Facsímil 04

La caja de herramientas

Capítulo 01

Elegir la intervención correcta: prompt, RAG, tool o ajuste

686f6c61 · 2026

Capítulo 01: Elegir la intervención correcta: prompt, RAG, tool o ajuste

La herramienta no es el punto de partida

El error más frecuente en IA aplicada es elegir la herramienta antes que el problema: «hagamos RAG», «probemos fine-tuning», «pongamos un agente», «llevémoslo a local». En ingeniería ese salto tiene nombre: es resolver sin haber hecho el análisis de requisitos. Una solución elegida antes del diagnóstico no es una decisión, es una preferencia.

Una herramienta es la respuesta a un síntoma concreto, y cada síntoma apunta a una capa distinta del sistema. Si el modelo no respeta un formato, falla el contrato de salida: salida estructurada. Si no conoce la normativa interna, falta evidencia: RAG. Si debe consultar stock real, falta estado externo: una tool. Si responde bien pero con criterio inestable en miles de casos repetidos, falla la conducta aprendida: ajuste. Si los datos no pueden salir de una máquina, manda el entorno de despliegue: local. La herramienta no se elige por su prestigio, sino por la capa que repara.

Este capítulo, por tanto, se ordena alrededor de una sola pregunta de diseño: **¿qué parte del sistema falla y cuál es el cambio mínimo que la corrige con la mejor relación entre mejora medible y coste de operación?**

Estado del arte con fecha de corte

Fecha de corte: 25 de mayo de 2026.
Fuentes consultadas ese día: documentación oficial de Anthropic sobre prompt engineering, OpenAI sobre salidas estructuradas y function calling, Hugging Face PEFT/LoRA, documentación de Ollama API y Cloud, y el paper original de RAG.

Esta foto cambia rápido. Lo estable no es el nombre de una herramienta concreta, sino el patrón de decisión: contexto, evidencia, schema, tool, ajuste, local, evaluación y coste.

En 2026, las plataformas de modelos ya no ofrecen solo “texto dentro, texto fuera”. Las APIs modernas permiten schemas, tool calls, streaming, batch, cache y distintos modelos para coste o calidad. OpenAI documenta salidas estructuradas para forzar que una respuesta siga un schema.[1] Anthropic mantiene guías de prompt engineering como primera capa de control antes de añadir arquitectura.[2]

También se consolidó una separación práctica: RAG para conocimiento externo, PEFT/LoRA para adaptar comportamiento con pocos parámetros y modelos locales o cloud local-compatible cuando la restricción está en privacidad, coste, portabilidad o experimentación.[3] Ollama documenta una API local y una variante cloud con una experiencia parecida para modelos que no caben en hardware personal.[4][5]

A 2026 hay además un consenso práctico sobre el orden por defecto: prompt y salida estructurada como primera capa de control, RAG como forma habitual de aportar conocimiento que cambia, tools para estado real y cálculo, y ajuste de pesos solo cuando una conducta repetida no se estabiliza de otra forma. Los flujos agentic (varios pasos, herramientas y memoria) ganan terreno para tareas complejas, pero no cambian la regla de fondo: no se empieza por la pieza más cara, se llega a ella cuando una evaluación lo justifica. La novedad de los últimos años no es una herramienta nueva que sustituya a las demás, sino mejores contratos entre capas (schemas, tool calls tipados, trazas) que hacen que combinar varias sea más fiable y más fácil de depurar.

Qué no es una caja de herramientas de IA

Una caja de herramientas no es una lista de marcas. Si no sabes qué síntoma resuelve cada pieza, el catálogo solo añade ruido. La persona que sabe usar herramientas no es quien instala más librerías, sino quien simplifica antes de complicar.

Tampoco es una escalera de prestigio. Prompt no es “poco serio” y fine-tuning no es “más profesional” por defecto. RAG no es superior a una consulta SQL si la pregunta exige datos exactos. Un modelo local no es mejor que una API si no cumple calidad o mantenimiento. Cada elección compra algo y paga algo.

Y una caja de herramientas no sustituye a evaluación. Puedes montar RAG, tool calling, LoRA y modelo local; si no tienes casos de prueba, no sabes si has mejorado o solo has construido una máquina más difícil de entender.

La pregunta correcta: qué quieres cambiar

Antes de elegir intervención, separa cuatro planos. El primero es **entrada**: quizá el modelo ya puede hacer la tarea si le das mejores instrucciones, ejemplos y formato esperado. El segundo es **contexto**: quizá necesita documentos, datos o memoria externa. El tercero es **acción**: quizá debe consultar una API, calcular algo o escribir en un sistema. El cuarto es **comportamiento aprendido**: quizá quieres que responda de una forma estable en una tarea repetida.

Esa separación evita mezclar herramientas:

SI EL PROBLEMA ES...	PRIMERA INTERVENCIÓN RAZONABLE	POR QUÉ
Formato irregular	Prompt claro y salida estructurada	El modelo sabe responder, pero falta contrato de salida.
Conocimiento vivo	RAG o consulta a fuente externa	El dato cambia y debe poder citarse o verificarse.
Cálculo, estado o acción	Tool con schema y permisos	El modelo no debe inventar resultados de sistemas externos.
Tono o taxonomía repetida	Ejemplos, luego fine-tuning si escala	Cambias comportamiento estable, no conocimiento vivo.
Privacidad u offline	Modelo local o entorno privado	La restricción vive en dónde se ejecuta.
Coste o latencia	Modelo más pequeño, cache, batch o local	El problema puede estar en serving, no en “inteligencia”.

Para verlo con un caso cercano: una universidad quiere contestar dudas de matrícula. Si las reglas cambian cada curso, RAG o consulta a la base oficial. Si el problema es devolver siempre JSON para integrarlo en una app, salida estructurada. Si debe comprobar si una persona ya pagó, tool contra el sistema de pagos. Si cada respuesta debe tener tono institucional, ejemplos y quizá ajuste más adelante.

El criterio de decisión: utilidad esperada y valor de la información

Elegir una intervención bajo incertidumbre no es una corazonada: es un problema clásico de teoría de la decisión, y conviene usar su formalismo en lugar de una regla casera.

El primer pilar es la **utilidad esperada**. Una decisión racional elige la acción que maximiza la utilidad esperada sobre los resultados posibles, formalizada por von Neumann y Morgenstern.[6]

$$\mathbb{E}[U(a)] = \sum_{s} P(s)\,U(a, s)$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
a	Acción o intervención que evaluamos.	Añadir RAG, una tool o un schema.
s	Estado o resultado posible del sistema.	El dato estaba vigente o no, el formato se valida o no.
$P(s)$	Probabilidad de cada resultado.	Estimada con casos reales, no supuesta.
$U(a, s)$	Utilidad de la acción si ocurre s.	Respuesta correcta y citada frente a respuesta inventada.
$\mathbb{E}[U(a)]$	Utilidad esperada de la acción.	Promedio ponderado de los resultados.

