Reduce costos y errores de agentes de IA con seleccion semantica de herramientas

Filtra a las 3 herramientas que encajan con la consulta, antes de que el modelo las vea

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

Elizabeth Fuentes Leone

Developer Advocate @ AWS

Bilingue (EN/ES) · 107+ articulos en dev.to · Maestria en Data Science

elifuentes.tech

6 tecnicas para evitar que los agentes de IA fallen

Parte de una serie sobre patrones de produccion para agentes confiables:

01 **Graph-RAG**

Grafos de conocimiento para respuestas fundamentadas

02 **Seleccion semantica de herramientas**

Filtrado con FAISS para reducir tokens y errores

03 **Validacion multi-agente**

Pipeline Executor, Validator, Critic

04 **Guardrails neurosimbolicos**

Reglas que el LLM no puede saltarse

05 **Agent Control (Steering)**

Autocorreccion en lugar de fallar

06 **Despliegue en produccion**

AgentCore Gateway, serverless, observabilidad

Hoy: seleccion semantica de herramientas

01 **Graph-RAG**

Grafos de conocimiento para respuestas fundamentadas

02 **Seleccion semantica de herramientas**

Filtrado con FAISS para reducir tokens y errores

03 **Validacion multi-agente**

Pipeline Executor, Validator, Critic

04 **Guardrails neurosimbolicos**

Reglas que el LLM no puede saltarse

05 **Agent Control (Steering)**

Autocorreccion en lugar de fallar

06 **Despliegue en produccion**

AgentCore Gateway, serverless, observabilidad

El problema dual: tokens y herramientas equivocadas

Tu agente tiene 29 herramientas. En cada llamada, las 29 descripciones entran al contexto:

$ Desperdicio de tokens

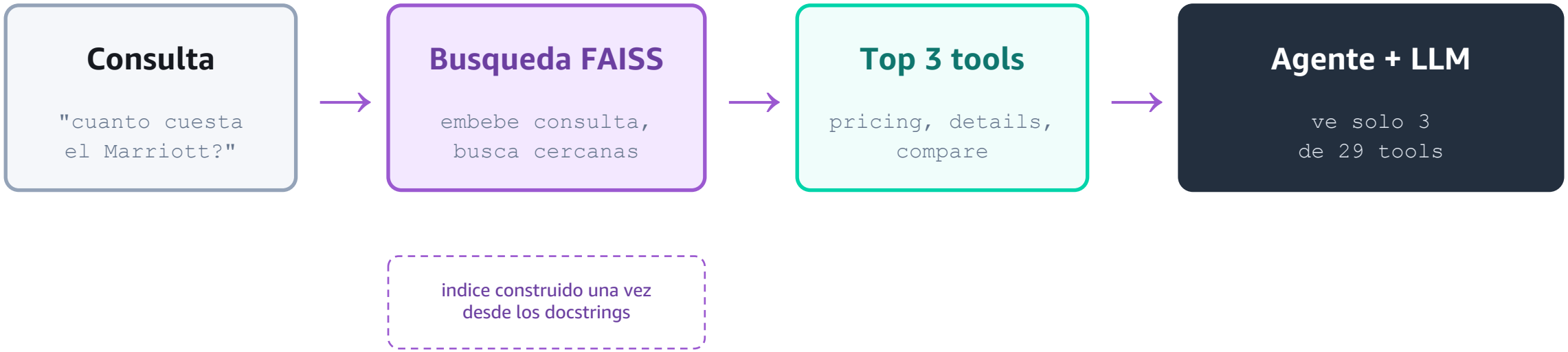
- Las 29 herramientas en cada llamada
- Miles de tokens por consulta
- Se pagan se usen o no
- El costo crece con cada herramienta

? Seleccion equivocada

- Las genericas compiten con las especificas
- search vs search_hotels vs search_flights
- Mas herramientas, mas confusion
- Elegir mal se vuelve responder mal

