Elizabeth Fuentes Leone

Developer Advocate

elifuentes.tech

elifuentes.tech

Semantic Tool Selection

Reduce los costos de agentes IA filtrando herramientas antes de que el LLM las vea

Basado en: Internal Representations as Indicators of Hallucinations in Agent Tool Selection (arxiv 2601.05214)

6 tecnicas para evitar que los agentes IA fallen

Esta charla es parte de una serie sobre patrones de produccion para agentes IA confiables:

01

Graph-RAG

Grafos de conocimiento para respuestas fundamentadas

02

Semantic Tool Selection

Filtrado FAISS para reducir tokens y errores

03

Multi-Agent Validation

Pipeline Ejecutor, Validador, Critico

04

Neurosymbolic Guardrails

Reglas duras que el LLM no puede evadir

05

Agent Control Steering

Auto-correccion en lugar de fallo

06

Production Deployment

MCP Gateway, serverless, observabilidad

Hoy: Semantic Tool Selection

01 **Graph-RAG**

Grafos de conocimiento para respuestas fundamentadas

02 **Semantic Tool Selection**

Filtrado FAISS para reducir tokens y errores

03 **Multi-Agent Validation**

Pipeline Ejecutor, Validador, Critico

04 **Neurosymbolic Guardrails**

Reglas duras que el LLM no puede evadir

05 **Agent Control Steering**

Auto-correccion en lugar de fallo

06 **Production Deployment**

MCP Gateway, serverless, observabilidad

El problema dual: tokens + alucinaciones

Tu agente tiene 29 herramientas. En cada llamada, las 29 descripciones se serializan en el contexto:

$$ Desperdicio de tokens

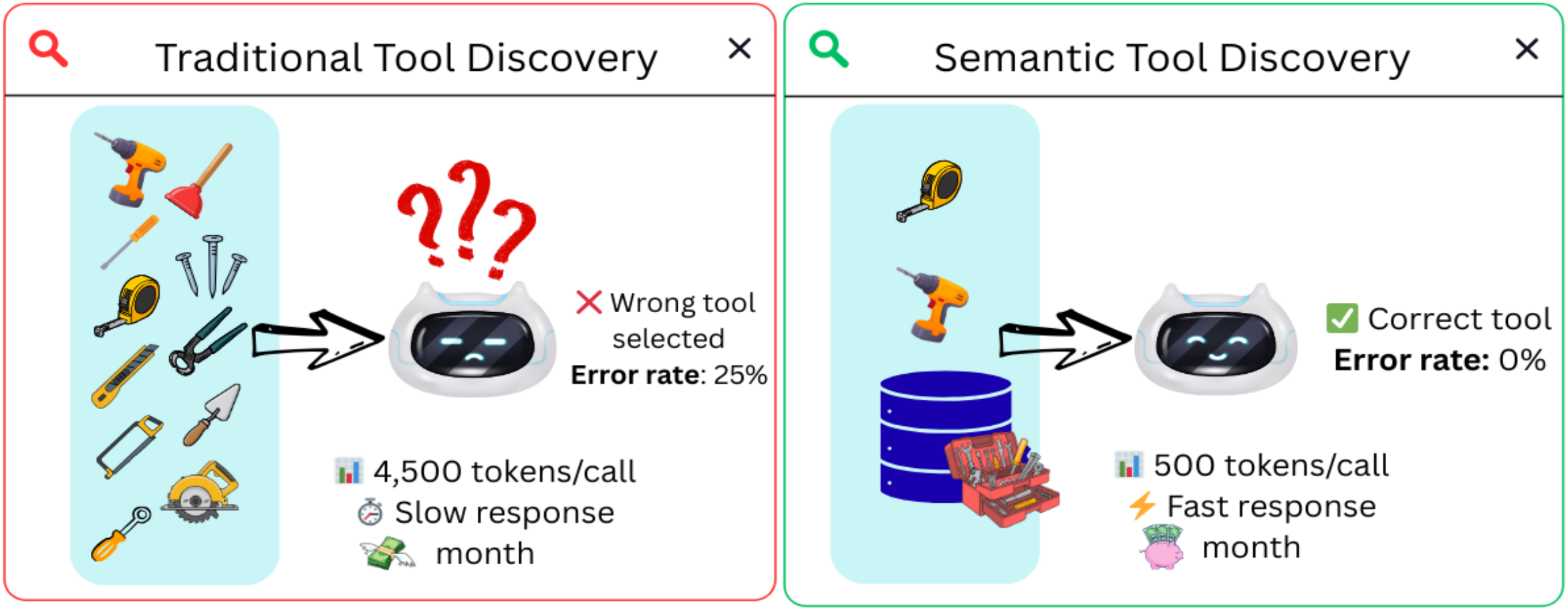
29 herramientas x 50 tokens
= ~1,450 tokens por consulta
Se paga en cada llamada
Costo escala linealmente

?! Herramienta incorrecta

Herramientas genericas compiten con las especificas (search vs search_hotels vs search_flights)
15% tasa de error a escala

La investigacion identifica 5 modos de fallo al escalar herramientas: errores de seleccion, errores de idoneidad, errores de parametros, errores de completitud y bypass de herramientas. Mas herramientas = mas confusion para el LLM.

