Facsímil 04

La caja de herramientas

Capítulo 10

Evaluar RAG: retrieval, groundedness y abstención

686f6c61 · 2026

Capítulo 10: Evaluar RAG: retrieval, groundedness y abstención

La demo no cuenta como evaluación

En el capítulo 09 montamos el primer RAG serio: corpus, chunks, embeddings, retrieval, contexto, citas y abstención. Ahora viene la parte que separa una demo de un sistema profesional: demostrar que mejora, saber dónde falla y decidir cuándo no debe responder.

Un RAG puede fallar de formas muy distintas. Puede no recuperar el documento correcto. Puede recuperarlo y no meterlo en el contexto final. Puede meterlo y redactar algo que no está sostenido. Puede citar una fuente que no dice eso. Puede contestar cuando debería abstenerse. Si solo miras la respuesta final, llegas tarde: ves el síntoma, pero no el órgano que falló.

Evaluar RAG es medir la cadena completa:

1. ¿La pregunta está en un conjunto de evaluación representativo?
2. ¿El corpus contiene la evidencia necesaria?
3. ¿El retrieval trae esa evidencia entre los primeros resultados?
4. ¿El context builder la mete en el prompt sin ahogarla en ruido?
5. ¿La respuesta está apoyada por el contexto?
6. ¿Las citas apuntan a fragmentos que sostienen lo dicho?
7. ¿El sistema se abstiene cuando no hay evidencia?
8. ¿El coste, la latencia y la operación siguen siendo aceptables?

La evaluación no es una fase final. Es una pieza del producto.

Estado del arte con fecha de corte

Fecha de corte: 25 de mayo de 2026.
Fuentes consultadas ese día: documentación de Ragas, TruLens, LangSmith, LlamaIndex, Phoenix y OpenAI Graders; y referencias académicas sobre RAG, retrieval y benchmarks como BEIR/MTEB.

RAGAS propuso evaluar RAG separando componentes como recuperación, relevancia y fidelidad de la respuesta.[1] La documentación actual de Ragas organiza métricas de RAG como context precision, context recall, response relevancy, faithfulness y métricas multimodales.[2]

TruLens populariza una tríada muy práctica: relevancia del contexto, groundedness y relevancia de la respuesta.[3] LangSmith, LlamaIndex y Phoenix empujan una idea parecida desde producto: datasets, experimentos, trazas, evaluadores y comparación entre versiones.[4]

OpenAI Graders documenta la idea de evaluadores configurables con criterios, escalas y umbrales, incluyendo verificaciones de texto, similitud, evaluadores de modelo y ejecución de código.[5] La lección importante para nuestro facsímil no es “usa esta herramienta”, sino “define la rúbrica y conserva la traza”.

FAMILIA	QUÉ MIDE	CUÁNDO USARLA
Métricas clásicas de retrieval	Si los chunks esperados aparecen y en qué posición.	Antes de mirar la respuesta del LLM.
Métricas de contexto	Si el contexto final es útil, completo y no ruidoso.	Cuando el retrieval trae candidatos pero la respuesta falla.
Métricas de groundedness	Si las afirmaciones están sostenidas por el contexto.	Para respuestas citadas, informes y asistentes de documentación.
Métricas de abstención	Si responde cuando debe y se calla cuando toca.	En dominios donde inventar cuesta confianza.
Métricas de operación	Latencia, coste, errores, cobertura y deriva.	Cuando el RAG ya vive en una aplicación.

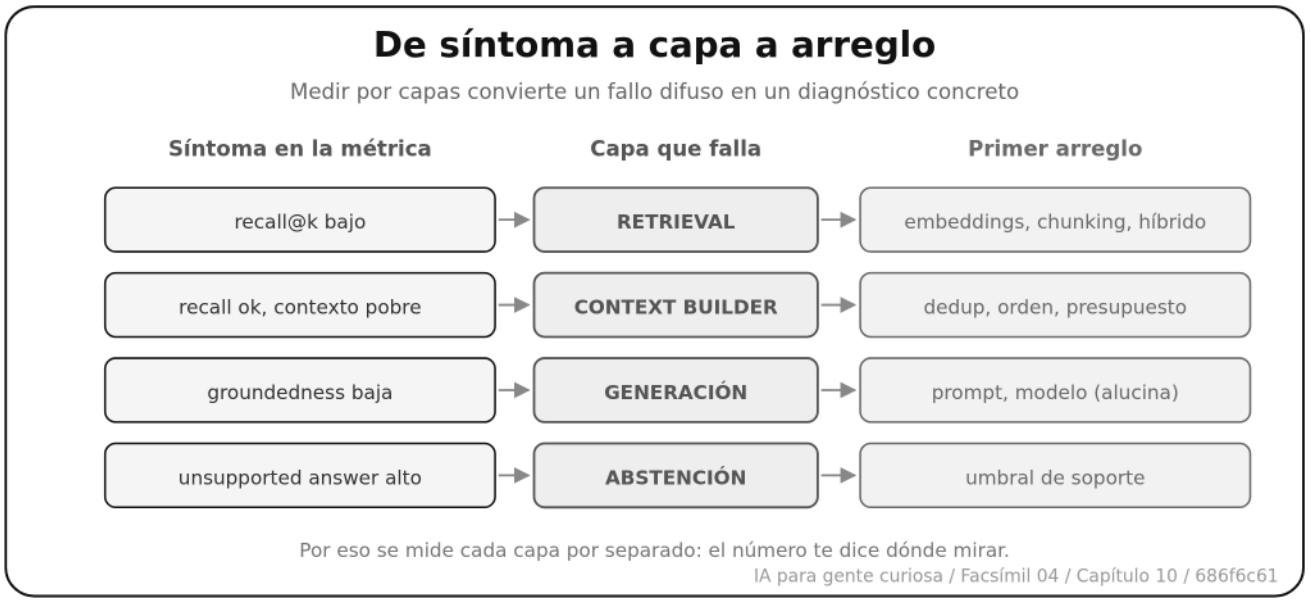
Qué significa evaluar por capas

Un RAG se evalúa por capas porque cada capa tiene una pregunta distinta. Si solo mides “respuesta correcta”, no sabes si mejorar embeddings, chunking, prompt, reranker o corpus.

