EDICIÓN #018

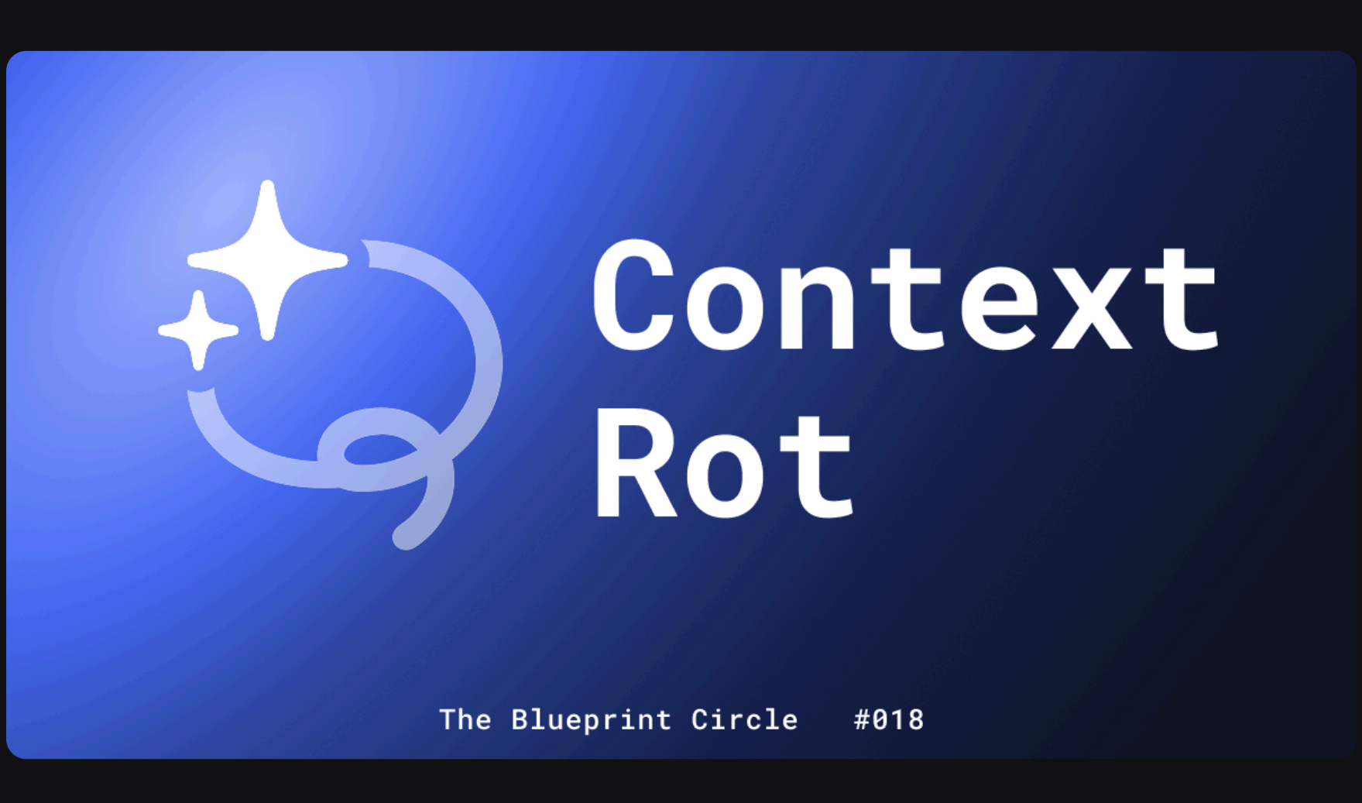
Context Rot: Más Tokens es... ¿peor?

25 de Enero de 2026

Context Rot: Más Tokens es... ¿peor?

Guía técnica sobre Context Rot: por qué los modelos fallan con mucha información y cómo usar el context engineering para mejorar la fiabilidad de tus aplicaciones con IA

25 January 2026



¿Alguna vez te has preguntado por qué ese script de RAG que funcionaba tan bien con tres documentos PDF empieza a alucinar cuando le metes toda la documentación técnica de tu empresa? **Nos han vendido la idea de que los contextos infinitos son la solución a todos nuestros problemas**. Primero fueron los 32k tokens, luego 128k, y ahora ya estamos hablando con total normalidad de ventanas de un millón o incluso diez millones de tokens con Gemini o las nuevas versiones de Llama. La promesa del marketing es que ya no necesitas vector databases, tíralo todo al prompt y que el modelo se apañe.

Eso que nos intentan vender suena muy bonito pero la realidad técnica es bastante menos brillante. En cuanto trasteas con configuraciones de contextos largos empiezas a darte cuenta que las respuestas se degradan, muchas instrucciones se dejan de tener en cuenta y te da la sensación que el modelo trabaja peor.

Acaba de salir un informe técnico de Chroma, bastante denso pero fascinante, que pone nombre y apellidos a este fenómeno: **Context Rot**. Y los experimentos que han hecho son preocupantes para cualquiera que esté construyendo aplicaciones sobre LLMs hoy en día.