La regla de decisión es directa: adoptas una intervención frente al baseline solo si su utilidad esperada, descontado el coste total, supera a la de no hacer nada.

$$\mathbb{E}[U(a)] - K(a) > \mathbb{E}[U(\text{baseline})]$$

Aquí $K(a)$ agrega el coste real de operar la intervención: económico, latencia, mantenimiento y riesgo. Esto evita la mala costumbre de contar solo la mejora y olvidar lo que habrá que sostener después.

El segundo pilar explica por qué RAG, las tools o cualquier recuperación de evidencia tienen sentido. No cambian el modelo: le dan información antes de decidir. La teoría del **valor de la información** de Howard cuantifica cuánto vale esa evidencia.[7]

$$\mathrm{VoI}(e) = \mathbb{E}_{e}\Big[\max_{a} \mathbb{E}[U(a) \mid e]\Big] - \max_{a} \mathbb{E}[U(a)] \geq 0$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
e	Evidencia que se recupera antes de responder.	El fragmento de normativa vigente, el stock real.
$\mathbb{E}[U(a) \mid e]$	Utilidad esperada de la acción dada esa evidencia.	Decidir con el documento delante.
$\mathrm{VoI}(e)$	Valor de la información de esa evidencia.	Cuánto mejora la decisión por tenerla.

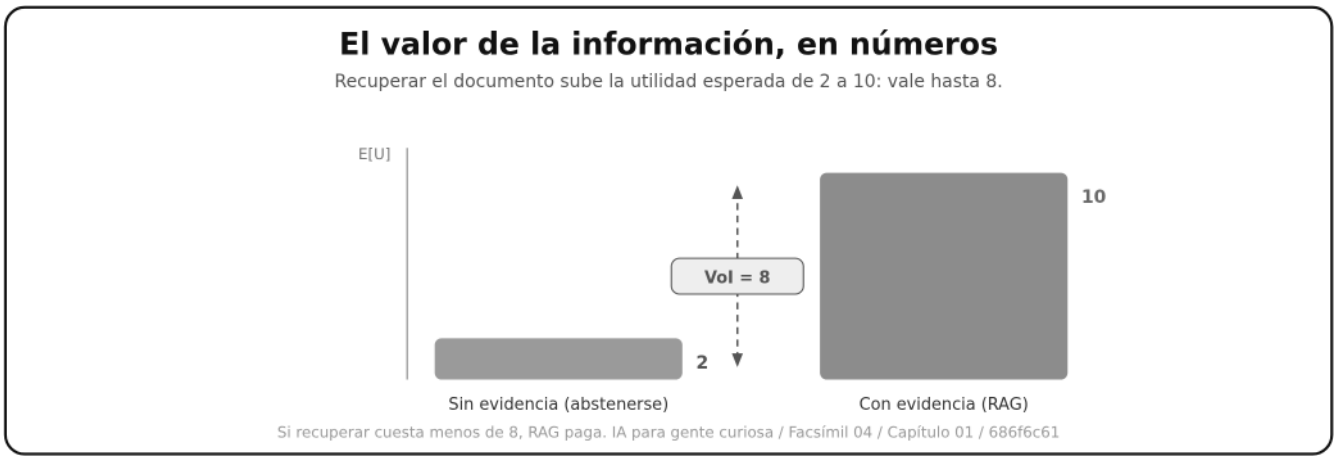
En palabras: añadir RAG o una tool solo merece la pena si la evidencia que aportan **cambia lo que el sistema haría**. Si la respuesta es la misma con o sin ese documento, su valor de información es cero, aunque la recuperación funcione a la perfección. Por eso el diagnóstico no pregunta «¿puedo recuperar algo?», sino «¿hay una decisión que esa evidencia modifique?».

Veámoslo con números. Un asistente debe decidir si una solicitud encaja en una política que puede decir «aprobar» o «denegar». Sin leer el documento, su creencia a priori es $P(\text{aprobar}) = 0{,}6$ y $P(\text{denegar}) = 0{,}4$. Las utilidades del caso: acertar vale $+10$, equivocarse -20 (una respuesta de política errónea es cara) y abstenerse para escalar a una persona vale $+2$.

ACCIÓN SIN EVIDENCIA	UTILIDAD ESPERADA
Decir «aprobar»	$0{,}6 \cdot 10 + 0{,}4 \cdot (-20) = -2$
Decir «denegar»	$0{,}6 \cdot (-20) + 0{,}4 \cdot 10 = -8$
Abstenerse y escalar	$+2$

Sin evidencia, la mejor acción es **abstenerse**, con $\mathbb{E}[U] = 2$. Ahora recuperamos el documento, que revela el estado real: con él, el asistente siempre elige la acción correcta y obtiene $+10$ en cualquiera de los dos estados, así que $\mathbb{E}[U \mid e] = 10$. El valor de la información de ese documento es:

$$\mathrm{VoI}(e) = 10 - 2 = 8$$



Ese 8 es el techo de lo que deberías estar dispuesto a pagar por recuperar: si el coste de RAG (tokens, latencia, mantener el índice) es menor que 8, la intervención paga; si es mayor, no. Y observa el otro extremo: si equivocarse costara solo -1 en vez de -20, responder «aprobar» sin leer nada ya daría $0{,}6 \cdot 10 + 0{,}4 \cdot (-1) = 5{,}6$, mejor que abstenerse, y la evidencia apenas cambiaría la decisión. El valor de la información cae cuando el coste de equivocarse es bajo. Esa es la diferencia entre añadir RAG porque toca y añadirlo porque mueve una decisión cara.

La fórmula no pretende dar una verdad exacta. Pretende impedir una mala costumbre: contar solo la mejora y olvidar todo lo que tendrás que operar después.

Las cinco intervenciones básicas

Primero hay que explicar el síntoma. Luego la herramienta empieza a tener sentido. Estas cinco piezas vuelven durante todo el facsímil.

La tabla siguiente es una síntesis didáctica, no una taxonomía oficial única. Se apoya en prompt engineering[8] para controlar tarea, contexto y ejemplos; en salidas estructuradas[9] para imponer contratos de respuesta; en RAG[10] para añadir evidencia recuperada; en function calling[11] para conectar el modelo con funciones externas; y en LoRA/PEFT[12] para adaptar comportamiento mediante pocos parámetros entrenables.

