El precipicio era la ventana de contexto

Un recorte silencioso del prompt que parecía un límite de herramientas

Ricardo Martínez

Cloud Security & Applied AI

Noviembre 2026

Resumen

Un modelo de pesos abiertos de 9.7B, detrás de un servidor de inferencia local, pasó una prueba de humo con una herramienta y después no llamó ninguna herramienta al conectarlo a un agente con 24 esquemas. Al barrer el número de herramientas pareció haber un precipicio entre 8 y 16. No lo había. La petición del agente pesaba 14,095 tokens, la ventana por defecto del servidor era de 4,096, y el servidor recortó el prompt a unos 2,050 tokens sin avisarle al cliente, descartando el inicio del prompt, que es donde van los esquemas de las herramientas. La petición había pedido una ventana de 262,144 tokens mediante una opción que el endpoint compatible con OpenAI no respetó. Con la ventana en 32,768, la misma petición produjo una llamada correcta en 10 de 10 intentos con muestreo por defecto, y el barrido quedó plano en 10 de 10 de 1 a 24 herramientas. Bajar la temperatura no cambió nada en ninguna de las dos ventanas. Un control sintético lo confirma: una sola herramienta con el prompt alargado al tamaño de 24 falla, y acierta, exactamente donde lo hace la petición de 24. Antes de concluir que un modelo pequeño no sabe usar herramientas, revisa el contexto efectivo.

1 Qué pasó

Un modelo de pesos abiertos (Qwen3.5 9B, Q4_K_M, 9.7B parámetros) servido con Ollama 0.33.2 en una sola máquina con 17.8 GiB de VRAM se conectó a un framework de agentes de código abierto por el endpoint compatible con OpenAI. Al pedirle que corriera un script, el agente respondió en prosa: escribió el comando en un bloque de código y describió lo que haría, sin llamar la herramienta de terminal que venía en su carga. Una prueba directa con una sola herramienta y el mismo modelo sí pasó.

Había tres explicaciones plausibles. El modelo es demasiado pequeño para usar herramientas. Los valores de muestreo por defecto (temperatura 1, penalización de presencia 1.5) lo hacen improvisar. O 24 herramientas son demasiadas para él. Una variante con temperatura 0.1 y sin penalización de presencia no cambió el comportamiento del agente.

2 Capturar la petición

Apuntamos el framework a un proxy local que guarda el cuerpo de cada petición y responde con texto fijo, de modo que no se ejecutó nada. La petición traía 24 esquemas de herramientas, un prompt de sistema de 18,536 caracteres, un mensaje de usuario de 20 tokens y `options.num_ctx = 262144`. Contada en un servidor con ventana suficiente para no recortar, pesaba 14,095 tokens: 9,686 de herramientas, 4,372 de instrucciones del agente, 20 del mensaje y 17 de plantilla de chat (Figura 1).

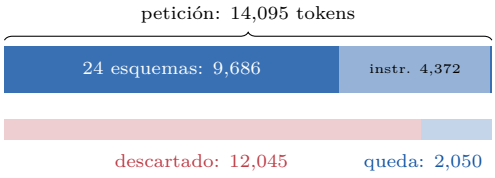


Figura 1: La petición capturada con la ventana por defecto de 4,096 tokens (a escala). El servidor conserva los primeros tokens y la cola, y los esquemas de herramientas van al inicio.

3 Reproducirla

Reprodujimos la petición capturada contra el mismo modelo, diez intentos por condición (Tabla 1).

Ventana	Muestreo	Herramientas	Tokens	Correctas	IC 95%
4,096	por defecto	24	2,050	0/10	0–31%
4,096	temp. baja	24	2,050	0/10	0–31%
4,096	por defecto	1	1,317	10/10	69–100%
4,096	temp. baja	1	1,317	10/10	69–100%
32,768	por defecto	24	14,095	10/10	69–100%
32,768	temp. baja	24	14,095	9/10	55–100%
32,768	por defecto	1	1,317	10/10	69–100%
32,768	temp. baja	1	1,317	10/10	69–100%

Tabla 1: Mismo modelo, misma petición, dos ventanas del servidor. “Llamada correcta” = llamada a la herramienta correcta con el comando correcto.