CAPA	PREGUNTA	EVIDENCIA QUE NECESITAS
Corpus	¿Existe la fuente correcta?	Documento vigente, versión, propietario y fecha.
Parsing	¿El texto extraído es fiel?	Comparación contra PDF, HTML, tabla o fuente original.
Chunking	¿La unidad recuperable sostiene una respuesta?	Chunks con título, sección, página y hash.
Retrieval	¿Aparece la evidencia en top-k?	Ranking de chunks y qrels.
Reranking	¿Sube la evidencia buena?	Ranking antes/después y relevancia graduada.
Context builder	¿El modelo recibe lo necesario y no demasiado ruido?	Contexto final enviado al LLM.
Generación	¿La respuesta contesta con contrato?	Respuesta, citas, formato y abstención.
Groundedness	¿Cada afirmación está sostenida?	Claims separados y evidencia por claim.
Operación	¿El sistema aguanta uso real?	Trazas, latencia, coste, errores y feedback.

La regla de ingeniería es sencilla: primero evalúa retrieval sin LLM; luego evalúa generación con contexto fijo; después evalúa el sistema completo. Si mezclas todo desde el principio, cada fallo parece misterioso.

Medir por capas tiene un beneficio inmediato: convierte un síntoma en un diagnóstico. Cada métrica que baja apunta a una capa concreta y a un primer arreglo, en vez de obligarte a adivinar.



Dataset de evaluación: el corazón del sistema

Un dataset de evaluación no es un conjunto de preguntas bonitas. Es un contrato de verdad para comparar versiones. Debe contener preguntas que representen el uso real, preguntas que el sistema debe responder, preguntas que debe rechazar por falta de evidencia y preguntas donde los filtros importan.

Un ejemplo de fila mínima:

CAMPO	PARA QUÉ SIRVE
`id`	Identificador estable de la pregunta.
`pregunta`	Lo que una persona o sistema preguntaría.
`answerable`	Si el corpus contiene evidencia suficiente.
`gold_chunks`	Chunks esperados o aceptables.
`gold_answer`	Respuesta de referencia, si existe.
`gold_citations`	Citas esperadas.
`filtros`	Curso, rol, vigencia, idioma, cliente o tenant.
`tipo`	Single-hop, multi-hop, tabla, código, imagen, temporal, etc.
`dificultad`	Fácil, media, difícil, o escala propia.
`criterio`	Qué debe ocurrir para considerar la respuesta válida.

No todas las preguntas necesitan una respuesta literal de referencia. Para retrieval basta con qrels: juicios de relevancia entre pregunta y chunks. Para groundedness necesitas contexto y respuesta. Para abstención necesitas casos sin evidencia suficiente.

TIPO DE PREGUNTA	EJEMPLO	QUÉ PRUEBA
Directa	“¿Cuándo se abre la ampliación de matrícula?”	Retrieval simple y cita directa.
Con filtro	“¿Qué aplica al curso 2026?”	Metadata y vigencia.
Multi-hop	“¿Puedo ampliar si tengo pagos pendientes y cómo se solicita?”	Recuperar más de un fragmento.
Tabla	“¿Qué plazo corresponde a segunda matrícula?”	Parsing y estructura tabular.
No respondible	“¿Cuál es el horario de cafetería?” si no está en corpus.	Abstención.
Contradicción documental	Dos fuentes con fechas distintas.	Vigencia, prioridad y explicación.
Texto largo	Preguntas que requieren contexto distribuido.	Recall, deduplicación y presupuesto.

Ejemplos de datasets de evaluación que puedes construir en un proyecto real:

DATASET	QUÉ CONTIENE	POR QUÉ MERECE EXISTIR
FAQ real	Preguntas frecuentes, respuesta esperada y fuente exacta.	Mide si el RAG resuelve lo que más se pregunta.
Normativa vigente	Preguntas con `curso`, `fecha`, `rol` y chunks esperados.	Mide filtros y prioridad documental.
Casos sin evidencia	Preguntas plausibles cuya respuesta no está en el corpus.	Mide abstención correcta.
Multi-hop	Preguntas que exigen dos o más fuentes.	Mide si el contexto compone evidencia sin mezclar.
Tablas	Preguntas sobre filas, columnas, importes, plazos o unidades.	Mide parsing y recuperación estructurada.
Citas difíciles	Respuestas donde una cita parcial no basta.	Mide si la cita sostiene la afirmación completa.
Regresión	Casos que ya fallaron en producción y se corrigieron.	Evita reintroducir errores.
Segmentos críticos	Preguntas por idioma, área, producto, cliente o perfil.	Evita que la media esconda fallos locales.

Si vienes del facsímil 3, capítulo 06, la diferencia es esta: datasets como FLAN, Dolly-15K, HH-RLHF o LAION-5B pueden entrenar o adaptar modelos; este dataset de evaluación no debería entrenar el sistema que estás midiendo. Su trabajo es ponerle un espejo fiable.

El dataset debe crecer desde producción. Las primeras 30 preguntas sirven para empezar. Las siguientes 300 salen de dudas reales, tickets, búsquedas sin respuesta, feedback de usuarios y revisiones de profesorado o equipo de dominio.

Generar el dataset: a mano y sintético

Construir 300 preguntas a mano es lento, así que existe un atajo: **generación sintética**. Un LLM lee cada chunk del corpus y propone preguntas que ese chunk podría responder, junto con el chunk de oro asociado. Es lo que automatiza RAGAS con su generación de conjuntos de prueba, capaz de producir preguntas de distinta dificultad y tipo (directas, multi-hop, de razonamiento) a partir de tus propios documentos.[6]

Dos cautelas de ingeniería, porque lo sintético tiene trampa:

- **Revísalo a mano antes de usarlo.** Una pregunta sintética puede ser trivial («¿qué dice el artículo 14?» copiando el texto), ambigua o estar mal emparejada con su oro. Un dataset sin revisar mide la capacidad del generador, no la de tu RAG.

- **Lo sintético no sustituye lo real.** Las preguntas que de verdad importan (las raras, las mal formuladas, las que nadie anticipó) salen de producción, no de un generador. Usa lo sintético para arrancar y para cubrir tipos; usa lo real para decidir.

Y un caso que el generador casi nunca produce solo y tienes que añadir tú: las **preguntas no respondibles**, sin evidencia en el corpus, que son las que prueban la abstención. Sin ellas, tu dataset premia a un sistema que lo contesta todo.

Qrels y relevancia graduada

Un qrel es un juicio de relevancia. Dice que una pregunta necesita tal documento o tal chunk. Puede ser binario o graduado.

RELEVANCIA	SIGNIFICADO
0	No ayuda a responder.
1	Relacionado, pero insuficiente.
2	Útil para parte de la respuesta.
3	Evidencia central.

Ejemplo:

PREGUNTA	CHUNK	RELEVANCIA
`q1` ampliación con pagos pendientes	`norm-2026#c1`	3
`q1` ampliación con pagos pendientes	`faq-campus#c1`	0
`q2` acceso al campus	`faq-campus#c1`	3
`q3` horario de cafetería	ningún chunk	no respondible

Esta tabla permite evaluar retrieval sin llamar al LLM. Eso ahorra coste y te dice si la base del sistema funciona.