INTERVENCIÓN	CAMBIA	SIRVE CUANDO	NO SIRVE PARA
Prompt y ejemplos	La entrada	Falta claridad, formato o criterio de respuesta.	Datos vivos grandes o acciones externas.
Salida estructurada	El contrato de salida	Necesitas JSON, campos obligatorios o validación automática.	Saber más contenido.
RAG	El contexto recuperado	Hay documentos, políticas o conocimiento cambiante.	Cambiar el estilo profundo del modelo.
Tool	La capacidad de consultar o actuar	Hay estado real, cálculo, base de datos o sistema externo.	Sustituir permisos o validación.
Fine-tuning/LoRA	Algunos pesos o adaptadores	Tarea repetida, estable y medible.	Actualizar información que cambia cada día.

RAG nació como una forma de combinar memoria paramétrica con recuperación externa en tareas intensivas en conocimiento.[13] LoRA propuso adaptar modelos grandes entrenando matrices pequeñas de bajo rango.[14] QLoRA redujo aún más memoria al ajustar sobre modelos cuantizados.[15]

Para entenderlo antes de tocar código

Antes de escribir una línea de código conviene hacer una prueba mental muy simple: describe el fallo como lo vería una persona usuaria, no como lo nombraría una librería. Después tradúcelo a una capa del sistema. Esa traducción es el músculo que queremos entrenar en este capítulo.

La misma frase “el modelo falla” puede significar cinco cosas distintas. Puede fallar porque no recibió instrucciones claras, porque le falta evidencia, porque necesita consultar un sistema externo, porque debe repetir un criterio muy específico o porque el entorno de despliegue impone límites. Si llamas a todo “modelo malo”, acabarás cambiando piezas equivocadas.

CASO	SEÑAL OBSERVABLE	DIAGNÓSTICO	PRIMERA INTERVENCIÓN	QUÉ MEDIR
Asesoría con normativa cambiante	Responde bien hasta que cambia una norma.	Falta evidencia viva y trazable.	RAG o consulta a fuente oficial con cita.	Porcentaje de respuestas con documento correcto y fecha vigente.
CRM que necesita campos exactos	El texto es bueno, pero el backend rompe al parsear.	Falta contrato de salida.	Salida estructurada con schema y validación.	Campos válidos, errores de schema y casos que requieren revisión.
Tienda con stock real	Preguntan por talla, precio o disponibilidad.	Hace falta estado externo.	Tool pequeña contra inventario o pedidos.	Exactitud del dato, latencia de consulta y manejo de “no disponible”.
Soporte con miles de tickets parecidos	Responde con criterio parecido, pero formato y tono varían.	Conducta repetida poco estable.	Prompt con ejemplos; si escala, SFT, LoRA o adapter.	Consistencia de rúbrica, coste por ticket y regresiones.
Equipo con documentos sensibles	La mejor API externa no puede recibir ciertos textos.	Restricción de entorno.	Modelo local, entorno privado o arquitectura híbrida.	Calidad mínima aceptable, latencia, coste operativo y trazabilidad.

Fíjate en el tercer caso. Si una persona pregunta “¿quedan zapatillas talla 42?”, el modelo puede escribir una respuesta convincente, pero no conoce el almacén. La intervención correcta no es pedirle que “razone mejor”. Es darle una función con un contrato estrecho: `consultar_stock(producto, talla, tienda)`. La inteligencia del sistema no está solo en el modelo; está en saber cuándo el modelo debe dejar de completar texto y pedir un dato.

Ahora mira el primer caso. Si la asesoría trabaja con normas que cambian, ajustar pesos puede incluso empeorar la situación: convierte conocimiento vivo en memoria opaca. RAG no entra porque sea una palabra de moda, sino porque necesitamos tres cosas muy concretas: recuperar el fragmento vigente, citarlo y poder actualizarlo sin entrenar de nuevo.

El caso del CRM enseña otra lección. A veces el modelo “entiende” perfectamente la tarea, pero el producto necesita datos que se puedan validar. Ahí no queremos literatura: queremos `{categoria, prioridad, siguiente_paso, confianza}` y una regla clara para rechazar respuestas inválidas. La salida estructurada no mejora el mundo interno del modelo; mejora el contrato entre el modelo y el software que lo rodea.

Y el equipo de soporte muestra cuándo empieza a tener sentido ajustar. Si el conocimiento ya está resuelto, los documentos aparecen bien y el schema se cumple, pero la respuesta no respeta una rúbrica interna en miles de ejemplos parecidos, entonces sí: quizá toca entrenar una adaptación pequeña. Pero esa decisión llega después de medir, no antes.

Una regla útil: **si puedes arreglarlo cambiando entrada, contexto o contrato, no empieces cambiando pesos**. Los pesos se tocan cuando quieres estabilizar una conducta repetida, tienes ejemplos buenos, sabes medir el resultado y aceptas mantener otra versión del sistema.

Criterios de elección: la matriz que decide contigo

Si esto fuera una asignatura universitaria, aquí no bastaría con decir “depende”. El “depende” tiene que descomponerse en variables observables. Una buena decisión técnica no es la que suena más moderna, sino la que explica qué restricción pesa más y qué evidencia aceptaríamos para cambiar de opinión.

Esta es la forma práctica de la utilidad esperada cuando hay varios criterios en juego: el **análisis multicriterio**. Su versión más simple, el Weighted Sum Model, puntúa cada intervención como la suma de sus valoraciones por criterio, ponderadas por la importancia de cada criterio.[16][17]

$$\text{score}(a) = \sum_{j} w_j \, r_{aj}$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
a	Intervención evaluada.	RAG, tool, ajuste, local.
j	Criterio de decisión.	Conocimiento vivo, coste, privacidad.
w_j	Peso o importancia del criterio j.	Cuánto pesa la privacidad en tu caso.
r_{aj}	Valoración de la intervención a en el criterio j.	Del 0 al 5.
$\text{score}(a)$	Puntuación total de la intervención.	La que ordena el ranking.

Es justo lo que practicas en el cuaderno del facsímil: defines los pesos w_j de tu caso y dejas que el modelo ordene. Su límite académico también conviene decirlo: el Weighted Sum Model solo es válido si los criterios son comparables y están en la misma escala; si un criterio es de naturaleza distinta (por ejemplo, un requisito legal que no se puede compensar con coste), no entra en la suma, sino que actúa como restricción dura.

La siguiente matriz sirve para discutir una intervención en clase, en un equipo o en una revisión de arquitectura. No da una respuesta automática, pero obliga a justificarla.