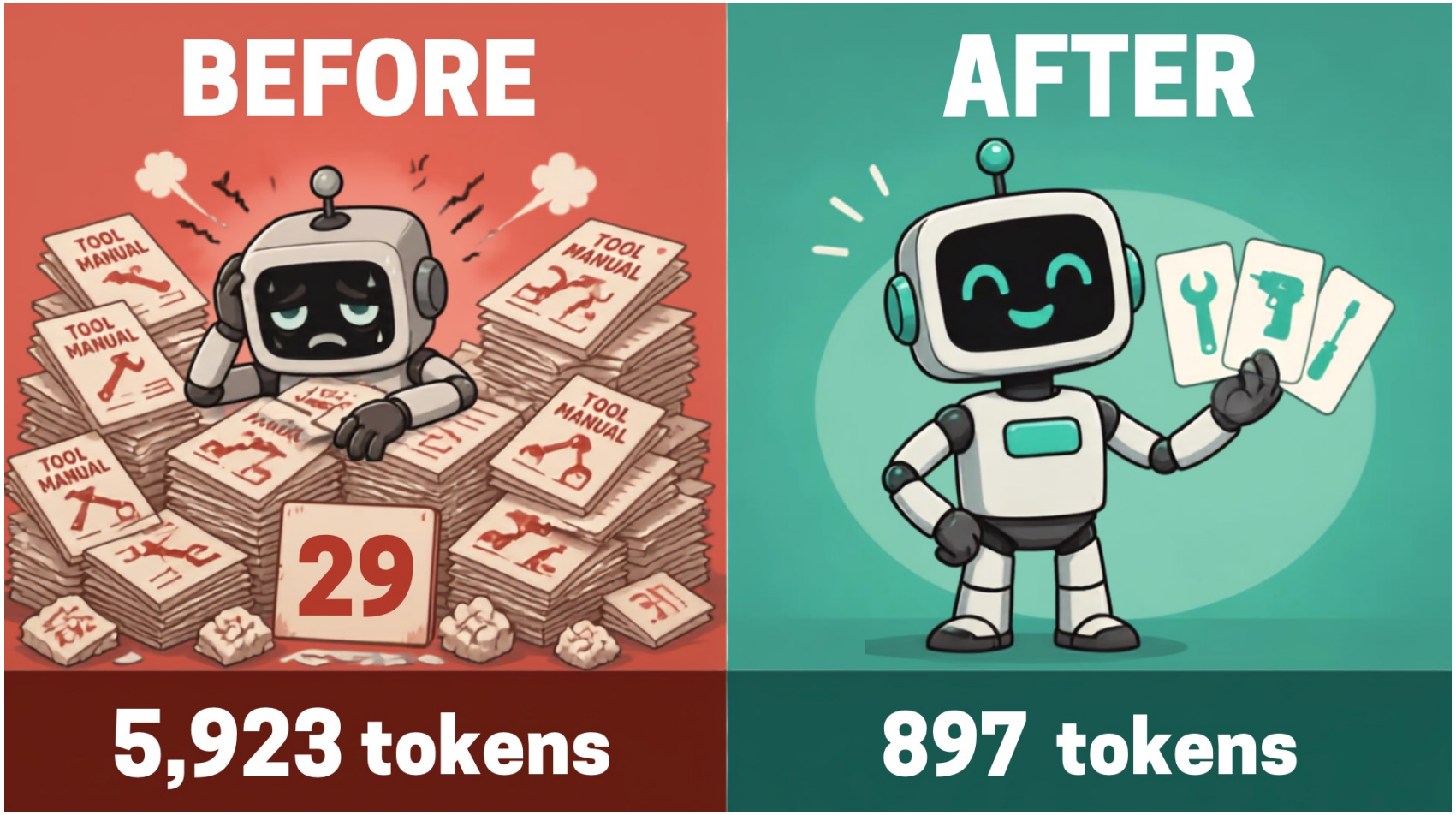
La investigacion (arXiv 2601.05214) identifica 5 modos de falla a medida que crecen las herramientas: seleccion de funcion, apropiacion, parametros, completitud y bypass. Menos herramientas frente al modelo significa menos de estos.

Como funciona: filtrar antes del modelo



La consulta se embebe una vez. FAISS devuelve las herramientas mas cercanas. Solo esas llegan al modelo, asi cada llamada carga una fraccion de los tokens.