Métricas de retrieval

Las métricas de retrieval responden a una pregunta: ¿la evidencia correcta aparece en el ranking? No hay que inventarlas: son las métricas estándar de la recuperación de información, con décadas de uso en las campañas TREC, y `nDCG` añade el descuento por posición para relevancia graduada.[7]

Sea G_q el conjunto de chunks relevantes para una pregunta q, y sea $R_k(q)$ la lista de los k primeros chunks recuperados.

$$\text{Precision@k}(q) = \frac{|R_k(q) \cap G_q|}{k}$$

$$\text{Recall@k}(q) = \frac{|R_k(q) \cap G_q|}{|G_q|}$$

$$\text{Hit@k}(q) = \begin{cases} 1, & \text{si } R_k(q) \cap G_q \neq \varnothing \\ 0, & \text{si } R_k(q) \cap G_q = \varnothing \end{cases}$$

MÉTRICA	QUÉ TE DICE	CUIDADO
Precision@k	De lo que recuperas, cuánto sirve.	Puede ser baja si necesitas traer contexto amplio.
Recall@k	De lo que necesitabas, cuánto aparece.	Puede subir trayendo demasiado ruido.
Hit@k	Si aparece al menos una evidencia.	No mide si aparece toda la evidencia.
MRR	Qué tan pronto aparece la primera evidencia.	No basta para preguntas multi-hop.
nDCG@k	Si lo más relevante aparece arriba.	Requiere relevancia graduada.

MRR se calcula con la posición de la primera evidencia relevante:

$$\text{RR}(q) = \frac{1}{\text{rank}_q}$$

Si el primer chunk relevante aparece en posición 1, RR vale 1. Si aparece en posición 5, vale 0,2. Si no aparece, vale 0. El MRR es la media de RR en muchas preguntas.

nDCG usa relevancia graduada:

$$\text{DCG@k} = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

$$\text{nDCG@k} = \frac{\text{DCG@k}}{\text{IDCG@k}}$$

SÍMBOLO	SIGNIFICADO
rel_i	Relevancia del resultado en posición i.
DCG	Ganancia descontada por posición.
$IDCG$	DCG ideal si los mejores resultados estuvieran arriba.

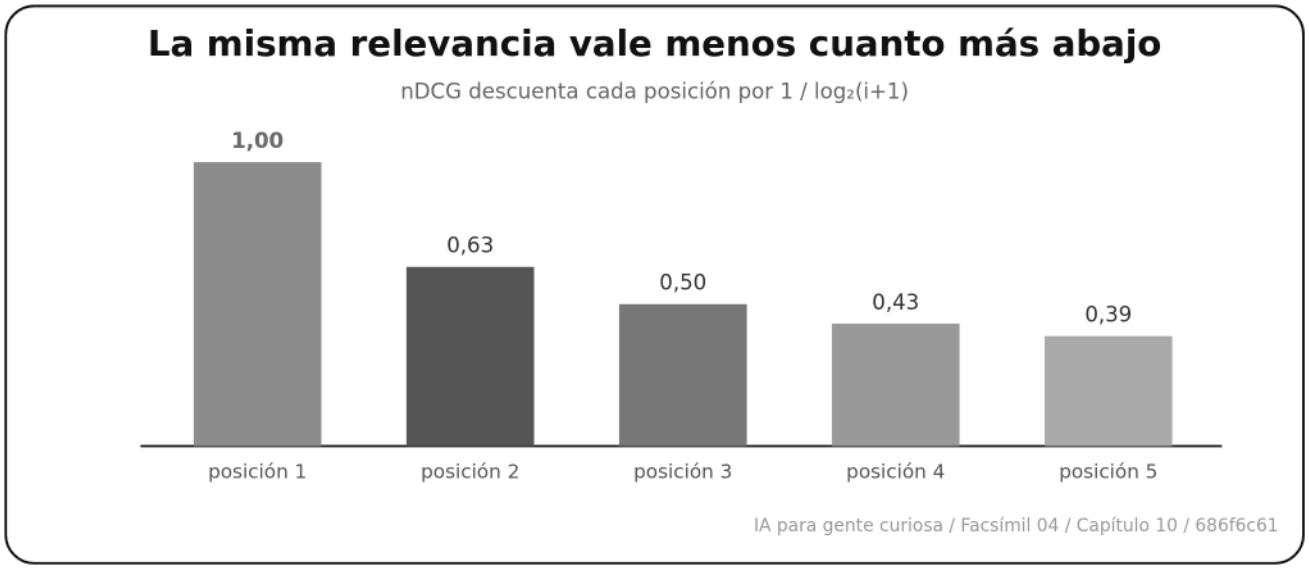
Veámoslo con números, porque es la métrica que más confunde. Recuperas 3 chunks y un humano les da relevancia graduada (0 = no sirve, 1 = ayuda, 2 = relevante, 3 = clave). El ranking real trae las relevancias $[2, 0, 3]$: un chunk relevante arriba, ruido en medio y el chunk clave enterrado en la posición 3.

$$\text{DCG@3} = \frac{2^2 - 1}{\log_2 2} + \frac{2^0 - 1}{\log_2 3} + \frac{2^3 - 1}{\log_2 4} = 3 + 0 + 3{,}5 = 6{,}5$$

El ranking ideal pondría el clave primero, $[3, 2, 0]$, de modo que $\text{IDCG@3} = 7 + 1{,}893 + 0 = 8{,}893$. Normalizando:

$$\text{nDCG@3} = \frac{6{,}5}{8{,}893} = 0{,}73$$

La lectura de ingeniería: 0,73 no es desastroso, pero toda la penalización viene de tener el chunk clave en la posición 3 en lugar de la 1. Eso es lo que mide nDCG y no las métricas binarias: **enterrar lo bueno cuesta**, y cuesta porque cada posición descuenta por $1/\log_2(i+1)$, un factor que cae rápido.



Precision@k y recall@k vienen de la tradición de recuperación de información. BEIR y MTEB son recordatorios útiles: un retriever puede brillar en un benchmark y fallar en tu dominio.[8] Por eso el benchmark público orienta, pero el dataset interno decide.

Métricas de contexto

El retrieval devuelve candidatos. El contexto final es lo que realmente lee el modelo. Entre una cosa y otra puede haber deduplicación, recorte, reordenación, filtros, prioridad de fuentes y presupuesto de tokens.

Ragas llama context precision y context recall a dos ideas útiles.[9] Las traduzco de forma operativa:

MÉTRICA	PREGUNTA
Context precision	¿El contexto incluido es útil o está lleno de ruido?
Context recall	¿El contexto contiene toda la evidencia necesaria?