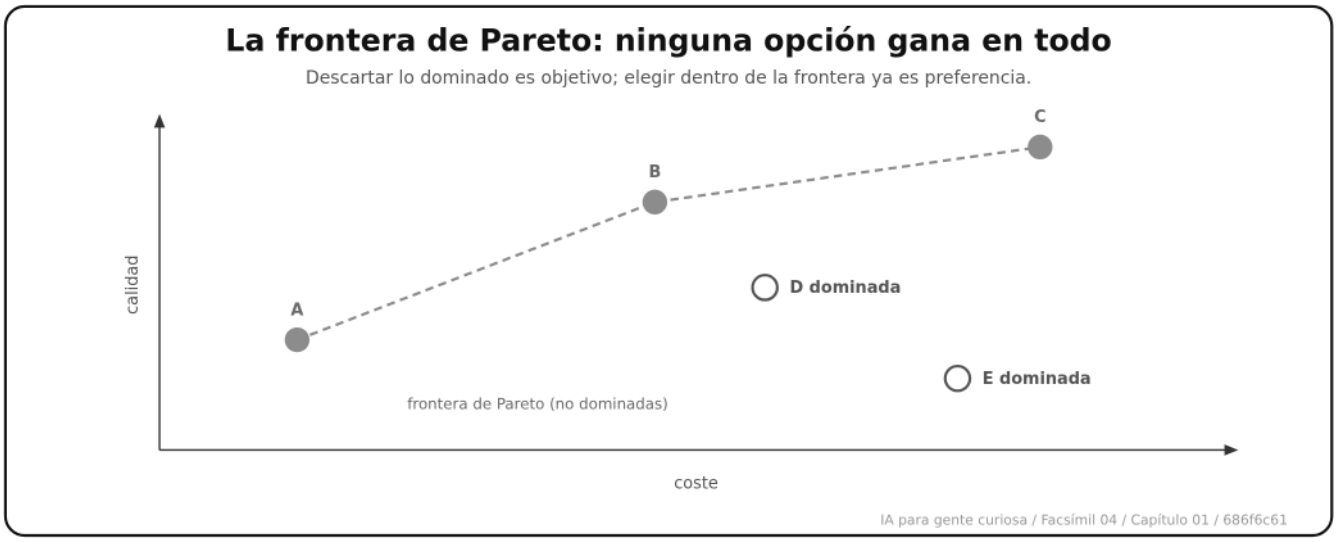
CRITERIO	PREGUNTA QUE DEBES HACER	SI PESA MUCHO, SUELE EMPUJAR HACIA...	EVIDENCIA MÍNIMA
Conocimiento cambiante	¿La respuesta depende de datos que cambian con frecuencia?	RAG, base de datos o tool.	Documentos con fecha, fuente y casos donde el dato cambia.
Necesidad de cita	¿La persona debe poder revisar de dónde sale la respuesta?	RAG con citas o consulta verificable.	Porcentaje de respuestas con evidencia correcta.
Estado real	¿Hace falta mirar inventario, pagos, agenda, expediente o cálculo externo?	Tool con schema estrecho.	Función definida, entrada validada y salida comprobable.
Formato estricto	¿Otro software consume la respuesta?	Salida estructurada.	JSON válido, campos obligatorios y tests de schema.
Conducta repetida	¿La misma tarea aparece miles de veces con criterios estables?	Prompt con ejemplos; si no basta, SFT, LoRA o adapter.	Dataset pequeño pero limpio, rúbrica y comparación contra baseline.
Privacidad o despliegue	¿Dónde puede ejecutarse el sistema y dónde pueden vivir los datos?	Modelo local, entorno privado o arquitectura híbrida.	Política de datos, latencia aceptable y calidad mínima medida.
Coste y latencia	¿El problema es calidad o servirlo sin arruinar la experiencia?	Modelo menor, cache, batch, cuantización o local.	Coste por consulta, TTFT, tokens/s y percentiles de latencia.
Reversibilidad	¿Podemos deshacer el cambio si empeora?	Prompt, schema, RAG o tool antes que ajuste de pesos.	Plan de rollback y comparación antes/después.

Esta matriz también evita discusiones tramposas. Si alguien propone fine-tuning para un problema de stock, puedes preguntar: “¿qué criterio de la tabla justifica cambiar pesos?”. Si nadie puede responder, todavía no hay diagnóstico.

La frontera de Pareto: cuando ninguna opción gana en todo

El Weighted Sum Model da un único ganador porque colapsa todo en un número. Pero esconde una decisión: los pesos. A veces no existe una opción que domine a las demás en todos los criterios; una es más barata, otra más precisa. Para esos casos la herramienta correcta no es la suma ponderada, sino la **frontera de Pareto**, el concepto central de la optimización multiobjetivo.[18]

Una opción está **dominada** si otra es al menos igual de buena en todos los criterios y estrictamente mejor en alguno. Las opciones que nadie domina forman la frontera de Pareto. Lo útil es que descartar lo dominado no exige elegir pesos: es objetivo. Solo cuando te quedas en la frontera aparece el juicio, porque allí toda mejora en un criterio cuesta empeorar otro.



En el dibujo, B domina a D: es más barata y de más calidad, así que D no debería ni discutirse. A, B y C, en cambio, son defendibles: A es la más barata, C la de más calidad y B el punto intermedio. Elegir entre ellas es declarar cuánto vale para ti un punto extra de calidad. La frontera no decide, pero limpia la mesa de las opciones que nadie debería elegir.

Un mismo caso, cinco soluciones posibles

Tomemos un caso único para no perdernos: una universidad quiere un asistente de matrícula. El objetivo superficial parece uno solo, “responder dudas”, pero debajo hay varios problemas distintos. Según cuál duela, la intervención cambia.

LECTURA DEL PROBLEMA	SOLUCIÓN RAZONABLE	QUÉ MEJORA	QUÉ NO ARREGLA
El alumnado pregunta de formas muy distintas y el sistema responde desordenado.	Prompt con ejemplos y criterios de estilo.	Claridad, tono, estructura inicial.	Normativa nueva o datos personales.
La app necesita guardar la respuesta en una ficha.	Salida estructurada con campos como `tema`, `respuesta`, `fuente`, `confianza`.	Integración con software y validación.	Saber si la norma está vigente.
Las normas de matrícula cambian cada curso.	RAG sobre normativa oficial, con fecha y cita.	Respuestas trazables y actualizables.	Consultar si una persona concreta pagó.
El alumno pregunta “¿me falta pagar algo?”.	Tool contra el sistema académico o de pagos.	Estado real de esa persona.	Explicar bien una norma general.
El equipo responde miles de tickets con una rúbrica estable.	Ajuste ligero si prompt, schema y RAG ya no bastan.	Consistencia en una tarea repetida.	Conocimiento que cambia cada semana.

