Facsímil 07

Evaluar, calibrar e interpretar

Capítulo 03

Evaluar RAG: retrieval, groundedness y abstención

686f6c61 · 2026

Capítulo 03: Evaluar RAG: retrieval, groundedness y abstención

Entrando en el tema

En el facsímil 4, capítulo 09 montamos un RAG básico: documentos, chunks, embeddings, búsqueda, contexto y respuesta. En el facsímil 4, capítulo 10 vimos una primera idea de evaluación: no mirar solo si la respuesta "suena bien". Ahora toca hacerlo con mentalidad de ingeniería.

Un RAG no es un modelo que responde con magia documental. Es una cadena. Entra una pregunta, se busca evidencia, se ordenan fragmentos, se empaqueta contexto, el modelo genera una respuesta, se citan fuentes y se decide si había base suficiente para responder. Si una pieza falla, la respuesta final puede seguir pareciendo bonita. Ese es el peligro.

La idea central del capítulo es esta: **evaluar RAG es seguir la cadena de custodia de la evidencia**. No basta con preguntar "¿la respuesta está bien?". Hay que poder decir dónde se rompió el sistema: corpus, chunking, retrieval, ranking, contexto, generación, citas, abstención, coste o latencia.

Al terminar deberías poder hacer algo concreto: coger trazas de un RAG, calcular métricas de recuperación, revisar si las afirmaciones están sostenidas, medir si las citas cubren lo importante, comprobar si el sistema se abstiene cuando toca y producir una decisión de release. Esto conecta directamente con el capítulo 01: una eval no existe para decorar una demo, existe para permitir una decisión.

Fecha de corte del estado del arte

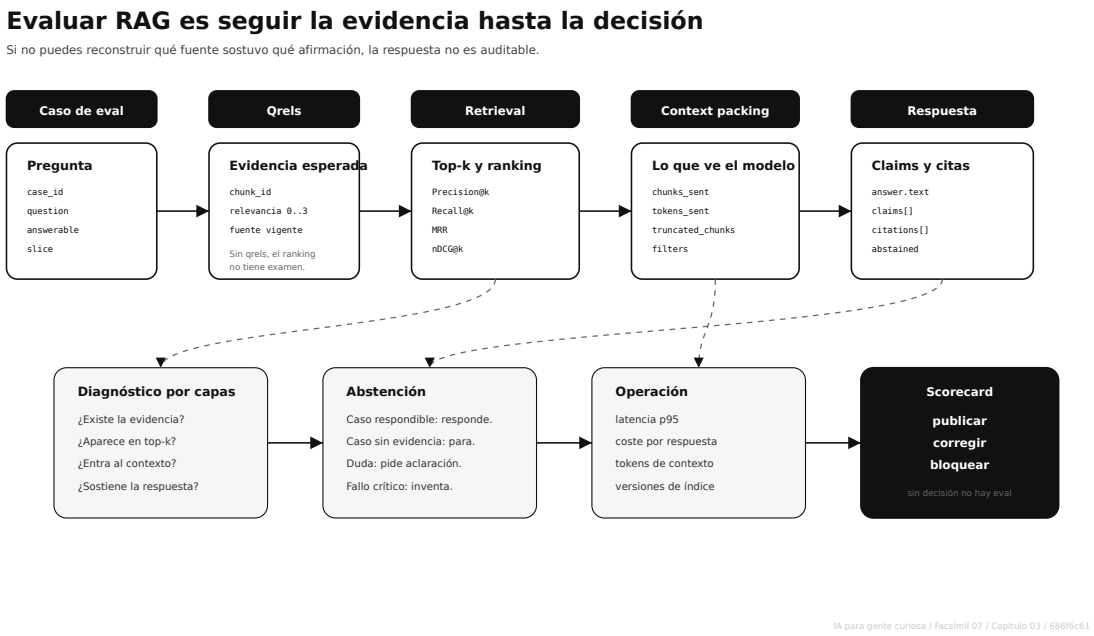
Fecha de corte: 31 de mayo de 2026.