Estas son las métricas de contexto que formaliza el marco RAGAS para evaluar RAG de forma automática; aquí las usamos en su versión más simple:[10]

$$\text{ContextPrecision} = \frac{\text{chunks útiles en contexto}}{\text{chunks en contexto}}$$

$$\text{ContextRecall} = \frac{\text{evidencias esperadas presentes}}{\text{evidencias esperadas}}$$

La diferencia con retrieval es sutil pero importante. Retrieval@k mide el ranking bruto. Context precision/recall mide el paquete de evidencia que llegó al prompt.

FALLO	RETRIEVAL	CONTEXTO
El chunk bueno no aparece en top 20.	Falló retrieval.	No tiene oportunidad.
El chunk bueno aparece en top 5 pero se recorta.	Retrieval bien.	Falló context builder.
Hay cinco chunks repetidos.	Retrieval dudoso.	Falló deduplicación.
Entra una fuente antigua y otra vigente.	Filtros dudosos.	Falló prioridad o explicación.

Groundedness, faithfulness y citas

Groundedness significa que la respuesta está apoyada por el contexto recuperado. Faithfulness suele usarse de forma cercana: la respuesta no añade hechos que no se desprenden del contexto. TruLens lo conecta con la tríada: contexto relevante, respuesta apoyada en el contexto y respuesta relevante para la pregunta.[11]

La forma práctica de evaluarlo es separar la respuesta en afirmaciones.

RESPUESTA	CLAIMS
“La ampliación se abre en septiembre y no puede haber pagos vencidos.”	1. La ampliación se abre en septiembre. 2. No puede haber pagos vencidos.

Cada claim se evalúa contra el contexto:

CLAIM	EVIDENCIA	RESULTADO
La ampliación se abre en septiembre.	Chunk `norm-2026#c1`.	Sostenido.
No puede haber pagos vencidos.	Chunk `norm-2026#c1`.	Sostenido.
Se aprueba automáticamente.	No aparece en contexto.	No sostenido.

Esta es la idea de la faithfulness de RAGAS: descomponer la respuesta en afirmaciones y medir qué fracción se sostiene en el contexto recuperado.[12]

$$\text{Groundedness} = \frac{\text{claims sostenidos}}{\text{claims totales}}$$

La cita añade otra capa. No basta con que la respuesta sea cierta: debe citar el fragmento correcto.

$$\text{CitationPrecision} = \frac{\text{citas válidas usadas}}{\text{citas usadas}}$$

$$\text{CitationRecall} = \frac{\text{evidencias citadas}}{\text{evidencias necesarias}}$$

CASO	GROUNDEDNESS	CITAS	DIAGNÓSTICO
Respuesta correcta y cita correcta.	Alta.	Alta.	Bien.
Respuesta correcta sin cita.	Alta.	Baja.	Falta trazabilidad.
Respuesta correcta con cita equivocada.	Puede parecer alta.	Baja.	Interfaz de confianza rota.
Respuesta inventada con cita real.	Baja.	Engañosa.	El chunk citado no sostiene la frase.

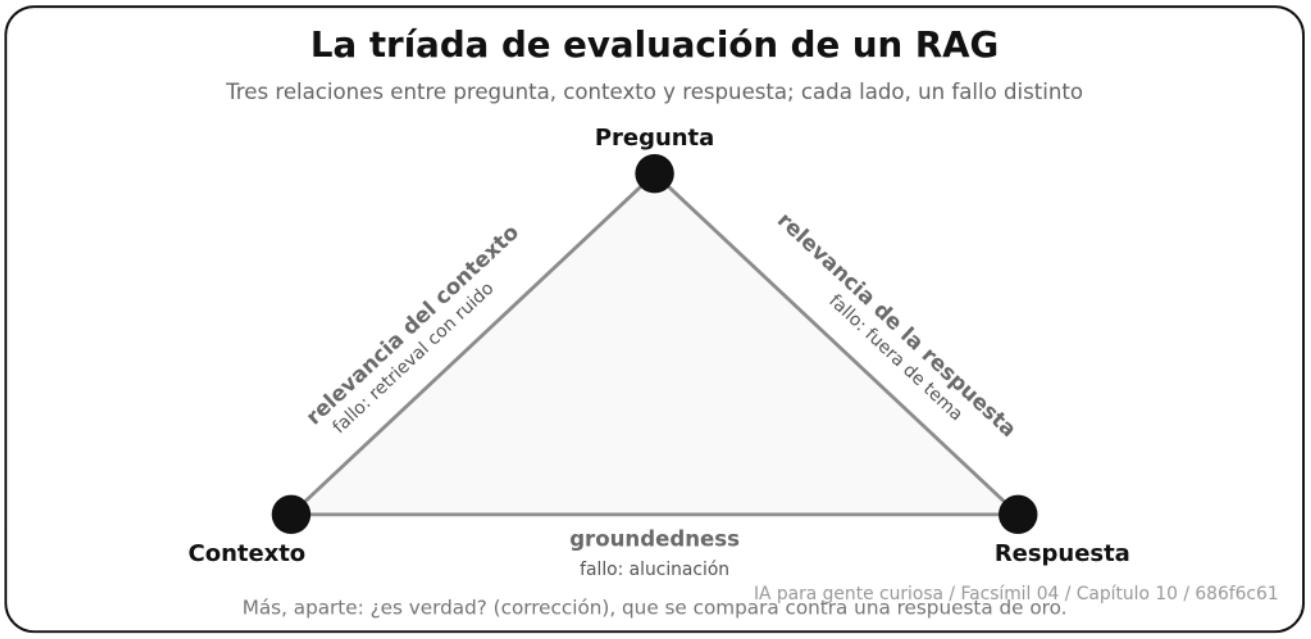
Un evaluador LLM puede ayudar a evaluar groundedness, pero no es oráculo. Debe recibir rúbrica clara, contexto, respuesta y, si existe, respuesta de referencia. Sus resultados deben compararse con revisión humana en una muestra. OpenAI Graders documenta distintos tipos de evaluadores y la idea de devolver una puntuación numérica contra criterios.[13]

Estar fundamentado no es estar en lo cierto

Cuidado con un atajo peligroso: confundir groundedness con corrección. Una respuesta puede apoyarse en el chunk correcto y, aun así, ser falsa, porque lo leyó mal, mezcló dos cláusulas o invirtió una condición («se concede» en vez de «no se concede»). Por eso un RAG serio mide tres cosas distintas, que son la **tríada de evaluación** que populariza TruLens:[14]

PREGUNTA	QUÉ COMPARA	FALLO QUE DETECTA
Relevancia del contexto	Pregunta ↔ contexto recuperado	El retrieval trajo ruido.
Groundedness	Contexto ↔ respuesta	La respuesta alucina más allá del contexto.
Relevancia de la respuesta	Pregunta ↔ respuesta	La respuesta se va por las ramas.