Lo importante es que ninguna fila “gana” siempre. En un producto real quizá uses varias: RAG para normativa, tool para expediente, salida estructurada para integrar y prompt para tono. Pero las añades por capas, no por entusiasmo.

Combinar intervenciones: el orden importa

Un sistema real casi nunca usa una sola intervención. Un asistente de atención puede tener prompt cuidado, salida estructurada para integrar, RAG para políticas y una tool para consultar pedidos. La pregunta no es «¿cuál uso?», sino «¿en qué orden las añado y cómo sé que cada capa ayuda?».

La regla práctica es añadir de barata a cara, y medir entre capa y capa. Cada intervención nueva tiene que justificar su coste con una mejora visible en la evaluación; si no la mueve, sobra. Así se evita el sistema que tiene de todo y no se sabe por qué funciona ni por qué falla.

ORDEN	CAPA QUE AÑADES	ANTES DE AÑADIRLA, COMPRUEBA	LA EVAL QUE LA JUSTIFICA
1	Prompt y ejemplos	Que el modelo base no resuelve ya la tarea con instrucciones claras.	Tasa de respuestas útiles sobre 30 casos.
2	Salida estructurada	Que otro software consume la respuesta y el formato rompe.	JSON válido y campos correctos.
3	RAG	Que el fallo es de conocimiento vivo, no de formato ni de tono.	Evidencia correcta y abstención cuando falta fuente.
4	Tool	Que hace falta estado real o cálculo que el modelo no debe inventar.	Exactitud de la llamada y manejo de «no disponible».
5	Ajuste	Que prompt, schema y RAG ya no mueven la conducta repetida.	Mejora contra baseline sin regresiones.

El orden no es dogma, pero su lógica sí: las capas de abajo son más baratas y reversibles, y muchas veces hacen innecesarias las de arriba. Si añades RAG y la tarea mejora, quizá ya no necesitas ajustar. Si saltas directo a fine-tuning, puede que estés pagando la capa más cara para arreglar un problema que un schema resolvía.

Una advertencia: combinar no es acumular. Cada capa añade superficie que mantener, evaluar y depurar. Dos intervenciones bien medidas valen más que cinco puestas por entusiasmo. La señal de un buen sistema no es cuántas herramientas tiene, sino que cada una se pueda explicar con la eval que la sostiene.

Cómo evaluar cada intervención

Una intervención no está terminada cuando funciona en la demo. Está terminada cuando puedes repetir una prueba y decidir si mejoró. Si no sabes qué medir, la arquitectura se convierte en opinión.

INTERVENCIÓN	MÉTRICA PRINCIPAL	PRUEBA MÍNIMA	SEÑAL DE QUE NO BASTA
Prompt y ejemplos	Tasa de respuestas útiles según rúbrica.	30 casos reales antes/después.	Mejora solo en ejemplos vistos o se rompe con variaciones simples.
Salida estructurada	Validez de schema y tasa de campos correctos.	Tests con campos obligatorios, tipos y casos incompletos.	El JSON es válido pero el contenido sigue siendo incorrecto.
RAG	Evidencia correcta, cobertura y abstención cuando falta fuente.	Preguntas con documento esperado y preguntas sin respuesta en corpus.	Recupera texto parecido pero no el fragmento que justifica la respuesta.
Tool	Exactitud de llamada, validación de argumentos y latencia.	Casos con entradas válidas, incompletas y ambiguas.	La tool se invoca cuando no toca o con parámetros mal formados.
Fine-tuning/LoRA	Mejora contra baseline sin perder casos importantes.	Eval fija antes/después y revisión de regresiones.	Mejora el formato pero empeora factualidad o flexibilidad.
Modelo local o privado	Calidad mínima, latencia, coste y mantenimiento.	Misma eval que la API base, con medición de recursos.	Cumple privacidad pero no alcanza la calidad necesaria.

Esas métricas no son vagas: tienen definición formal y conviene usar la real. Para recuperación, la métrica estándar es el `recall@k`, la fracción de consultas cuyo documento relevante aparece entre los k recuperados.[19] Para RAG, la *groundedness* mide qué fracción de las afirmaciones de la respuesta están soportadas por la evidencia citada.[20]

$$\text{recall@}\,k = \frac{1}{N}\sum_{i=1}^{N} 1[d_i^* \in R_k(q_i)]$$

INTERVENCIÓN	MÉTRICA FORMAL	DEFINICIÓN
Salida estructurada	Tasa de validez de schema	Respuestas que validan contra el schema, sobre el total.
RAG (recuperación)	`recall@k`	Consultas con el documento relevante en el top-k, sobre el total.
RAG (generación)	Groundedness	Afirmaciones soportadas por la evidencia citada, sobre el total.
Tool	Exactitud de llamada	Llamadas con nombre y argumentos correctos, sobre el total.

Y un punto que separa una medición de una anécdota: toda métrica es una **proporción estimada con una muestra**, así que tiene incertidumbre. Para n casos y una tasa observada $\hat{p}$, el intervalo de confianza al 95 % por la aproximación normal (intervalo de Wald) es:[21]

$$\hat{p} \pm 1{,}96 \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

Con $\hat{p} = 0{,}8$ y $n = 100$ el margen es $\pm 0{,}08$; con $n = 30$, $\pm 0{,}14$. Por eso una mejora de dos puntos medida sobre 30 casos no es señal: cabe entera dentro del ruido. La eval mínima de 30 casos vale para cazar fallos gruesos; afirmar una mejora fina exige más muestra o una comparación pareada.

En clase, yo pediría siempre tres números antes de aceptar una propuesta: calidad, coste y reversibilidad. Calidad sin coste puede ser inviable. Coste sin calidad no sirve. Y una mejora que no puedes revertir exige mucha más evidencia.

Errores de diagnóstico que conviene detectar

Estos errores parecen razonables cuando tienes prisa, por eso conviene nombrarlos. No son fallos de principiante: aparecen en equipos buenos cuando el problema se describe demasiado pronto con el nombre de una herramienta.

ERROR DE DIAGNÓSTICO	CÓMO SE VE	CÓMO CORREGIRLO
Confundir conocimiento con comportamiento	“Ajustemos el modelo para que sepa la normativa nueva”.	Si cambia con frecuencia, sácalo a documentos, base de datos o tool.
Confundir formato con inteligencia	“El modelo no entiende”, cuando lo único que falla es el JSON.	Primero schema, validación y ejemplos de salida.
Confundir búsqueda con respuesta	El retrieval trae documentos parecidos, pero no justifican la conclusión.	Evaluar recuperación por fragmento correcto, no solo por similitud.
Confundir acción con texto	El modelo “dice” que ha comprobado algo, pero no ha consultado ningún sistema.	Tool real, logs y permisos mínimos.
Confundir benchmark con caso propio	Se elige modelo por ranking general.	Eval con idioma, dominio, coste y latencia del proyecto.
Confundir privacidad con peor producto	“Local” se acepta sin medir calidad.	Comparar local, privado y API con la misma rúbrica.