RAG aparece formalizado en el trabajo de Lewis y otros como una forma de combinar recuperación y generación para tareas intensivas en conocimiento.[1] La evaluación de retrieval se apoya en una tradición anterior de recuperación de información, rankings y relevancia graduada. BEIR y MTEB son referencias modernas para comparar modelos y tareas de retrieval/embeddings en dominios variados.[2]

En aplicaciones RAG actuales, herramientas como RAGAS, TruLens, LangSmith, LlamaIndex y Phoenix separan señales como relevancia del contexto, faithfulness/groundedness, respuesta, citas y trazas.[3]

Conviene distinguir dos cosas. Las métricas de ranking como Precision@k, Recall@k, MRR o nDCG vienen de recuperación de información y tienen una formulación académica estable. En cambio, “groundedness”, “faithfulness”, “citation quality” o “abstention quality” suelen ser definiciones operativas de una herramienta o de un equipo. Son útiles, pero no deben presentarse como si fueran una ley universal. En este capítulo las uso como señales de ingeniería: se definen, se auditan y se conectan con una decisión.

Anatomía de una evaluación RAG



> La evaluación RAG útil no empieza en la respuesta final. Empieza en el caso, los qrels y la evidencia que debería aparecer.

El diagrama resume la disciplina que vamos a seguir. Primero se define el caso de evaluación. Después se declara qué evidencia sería válida. Luego se mira si el retriever la encuentra, si el ranking la coloca arriba, si el contexto que entra al modelo la conserva, si la respuesta la usa y si las citas permiten auditar lo dicho.

Esto tiene una consecuencia muy práctica: cuando una respuesta falla, no dices “el RAG es malo”. Dices algo más útil: “el chunk correcto no existía”, “existía pero no fue recuperado”, “fue recuperado pero quedó fuera del contexto”, “entró al contexto pero el modelo añadió una afirmación sin soporte”, o “no había evidencia y aun así respondió”.

Qué se mide antes de llamar al modelo

La primera tentación es evaluar la respuesta final. Al usuario, al fin y al cabo, le importa la respuesta. Pero para ingeniería es demasiado tarde. Si la evidencia no llega al contexto, el generador no puede resolverlo de forma fiable. Por eso el primer bloque de evaluación se hace solo con ranking.

Definimos estos símbolos:

SÍMBOLO	SIGNIFICADO	EJEMPLO
q	Pregunta evaluada.	"¿Puedo ampliar matrícula si tengo un pago pendiente?"
Q	Conjunto de preguntas de evaluación.	Dataset de 200 preguntas reales o revisadas.
G_q	Chunks relevantes esperados para q.	`{normativa#plazos, normativa#pagos}`
$R_k(q)$	Lista de los k primeros chunks recuperados.	Top 3 devuelto por el retriever.
rel_i	Relevancia graduada del resultado en la posición i.	0, 1, 2 o 3.
$rank_q$	Posición de la primera evidencia relevante.	1 si aparece arriba del todo.

Con eso podemos usar métricas académicas de recuperación de información:

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

En palabras: de los k documentos que has traído, cuántos eran realmente evidencia esperada. Si top-3 trae un chunk bueno y dos irrelevantes, Precision@3 es $1/3$.

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

En palabras: de toda la evidencia que hacía falta, cuánta apareció entre los k primeros. Si la pregunta necesitaba dos chunks y solo aparece uno, Recall@3 es $1/2$. Esta métrica es muy importante en preguntas que necesitan varias piezas.

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

En palabras: basta con que aparezca una evidencia válida. Es una métrica rápida, pero puede engañar: una sola fuente útil no basta si la respuesta necesitaba dos.

$$\mathrm{RR}(q) = \frac{1}{rank_q}$$

$$\mathrm{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \mathrm{RR}(q)$$

En palabras: premia que la primera evidencia útil aparezca pronto. Si la primera evidencia buena aparece en posición 1, RR vale 1. Si aparece en posición 4, vale 0,25.