Con la ventana por defecto del servidor, de 4,096 tokens, el modelo no hizo ninguna llamada correcta en 20 de 20 intentos, con muestreo por defecto y con temperatura baja. Con 32,768 hizo una en 10 de 10 y en 9 de 10 (el único fallo llamó otra herramienta). La petición con una herramienta pasó en las cuatro condiciones, incluido el muestreo por defecto, así que la impresión previa de que bajar la temperatura la rescataba no se reprodujo con intentos repetidos.

El servidor explica la diferencia. Con 4,096 registró `truncating input prompt` con un límite de 2,050 para un prompt de 14,095 tokens, 50 veces durante la corrida, y ninguna con 32,768. La respuesta no traía error, y su campo de uso reportaba 2,050 tokens de prompt.

Una sonda muestra qué sobrevivió. Con 4,096, al pedirle que repitiera dos marcadores puestos al inicio y al final del mensaje de sistema, el modelo dijo no ver el primero y leyó el segundo; al pedirle que listara sus herramientas, listó nombres de skills que viven en el prompt de sistema, no las funciones. Con 32,768 leyó los dos marcadores y listó los 24 nombres de función. La plantilla de chat del modelo parece poner el bloque de herramientas antes del texto de sistema (no imprimimos el prompt renderizado), lo que encaja con la sonda: el corte quita el principio, así que los esquemas se van primero.

4 Parece un límite de herramientas

La Figura 2 barre el número de herramientas con un prompt de sistema corto, usando subconjuntos de los 24 esquemas reales.

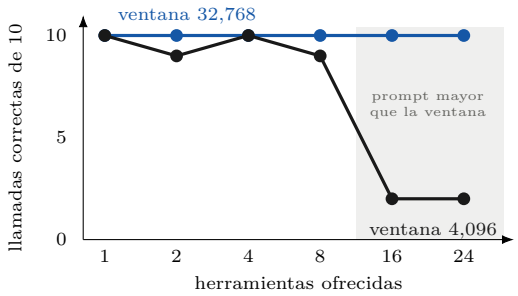


Figura 2: Llamadas correctas de 10 con cada número de herramientas, mismo modelo, mismos subconjuntos, mismos prompts. Solo cambia la ventana del servidor. Con 4,096 la curva cae entre 8 y 16 herramientas, justo donde el prompt empieza a exceder la ventana; con 32,768 es plana.

Quien se hubiera detenido en la curva oscura habría reportado un precipicio de herramientas. La curva azul es el mismo experimento con una sola configuración distinta.

5 Control: herramientas contra tamaño

El barrido anterior cambia a la vez el número de herramientas y el tamaño del prompt. Para separarlos, un arnés público sin datos privados construye herramientas sintéticas de tamaño realista (unos 420 tokens cada una) y compara dos brazos con el mismo total de tokens. El brazo A ofrece k herramientas con un prompt de sistema corto. El brazo B ofrece una sola herramienta y alarga el prompt de sistema con texto neutro hasta el tamaño del brazo A (Tabla 2).

Con 4,096 los dos brazos se comportan igual: 10 de 10 mientras el prompt cabe (hasta 3,243 tokens) y 0 de 10 cuando no cabe (6,594 y 9,943 tokens, recortados a 2,050). En esas celdas fallidas el modelo escribió el comando como texto en 34 de 40 intentos e hizo una llamada a herramienta en solo 2 de 40, ninguna correcta. Con 32,768 todas las celdas son 10 de 10. Una herramienta es tan vulnerable como 24 cuando el prompt rebasa la ventana, y 24 herramientas son tan seguras como una mientras no la rebasa. Lo que importa es el tamaño del prompt contra la ventana; las herramientas solo importan por ocupar espacio.

Herram.	Tokens	A, 4,096	B, 4,096	A, 32,768	B, 32,768
4	1,566	10/10	10/10	10/10	10/10
8	3,243	10/10	10/10	10/10	10/10
16	6,594	0/10	0/10	10/10	10/10
24	9,943	0/10	0/10	10/10	10/10

Tabla 2: Control sintético con el mismo total de tokens. A: k herramientas. B: una herramienta, prompt de sistema alargado al tamaño de A. Llamadas correctas de 10.