El antídoto común es escribir una frase de diagnóstico antes de escribir una frase de solución: “El sistema falla porque...”. Si esa frase ya contiene el nombre de una herramienta, sospecha un poco.

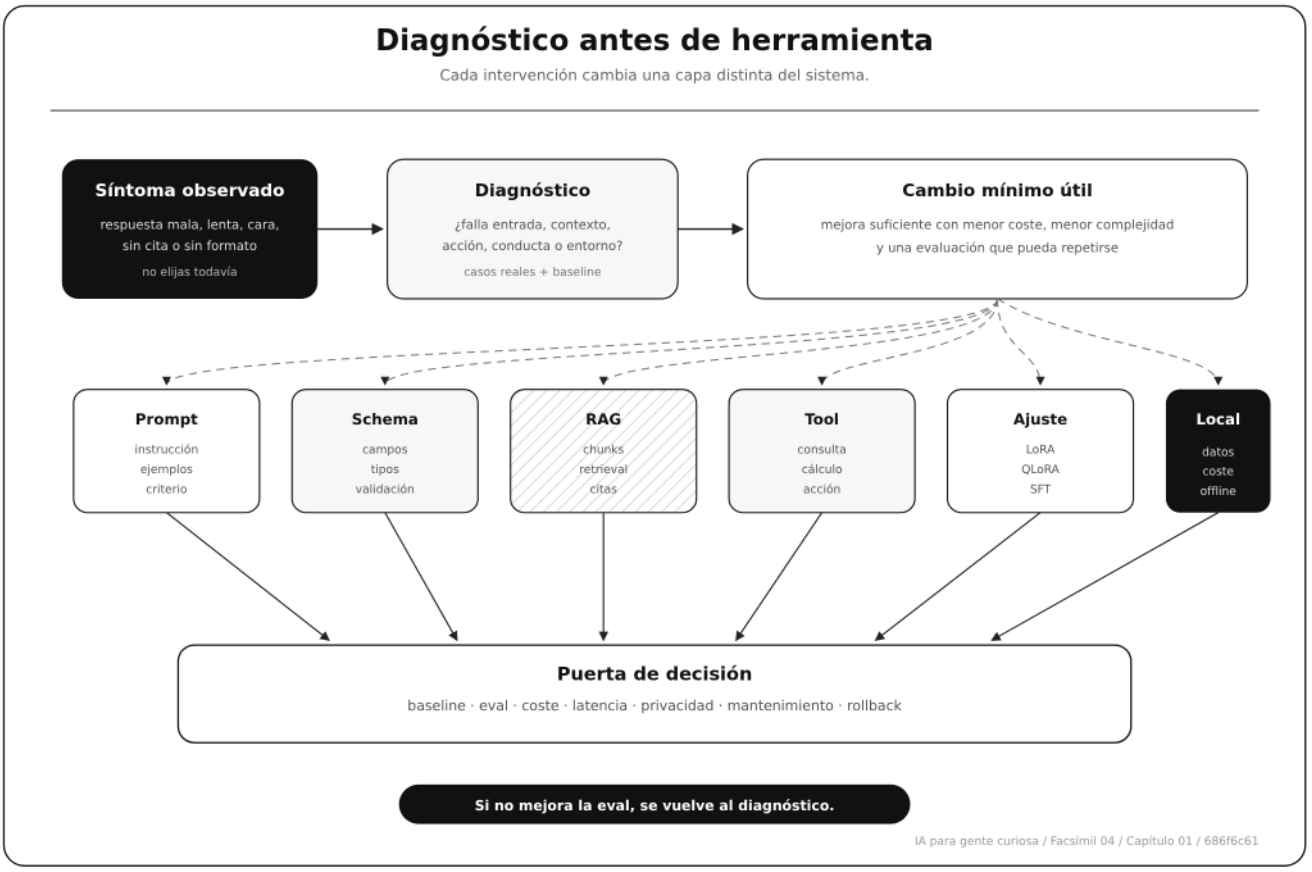
Mini práctica de decisión

Para entrenar el criterio, resuelve estos cuatro casos sin programar. En cada uno escribe: síntoma, intervención principal, alternativa descartada y métrica de aceptación. Después compara con la solución modelo.

CASO	SÍNTOMA	INTERVENCIÓN PRINCIPAL	ALTERNATIVA DESCARTADA	MÉTRICA DE ACEPTACIÓN
Biblioteca universitaria con horarios cambiantes.	Responde horarios antiguos.	RAG o consulta a fuente oficial.	Fine-tuning.	Respuestas con horario vigente y fuente correcta.
App médica que necesita clasificar mensajes en tres colas internas.	El texto es bueno, pero la integración falla.	Salida estructurada.	RAG.	JSON válido y cola correcta según rúbrica humana.
Ecommerce con preguntas de disponibilidad por tienda.	El modelo estima stock.	Tool de inventario.	Prompt más insistente.	Stock correcto, latencia aceptable y manejo de ausencia de dato.
Equipo legal con contratos sensibles en portátiles sin conexión.	No puede enviar documentos fuera y necesita asistencia básica.	Modelo local cuantizado o entorno privado.	API externa directa.	Calidad mínima en una eval interna y tiempo de respuesta usable.

La solución modelo no pretende cerrar todos los matices. Pretende mostrar el razonamiento: cada respuesta identifica la capa que falla. Si cambias el enunciado, puede cambiar la intervención. Si la biblioteca también necesita guardar campos exactos, añadirías salida estructurada. Si el ecommerce además debe explicar políticas de devolución, quizá combinarías tool con RAG. La arquitectura final puede tener varias piezas; el diagnóstico inicial decide cuál entra primero.

Mapa visual de diagnóstico



En el día a día

En un equipo real, este capítulo se usa antes de abrir el editor. Pones encima de la mesa tres cosas: el caso de uso, diez ejemplos reales y una forma de medir. Solo entonces eliges herramienta.

Si un jefe de producto pide “un chatbot con todos los documentos”, tradúcelo: quizá quiere búsqueda con citas, quizá quiere navegación guiada, quizá quiere reducir tickets, quizá quiere extraer campos. Cada objetivo produce una arquitectura distinta.

Si un equipo técnico dice “hagamos RAG”, pregunta por el corpus, el tipo de preguntas, el criterio de cita, los permisos por documento y cómo sabremos que el retrieval encontró evidencia. Si esas respuestas no existen, todavía no hay arquitectura: hay deseo.