Cuando la relevancia no es solo “sí/no”, usamos ganancia acumulada. Järvelin y Kekäläinen formalizaron DCG y nDCG para rankings con relevancia graduada.[4]

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

$$\mathrm{nDCG@k}(q) = \frac{\mathrm{DCG@k}(q)}{\mathrm{IDCG@k}(q)}$$

En palabras: un chunk de relevancia 3 en la posición 1 pesa mucho más que un chunk de relevancia 1 en la posición 5. nDCG divide por el ranking ideal para que el resultado quede normalizado.

MÉTRICA	QUÉ DETECTA	QUÉ NO DETECTA
Precision@k	Ruido en los primeros resultados.	Si falta una segunda fuente necesaria.
Recall@k	Cobertura de evidencia esperada.	Si la evidencia aparece demasiado abajo para tu top-k real.
Hit@k	Si aparece al menos una fuente útil.	Si la respuesta necesita varias fuentes.
MRR	Si lo primero útil aparece pronto.	Si el resto del contexto está lleno de ruido.
nDCG@k	Orden con relevancia graduada.	Si la respuesta final cita bien.

Estas fórmulas no son decoración: sirven para separar retrieval de generación. Si Recall@k ya es bajo antes de llamar al LLM, no tiene sentido gastar diez horas ajustando el prompt. Primero debes mirar corpus, chunking, filtros, embeddings, búsqueda híbrida, reranking o reescritura de consulta.

Qrels: el examen del retrieval

Un qrel es un juicio de relevancia. En lenguaje sencillo: para una pregunta concreta, alguien declara qué chunks deberían aparecer y con qué fuerza. No es glamuroso, pero sostiene toda la evaluación.

Un qrel mínimo debería guardar:

CAMPO	QUÉ GUARDA	POR QUÉ IMPORTA
`case_id`	Identificador estable.	Permite comparar versiones sin perder trazabilidad.
`question`	Pregunta evaluada.	Debe parecerse a uso real, no a una frase perfecta de laboratorio.
`answerable`	Si el corpus permite responder.	Activa métricas de abstención.
`gold_chunks`	Chunks esperados y relevancia.	Permite Recall@k, MRR y nDCG@k.
`slice`	Segmento de análisis.	Evita que la media esconda fallos por fuente, idioma o producto.
`why_it_exists`	Motivo de incluir el caso.	Evita datasets decorativos.

La relevancia graduada suele bastar con cuatro niveles:

VALOR	LECTURA PRÁCTICA	EJEMPLO
0	No aporta evidencia.	Documento parecido, pero de otro curso.
1	Relacionado, insuficiente.	FAQ general sin la condición clave.
2	Útil para parte de la respuesta.	Fragmento con el plazo, pero no con excepciones.
3	Evidencia central.	Fragmento que sostiene la afirmación principal.

En un proyecto serio, los qrels se revisan como código: diff, propietario, fecha, fuente, motivo y vínculo al documento. Si cambian los qrels, puede cambiar la métrica aunque el sistema no haya cambiado. Por eso el dataset de evaluación también debe versionarse.

Groundedness, citas y abstención

Groundedness no significa que una respuesta “suene bien”. Significa que las afirmaciones importantes de la respuesta están sostenidas por evidencia recuperada. RAGAS habla de faithfulness y otras métricas de contexto/respuesta; TruLens resume su evaluación RAG con una tríada: relevancia de contexto, groundedness y relevancia de respuesta. La idea común es separar “me gusta la respuesta” de “puedo defender cada afirmación con el contexto”.

