Observabilidad aplicada a LLMs

¿Qué es la Observabilidad?

- Permite entender el estado interno de un sistema a basándose en los resultados
- Propiedad medible
- Percepción que tienen tanto usuarios como desarrolladores

¿Por qué?

- Problemas no medibles “Debuggeando”.
- Permite ver problemas sin cambiar código
- Simplifica las nuevas funcionalidades

Large Language Models (LLMs)

Perspectiva de usuario

- Modelo de la caja Negra
- Tarea → Información
- No se sabe lo que ocurre por dentro
- Extensas o concretas

Definición fundamental

Arquitectura transformacional

Transformar la experiencia del usuario y mejorar la calidad de vida

Problemas semánticos complejos computacionalmente

Se guardan en memoria

Avances

- Desde 2017
- Memoria más potente → Lenguaje más potente

Fases de desarrollo

TRAINING →
INFORMACIÓN

ALLINGMENT →
OBJETIVOS

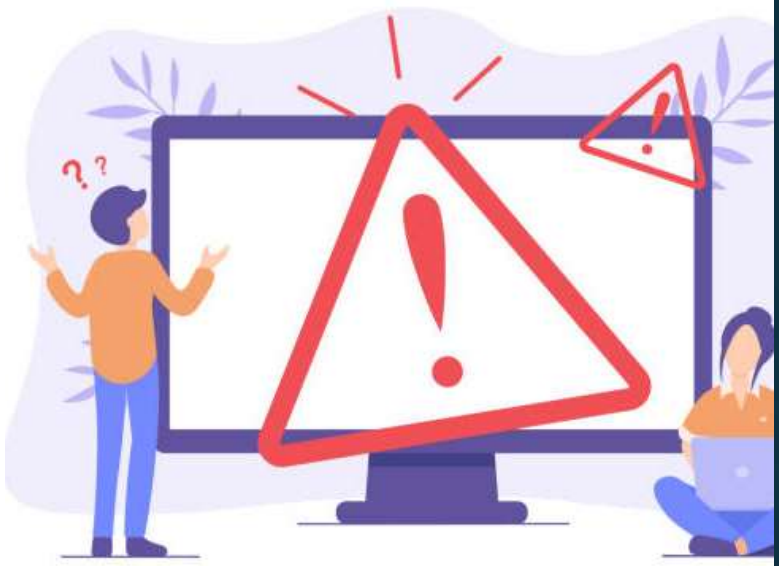
SIGUE HABIENDO
PROBLEMAS

FINE-TUNING

ESPECIALIZACIÓN CON
TU PROPIA INFORMACIÓN

Problemas en los LLMs

- Alta latencia
- Poca coherencia
- Estamos acostumbrados **AHORA**



Solución

- Fine-tuning
- Prompt engineeering
- Observability Driven Development (ODD)

Fine-tuning

Beneficios

- Especialización
- Muy potente

Problemas

- Demasiado profundo
- Debilita experiencia

Fine-tuning: Ejemplo

GPT-4: General

Copilot: Especializado en programación

Prompt engineering

- Experimentar con diferentes entradas
- No se modifica el modelo
- Inconveniente: prueba y error

Prompt engineering: Ejemplo

- "Escribe un código que ordene una lista."

- Demasiado general:
  - ¿Qué algoritmo va a usar?
  - ¿En qué lenguaje se va a usar?
  - El código puede estar bien estructurado o no

- "Genera un script en Python que implemente el algoritmo de ordenación rápida (*Quicksort*) para ordenar una lista de números enteros. El código debe incluir comentarios explicativos en cada paso y una función principal con un ejemplo de uso."

- Más específico:
  - Se especifica el algoritmo y el lenguaje a usar
  - Se piden comentarios explicativos

Observability Driven Development (ODD)

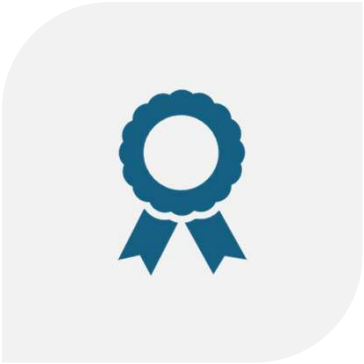
- Diseño proactivo
- Herramientas de observabilidad
  - OpenTelemetry
  - Honeycomb
  - Prometheus
- Desarrollo guiado por datos
- Reducción de la depuración a ciegas

El cambio en el proceso de desarrollo

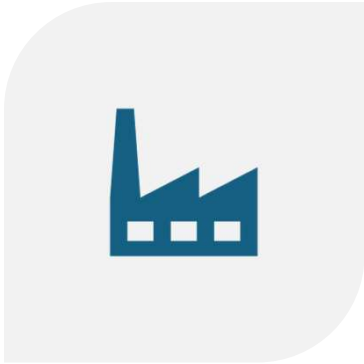
- Desarrollo tradicional y sus problemas respecto a la IA.
- Cambios en las empresas.
- El nuevo proceso de desarrollo.
- Observabilidad en el desarrollo.
- Observabilidad en la detección de errores.