Parece que tratar a los modelos de lenguaje como si fueran discos duros infinitos donde volcar información sin criterio tiene un coste de rendimiento muy grande. Hoy quiero desgranar este informe, explicarte por qué ocurre este deterioro y, lo más importante, qué es el famoso **Context Engineering** del que tanto habla **Andrej Karpathy** y por qué **vas a tener que aprenderlo** si quieres que tus aplicaciones funcionen en producción en 2026.

Resumen ejecutivo (TL;DR)

Este análisis explica por qué **aumentar el tamaño del contexto en los LLMs degrada su capacidad de razonamiento y recuperación de información**, un fenómeno conocido como **Context Rot**.

- El **rendimiento de los modelos no es uniforme**: procesar 100.000 tokens no es igual de fiable que procesar 1.000. La precisión cae drásticamente a medida que aumenta la longitud.
- Las pruebas estándar tipo **Needle in a Haystack** (buscar una aguja en un pajar) son un poco engañosas porque se basan en búsquedas exactas. Cuando la búsqueda requiere razonamiento semántico, los modelos fallan mucho más.
- Los **distractores** (información relacionada pero incorrecta) **hacen mucho más daño en contextos largos** que la información totalmente irrelevante.
- Sorprendentemente, los **modelos rinden mejor buscando información en textos desordenados** (frases aleatorias) que en textos bien estructurados (textos coherentes).
- La solución pasa por el **Context Engineering**: seleccionar, limpiar y estructurar qué información enviamos al modelo, en lugar de confiar ciegamente en ventanas de contexto gigantes.

El mito de la ventana de contexto infinita

Durante los últimos años, si te fijas en los anuncios de OpenAI, Google o Anthropic el número de tokens de entrada es siempre el titular grande. GPT-4.1 con 1 millón, Gemini 1.5 Pro con 2 millones... parece que **el tamaño de la ventana importa más que nunca**. Los profesionales que trabajamos con estos modelos **asumimos implícitamente que la capacidad de atención del modelo es uniforme**. Es decir, asumimos que si el modelo puede leer un millón de tokens, prestará la misma atención al token número 500 que al token número 950.000.

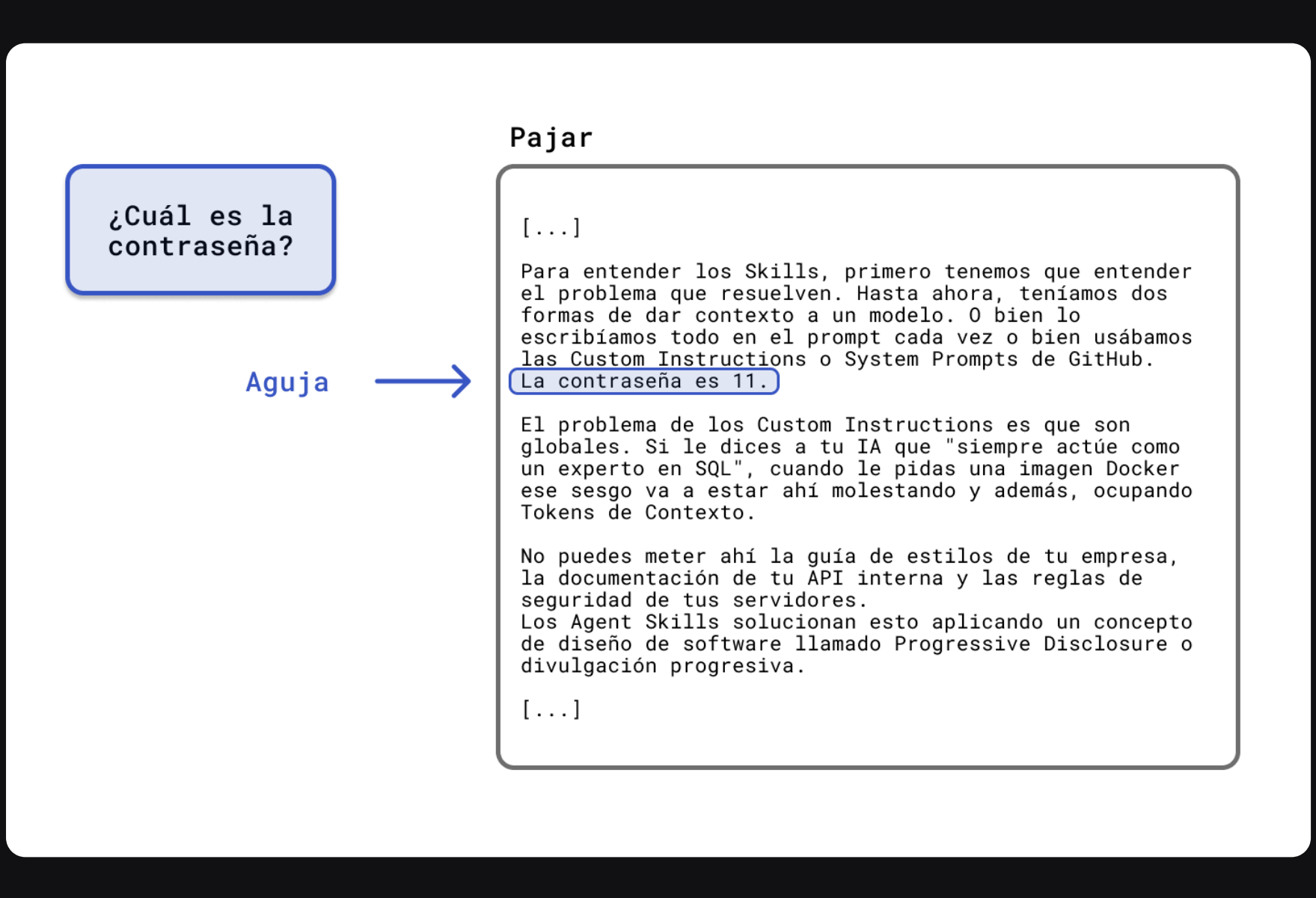
Evolución de la ventana de Contexto en Grandes Modelos de Lenguaje