Para trabajar con groundedness de forma auditable, no empieces por una nota global. Empieza por claims:

PASO	QUÉ HACES	EVIDENCIA QUE PRODUCES
1	Separas la respuesta en afirmaciones verificables.	`claims[]`
2	Para cada afirmación, apuntas qué chunk la sostiene.	`supporting_chunks[]`
3	Marcas afirmaciones sin soporte.	diagnóstico por caso
4	Revisas si las citas cubren lo importante.	`citation_precision`, `citation_recall` en la práctica
5	Bloqueas si hay una respuesta sin evidencia en un caso no respondible.	fallo crítico

Aquí no voy a disfrazar una definición operativa de fórmula universal. En el cuaderno del facsímil, `groundedness` se calcula como proporción de claims con soporte explícito. Es una decisión práctica para enseñar el mecanismo. En una herramienta real, ese cálculo puede hacerse con reglas, revisión humana, evaluadores LLM, embeddings o combinaciones. Lo importante es que la traza permita auditar por qué una afirmación se considera sostenida.

Las citas también tienen dos lecturas distintas:

SEÑAL	PREGUNTA QUE RESPONDE	ERROR TÍPICO
Precisión de cita	De las citas usadas, ¿cuántas apuntan a evidencia válida?	Citar un documento real que no sostiene la frase.
Cobertura de cita	De la evidencia esperada, ¿cuánta aparece citada?	Responder con una fuente parcial y olvidar la condición importante.

La abstención es la tercera pata. Un RAG serio no solo responde. También sabe decir “con la evidencia disponible no puedo afirmarlo”. Esto no es un gesto de humildad estética: es una política de producto. En un asistente de normativa, soporte técnico, salud, finanzas, compliance o documentación interna, responder sin evidencia puede ser peor que no responder.

CASO	QUÉ DEBERÍA PASAR	QUÉ MIDES
Pregunta respondible y evidencia recuperada.	Responder con citas.	Ranking, groundedness, citas, coste y latencia.
Pregunta respondible, pero evidencia fuera de top-k.	Abstenerse o pedir más contexto.	Retrieval y política de abstención.
Pregunta no respondible por el corpus.	Abstenerse con explicación breve.	Abstención correcta y fallos críticos.
Pregunta ambigua.	Pedir aclaración.	Contrato conversacional y trazas.

Diagnóstico por capas

Cuando un RAG falla, no cambies todo a la vez

El síntoma te dice qué capa mirar primero y qué acción tiene sentido.

Síntoma	Capa probable	Señal	Acción técnica	No hagas esto primero
Recall bajo falta evidencia esperada	corpus, chunking, filtros embedding o búsqueda híbrida	Recall@k, Hit@k qrels por slice	revisar índice y fuentes probar BM25 + vector	cambiar el modelo generador sin mirar retrieval
nDCG bajo la evidencia aparece tarde	ranking o reranking orden de contexto	MRR, nDCG@k posición de gold chunks	añadir reranker o RRF medir latencia p95	subir top-k sin mirar ruido ni coste de contexto
Claims sin soporte la respuesta añade de más	prompt, contrato de salida o contexto incompleto	groundedness operativo claims con soporte	cita por afirmación validar salida estructurada	aceptar una cita al final como si cubriera todo
No se abstiene responde sin evidencia	política de respuesta y casos no respondibles	abstention accuracy critical failures	bloquear release añadir regresiones	premiar respuestas largas sin evidencia verificable
P95 o coste sube la mejora no cabe en operación	top-k, reranker, tokens y modelo generador	p95, tokens, coste por respuesta aceptada	comparar variantes con restricciones	elegir la métrica más alta ignorando el SLO

Una evaluación RAG debe producir diagnóstico accionable. Si la salida solo dice “calidad 0,72”, todavía no te ayuda a arreglar nada.

Cuando un RAG falla, el orden de trabajo importa. Si Recall@k es bajo, no empieces retocando el prompt. Si Recall@k es alto pero groundedness baja, mira el contexto final y el contrato de salida. Si la respuesta tiene citas, pero no sostienen las afirmaciones, el problema no es “falta citar”, sino “falta trazabilidad claim-cita”. Si todo mejora pero p95 y coste se disparan, el candidato quizá sirve para un flujo de revisión, pero no para producción.

