Reduce AI Agent Costs and Mistakes with Semantic Tool Selection

Filter to the 3 tools that fit the query, before the model ever sees them

Based on: Internal Representations as Indicators of Hallucinations in Agent Tool Selection (arXiv 2601.05214)

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Elizabeth Fuentes Leone

Developer Advocate @ AWS

Bilingual (EN/ES) · 107+ articles on dev.to · MS in Data Science

elifuentes.tech

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

6 Techniques to Stop AI Agents from Failing

Part of a series on production patterns for reliable AI agents:

01 **Graph-RAG**

Knowledge graphs for grounded answers

02 **Semantic Tool Selection**

FAISS filtering to cut tokens and errors

03 **Multi-Agent Validation**

Executor, Validator, Critic pipeline

04 **Neurosymbolic Guardrails**

Hard rules the LLM cannot bypass

05 **Agent Control Steering**

Self-correction instead of failure

06 **Production Deployment**

AgentCore Gateway, serverless, observability

aws

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Today: Semantic Tool Selection

01 **Graph-RAG**

Knowledge graphs for grounded answers

02 **Semantic Tool Selection**

FAISS filtering to cut tokens and errors

03 **Multi-Agent Validation**

Executor, Validator, Critic pipeline

04 **Neurosymbolic Guardrails**

Hard rules the LLM cannot bypass

05 **Agent Control Steering**

Self-correction instead of failure

06 **Production Deployment**

AgentCore Gateway, serverless, observability

aws

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The Dual Problem: Tokens and Wrong Tools

Your agent has 29 tools. On every call, all 29 descriptions get packed into context:

$ Token Waste

- 29 tools packed on every call
- Thousands of tokens per query
- Paid whether or not they are used
- Cost grows with each tool added

? Wrong Tool Selection

- Generic tools compete with specific
- search vs search_hotels vs search_flights
- More tools means more confusion
- Wrong pick becomes a wrong answer

Research (arXiv 2601.05214) finds 5 failure modes as tools scale: function selection, appropriateness, parameter, completeness, and tool bypass. Fewer tools in front of the model means fewer of these.

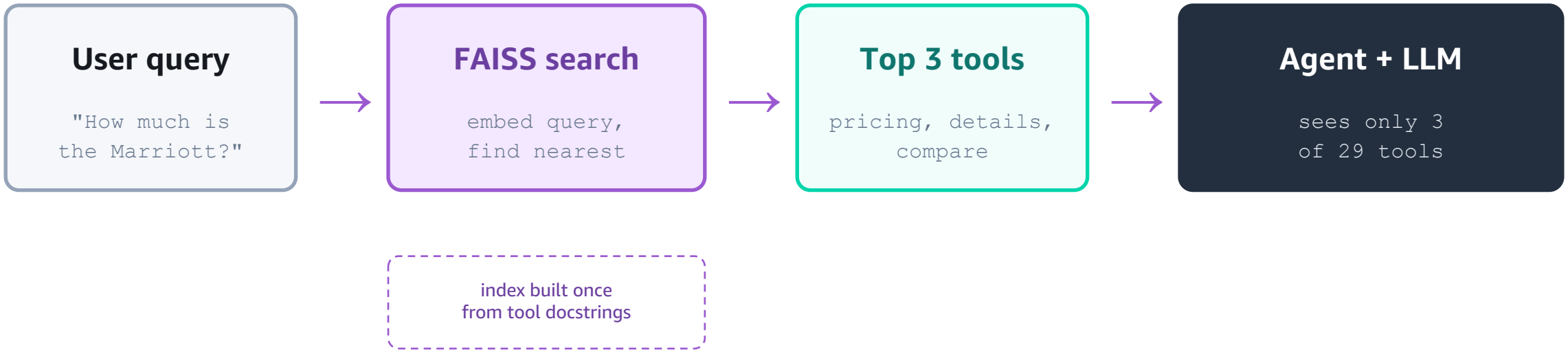
aws
© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