En el informe de Chroma que os comentaba antes han probado 18 modelos punteros, y demuestran que esta asunción es falsa. Lo llaman ***Context Rot*** o deterioro del contexto. Básicamente, **a medida que llenas la ventana de contexto, la capacidad del modelo para realizar tareas simples no se mantiene estable, sino que decae**. Y no decae de forma lineal o predecible, sino que a veces se desploma.

Esto rompe el paradigma actual de no preocuparse por optimizar la ventana siempre y cuando quepa en el contexto. Si estás construyendo un asistente legal y le metes 500 páginas de un caso, la probabilidad de que alucine o pase por alto un detalle crucial aumenta exponencialmente con cada página extra que añades, aunque técnicamente te quepa en la ventana. **No es que el modelo no pueda leerlo, es que no puede procesarlo con la misma fidelidad**.

¿Qué es el “Needle in a Haystack" en los LLMs y por qué no es fiable?

Hasta ahora, la forma estándar de medir si un modelo aguantaba bien los contextos largos era una prueba llamada **Needle in a Haystack** o por sus siglas NIAH, en castellano sería como encontrar la aguja en el pajar. Esta prueba consiste en coger un dato aleatorio (lo que sería la aguja), como "*La contraseña secreta es '11'*", y lo escondes en medio de un texto enorme irrelevante (el pajar), por ejemplo cientos de informes financieros. Luego le preguntamos al modelo: *¿Cuál es la contraseña?*.



[...]

Para entender los Skills, primero tenemos que entender el problema que resuelven. Hasta ahora, teníamos dos formas de dar contexto a un modelo. O bien lo escribíamos todo en el prompt cada vez o bien usábamos las Custom Instructions o System Prompts de GitHub.

La contraseña es 11.

El problema de los Custom Instructions es que son globales. Si le dices a tu IA que "siempre actúe como un experto en SQL", cuando le pidas una imagen Docker ese sesgo va a estar ahí molestando y además, ocupando Tokens de Contexto.

No puedes meter ahí la guía de estilos de tu empresa, la documentación de tu API interna y las reglas de seguridad de tus servidores.
Los Agent Skills solucionan esto aplicando un concepto de diseño de software llamado Progressive Disclosure o divulgación progresiva.

[...]

Prueba de Needle in a Haystack para evaluar LLMs

Los modelos modernos sacan casi un 100% en esta prueba. Uno puede hacer alguna prueba, ver el resultado y pensar que el problema está resuelto pero aquí viene el problema que señala el informe y es que el ***Needle in a Haystack*** **es una prueba de recuperación léxica pero no semántica.** Es básicamente como hacer un Ctrl+F. El modelo solo tiene que hacer coincidir patrones de palabras exactas.

El equipo de Chroma decidió complicar las cosas. En lugar de pedir una coincidencia exacta, probaron con algo que llaman *agujas semánticas*. Por ejemplo, esconden la frase "Álvaro vive al lado del Museo del Prado". Y la pregunta no es "¿Dónde vive Álvaro?", sino "¿Quién ha estado en Madrid?". Para responder, el modelo tiene que saber (o deducir) que el museo del Prado está en Madrid y conectar los dos puntos.