SÍNTOMA	QUÉ MIRAR PRIMERO	CAMBIO RAZONABLE
Recall@k bajo.	Corpus, filtros, embeddings, búsqueda híbrida, query rewriting.	Mejorar qrels, chunking, índice o estrategia de búsqueda.
Recall alto, nDCG bajo.	Orden de resultados.	Añadir reranker, RRF o señales de metadata.
Retrieval correcto, groundedness baja.	Prompt, contrato de citas, claims y contexto final.	Exigir respuesta con soporte y validar afirmaciones.
Citas presentes pero débiles.	Asociación claim-cita.	Citas por afirmación, no bibliografía decorativa.
Buena calidad, coste alto.	Top-k, reranker, tamaño de chunks, cache, modelo.	Reducir contexto, cachear retrieval o usar ruta escalonada.
Responde sin evidencia.	Casos no respondibles, umbral y contrato de abstención.	Endurecer política y añadir regresiones.
Funciona en media, falla en un grupo.	Slices.	Métricas por idioma, producto, fuente, fecha o perfil.

Trazas de depuración

Una evaluación sin traza solo dice “falló”. Una evaluación con traza te dice dónde mirar. Una traza útil no guarda solo la respuesta final; guarda versiones, filtros, scores, tokens, latencia y coste.

```
{
  "run_id": "rag-run-2026-05-31-00042",
  "case_id": "rag_002",
  "pipeline_version": "rag-pipeline-v0.7.3",
  "corpus_version": "normativa-campus-2026-05-30",
  "index_version": "hnsw-embeddings-2026-05-30",
  "retriever": {
    "type": "hybrid_rrf",
    "top_k_sparse": 20,
    "top_k_dense": 20,
    "top_k_final": 5,
    "filters": {"course": "2026", "document_status": "vigente"}
  },
  "retrieved_chunks": [
    {
      "rank": 1,
      "chunk_id": "normativa-2026#plazos-ampliacion",
      "score_dense": 0.88,
      "score_sparse": 12.4,
      "tokens": 170
    }
  ],
  "context": {
    "chunks_sent": ["normativa-2026#plazos-ampliacion"],
    "tokens_sent": 170,
    "truncated_chunks": []
  },
  "answer": {
    "abstained": false,
    "citations": ["normativa-2026#plazos-ampliacion"],
    "claims_count": 2,
    "output_tokens": 78
  },
  "latency_ms": {"retrieval": 42, "generation": 920, "total": 1115},
  "estimated_cost": {"generation": 0.0038, "total": 0.004}
}
```

Con una traza así puedes contestar preguntas reales:

PREGUNTA	CAMPO QUE LA RESPONDE
¿Cambiamos el índice sin darnos cuenta?	`index_version`
¿El filtro de curso estaba activo?	`retriever.filters`
¿La evidencia se recuperó pero quedó fuera del contexto?	`retrieved_chunks` y `context.chunks_sent`
¿El contexto se recortó?	`truncated_chunks`
¿La respuesta se encareció por salida larga?	`answer.output_tokens`
¿El fallo viene de generación o retrieval?	comparación entre `retrieved_chunks`, `context` y `answer`

Comparar variantes sin hacerse trampas

Un RAG no mejora por intuición. Mejora comparando variantes controladas. La palabra importante es **controladas**: si cambias embeddings, chunking, reranker, prompt y modelo a la vez, quizá suba una métrica, pero no sabrás qué pieza produjo la mejora.

Una matriz mínima de experimentos:

VARIANTE	QUÉ CAMBIA	QUÉ QUEDA FIJO	MÉTRICAS QUE MIRAS
`bm25_base`	Búsqueda léxica.	Corpus, chunks, prompt, modelo.	Recall@k, nDCG, latencia.
`dense_base`	Embedding denso.	Corpus, chunks, top-k, prompt, modelo.	Recall@k, MRR, coste de índice.
`hybrid_rrf`	Fusión BM25 + vector.	Corpus, chunks, prompt, modelo.	Recall@k, precision@k, nDCG.
`hybrid_rerank`	Añade reranker.	Corpus, chunks, generador.	nDCG, groundedness, latencia p95.
`chunk_300`	Chunks más pequeños.	Índice, prompt, modelo.	Recall@k, citation recall, tokens.
`chunk_900`	Chunks más grandes.	Índice, prompt, modelo.	Groundedness, ruido, coste.
`topk_8`	Más contexto.	Retriever, chunks, prompt, modelo.	Recall, precisión de contexto, tokens.
`strict_abstain`	Umbral más exigente.	Retriever, corpus, respuesta.	Abstención correcta, cobertura y satisfacción.