Descubrimiento tradicional vs semantico



84.9% menos tokens

medido en corridas reales de gpt-4o-mini, no estimado

La matematica de tokens (medida)

Mismas 3 consultas, mismo agente, solo cambia la lista de herramientas:

TRADICIONAL

5,923

tokens totales

las 29 herramientas cada consulta

→

SEMANTICO

897

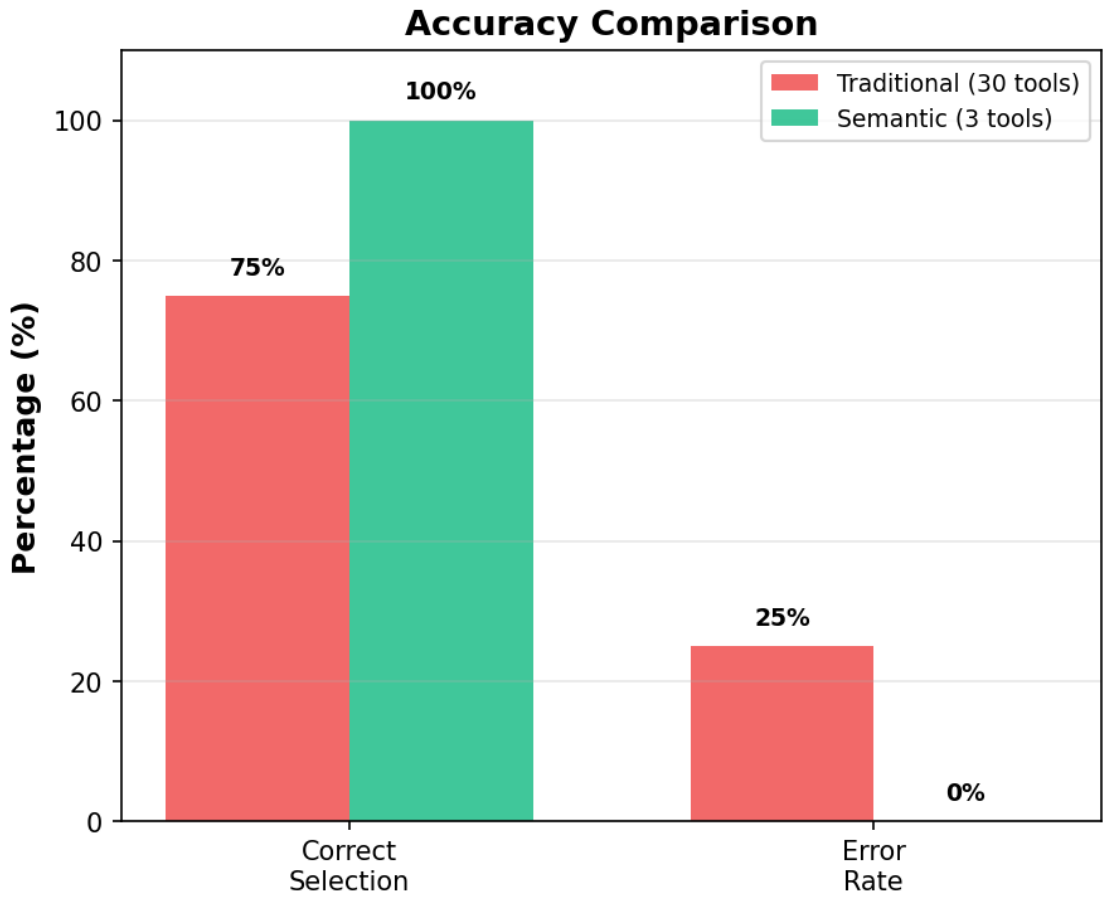
tokens totales

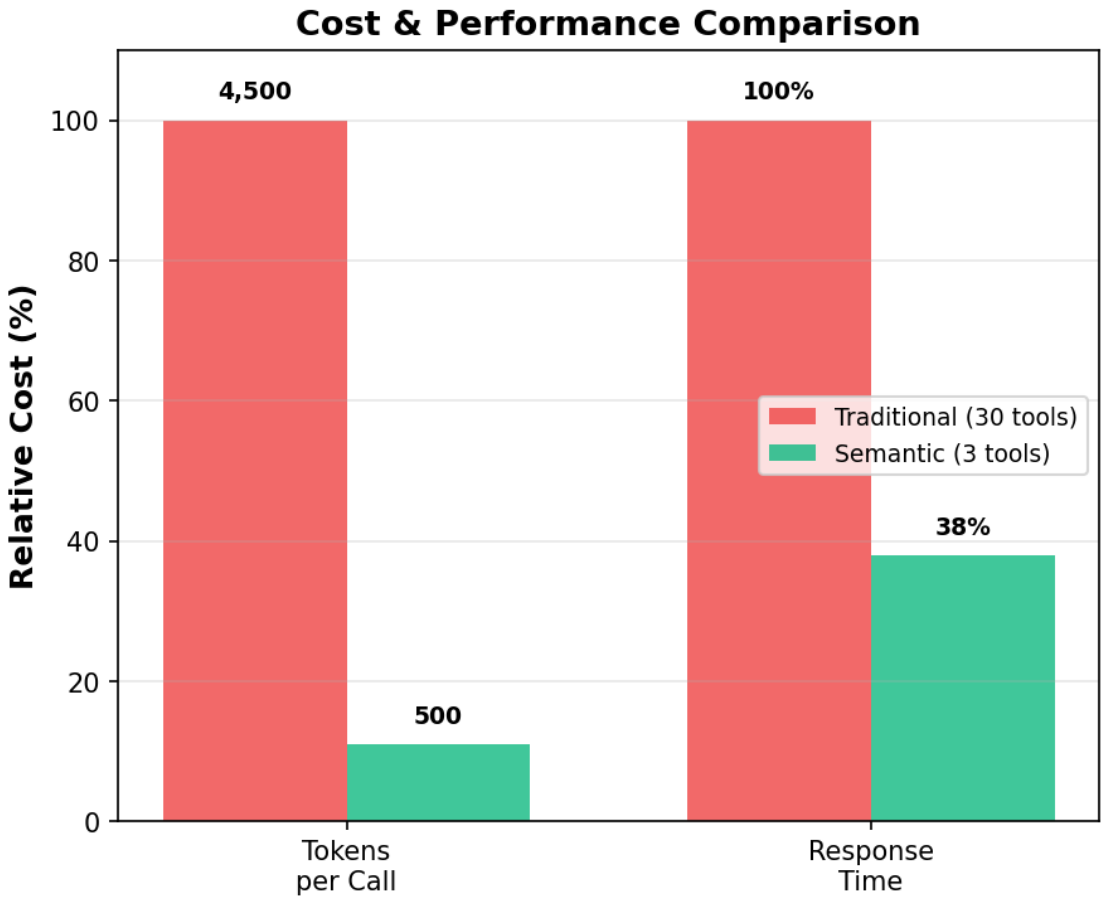
top 3 herramientas filtradas

84.9% menos tokens

medido en corridas reales de gpt-4o-mini, no estimado

Precision y costo de tokens, lado a lado





Demo en vivo

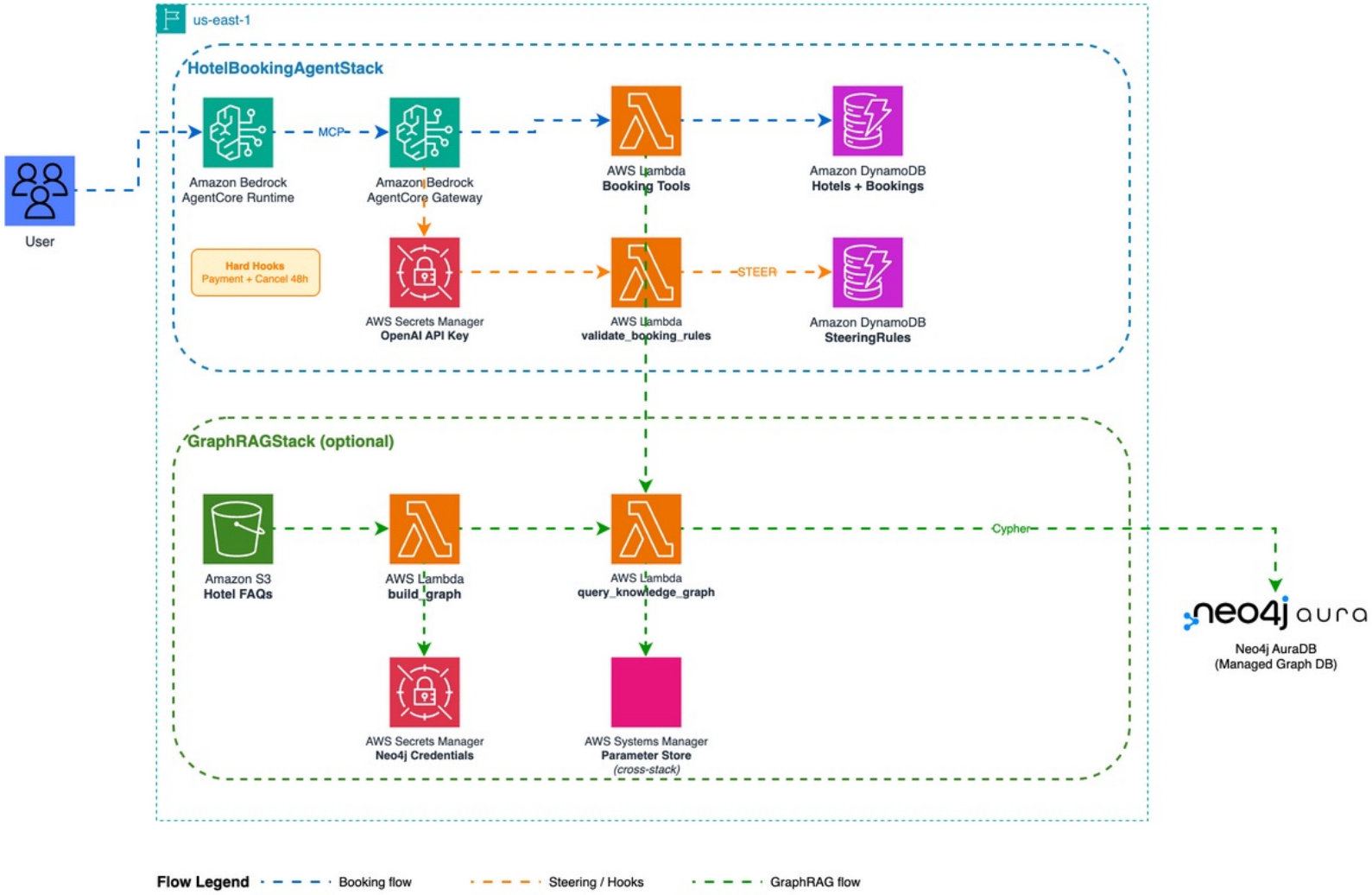
29 herramientas, conteo de tokens y precision en consultas identicas

Esa fue la demo local.