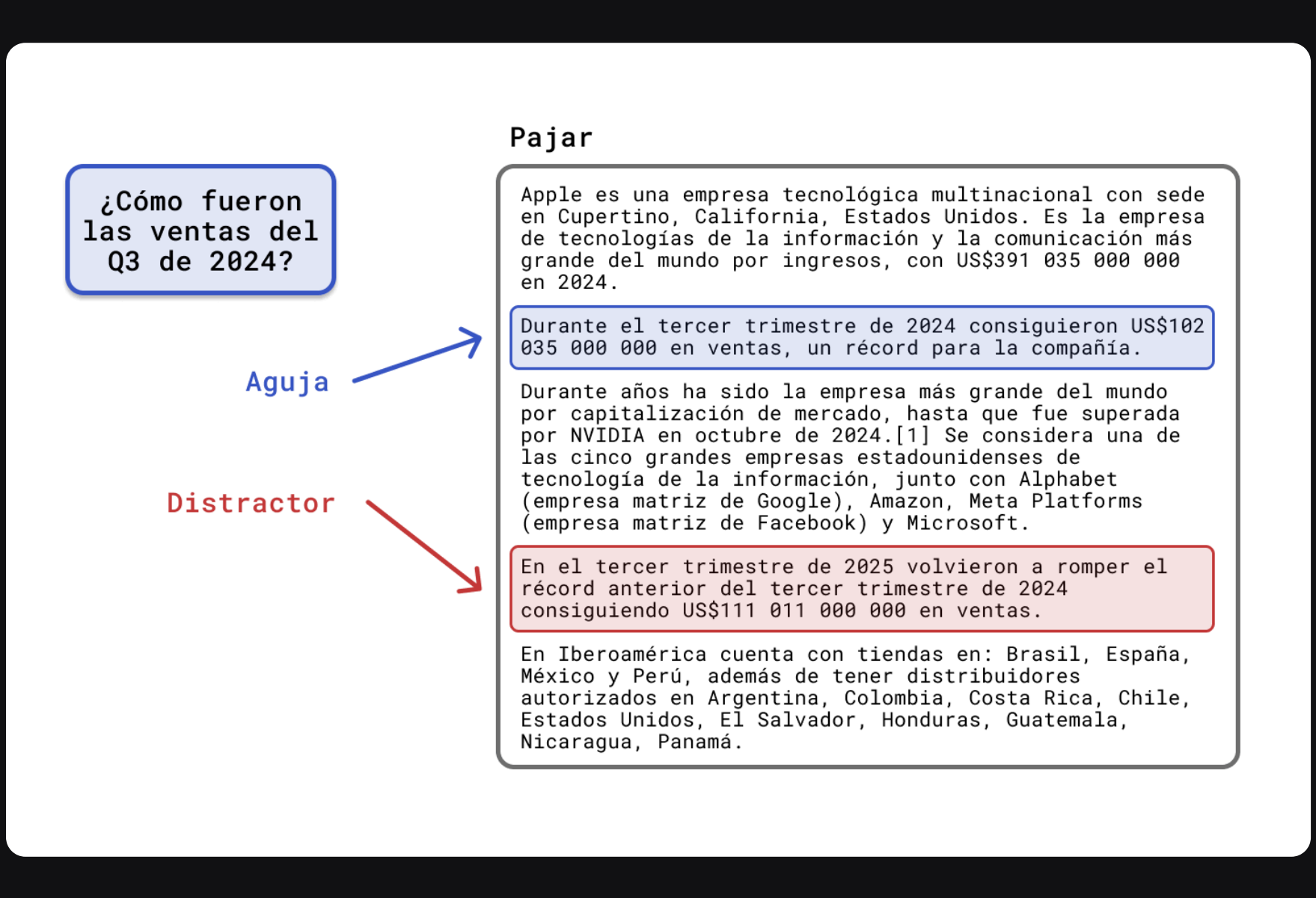
Cuando cambiamos la recuperación por razonamiento el rendimiento es mucho peor, os recomiendo leer el análisis de Chroma para ver ejemplos reales. **El rendimiento cae en picado a medida que aumenta el contexto**. Esto es gravísimo porque es así como usamos la IA en la vida real. Rara vez buscamos una palabra exacta, buscamos conceptos, ideas o respuestas que requieren sintetizar información dispersa. Si tu sistema de RAG depende de que el modelo entienda el matiz en un documento de 50 páginas, y solo has probado que sabe encontrar una palabra clave, te vas a llevar una sorpresa desagradable en producción cuando los usuarios lo prueben.

El peligro de los distractores y la información casi correcta

Otro punto muy bueno del estudio es **cómo manejan los modelos la información que está diseñada para confundir**. En un entorno de laboratorio, el pajar suele ser texto basura o irrelevante. Pero en tu base de datos vectorial o en tus documentos de empresa, tienes un montón de **información que se parece mucho a lo que buscas**, pero no es exactamente lo que buscas.

Imagina que preguntas por las ventas del Q3 de 2024. El documento correcto está ahí. Pero también tienes documentos sobre las ventas del Q3 de 2023, las proyecciones del Q4, y las ventas del Q3 de la competencia. A esto se le llama **distractores**: información semánticamente muy similar pero no igual.

El informe muestra que los distractores tienen un impacto no uniforme y devastador. **Si añades información que es temáticamente similar a la respuesta correcta pero que es errónea para la consulta específica, el modelo tiende a alucinar con mucha más frecuencia a medida que el contexto crece**.



Apple es una empresa tecnológica multinacional con sede en Cupertino, California, Estados Unidos. Es la empresa de tecnologías de la información y la comunicación más grande del mundo por ingresos, con US$391 035 000 000 en 2024.

Durante el tercer trimestre de 2024 consiguieron US$102 035 000 000 en ventas, un récord para la compañía.

Durante años ha sido la empresa más grande del mundo por capitalización de mercado, hasta que fue superada por NVIDIA en octubre de 2024.[1] Se considera una de las cinco grandes empresas estadounidenses de tecnología de la información, junto con Alphabet (empresa matriz de Google), Amazon, Meta Platforms (empresa matriz de Facebook) y Microsoft.

En el tercer trimestre de 2025 volvieron a romper el récord anterior del tercer trimestre de 2024 consiguiendo US$111 011 000 000 en ventas.

En Iberoamérica cuenta con tiendas en: Brasil, España, México y Perú, además de tener distribuidores autorizados en Argentina, Colombia, Costa Rica, Chile, Estados Unidos, El Salvador, Honduras, Guatemala, Nicaragua, Panamá.