RRF, Reciprocal Rank Fusion, es una técnica sencilla para fusionar rankings de varios sistemas y suele usarse al combinar búsqueda léxica y vectorial.[5] No necesitas venderla como solución mágica. Necesitas medir si, para tu corpus, sube cobertura sin disparar latencia, coste o ruido.

También debes protegerte de la fuga de evaluación. Si usas los mismos casos para ajustar prompts, retocar chunks, cambiar filtros y declarar victoria, has entrenado contra el examen. Separa al menos tres conjuntos:

CONJUNTO	USO CORRECTO	QUÉ NO DEBERÍAS HACER
`dev`	Ajustar prompts, top-k, chunking y umbrales.	Presentarlo como resultado final.
`regression`	Vigilar errores conocidos que no deben volver.	Usarlo como único dataset de calidad.
`holdout`	Estimar rendimiento antes de publicar.	Mirarlo cada vez que retocas el sistema.

Si el dataset es pequeño, una mejora de 2 puntos puede ser ruido. Para comparaciones pareadas puedes usar bootstrap para estimar incertidumbre o McNemar cuando comparas dos sistemas en aciertos/errores emparejados.[6]

En el día a día

Imagina un asistente de normativa académica. La pregunta es: “¿Puedo ampliar matrícula si tengo un pago pendiente?”. La respuesta correcta necesita dos evidencias: el plazo de ampliación y la regla sobre pagos pendientes. Si el RAG recupera solo el plazo, Hit@3 puede ser 1, pero Recall@3 será 0,5. Si responde “sí, puedes” citando solo el plazo, la respuesta parece útil, pero no está completa.

Ahora imagina soporte interno. Una persona pregunta cómo rotar una clave de producción. El corpus tiene una guía antigua y una nueva. Si el retriever trae la guía antigua porque tiene más coincidencias léxicas, la respuesta puede ser peligrosa aunque esté “citada”. La cita no basta: tienes que comprobar vigencia, metadata, filtros y prioridad.

Otro caso muy común aparece en documentación de producto. El usuario pregunta por una funcionalidad que todavía no existe. El RAG encuentra documentos parecidos, el modelo extrapola y da instrucciones inventadas. Aquí la métrica importante no es solo groundedness: es abstención. Si el sistema no sabe decir “no encuentro evidencia suficiente”, no está listo para producción.

Por qué debería importarte

Porque RAG suele entrar en sistemas donde la fuente importa: políticas internas, normativa, contratos, documentación técnica, manuales, tickets, soporte, compliance o conocimiento cambiante. Si no evalúas la cadena de evidencia, no sabes si tu sistema responde por documentos o por memoria estadística del modelo.

Para ingeniería, esto cambia decisiones concretas:

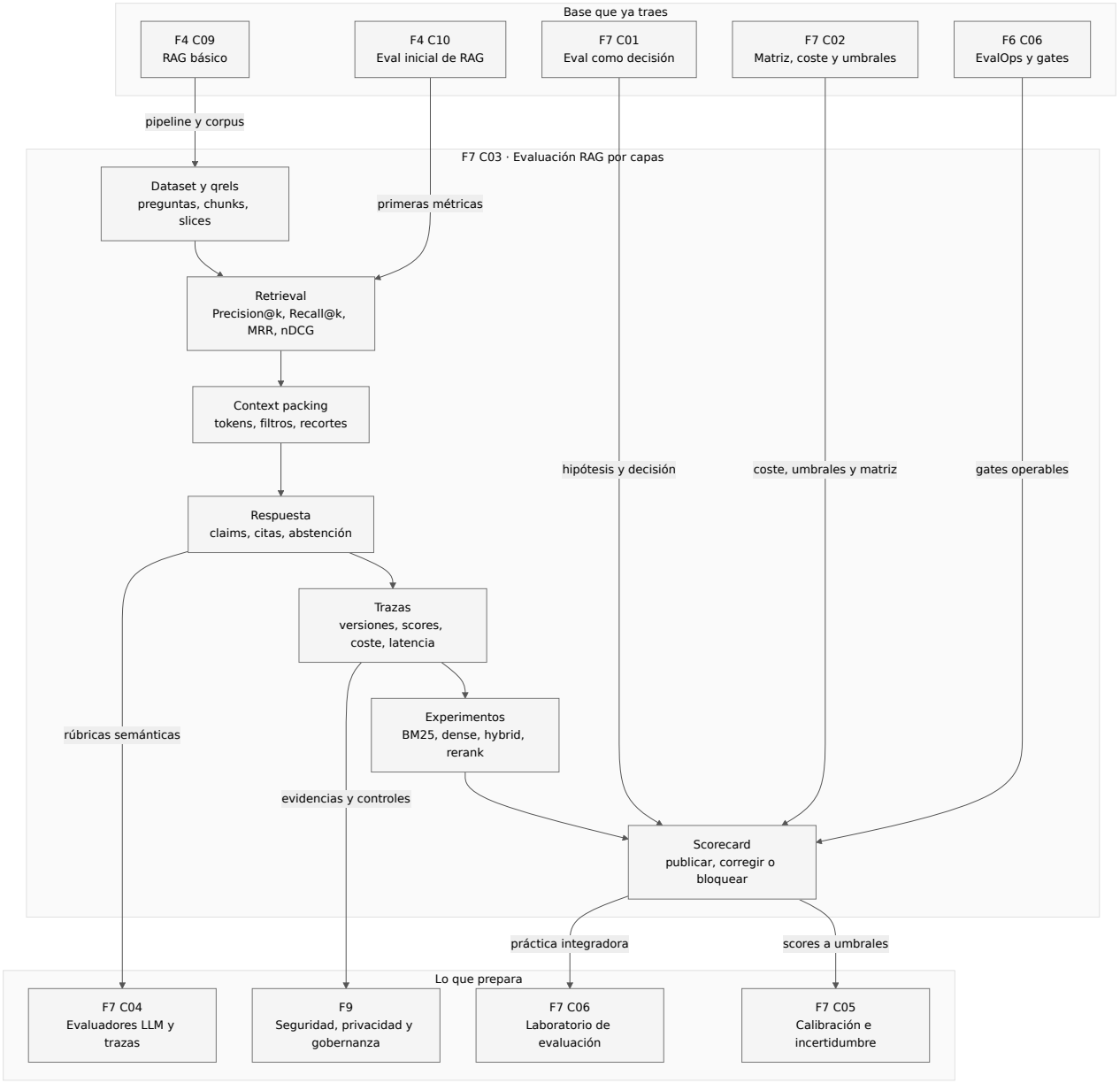
DECISIÓN	SIN EVALUACIÓN POR CAPAS	CON EVALUACIÓN POR CAPAS
Cambiar embeddings	Lo haces porque “parece mejor”.	Lo haces si mejora Recall@k/nDCG por slice sin romper coste.
Subir top-k	Metes más contexto y cruzas dedos.	Mides recall, ruido, tokens y p95.
Añadir reranker	Lo vendes como mejora automática.	Lo publicas solo si sube ranking y cabe en SLO.
Ajustar prompt	Lo usas para tapar fallos de retrieval.	Lo tocas cuando ya sabes que el contexto llega.
Publicar una versión	Mirar ejemplos bonitos.	Scorecard con umbrales y fallos críticos.

Esta forma de trabajar también evita discusiones vagas. Si el equipo de producto dice “responde bien”, ingeniería puede preguntar “¿en qué slices, con qué qrels, con qué tasa de abstención y con qué p95?”. No es pedantería: es proteger la decisión.