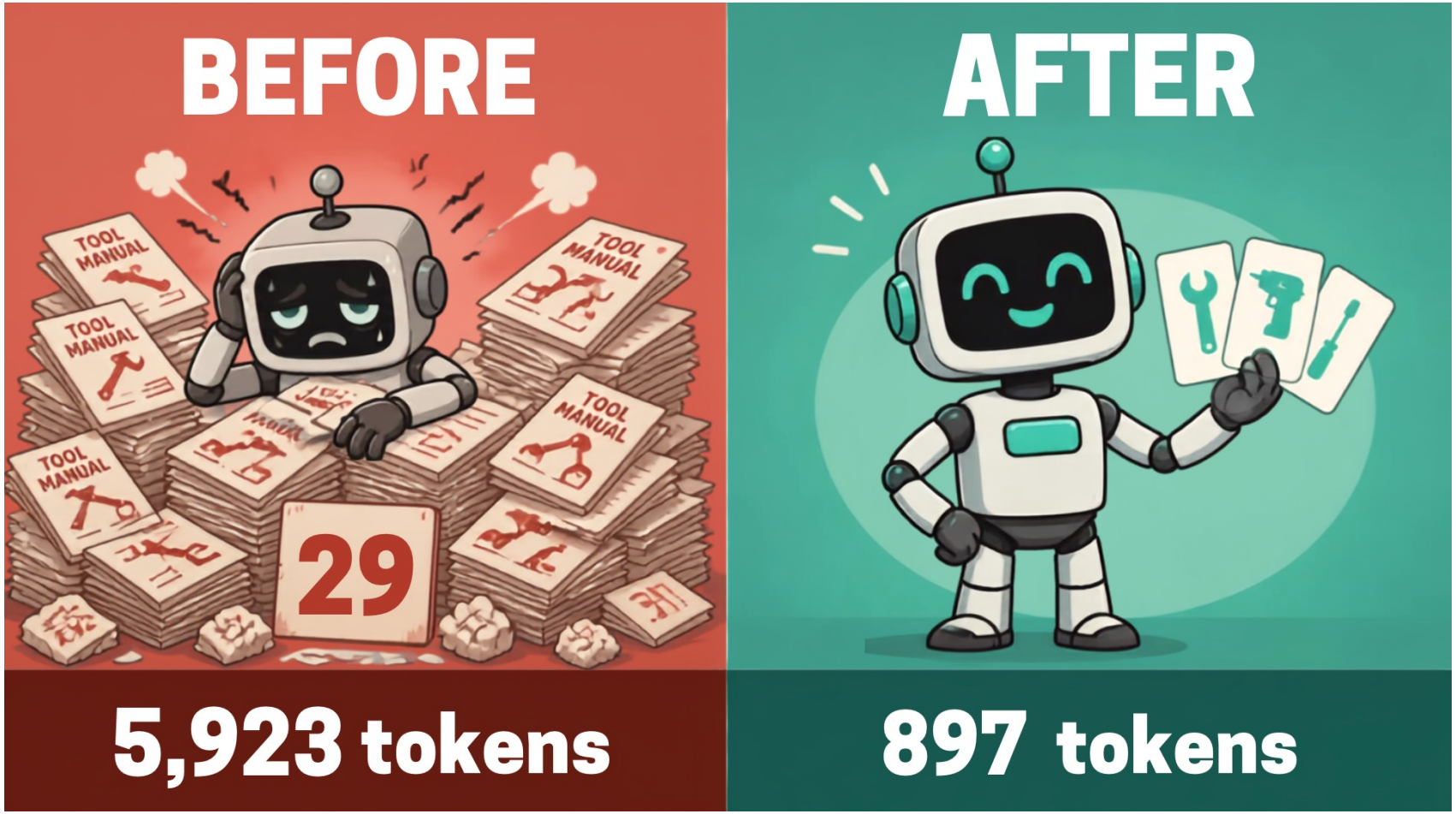
How It Works: Filter Before the Model



The query is embedded once. FAISS returns the closest tool embeddings. Only those reach the model, so each call carries a fraction of the tokens.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Traditional vs Semantic Tool Discovery



84.9% fewer tokens

measured on real gpt-4o-mini runs, not estimated

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The Token Math (Measured)

Same 3 queries, same agent, only the tool list changes:

TRADITIONAL

5,923

total tokens

all 29 tools every query

→

SEMANTIC

897

total tokens

top 3 filtered tools

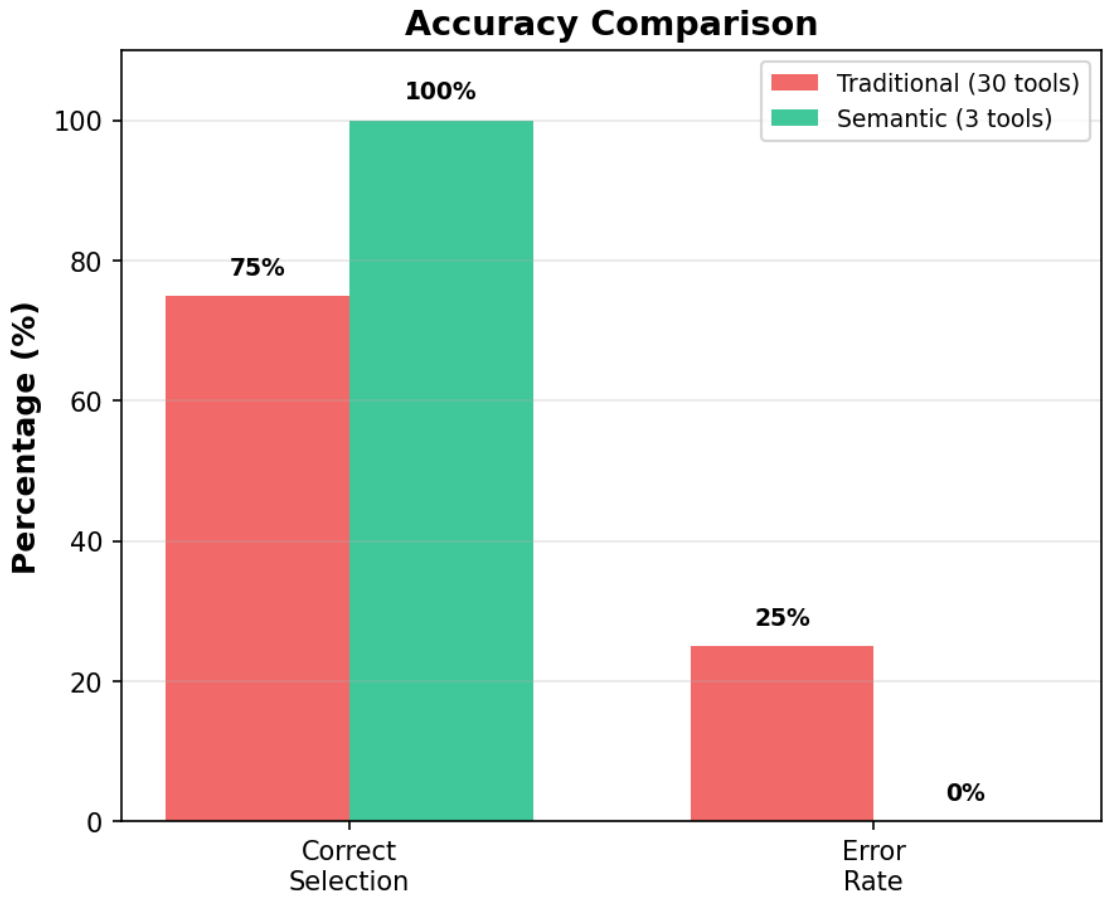
84.9% fewer tokens