Prueba de Needle in a Haystack con distracciones para evaluar LLMs

Lo curioso es ver cómo reaccionan las distintas familias de modelos. Los modelos de la familia Claude como **Opus y Sonnet tienden a ser más conservadores**. Cuando se ven abrumados por distractores en un contexto largo, suelen optar por decir "que no encuentran la respuesta o abstenerse. Es un comportamiento defensivo que, sinceramente, prefiero en entornos críticos.

Por otro lado, la familia **GPT tiende a ser más atrevida** y ante la duda provocada por los distractores y el contexto largo, es más probable que intenten adivinar o inventarse una respuesta mezclando la información correcta con la incorrecta. Esto es lo que llamamos **alucinación inducida por contexto**.

Para un desarrollador, esto significa que la limpieza de datos antes de enviarlos al modelo (el **pre-procesamiento** en tu pipeline de RAG) **es mucho más crítica de lo que pensábamos**. No puedes confiar en que el modelo discierna el grano de la paja si hay demasiada paja que parece grano.

¿Debo organizar mis datos antes de dárselos al LLM? ¿Por qué el caos funciona mejor?

En una de las pruebas, los investigadores compararon dos tipos de pajar para esconder la información:

1. **Pajar Estructurado:** Una colección de ensayos y papers coherentes, con su introducción, desarrollo, conclusión y flujo lógico natural.
2. **Pajar Barajado:** Las mismas frases de los ensayos, pero mezcladas aleatoriamente, rompiendo cualquier coherencia narrativa o lógica: un sinsentido.

La intuición nos dice que **el modelo debería rendir mejor en el texto estructurado**. Al fin y al cabo, estos modelos se entrenan para entender el lenguaje humano, la gramática y el flujo de las ideas.

Pues bien, resulta que **los modelos rinden mejor encontrando información en el pajar barajado, el texto caótico y sin sentido**, que en el texto bien escrito y estructurado.

¿Cómo es posible? La hipótesis (y esto es especulación técnica basada en cómo funcionan los Transformers) es que **la estructura narrativa distrae al mecanismo de atención del modelo**. Cuando el texto tiene sentido y flujo, el modelo dedica recursos a seguir ese hilo argumental, a predecir la siguiente palabra basándose en la lógica del discurso. La información insertada (la aguja) puede quedar diluida o interpretada como una digresión dentro de esa narrativa fuerte.

En cambio, cuando el texto es una combinación de frases aleatorias, no hay ningún hilo que seguir. El mecanismo de atención no se engancha a ninguna estructura global, por lo que **la aguja destaca más o es más fácil de recuperar** aisladamente porque no compite con una narrativa coherente. Es como si fuera más fácil encontrar una pelota roja en una piscina de bolas grises desordenadas que encontrar una frase específica en una novela bien escrita donde esa frase podría tener un doble sentido o ser parte de una metáfora.

Esto tiene implicaciones interesantes para cómo diseñamos nuestros prompts. A veces nos esforzamos muchísimo en darle al modelo un contexto super ordenado, con formato bonito y narrativo, y quizás, solo quizás, **para tareas de pura extracción, estaríamos mejor pasándole una lista de hechos atómicos desconectados**.

Si queréis profundizar en el experimento en el que barajaron los textos os dejo el anchor a la parte del estudio.

La prueba de las palabras repetidas: cuando copiar es difícil

Para demostrar que esto no es solo un problema de comprensión compleja, sino algo fundamental en la arquitectura de los LLMs actuales, hicieron una prueba que a priori podría ser un poco absurda por su simplicidad que consistió en **pedirle al modelo que repita una secuencia de palabras**.