Por qué debería importarte

Porque cada intervención mal elegida deja deuda. Un RAG innecesario añade índices, chunks y evals. Un fine-tuning prematuro añade dataset, entrenamiento y versiones. Una tool amplia añade permisos y validación. Un modelo local añade soporte, hardware y medición de calidad.

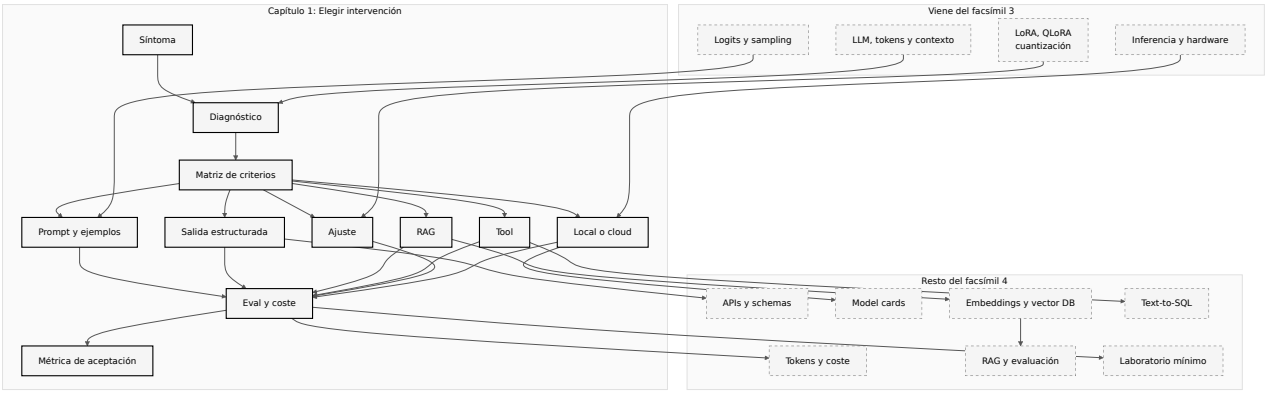
La buena noticia es que una intervención bien elegida suele simplificar. Un schema puede eliminar cientos de líneas de parsing frágil. Una tool puede evitar que el modelo invente un dato. Un RAG con citas puede convertir una respuesta bonita en una respuesta revisable.

Dónde solía tropezar yo

Estos tropiezos aparecen mucho al empezar proyectos con IA. Casi todos nacen de confundir síntoma con solución.

ERROR	POR QUÉ ES UN ERROR	ANTÍDOTO
Empezar por RAG sin corpus claro	Si no sabes qué documentos, preguntas y citas necesitas, el índice será decoración cara.	Definir 30 preguntas reales antes de indexar.
Ajustar pesos para datos vivos	El dato que cambia mañana no debería quedar escondido en un adapter.	Usar RAG, base de datos o tool para conocimiento cambiante.
Pedir formato con súplicas	“Responde en JSON” no es contrato suficiente si el backend depende de campos exactos.	Usar salida estructurada y validación.
Usar una tool para todo	Una función gigante es difícil de validar y fácil de usar mal.	Tools pequeñas, schemas claros y permisos mínimos.
Comparar modelos sin tarea	Un ranking genérico no sabe qué coste, idioma, latencia o riesgo tiene tu producto.	Comparar con tus casos, tus métricas y tus límites.

Cómo encaja todo



Vocabulario aprendido

Estas palabras aparecerán en todo el facsímil. Conviene que queden limpias desde el principio.

TÉRMINO	DEFINICIÓN
Intervención	Cambio concreto en el sistema para mejorar un síntoma medible.
Prompt	Entrada que define tarea, contexto, ejemplos y criterio.
Schema	Estructura esperada de una salida o una tool.
Salida estructurada	Respuesta obligada a cumplir un formato validable.
RAG	Recuperar evidencia externa y pasarla al modelo como contexto.
Tool	Función externa que consulta, calcula o modifica algo bajo reglas.
Fine-tuning	Ajustar pesos de un modelo con datos propios.
LoRA	Ajuste eficiente con matrices pequeñas de bajo rango.
Modelo local	Modelo ejecutado en una máquina o infraestructura controlada.
Baseline	Comparación mínima antes de añadir complejidad.
Eval	Prueba repetible que mide calidad, coste, latencia o seguridad del sistema.
Matriz de decisión	Tabla que obliga a justificar una elección con criterios observables.
Reversibilidad	Facilidad para volver atrás si una intervención empeora el sistema.
Abstención	Capacidad de reconocer que falta evidencia suficiente para responder.

Antes de pasar página

- [ ] ¿Puedo explicar por qué no se empieza eligiendo herramienta? (Si no, vuelve a «La herramienta no es el punto de partida».)
- [ ] ¿Sé distinguir si un problema es de entrada, contexto, acción, conducta o entorno? (Si no, vuelve a «La pregunta correcta: qué quieres cambiar».)
- [ ] ¿Puedo decir cuándo basta prompt/schema y cuándo hace falta RAG? (Si no, vuelve a «Las cinco intervenciones básicas».)
- [ ] ¿Entiendo por qué fine-tuning no es buena memoria para datos vivos? (Si no, vuelve a «Para entenderlo antes de tocar código».)
- [ ] ¿Sé cuándo una tool es más adecuada que pedirle al modelo que “razone”? (Si no, vuelve a «Para entenderlo antes de tocar código».)

- [ ] ¿Puedo explicar una decisión con utilidad esperada y valor de la información en vez de con una corazonada? (Si no, vuelve a «El criterio de decisión: utilidad esperada y valor de la información».)
- [ ] ¿Puedo justificar una intervención usando criterios como cita, estado real, formato, coste y reversibilidad? (Si no, vuelve a «Criterios de elección: la matriz que decide contigo».)
- [ ] ¿Sé resolver un mismo caso con prompt, schema, RAG, tool o ajuste según el síntoma? (Si no, vuelve a «Un mismo caso, cinco soluciones posibles».)
- [ ] ¿Puedo proponer una métrica mínima para evaluar cada intervención? (Si no, vuelve a «Cómo evaluar cada intervención».)
- [ ] ¿Sé detectar cuándo estoy confundiendo conocimiento, formato, acción o despliegue? (Si no, vuelve a «Errores de diagnóstico que conviene detectar».)
- [ ] ¿Sé ajustar los pesos de la matriz para que gane otra intervención y por qué? (Si no, repasa «Criterios de elección: la matriz que decide contigo».)

En resumen

La caja de herramientas empieza por diagnóstico. Si eliges bien el problema, la herramienta suele volverse evidente.