6 Mecanismo y arreglo

La documentación de Ollama da longitudes de contexto por defecto según la VRAM (4k por debajo de 24 GiB) y dice que la API de OpenAI no tiene forma de fijar el tamaño de contexto de un modelo, por lo que recomienda un Modelfile con `PARAMETER num_ctx` [5, 6]. Nuestro servidor registró una ventana de 4,096 para una petición que traía `num_ctx = 262144` en `options`. Una variante por Modelfile con `num_ctx 32768` en un servidor por lo demás por defecto devolvió una llamada correcta en 5 de 5 intentos y sin avisos de recorte. La documentación recomienda al menos 64,000 tokens para agentes; esta sola carga ocupa 14,095 antes de cualquier conversación.

7 Qué revisar primero

- Compara `usage.prompt_tokens` de la respuesta con lo que enviaste. Con 4,096 el servidor reportó 2,050 para una petición de 14,095 tokens y no devolvió error.
- Busca `truncating input prompt` en el log del servidor y lee la columna `CONTEXT` de `ollama ps`.
- Si pasas por `/v1`, fija la ventana en el modelo (Modelfile), no en las opciones de la petición.
- Certifica el uso de herramientas con la carga de producción, no con la prueba de humo. Pasar con una herramienta es evidencia sobre una herramienta.

8 Trabajo relacionado

La degradación por número de herramientas está documentada. En el benchmark ReAct de LangChain, GPT-4o cayó a 2% en agenda de calendario con siete dominios y Llama-3.3-70B no llamó una herramienta requerida en ningún caso [1]. Maxim AI reporta que reducir un conjunto de herramientas de GitHub y Notion de 48 a 25 mejoró a cuatro de cinco modelos, con Claude Sonnet 4 pasando de 66.67% a 73.33% [2]. Anthropic reporta a Claude Opus 4.5 pasando de 79.5% a 88.1% en una evaluación interna de MCP cuando las herramientas se descubren bajo demanda, una cifra del proveedor [3]. Un preprint de mayo de 2026 reporta a Phi-4 14B restaurado de 0% a 84.4% con 20 herramientas al compilar los esquemas, en el benchmark del propio autor [4]. Esos estudios miden la selección bajo un contexto creciente. Nuestro caso es otro mecanismo: las herramientas nunca llegaron al modelo. No revisamos si alguna corrida citada sufrió recorte; reportar la ventana efectiva permitiría a los lectores saberlo.

Límites

Un modelo (Qwen3.5 9B, Q4_K_M), una versión de servidor (Ollama 0.33.2), una máquina, una tarea (correr un comando de shell), diez intentos por celda. En el control, el relleno va antes de la instrucción y no probamos relleno después de ella; las herramientas sintéticas son genéricas y la herramienta objetivo queda a dos tercios de la lista. Con diez intentos, 0 de 10 y 10 de 10 se distinguen (intervalos exactos de 95% de 0–31% y 69–100%); las diferencias pequeñas no. El esfuerzo de razonamiento fue medio, como en la petición capturada. No registramos la configuración original del servidor el día de la falla: la reproducción da las mismas respuestas en prosa con la ventana por defecto, lo que es consistente con que las corridas originales se recortaran igual, pero no lo prueba. Las dos páginas de Ollama no coinciden en el valor por defecto (según la VRAM contra 4,096); nuestro servidor registró 4,096 con 17.8 GiB. Una versión anterior de este repositorio llamó a esto un precipicio de herramientas y un colapso categórico de los modelos pequeños. Esa afirmación era incorrecta, y estas mediciones la reemplazan.

Referencias

[1] LangChain. *Benchmarking Single Agent Performance*. 10 de febrero de 2025. blog.langchain.com/react-agent-benchmarking.

[2] Maxim AI. *How We're Measuring Model-Tool Accuracy in the Age of MCP*. Julio de 2025. getmaxim.ai/blog.

[3] Anthropic. *Introducing advanced tool use on the Claude Developer Platform*. 24 de noviembre de 2025. anthropic.com/engineering/advanced-tool-use.

[4] F. Sakizli. *TSCG: Deterministic Tool-Schema Compilation for Agentic LLM Deployments*. arXiv:2605.04107, mayo de 2026.

[5] Ollama. *Context length*. docs.ollama.com/context-length.

[6] Ollama. *OpenAI compatibility*. docs.ollama.com/api/openai-compatibility.