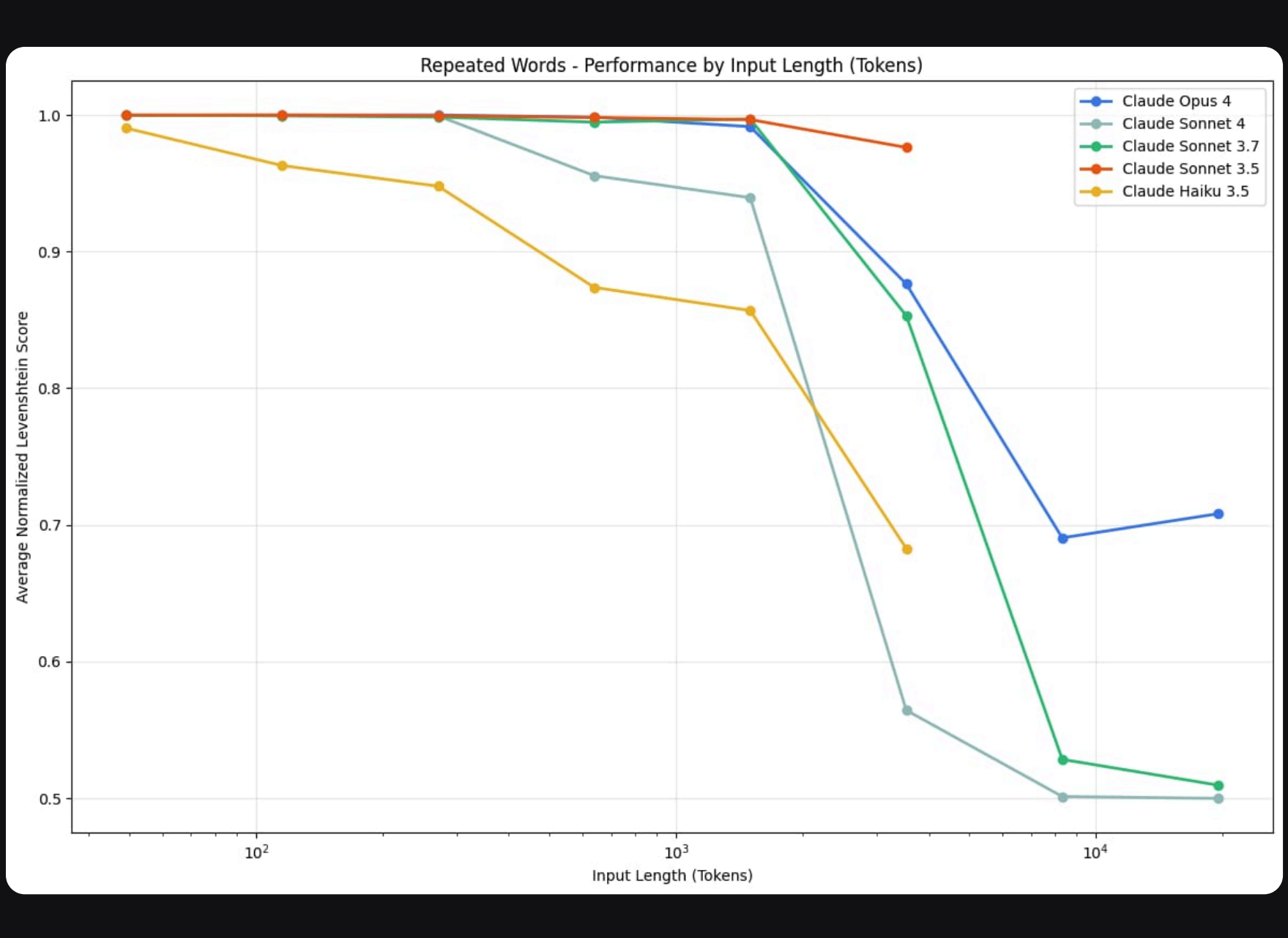
El prompt fue:

```
Simplemente replica el siguiente texto, emite exactamente el mismo texto: manzana
```

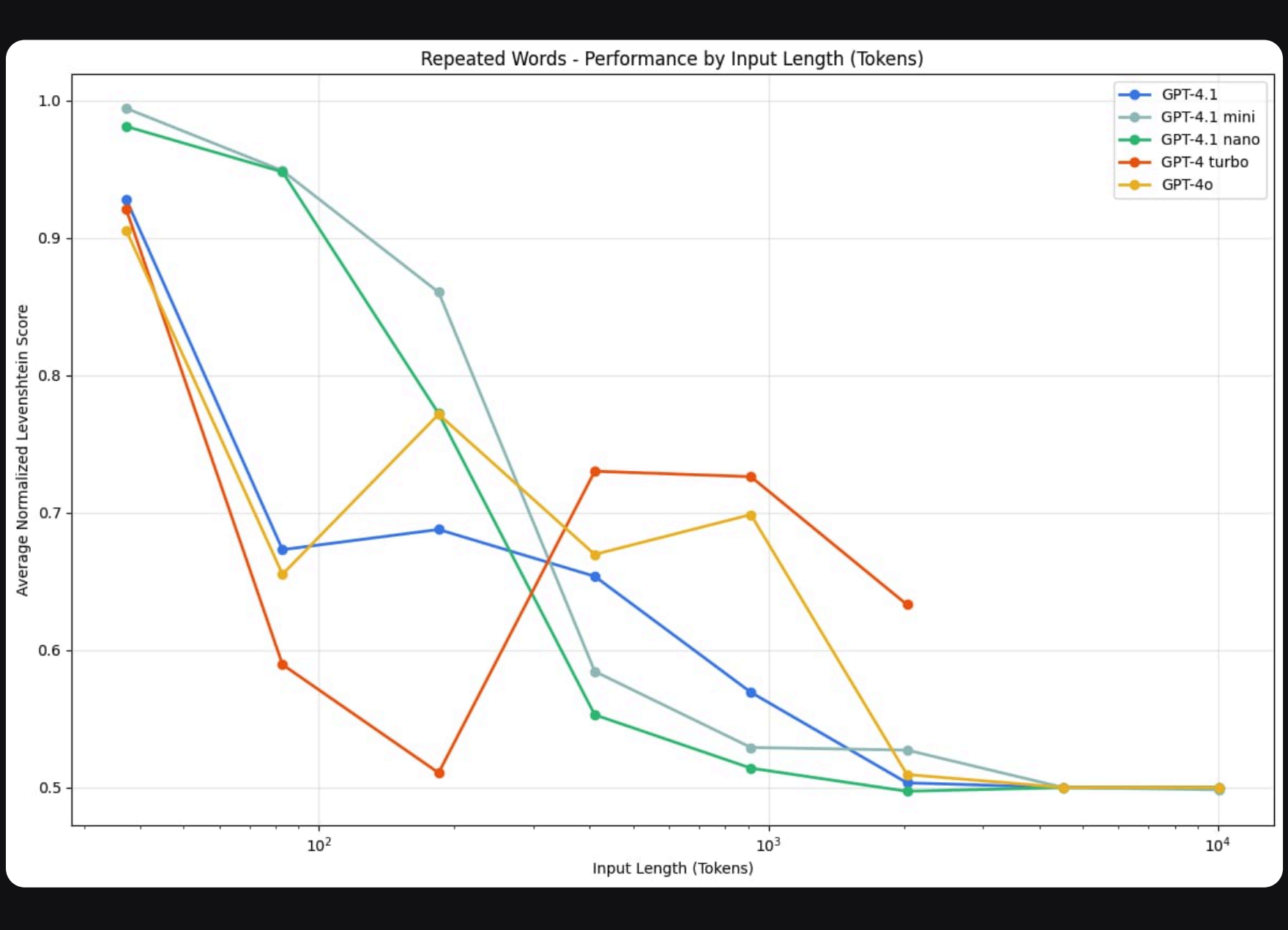
Uno pensaría que esto es trivial. Es una máquina. Copiar y pegar es lo que hacen los ordenadores. Pero para un LLM, que genera token a token probabilísticamente, esto no se vuelve tan fácil en contextos largos. Fijaros que una de las palabras es “manzanas” en lugar de “manzana”.