IDEA FUERZA	DETALLE
La herramienta no es el punto de partida.	Primero síntoma, casos reales y baseline.
Prompt/schema arreglan contrato de entrada y salida.	No hacen que el modelo sepa datos vivos.
RAG aporta evidencia externa.	Sirve para documentos cambiantes y respuestas citables.
Tool conecta con estado real.	Sirve para consultar, calcular o actuar sin inventar.
Ajustar pesos cambia comportamiento.	Tiene sentido en tareas repetidas, estables y medibles.
Local/cloud son decisiones de entorno.	Privacidad, coste, latencia y mantenimiento mandan.
Eval decide si la intervención se queda.	Sin medición, solo hay impresión.
Una buena decisión debe poder explicarse.	Criterio, alternativa descartada, métrica y rollback forman parte de la respuesta.

Para saber más

Anthropic. (2026). *Prompt engineering overview*. https://platform.claude.com/docs/en/build-with-claude/prompt-engineering/overview

Dettmers, T. y otros (2023). QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in Neural Information Processing Systems 36*. https://arxiv.org/abs/2305.14314

Hugging Face. (2026). *PEFT: LoRA developer guide*. https://huggingface.co/docs/peft/developer_guides/lora

Hu, E. J. y otros (2022). LoRA: Low-Rank Adaptation of Large Language Models. *International Conference on Learning Representations*. https://arxiv.org/abs/2106.09685

Lewis, P. y otros (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems 33*, 9459-9474. https://papers.nips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

Ollama. (2026). *Introduction to the Ollama API*. https://docs.ollama.com/api/introduction

Ollama. (2026). *Ollama Cloud*. https://docs.ollama.com/cloud

OpenAI. (2026). *Function calling*. https://developers.openai.com/api/docs/guides/function-calling

OpenAI. (2026). *Structured model outputs*. https://developers.openai.com/api/docs/guides/structured-outputs

Notas

1. OpenAI. (2026). *Structured model outputs*. https://developers.openai.com/api/docs/guides/structured-outputs. Consultado el 25 de mayo de 2026. La guía distingue entre salidas estructuradas para respuesta final y function calling cuando el modelo se conecta a herramientas o datos. ↩

2. Anthropic. (2026). *Prompt engineering overview*. https://platform.claude.com/docs/en/build-with-claude/prompt-engineering/overview. Consultado el 25 de mayo de 2026. La guía trata el prompt como mecanismo de especificación de tarea, contexto, ejemplos y restricciones. ↩

3. Hugging Face. (2026). *PEFT: LoRA developer guide*. https://huggingface.co/docs/peft/developer_guides/lora. Consultado el 25 de mayo de 2026. PEFT documenta LoRA y variantes para entrenar adaptadores pequeños sobre modelos base. ↩

4. Ollama. (2026). *Introduction to the Ollama API*. https://docs.ollama.com/api/introduction. Consultado el 25 de mayo de 2026. La documentación indica la URL local por defecto y la URL cloud compatible. ↩

5. Ollama. (2026). *Ollama Cloud*. https://docs.ollama.com/cloud. Consultado el 25 de mayo de 2026. La guía describe modelos cloud que se ejecutan sin requerir una GPU local potente. ↩

6. von Neumann, J. y Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press. El teorema de la utilidad esperada establece que, bajo axiomas de racionalidad, las preferencias se representan maximizando la esperanza de una función de utilidad. ↩

7. Howard, R. A. (1966). Information value theory. *IEEE Transactions on Systems Science and Cybernetics, 2*(1), 22-26. https://doi.org/10.1109/TSSC.1966.300074. El valor de la información es la mejora de utilidad esperada que produce observar una evidencia antes de decidir, y es siempre no negativa. ↩

8. Anthropic. (2026). *Prompt engineering overview*. https://platform.claude.com/docs/en/build-with-claude/prompt-engineering/overview. Consultado el 25 de mayo de 2026. ↩

9. OpenAI. (2026). *Structured model outputs*. https://developers.openai.com/api/docs/guides/structured-outputs. Consultado el 25 de mayo de 2026. ↩

10. Lewis, P. y otros (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems 33*, 9459-9474. https://papers.nips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html. ↩

11. OpenAI. (2026). *Function calling*. https://developers.openai.com/api/docs/guides/function-calling. Consultado el 25 de mayo de 2026. ↩

12. Hu, E. J. y otros (2022). LoRA: Low-Rank Adaptation of Large Language Models. *International Conference on Learning Representations*. https://arxiv.org/abs/2106.09685. Hugging Face. (2026). *PEFT: LoRA developer guide*. https://huggingface.co/docs/peft/developer_guides/lora. Consultado el 25 de mayo de 2026. ↩

13. Lewis, P. y otros (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems 33*, 9459-9474. https://papers.nips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html. El trabajo formuló modelos que recuperan documentos y los combinan con generación. ↩

14. Hu, E. J. y otros (2022). LoRA: Low-Rank Adaptation of Large Language Models. *International Conference on Learning Representations*. https://arxiv.org/abs/2106.09685. ↩

15. Dettmers, T. y otros (2023). QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in Neural Information Processing Systems 36*. https://arxiv.org/abs/2305.14314. ↩

.6. Triantaphyllou, E. (2000). *Multi-Criteria Decision Making Methods: A Comparative Study*. Springer. https://doi.org/10.1007/978-1-4757-3157-6. El Weighted Sum Model es el método multicriterio más conocido; su validez exige criterios comparables y normalizados. ↩

.7. Keeney, R. L. y Raiffa, H. (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley. Formaliza cómo combinar varios objetivos con funciones de valor y pesos. ↩

.8. Keeney, R. L. y Raiffa, H. (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley. La frontera eficiente de compromisos recoge las alternativas no dominadas entre las que la elección depende de las preferencias. ↩

.9. Manning, C. D., Raghavan, P. y Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. Define precisión, exhaustividad (recall) y sus variantes por corte, base de la evaluación de recuperación. ↩

20. Es, S., James, J., Espinosa-Anke, L. y Schockaert, S. (2023). *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. https://arxiv.org/abs/2309.15217. Formaliza métricas como faithfulness y context relevance para evaluar sistemas RAG. ↩

21. El intervalo de Wald usa la aproximación normal de la binomial; para n pequeño o $\hat{p}$ cercano a 0 o 1 conviene el intervalo de Wilson. Lawrence D. Brown, T. Tony Cai y Anirban DasGupta. (2001). *Interval Estimation for a Binomial Proportion*. Statistical Science, 16(2), 101-133. https://doi.org/10.1214/ss/1009213286. ↩