Y, por encima de las tres, la **corrección** (*answer correctness*): ¿la respuesta es verdad? Eso se compara contra una respuesta de oro, no solo contra el contexto. Un sistema puede tener groundedness alta y corrección baja a la vez; medir solo lo primero da una falsa sensación de seguridad.



Evaluar abstención

Abstenerse no significa que el sistema sea torpe. En RAG, abstenerse puede ser la respuesta correcta. Hay preguntas que el corpus no cubre, fuentes contradictorias, permisos insuficientes o evidencia demasiado débil.

La decisión es una regla de umbral, la **opción de rechazo** clásica de la clasificación: responder solo cuando la confianza supera un umbral, y abstenerse si no.[15]

$$d(q) = \begin{cases} \text{responder}, & \text{si } s(q) \geq \tau \\ \text{abstenerse}, & \text{si } s(q) < \tau \end{cases}$$

SÍMBOLO	SIGNIFICADO
q	Pregunta.
$s(q)$	Soporte estimado: evidencia, scores, citas, groundedness.
τ	Umbral mínimo para responder.
$d(q)$	Decisión final.

La matriz de abstención:

REALIDAD	EL SISTEMA RESPONDE	EL SISTEMA SE ABSTIENE
Hay evidencia suficiente.	Respuesta evaluable.	Abstención innecesaria.
No hay evidencia suficiente.	Respuesta no sostenida.	Abstención correcta.

Métricas útiles:

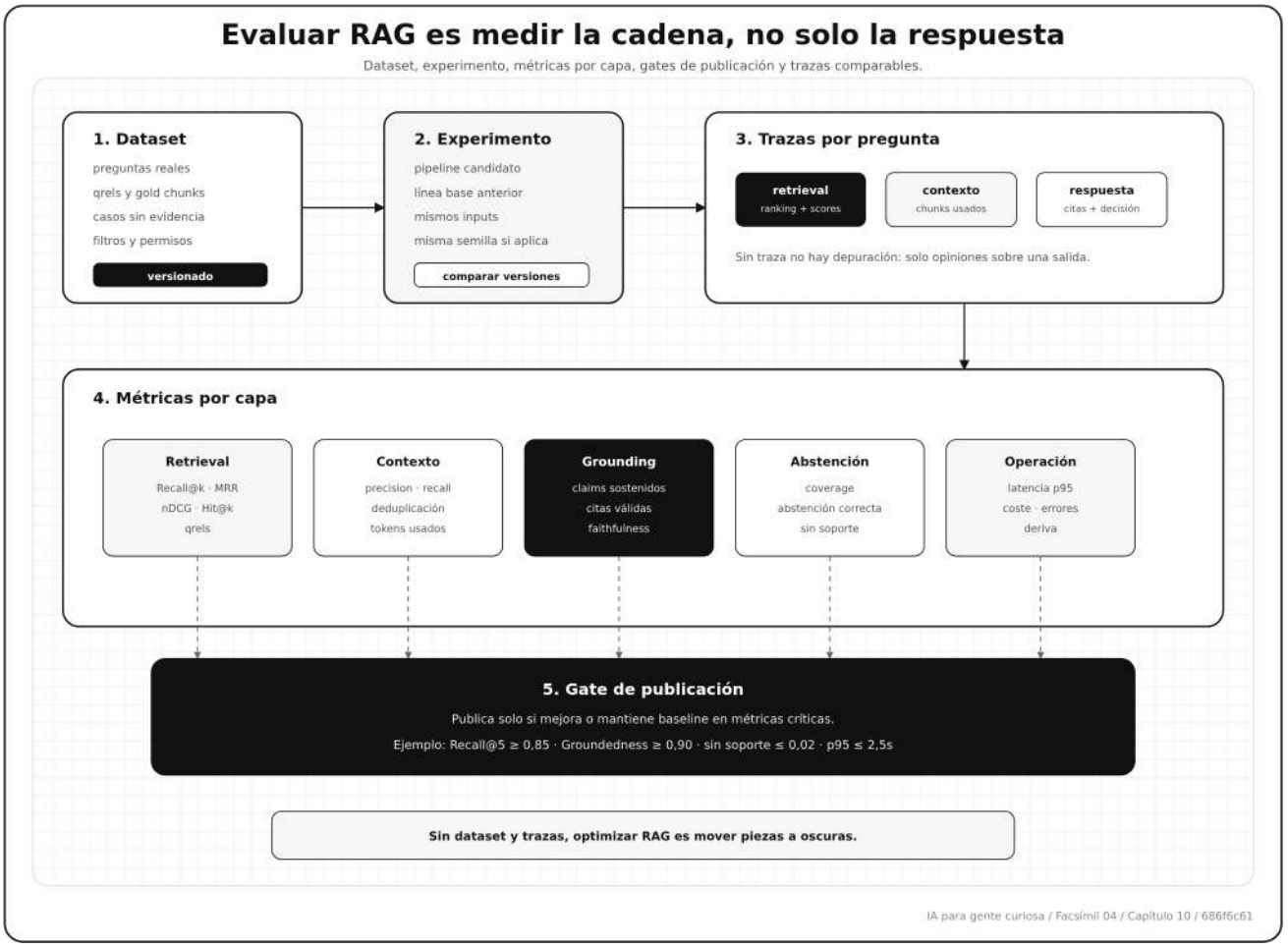
$$\text{Coverage} = \frac{\text{preguntas respondidas}}{\text{preguntas totales}}$$

$$\text{CorrectAbstentionRate} = \frac{\text{abstenciones correctas}}{\text{preguntas no respondibles}}$$

$$\text{UnsupportedAnswerRate} = \frac{\text{respuestas sin soporte}}{\text{preguntas no respondibles}}$$

Subir cobertura no siempre es bueno. Si responde más a costa de inventar más, el sistema empeora. El umbral debe calibrarse con curvas: cuánto ganas en cobertura y cuánto pierdes en precisión de respuesta.

Arquitectura de evaluación



El diagrama tiene una idea central: cada experimento debe producir trazas comparables. Si cambias embeddings, chunking o prompt, no basta con ver una respuesta bonita. Comparas contra baseline y miras qué capa se movió.