Repitieron el experimento con 25, 50, 75, 100, 250, 500, 750, 1000, 2500, 5000, 7500, 10000 palabras. A medida que aumentaba la longitud de la secuencia a repetir, **todos los modelos empezaban a fallar.** Algunos se olvidaban de la palabra única (manzanas), otros empezaban a repetir patrones extraños, y otros simplemente se rompían y empezaban a escupir basura o se negaban a continuar.

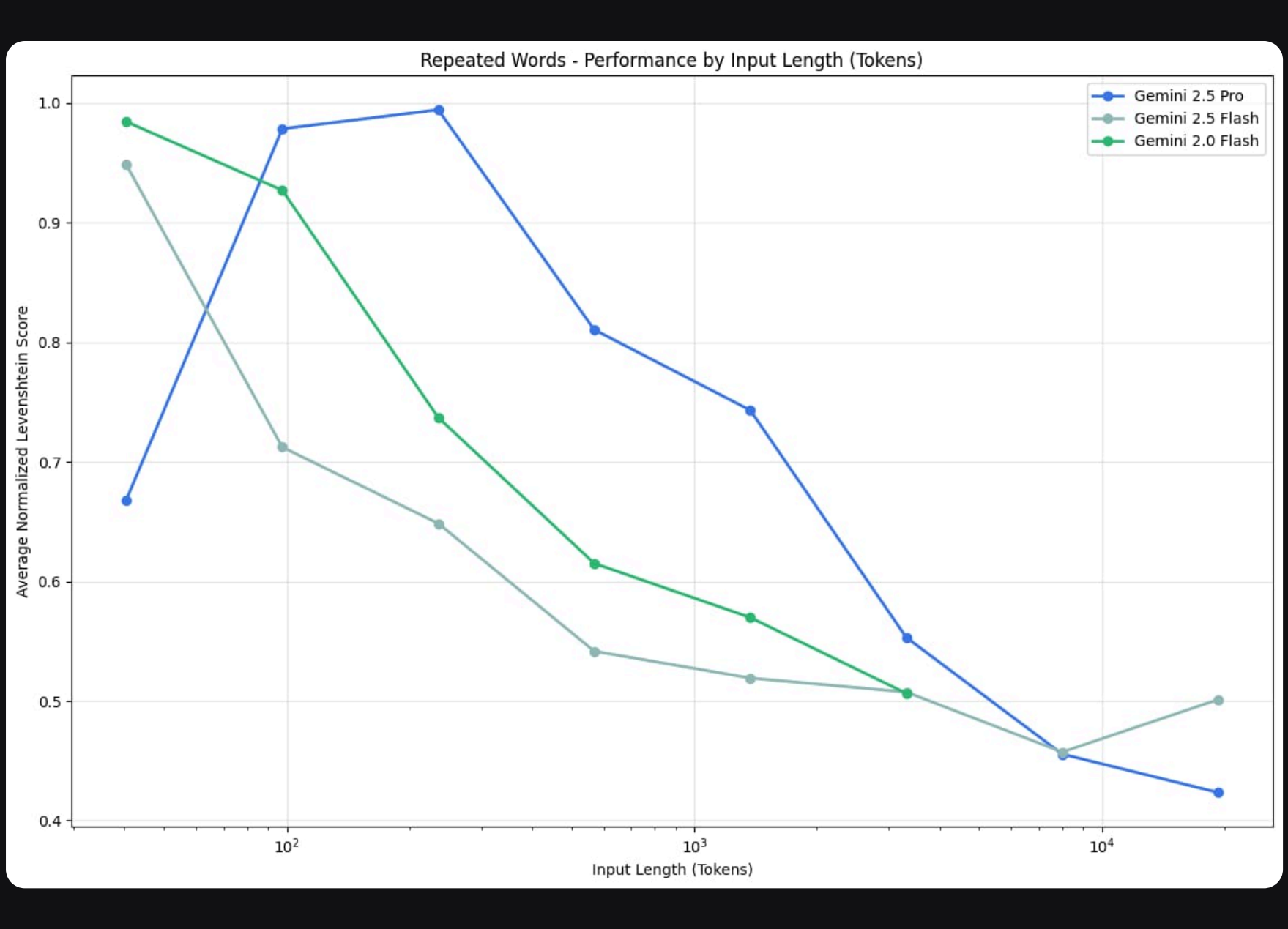
Os dejo los resultados extraídos del estudio de Chroma para los modelos de Claude, OpenAI y Google:



Resultados del experimento de palabras repetidas en LLM empleando modelos de Claude



Resultados del experimento de palabras repetidas en LLM empleando modelos de OpenAI



Resultados del experimento de palabras repetidas en LLM empleando modelos de Google

En mi caso, me ha pasado algo similar con Gemini y GPT al intentar generar JSONs muy largos. Llega un punto en el que el modelo olvida la estructura rígida y empieza a divagar y comete errores de sintaxis. El informe muestra que modelos como GPT-4 Turbo tienen picos de rendimiento extraños: funcionan bien con 50 palabras, fallan con 250, mejoran con 500 y vuelven a caer. **No es lineal**, aunque con modelos más nuevos como Claude Sonnet 4.5 esto ocurre cada vez menos.

Esto nos recuerda que, por muy avanzados que sean, **estos modelos no tienen una memoria RAM fiable como un programa tradicional**. Su memoria es una ventana de atención probabilística y difusa. Si tu aplicación depende de que el modelo mantenga una coherencia exacta sobre una salida de 10.000 palabras basándose en una entrada de 50.000, deberías buscar otra solución.

Context Engineering: La nueva habilidad imprescindible

Visto todo esto, queda claro que la estrategia de dejarlo todo a la suerte de la ventana de contexto no es una solución real que funcione en escala. Estamos entrando en la era de la **Context Engineering** o Ingeniería de Contexto. Si el Prompt Engineering era sobre cómo pedir las cosas, **el Context Engineering es sobre qué información le das al modelo y cómo se la estructuras**.

No basta con recuperar la información que necesita el modelo y soltarla al contexto. Tienes que pensar en:

1. **Densidad de información:** Eliminar la paja es más importante que nunca. Si puedes resumir un documento antes de meterlo en el contexto, hazlo. Menos tokens de mayor calidad siempre ganan a muchos tokens de baja calidad.
2. **Posición de la información:** Hay un concepto llamado *Lost in the Middle* que nos dice que la información en el medio del contexto se pierde más que la del principio o el final. El informe de Chroma refuerza que la estructura importa. Colocar las instrucciones críticas y los datos más relevantes al final del prompt (**lo más cerca posible de la generación de respuesta**) sigue siendo la mejor práctica.
3. **Filtrado de distractores:** El sistema de recuperación (el *Retriever* de tu RAG) tiene que ser mucho más preciso. Usar técnicas de *Re-ranking* (volver a ordenar los resultados de búsqueda con un modelo más pequeño pero preciso) ya es algo necesario si quieres evitar el **Context Rot**. Necesitas asegurarte de que lo que entra en el prompt es relevante de verdad, no solo parecido.

También es interesante ver cómo **los modelos que tienen modos de pensamiento o razonamiento extendido mitigan parte de este problema**, pero no lo solucionan del todo. Además para ciertos problemas seguramente no queramos incurrir en el gasto de un modelo razonador pudiendo usar uno que no lo es.

¿Qué hacemos entonces en 2026?

De cara a este año y al siguiente, mi apuesta es que vamos a ver un cambio de tendencia. **Dejaremos de obsesionarnos tanto con el tamaño bruto de la ventana de contexto y empezaremos a valorar más la gestión inteligente de ese contexto**.

Tendremos que organizar y seleccionar el contexto dinámicamente. El Context Rot del que te hablaba la semana pasada en el blog es algo muy nimportante. Los modelos se ‘cansan’, se confunden y se ‘aburren’ si les das demasiada información irrelevante. Si quieres buenos resultados dales la información masticada y limpia, y verás cómo ese modelo que rendía regular puede funcionar mejor.

Referencias técnicas

Aquí os dejo los enlaces directos al informe y a los recursos que menciono, por si queréis profundizar en las gráficas y la metodología, que merece mucho la pena.

1. Context Rot: How Increasing Input Tokens Impacts LLM Performance - Informe técnico oficial de Chroma (Julio 2025) con todas las gráficas y metodología detallada.
2. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory - Paper académico sobre evaluación de memoria a largo plazo en asistentes conversacionales.
3. NoLiMa: Long-Context Evaluation Beyond Literal Matching - Estudio sobre las limitaciones de la coincidencia literal en contextos largos.
4. Repositorio de Chroma - Base de datos vectorial Open Source utilizada para la investigación y fundamental para implementar RAG con filtrado avanzado.
5. Lost in the Middle: How Language Models Use Long Contexts - Paper fundacional de la Universidad de Stanford que explica por qué la posición de la información en el prompt es crítica.