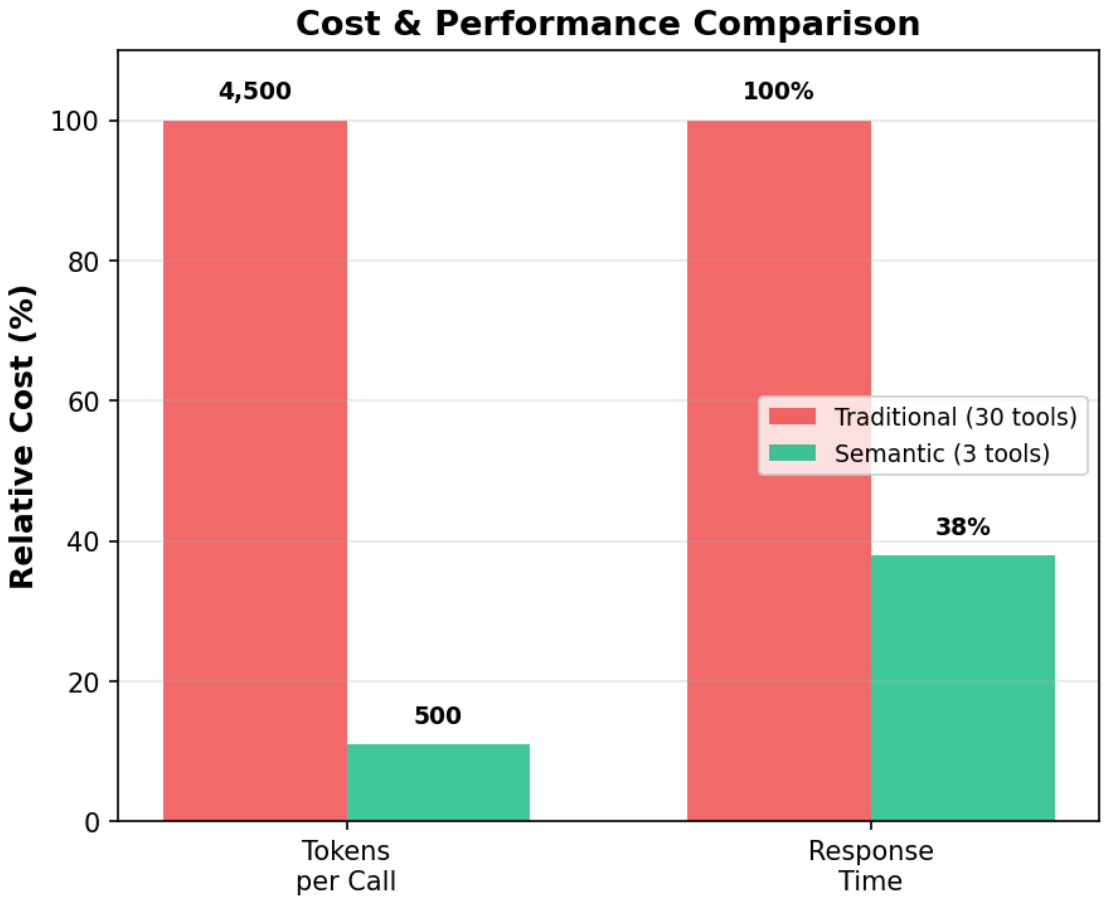
measured on real gpt-4o-mini runs, not estimated

aws

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Accuracy and Token Cost, Side by Side





© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Demo Time

29 tools, token counts, and accuracy on identical queries

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

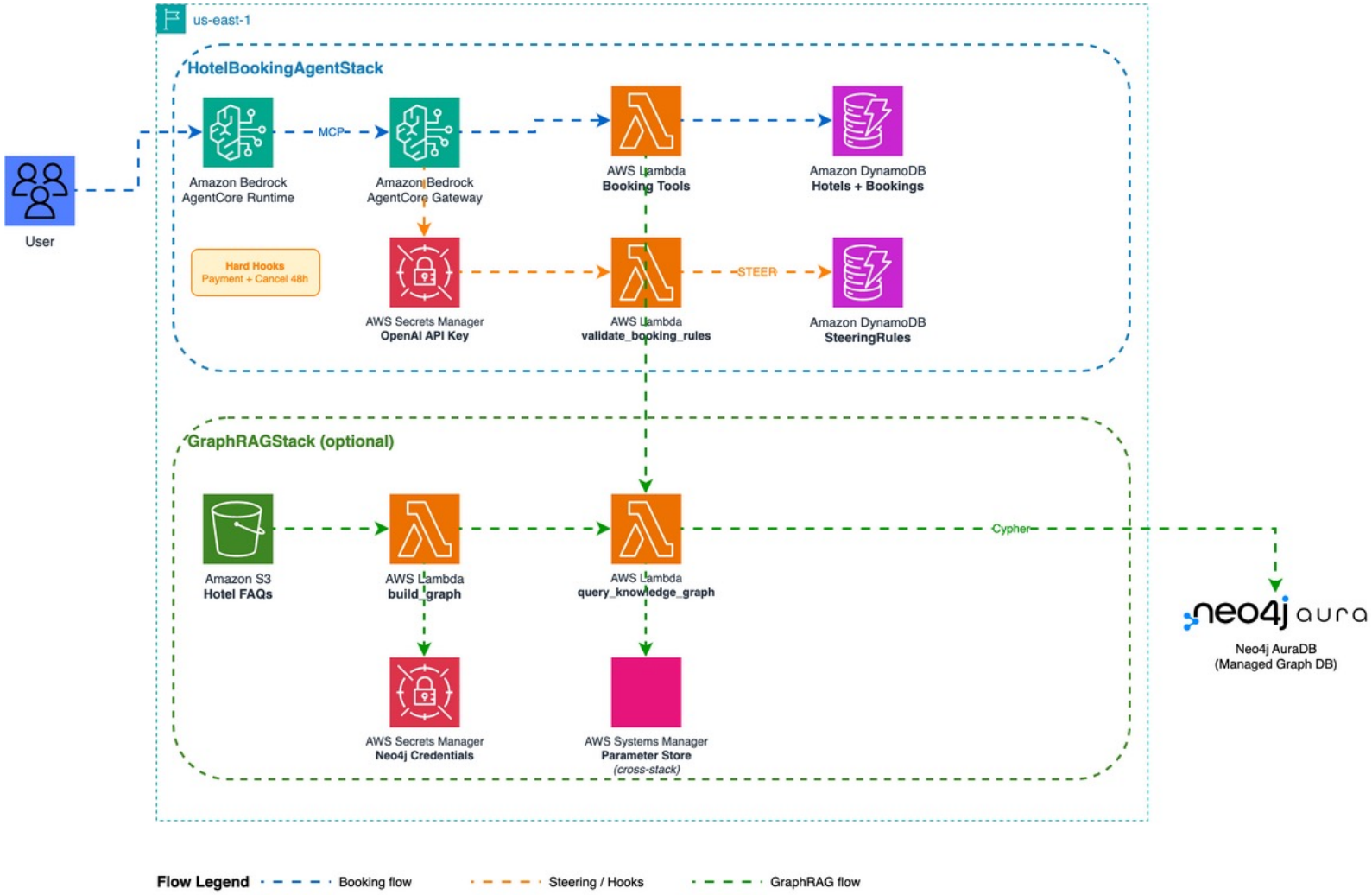
That was the local demo.

How does this run in production?

Local notebook	→	**Managed AgentCore Runtime**
FAISS in memory	→	**Gateway semantic search over MCP**
print() statements	→	**CloudWatch + OpenTelemetry**
No guardrails	→	**Framework-level hooks**

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Production: AgentCore Gateway Semantic Search



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Local FAISS vs AgentCore Gateway

Aspect	Local (this demo)	Production (Gateway)
Filtering	`FAISS + SentenceTransformer`	`x_amz_bedrock_agentcore_search`
Tool hosting	`Python functions in process`	`Lambda functions (auto-scale)`
Memory	`swap_tools + agent.messages`	`MCP session state`
Guardrails	`None`	`Framework hooks (cannot bypass)`
Observability	`print() statements`	`OpenTelemetry + CloudWatch`
Index upkeep	`You build and refresh it`	`Managed by the Gateway`

Same pattern, managed infrastructure. The Gateway replaces your FAISS index.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Key Takeaways

1. Filter tools before the model with FAISS + SentenceTransformers
2. Fewer tools in context cut cost and wrong-tool errors at once
3. Measured 84.9% token reduction on identical queries
4. Preserve conversation memory while swapping tools per query
5. In production, AgentCore Gateway does semantic search for you

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Resources

Code	github.com/aws-samples/sample-why-agents-fail
Paper	Internal Representations as Indicators of Hallucinations (arXiv 2601.05214)
Blog	dev.to/elizabethfuentes12

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank You!

Elizabeth Fuentes Leone

Developer Advocate @ AWS

Blog & Socials

elifuentes.tech

Resources

bit.ly/4bVJUBw

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.