Las métricas tienen incertidumbre

Antes de comparar dos versiones, una advertencia que se ignora demasiado: **una métrica medida sobre un dataset es una estimación, no un valor exacto**. Cada métrica es una proporción sobre una muestra de n preguntas, así que tiene un intervalo de confianza, como vimos en el capítulo 1. Para una tasa observada $\hat{p}$ sobre n preguntas, el intervalo al 95 % por la aproximación normal (intervalo de Wald) es:[16]

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

Con $n = 40$ preguntas y un groundedness de 0,82, el margen es de $\pm$0,12: dos versiones que difieren en dos o tres puntos están **dentro del ruido**, y declarar ganadora a una es engañarse. Dos consecuencias prácticas. Primera: reporta siempre n y el intervalo, no solo la media. Segunda: para comparar dos versiones sobre las **mismas** preguntas, usa una prueba pareada (McNemar sobre aciertos por consulta, o un bootstrap), que detecta diferencias reales con menos muestra que comparar dos medias sueltas. Un gate honesto exige no solo superar el umbral, sino superarlo por más que el margen de error.

Gates: decidir si una versión publica

Un gate es una regla de publicación. Evita que un cambio que mejora una métrica rompa otra más importante.

MÉTRICA	UMBRAL EJEMPLO	QUÉ PROTEGE
Recall@5	$\geq 0,85$	Que la evidencia aparezca.
nDCG@5	$\geq 0,80$	Que aparezca arriba.
Groundedness	$\geq 0,90$	Que no añada afirmaciones sin soporte.
Citation precision	$\geq 0,95$	Que las citas sean revisables.
Correct abstention	$\geq 0,85$	Que no responda fuera del corpus.
Unsupported answer rate	$\leq 0,02$	Que no conteste sin evidencia.
Latencia p95	$\leq 2,5s$	Que sea usable.
Coste por respuesta	$\leq presupuesto$	Que sea sostenible.

Los umbrales no salen de una tabla universal. Salen del dominio. Un asistente de lectura puede aceptar más incertidumbre que un asistente que orienta trámites administrativos. Lo importante es que el equipo escriba el umbral antes de mirar si su cambio favorito pasa.

Evaluadores LLM y rúbricas

Los evaluadores LLM son útiles para escalar revisión, pero hay que usarlos con cuidado. Un evaluador no sustituye una rúbrica: ejecuta una rúbrica.

Una rúbrica mínima para groundedness:

```
Evalúa si la respuesta está sostenida por el contexto.

Entrada:
- Pregunta del usuario.
- Contexto recuperado con ids de chunk.
- Respuesta generada.

Devuelve JSON:
{
  "score": 0.0 a 1.0,
  "claims_no_sostenidos": ["..."],
  "citas_invalidas": ["..."],
  "decision": "pasa" | "revisar" | "falla"
}

Criterio:
- 1.0: todas las afirmaciones relevantes están sostenidas.
- 0.5: la respuesta mezcla evidencia con inferencias no citadas.
- 0.0: la respuesta contradice o inventa respecto al contexto.
```

Buenas prácticas:

PRÁCTICA	MOTIVO
Separar evaluación de retrieval y generación.	Un evaluador de respuesta no descubre por sí solo si faltó un chunk.
Guardar entradas del evaluador.	Sin prompt, contexto y respuesta no puedes auditar el score.
Usar muestras revisadas por personas.	El evaluador debe calibrarse contra criterio humano.
Evaluar con varios tipos de pregunta.	Una métrica media puede esconder fallos en tablas, fechas o multi-hop.
Repetir evaluaciones críticas.	Algunos evaluadores tienen variabilidad; mide estabilidad.
No entrenar el sistema para complacer al evaluador.	El objetivo es utilidad verificable, no ganar una métrica estrecha.

Hay una razón de fondo para no tratar al juez como oráculo: tiene **sesgos medidos**. Los evaluadores LLM favorecen la primera respuesta que ven (sesgo de posición), las respuestas más largas (sesgo de verbosidad) y el estilo del propio modelo (autopreferencia).[17] Por eso conviene barajar el orden, controlar la longitud y, sobre todo, **calibrar el juez contra etiquetas humanas** en una muestra antes de fiarte de él a escala. Cuando se usa bien, el

patrón estándar es G-Eval: dar al evaluador una rúbrica explícita y pedirle una cadena de razonamiento antes del score, lo que correlaciona mejor con el juicio humano que pedir una nota a secas.[18]

Herramientas de evaluación que ya existen

No hace falta escribir toda la evaluación desde cero. Hay un ecosistema maduro de herramientas, y conviene conocerlo para elegir en vez de reinventar. La distinción clave es **abierta** (una librería que corres tú, normalmente integrable en tu CI) frente a **gestionada** (un servicio que aloja datasets, trazas y paneles).

HERRAMIENTA	TIPO	FUERTE EN	CUÁNDO ELEGIRLA
RAGAS	Abierta	Métricas de RAG (context precision/recall, faithfulness, answer relevance) sin necesidad de respuesta de oro; generación de datasets sintéticos.[19]	Quieres métricas de RAG estándar rápido y un primer dataset.
TruLens	Abierta	La tríada de RAG y feedback functions sobre trazas.[20]	Quieres instrumentar groundedness y relevancia sobre tu app.
DeepEval	Abierta	Evals como tests unitarios (estilo `pytest`), con asserts sobre métricas.	Quieres que la evaluación falle el build como un test más.
Promptfoo	Abierta	Comparación lado a lado de prompts/modelos y evals en CI.[21]	Comparas versiones y quieres una matriz reproducible.
Arize Phoenix	Abierta	Observabilidad y trazas de LLM/RAG, análisis de fallos.	Quieres ver y depurar trazas, no solo números.
LangSmith	Gestionada	Datasets, trazas, experimentos y evaluadores integrados con LangChain.[22]	Ya usas LangChain y quieres datasets y CI gestionados.
OpenAI Evals / Graders	Mixta	Graders sobre la API y framework de evals reproducibles.[23]	Trabajas sobre la API de OpenAI y quieres graders nativos.

La regla de ingeniería es la de siempre: la herramienta acelera, pero **no decide por ti**. Sea cual sea, sigues necesitando un dataset que represente tu caso, métricas por capa y un gate honesto. Una herramienta que da un número bonito sin un dataset bueno detrás no es evaluación, es decoración.

Evaluación offline, online y sombra

Hay tres modos de evaluación que se complementan.

MODO	QUÉ HACE	CUÁNDO USARLO
Offline	Ejecuta un dataset fijo contra una versión del RAG.	Antes de publicar cambios.
Sombra	Ejecuta una versión candidata con tráfico real sin mostrarla.	Para medir deriva y casos reales sin afectar a usuarios.
Online	Mide interacción real, feedback, coste, latencia y errores.	Cuando el sistema está en uso.