Descubrimiento de herramientas: tradicional vs semantico



Solucion: seleccion semantica con FAISS

ANTES

~1,450

tokens por consulta

29 descripciones de herramientas

Se paga en cada llamada

→

DESPUES

~150

tokens por consulta

3 herramientas filtradas

Busqueda de similitud FAISS

89% de reduccion de tokens por consulta

Hora de la Demo

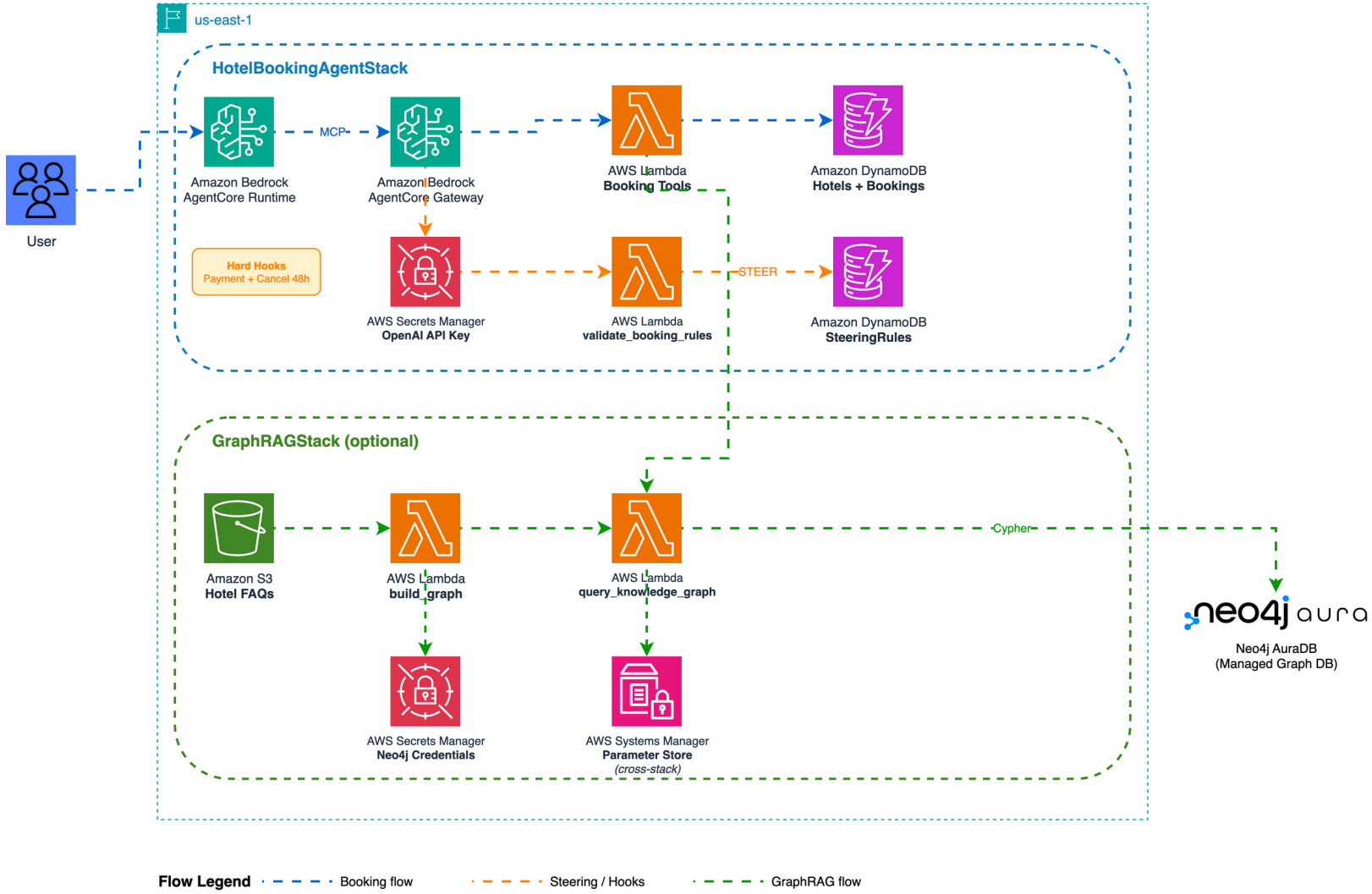
Codigo, conteo de tokens y resultados de precision

Eso fue la demo.

Como funciona esto en produccion?

Notebook local	→	**Runtime administrado**
FAISS en memoria	→	**Enrutamiento semantico MCP**
Print statements	→	**CloudWatch + OpenTelemetry**
Sin guardrails	→	**Hooks a nivel de framework**

Produccion: MCP Gateway con enrutamiento semantico



FAISS local vs MCP Gateway

Aspecto	Local (Demo)	Produccion (MCP Gateway)
Filtrado	`FAISS + SentenceTransformer`	**`search_type="SEMANTIC"`**
Herramientas	`Funciones Python en proceso`	**`Funciones Lambda (auto-escala)`**
Memoria	`swap_tools() + agent.messages`	**`Estado de sesion MCP`**
Guardrails	`Ninguno`	**`Hooks de framework (no evadible)`**
Observabilidad	`print() statements`	**`OpenTelemetry + CloudWatch`**
Escalabilidad	`Una sola maquina`	**`Serverless (pago por uso)`**

Mismo patron, infraestructura administrada. El Gateway reemplaza tu indice FAISS.

Conclusiones clave

1. Implementar seleccion semantica de herramientas con FAISS + SentenceTransformers
2. Construir un patron de registro de herramientas con embeddings y metadatos
3. Aplicar intercambio dinamico de herramientas preservando la memoria de conversacion
4. Desplegar a produccion con enrutamiento semantico de MCP Gateway
5. Evaluar enfoques local vs administrado con tradeoffs medidos

Recursos

Codigo	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

Gracias!

Elizabeth Fuentes Leone

elifuentes.tech

Blog y Redes

elifuentes.tech

Recursos

bit.ly/4bVJUBw