Elizabeth Fuentes Leone

Developer Advocate

elifuentes.tech

elifuentes.tech

Semantic Tool Selection

Reduce AI Agent Costs by Filtering Tools Before the LLM Sees Them

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

6 Techniques to Stop AI Agents from Failing

This talk is part of a series covering production patterns for reliable AI agents:

01 **Graph-RAG**

Knowledge graphs for grounded answers

02 **Semantic Tool Selection**

FAISS filtering to reduce 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**

MCP Gateway, serverless, observability

Today: Semantic Tool Selection

01 **Graph-RAG**

Knowledge graphs for grounded answers

02 **Semantic Tool Selection**

FAISS filtering to reduce 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**

MCP Gateway, serverless, observability

The Dual Problem: Tokens + Hallucinations

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

$$ Token Waste

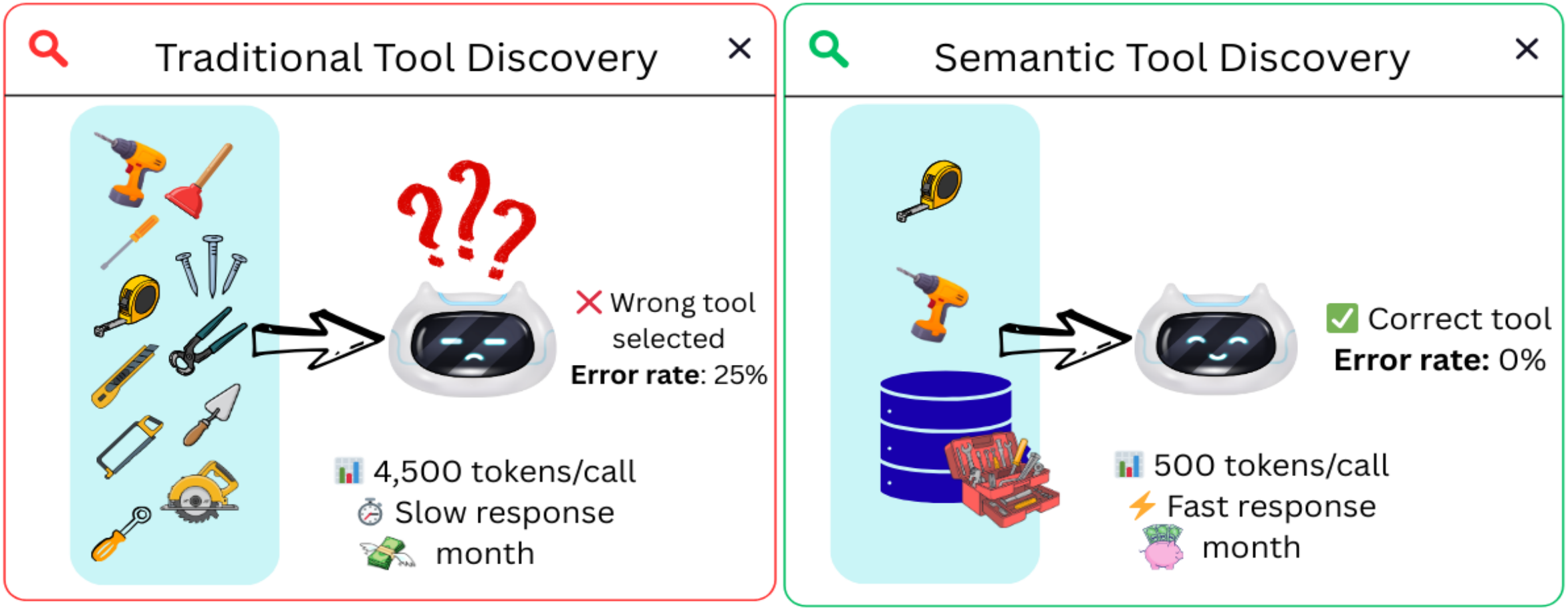
29 tools x 50 tokens each
= ~1,450 tokens per query
Paid on every single call
Cost scales linearly with tools

?! Wrong Tool Selection

Generic tools compete with specific ones (search vs search_hotels vs search_flights)
15% error rate at scale

Research identifies 5 failure modes as tools scale: function selection errors, appropriateness errors, parameter errors, completeness errors, and tool bypass behavior. More tools = more confusion for the LLM.