Offline te da repetibilidad. Sombra te da realidad sin exposición directa. Online te da señales de producto. Ninguna sustituye a las otras.

En el día a día

En el día a día, evaluar un RAG es lo que convierte «creo que funciona» en «sé que funciona, y aquí está la prueba». Es el dataset de 50 preguntas con su fuente esperada que corres antes de publicar cada cambio; es la métrica que distingue si una respuesta mala fue culpa del retrieval, del contexto o de la generación; es el número que enseñas cuando alguien pregunta «¿y esto cuánto acierta?». Sin evaluación, cada cambio en el chunking, el modelo o el prompt es una apuesta a ciegas.

La parte delicada es que evaluar bien cuesta disciplina: hay que construir el dataset con casos reales y no respondibles, medir por capas en vez de mirar solo la respuesta final, y resistir la tentación de quedarse con la media cuando el promedio alto esconde fallos graves por tipo de pregunta.

Por qué debería importarte

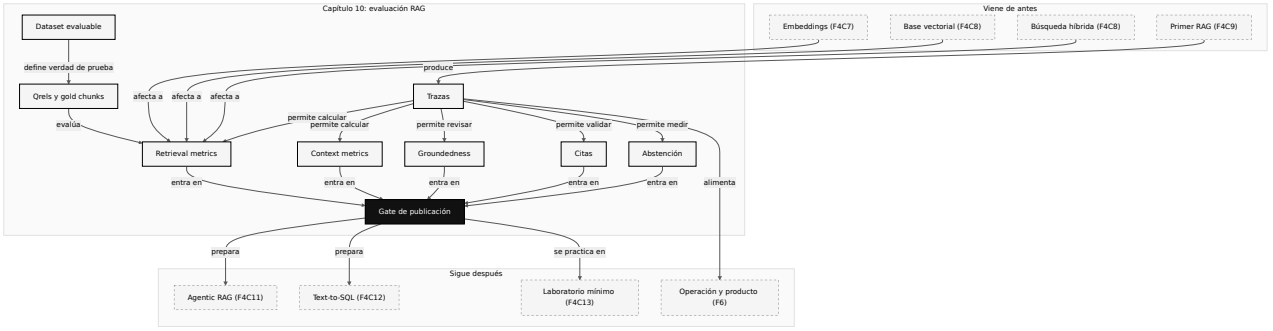
Porque es lo único que separa un RAG que mejora de uno que solo cambia. Sin un conjunto de evaluación, optimizar es mover parámetros y mirar tres ejemplos bonitos; con él, cada decisión (subir top-k, cambiar de embeddings, añadir un reranker) se valida con evidencia y se puede defender. La evaluación es también la condición para tener *gates*: reglas que deciden si una versión publica o no, en lugar de fiarlo todo a la intuición de quien la prueba.

Y porque medir por capas es lo que te hace eficiente diagnosticando. Cuando una respuesta sale mal, la evaluación te dice dónde mirar (retrieval, contexto, generación o abstención) en vez de obligarte a adivinar. Eso es la diferencia entre arreglar la causa y parchear el síntoma.

Dónde solía tropezar yo

ERROR	POR QUÉ ES UN ERROR	ANTÍDOTO
Evaluar solo la respuesta final	No sabes si falló retrieval, contexto o generación.	Guardar trazas y medir por capas.
Usar tres preguntas elegidas a mano	La demo se adapta a tus expectativas.	Crear dataset con casos reales y no respondibles.
Optimizar recall subiendo top-k sin límite	Traes más evidencia, pero también más ruido y coste.	Medir recall, precision, nDCG y tokens de contexto.
Confiar ciegamente en un evaluador LLM	El evaluador también se equivoca y depende de la rúbrica.	Calibrarlo con revisión humana y guardar entradas.
No medir abstención	El sistema aprende a contestarlo todo.	Incluir preguntas sin evidencia y umbrales.
No versionar corpus e índice	No puedes reproducir por qué respondió algo.	Guardar versión de corpus, embeddings, chunks y prompt.
Mezclar cambios	Si mejoras o empeoras, no sabes por qué.	Cambiar una variable por experimento.
Mirar solo medias	Un promedio alto oculta fallos graves por tipo de pregunta.	Reportar métricas por segmento: tabla, multi-hop, fecha, rol.

Cómo encaja todo



Vocabulario aprendido

TÉRMINO	RESPONDE A	DEFINICIÓN ÚTIL
Evaluación offline	¿Mejora antes de publicar?	Prueba repetible sobre dataset fijo.
Evaluación online	¿Qué ocurre en uso real?	Medición con tráfico, feedback, coste y latencia.
Evaluación sombra	¿Cómo pruebo sin mostrar al usuario?	Ejecutar versión candidata en paralelo y registrar resultados.
Qrel	¿Qué chunk debería recuperar?	Juicio de relevancia pregunta-documento.
Precision@k	¿Cuánto ruido hay en top-k?	Proporción de resultados relevantes entre los k primeros.
Recall@k	¿Apareció la evidencia esperada?	Proporción de evidencias recuperadas.
Hit@k	¿Aparece al menos una evidencia?	Indicador binario de recuperación suficiente mínima.
MRR	¿Cuán pronto aparece la primera evidencia?	Media del inverso de la primera posición relevante.
nDCG	¿Lo más útil aparece arriba?	Métrica con relevancia graduada y descuento por posición.
Context precision	¿El contexto final está limpio?	Proporción de chunks útiles dentro del contexto usado.
Context recall	¿El contexto contiene lo necesario?	Proporción de evidencia esperada incluida en el prompt.
Groundedness	¿La respuesta se apoya en contexto?	Claims sostenidos por evidencia recuperada.
Citation precision	¿Las citas usadas son válidas?	Citas que apuntan a evidencia real entre citas usadas.
Citation recall	¿Cité toda la evidencia necesaria?	Evidencias necesarias citadas entre evidencias esperadas.
Coverage	¿Cuánto responde el sistema?	Porcentaje de preguntas no abstendidas.
Gate	¿Publicamos esta versión?	Regla de aceptación con umbrales técnicos y de producto.

TÉRMINO	RESPONDE A	DEFINICIÓN ÚTIL
Evaluador LLM	¿Quién puntúa respuestas abiertas?	Modelo evaluador guiado por una rúbrica auditable.