Desarrollo tradicional del software

PRIMERO
FUNCIONALIDAD

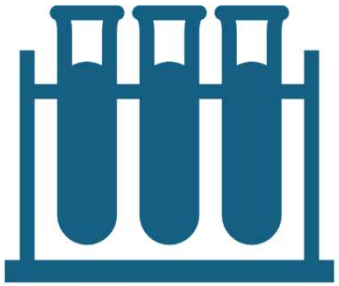


LUEGO QA



FINALMENTE
PRODUCCIÓN

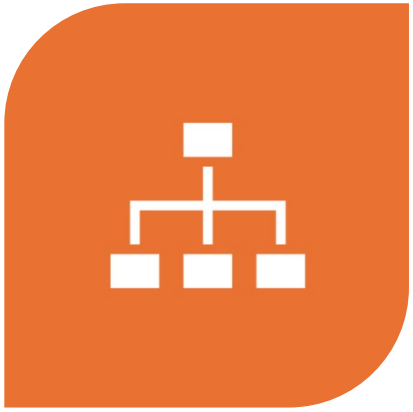
Problemas: ¿Por qué no es viable?



Pruebas de rendimiento

Pruebas básicas

Por otro lado, ¿Qué ha cambiado en las empresas?



DIRECTIVOS ENTUSIASMADOS.

EQUIPOS DE DESARROLLO.

El cambio en los equipos de desarrollo: El proceso ideal

Observar patrones diarios de uso.

Detectar errores y problemas de rendimiento.

Lanzar cambios pequeños cada día.

Monitoreo.

Herramientas clave (Objetivos de nivel de servicio - SLOs)

La importancia del feedback

Analisis del uso.

Identificación de necesidades.

Ajustado e integración de funcionalidades.

¿Cómo se implementa la observabilidad?

Registro de prompts, información y respuesta.

Inclusión de datos sobre decisiones en el tiempo.

Herramientas para Observabilidad

OpenTelemetry

15 minutos de instalación y dos días de implementación

Tipos de errores

Fallos críticos

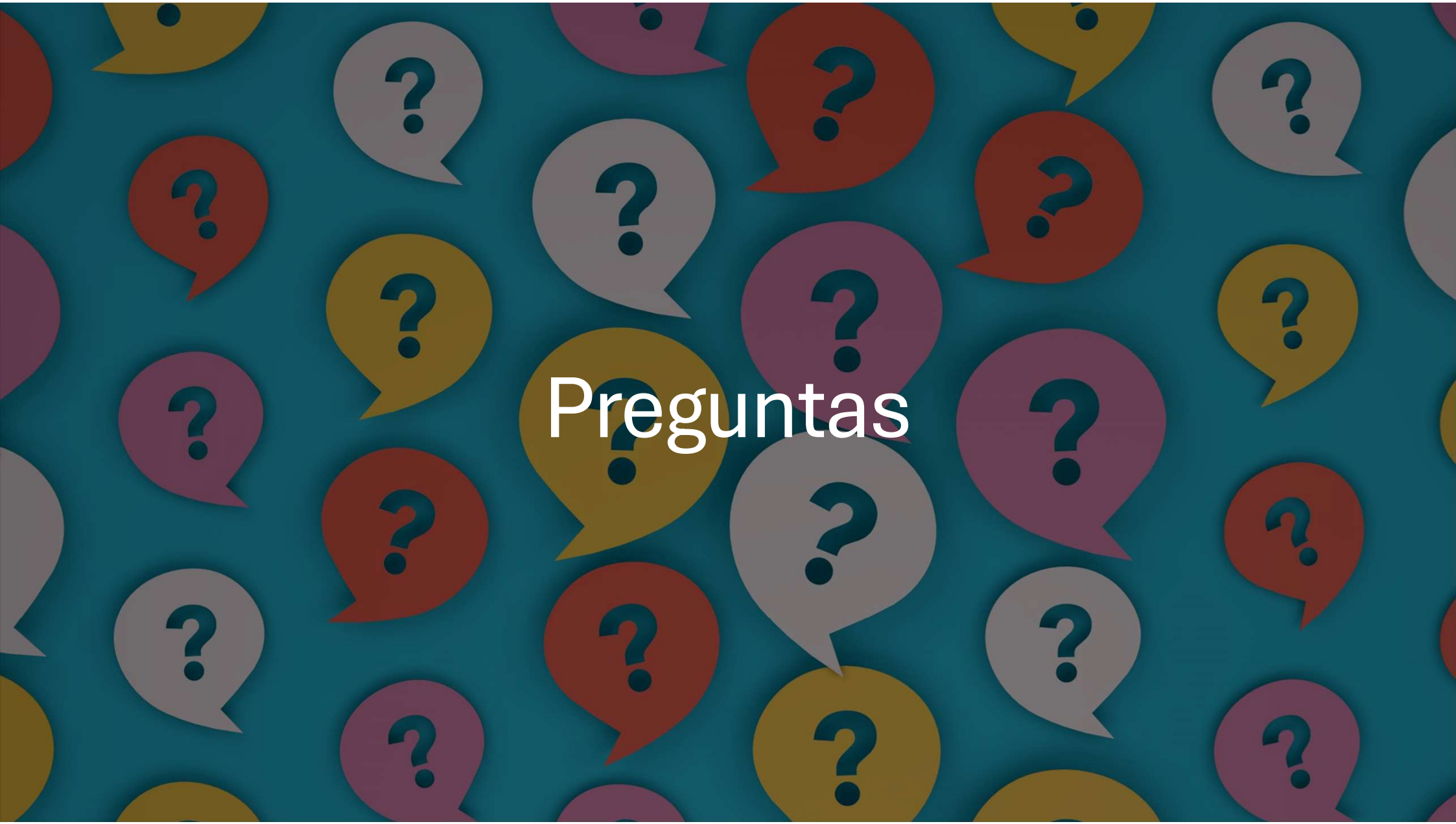
Errores corregibles.

Caso de Honeycomb

- Precisión de 65% a 90%

Desafíos en la observabilidad de modelos de lenguaje

- Falta de prácticas estandarizadas.
- Herramientas insuficientes.


Preguntas