Traditional vs Semantic Tool Discovery



Solution: Semantic Tool Selection with FAISS

BEFORE

~1,450

tokens per query

29 tool descriptions

Paid on every call

→

AFTER

~150

tokens per query

3 filtered tools

FAISS similarity search

89% token reduction per query

Demo Time

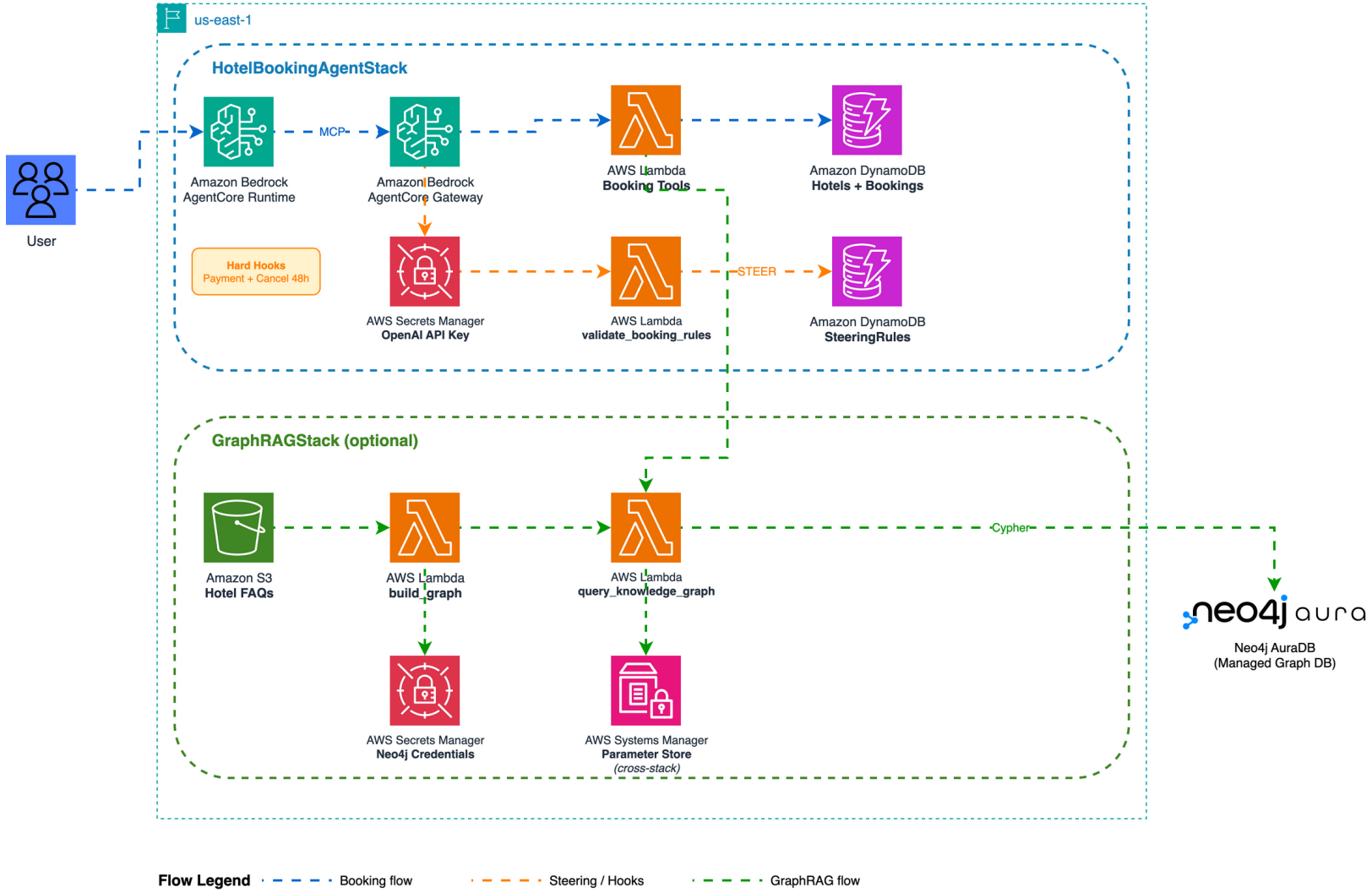
Code, token counts, and accuracy results

That was the demo.

How does this work in production?

Local notebook	→	**Managed runtime**
FAISS in memory	→	**MCP semantic routing**
Print statements	→	**CloudWatch + OpenTelemetry**
No guardrails	→	**Framework-level hooks**

Production: MCP Gateway with Semantic Routing



Local FAISS vs MCP Gateway

Aspect	Local (Demo)	Production (MCP Gateway)
Filtering	`FAISS + SentenceTransformer`	**`search_type="SEMANTIC"`**
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`**
Scaling	`Single machine`	**`Serverless (pay-per-use)`**

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

Key Takeaways

1. Implement semantic tool selection with FAISS + SentenceTransformers
2. Build a tool registry pattern with embeddings and metadata
3. Apply dynamic tool swapping while preserving conversation memory
4. Deploy to production with MCP Gateway semantic routing
5. Evaluate local vs managed approaches with measured tradeoffs

Resources

Code	github.com/aws-samples/sample-why-agents-fail
Paper	Internal Representations as Indicators of Hallucinations (arxiv 2601.05214)
Paper	Context Window Optimization (arxiv 2511.22729v1)
Blog	dev.to/elizabethfuentes12

Thank You!

Elizabeth Fuentes Leone

elifuentes.tech

Blog & Socials

elifuentes.tech

Resources

bit.ly/4bVJUBw