Antes de pasar página

- [ ] ¿Sé explicar por qué evaluar una respuesta final no basta? (Si no, vuelve a «Qué significa evaluar por capas».)
- [ ] ¿Sé construir una fila mínima de dataset de evaluación? (Si no, vuelve a «Dataset de evaluación: el corazón del sistema».)
- [ ] ¿Sé qué es un qrel y cuándo usar relevancia graduada? (Si no, vuelve a «Qrels y relevancia graduada».)
- [ ] ¿Sé calcular precision@k, recall@k, hit@k, MRR y nDCG? (Si no, vuelve a «Métricas de retrieval».)
- [ ] ¿Sé separar retrieval metrics de context metrics? (Si no, vuelve a «Métricas de contexto».)
- [ ] ¿Sé evaluar groundedness separando claims y evidencia? (Si no, vuelve a «Groundedness, faithfulness y citas».)
- [ ] ¿Sé medir precisión y recall de citas? (Si no, vuelve a «Groundedness, faithfulness y citas».)
- [ ] ¿Sé construir una matriz de abstención? (Si no, vuelve a «Evaluar abstención».)
- [ ] ¿Sé definir gates de publicación con umbrales explícitos? (Si no, vuelve a «Gates: decidir si una versión publica».)
- [ ] ¿Sé usar un evaluador LLM sin tratarlo como oráculo? (Si no, vuelve a «Evaluadores LLM y rúbricas».)
- [ ] ¿Sé guardar trazas suficientes para comparar versiones? (Si no, vuelve a «Arquitectura de evaluación».)

En resumen

IDEA FUERZA	DETALLE
Evaluar RAG es evaluar una cadena.	Corpus, parsing, retrieval, contexto, generación, citas y abstención.
Retrieval se mide antes del LLM.	Si no recuperas la evidencia, la generación no puede arreglarlo.
Groundedness exige claims.	No basta con una sensación global de respuesta correcta.
Citar también se evalúa.	Una cita debe sostener la frase que acompaña.
Abstenerse puede ser correcto.	Coverage alto con respuestas sin soporte es mala señal.
Un evaluador LLM necesita rúbrica.	La herramienta puntúa; el criterio lo diseña el equipo.
Un gate protege producto.	Publicas si la versión candidata supera umbrales críticos.
Sin trazas no hay aprendizaje.	Cada consulta debe dejar ranking, contexto, respuesta, citas y costes.

Para saber más

Arize Phoenix. (2026). *Evaluate RAG*. https://arize.com/docs/phoenix/cookbook/evaluation/evaluate-rag

Arize Phoenix. (2026). *Evaluation concepts*. https://arize.com/docs/phoenix/evaluation/concepts-evals/evaluation

Es, S., James, J., Espinosa-Anke, L. y Schockaert, S. (2023). *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. https://arxiv.org/abs/2309.15217

LangChain. (2026). *Evaluate a RAG application*. https://docs.langchain.com/langsmith/evaluate-rag-tutorial

LlamaIndex. (2026). *Evaluation modules*. https://developers.llamaindex.ai/python/framework/module_guides/evaluating/modules/

OpenAI. (2026). *Graders*. https://developers.openai.com/api/docs/guides/graders

Ragas. (2026). *List of available metrics*. https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/

Thakur, N. y otros (2021). *BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models*. https://arxiv.org/abs/2104.08663

TruLens. (2026). *RAG Triad*. https://www.trulens.org/getting_started/core_concepts/rag_triad/

Notas

1. Es, S., James, J., Espinosa-Anke, L. y Schockaert, S. (2023). *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. https://arxiv.org/abs/2309.15217. ↩

2. Ragas. (2026). *List of available metrics*. https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/. Consultado el 25 de mayo de 2026. ↩

3. TruLens. (2026). *RAG Triad*. https://www.trulens.org/getting_started/core_concepts/rag_triad/. Consultado el 25 de mayo de 2026. ↩

4. LangChain. (2026). *Evaluate a RAG application*. https://docs.langchain.com/langsmith/evaluate-rag-tutorial. LlamaIndex. (2026). *Evaluation modules*. https://developers.llamaindex.ai/python/framework/module_guides/evaluating/modules/. Arize Phoenix. (2026). *Evaluate RAG*. https://arize.com/docs/phoenix/cookbook/evaluation/evaluate-rag. Consultado el 25 de mayo de 2026. ↩

5. OpenAI. (2026). *Graders*. https://developers.openai.com/api/docs/guides/graders. Consultado el 25 de mayo de 2026. ↩

6. Shahul Es y otros (2023). *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. https://arxiv.org/abs/2309.15217. Documentación de generación de datasets: https://docs.ragas.io. ↩

7. Christopher D. Manning, Prabhakar Raghavan e Hinrich Schütze. (2008). *Introduction to Information Retrieval*. Cambridge University Press, cap. 8. https://nlp.stanford.edu/IR-book/. Para nDCG: Kalervo Järvelin y Jaana Kekäläinen. (2002). *Cumulated Gain-based Evaluation of IR Techniques*. ACM TOIS, 20(4), 422-446. https://doi.org/10.1145/582415.582418. ↩

8. Thakur, N. y otros (2021). *BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models*. https://arxiv.org/abs/2104.08663. Muennighoff, N. y otros (2023). *MTEB: Massive Text Embedding Benchmark*. https://arxiv.org/abs/2210.07316. ↩

9. Ragas, 2026. ↩

10. Shahul Es y otros (2023). *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. https://arxiv.org/abs/2309.15217. ↩

11. TruLens, 2026. ↩

12. Shahul Es y otros (2023). *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. https://arxiv.org/abs/2309.15217. ↩

.3. OpenAI, 2026. ↩

.4. TruLens. (2026). *RAG Triad*. https://www.trulens.org/getting_started/core_concepts/rag_triad/. ↩

.5. C. K. Chow. (1970). *On Optimum Recognition Error and Reject Tradeoff*. IEEE Transactions on Information Theory, 16(1), 41-46. https://doi.org/10.1109/TIT.1970.1054406. ↩

.6. El intervalo de Wald usa la aproximación normal de la binomial; para n pequeño 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. ↩

.7. Lianmin Zheng y otros (2023). *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. NeurIPS Datasets and Benchmarks. https://arxiv.org/abs/2306.05685. ↩

.8. Yang Liu y otros (2023). *G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment*. EMNLP. https://arxiv.org/abs/2303.16634. ↩

.9. Shahul Es y otros (2023). *RAGAS*. https://arxiv.org/abs/2309.15217. Documentación: https://docs.ragas.io. ↩

!0. TruLens. (2026). *RAG Triad*. https://www.trulens.org. ↩

!1. promptfoo. (2026). *LLM evals & red teaming*. https://www.promptfoo.dev. ↩

!2. LangChain. (2026). *Evaluate a RAG application*. https://docs.smith.langchain.com. ↩

!3. OpenAI. (2026). *Graders*. https://developers.openai.com/api/docs/guides/graders. ↩