliability tracking

Peak Usage Patterns
Capacity planning

Cost Optimization Strategies

01

Prompt Compression
Reduce input token count

02

Response Length Limits
Control output costs

03

Model Selection
Use cheaper models when appropriate

04

Caching
Store common responses

05

Batch Processing
Group similar requests

12

Key Success Metrics for LLM Observability

Quality KPIs

Response Relevance: > 85% of responses rated as relevant

Factual Accuracy: > 90% for factual questions

Toxicity Rate: < 1% of responses flagged as harmful

User Satisfaction: Average rating > 4.0/5.0

Performance KPIs

Average Latency: < 3 seconds for 95% of requests

Availability: > 99.5% uptime

Error Rate: < 2% of requests fail

Throughput: Handle target requests per second

Cost KPIs

01

Cost per Conversation: Stay within budget targets

02

Token Efficiency: Minimize unnecessary token usage

03

Model ROI: Demonstrate business value vs cost

04

Budget Adherence: Stay within monthly limits

13

Best Practices for LLM Observability

Implementation Best Practices

01

Start Simple: Begin with basic logging, add complexity gradually

02

Focus on Business Impact: Monitor metrics that matter to users

03

Automate Everything: Reduce manual monitoring overhead

04

Plan for Scale: Design for production volume from day one

05

Secure by Design: Protect sensitive conversation data

Monitoring Strategy

Real-time Alerts: Critical issues need immediate attention

Batch Analysis: Daily/weekly deep dives into patterns

Continuous Improvement: Use data to optimize prompts and performance

User Feedback Integration: Combine automated metrics with human input

Common Pitfalls to Avoid

Over- logging
Don't capture everything, focus on actionable metrics

Ignoring Context
Monitor conversations, not just individual responses

Cost Blindness
Always track and optimize token usage

Quality Assumptions
Measure, don't assume response quality

14

Real-World LLM Monitoring Scenarios

Scenario 1: Customer Support Chatbot

Monitoring Focus

- Response accuracy for common questions
- Escalation rate to human agents
- Customer satisfaction scores
- Average resolution time

Scenario 2: Content Generation Platform

Monitoring Focus

Content quality and originality

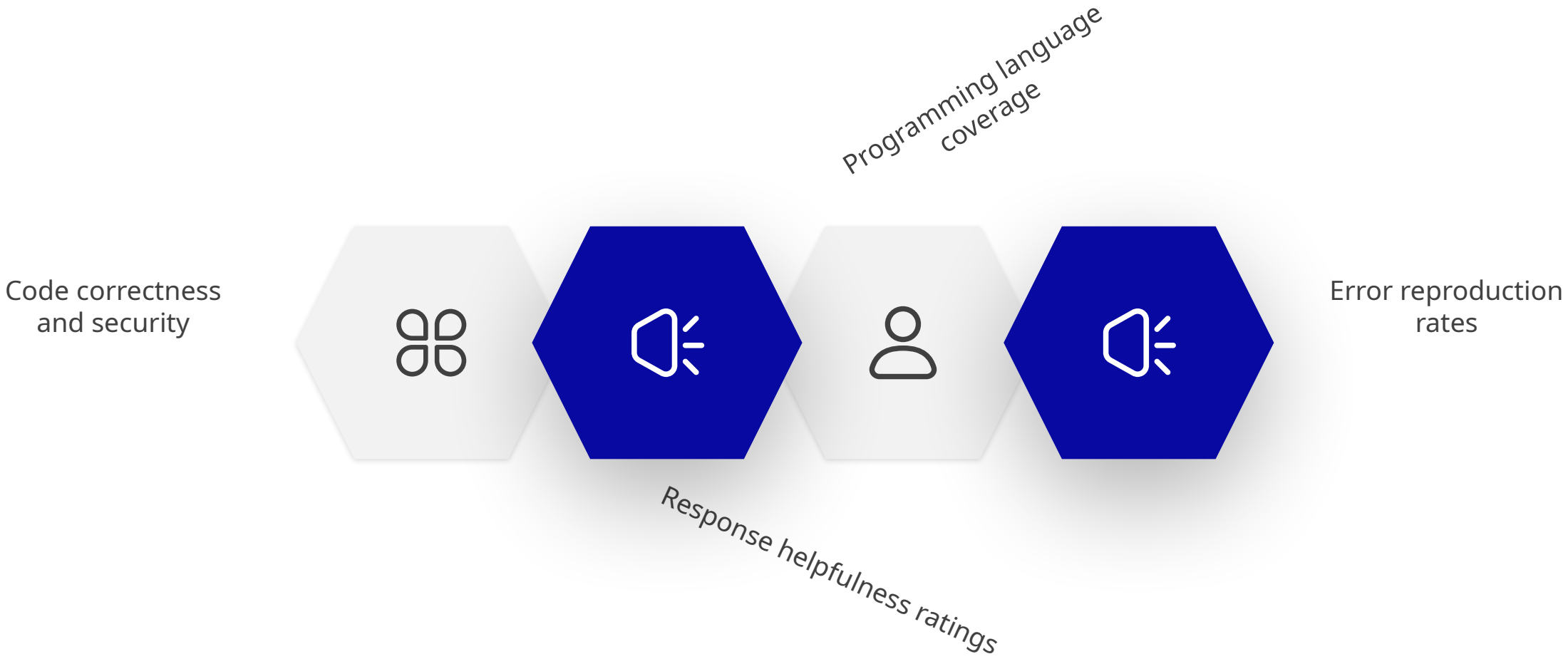
Brand voice consistency

Toxicity and compliance issues

Generation speed and costs

Scenario 3: Code Assistant

Monitoring Focus



15

Advanced LLM Observability Topics

Beyond Basic Monitoring

Prompt Injection Detection
Security monitoring

Bias and Fairness Tracking
Ethical AI monitoring

Multi- modal Monitoring
Images, audio, video inputs

Chain-of-Thought Tracing
Reasoning step analysis

A/B Testing Framework
Systematic prompt optimization

Thanks

Karthikeyan Vaiyapuri