Como corre esto en produccion?

Notebook local	→	**AgentCore Runtime gestionado**
FAISS en memoria	→	**Busqueda semantica del Gateway por MCP**
print() statements	→	**CloudWatch + OpenTelemetry**
Sin guardrails	→	**Hooks a nivel de framework**

Produccion: busqueda semantica en AgentCore Gateway



FAISS local vs AgentCore Gateway

Aspecto	Local (esta demo)	Produccion (Gateway)
Filtrado	`FAISS + SentenceTransformer`	**`x_amz_bedrock_agentcore_search`**
Alojamiento	`Funciones Python en proceso`	**Funciones Lambda (auto-escala)**
Memoria	`swap_tools + agent.messages`	**Estado de sesion MCP**
Guardrails	`Ninguno`	**Hooks de framework (sin bypass)**
Observabilidad	`print() statements`	**OpenTelemetry + CloudWatch**
Mantenimiento del indice	`Lo construyes y refrescas tu`	**Gestionado por el Gateway**

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

Puntos clave

1. Filtra las herramientas antes del modelo con FAISS + SentenceTransformers
2. Menos herramientas en contexto reduce costo Y errores a la vez
3. 84.9% menos tokens, medido, sobre consultas identicas
4. Preserva la memoria de conversacion mientras intercambias herramientas por turno
5. En produccion, AgentCore Gateway hace la busqueda semantica por ti

Recursos

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

Gracias!

Elizabeth Fuentes Leone

Developer Advocate @ AWS

Blog y redes

elifuentes.tech

Recursos

bit.ly/4bVJUBw