Dónde solía tropezar yo

TROPIEZO	POR QUÉ OCURRE	ANTÍDOTO
Medir solo la respuesta final	Si la respuesta falla, no sabes si arreglar corpus, chunking, retrieval, reranking, prompt o citas.	Medir por capas antes de tocar el sistema.
Celebrar Hit@k	Hit@k puede ser 1 aunque falte la segunda fuente necesaria.	Mirar Recall@k y casos multi-evidencia.
Confundir cita con evidencia	Una cita puede apuntar a un documento real y aun así no sostener la frase.	Validar claim por claim.
Subir top-k sin mirar coste	Más contexto puede traer evidencia, pero también ruido, tokens y latencia.	Medir recall, precision, nDCG y coste juntos.
No tener preguntas no respondibles	Si todas las preguntas tienen respuesta, nunca mides abstención.	Incluir casos plausibles sin evidencia.
Cambiar umbrales después de ver el resultado	Convierte la eval en ajuste manual.	Escribir la política antes de ejecutar.
Comparar variantes cambiándolo todo a la vez	Si sube la métrica, no sabes si fue por embeddings, reranker, prompt, chunks o corpus.	Usar una matriz de experimentos con una variable principal por variante.
No versionar el índice	Puedes creer que comparas dos pipelines cuando en realidad cambió el índice.	Guardar `corpus_version`, `chunker_version`, `embedding_model` e `index_version`.
Usar el holdout como zona de pruebas	Si miras el examen final cada vez que ajustas, deja de ser final.	Separar `dev`, `regression` y `holdout`.

Cómo encaja todo



Este capítulo es el puente entre construir un RAG y operar un RAG. El facsímil 4 te da la arquitectura; el facsímil 6 te recuerda que una métrica debe convertirse en gate; C1 y C2 de este facsímil te dan unidad de evaluación, coste y umbrales; C3 aplica todo eso a evidencia documental. C4 usará evaluadores LLM cuando haga falta criterio semántico, pero ahora ya tienes una base medible sin delegarlo todo en otro modelo.

Vocabulario aprendido

TÉRMINO	DEFINICIÓN BREVE
RAG	Arquitectura que combina recuperación de evidencia con generación.
Qrel	Juicio de relevancia entre pregunta y chunk.
Relevancia graduada	Puntuación que diferencia evidencia central, parcial o irrelevante.
Precision@k	Proporción de resultados útiles entre los k primeros.
Recall@k	Proporción de evidencias esperadas que aparecen en top-k.
Hit@k	Si al menos una evidencia esperada aparece en top-k.
MRR	Media del recíproco de la posición de la primera evidencia útil.
nDCG@k	Métrica de ranking que premia relevancia alta en posiciones altas.
Context packing	Selección y ordenación del contexto que entra al modelo.
Groundedness	Comprobación de si las afirmaciones están sostenidas por evidencia recuperada.
Cita válida	Cita que apunta a evidencia que sostiene lo afirmado.
Abstención	No responder cuando falta evidencia suficiente.
Fallo crítico	Respuesta sin evidencia en un caso que debía abstenerse.
Scorecard	Resumen de métricas y restricciones para decidir.
RRF	Técnica para fusionar rankings de varios retrievers.
Traza	Registro de una run: versiones, recuperación, contexto, respuesta, latencia y coste.
Holdout	Conjunto reservado para estimar rendimiento sin usarlo para ajustar.
Variante dominada	Configuración peor o igual en calidad y peor o igual en coste frente a otra.
Shadow	Ejecución paralela de una versión nueva sin responder todavía al usuario.

Antes de pasar página

Antes de avanzar, deberías poder responder:

1. ¿Por qué una respuesta correcta puede seguir siendo mala señal en un RAG?
2. ¿Qué diferencia hay entre Hit@k y Recall@k?
3. ¿Cuándo usarías nDCG@k en vez de solo Recall@k?
4. ¿Qué contiene un qrel útil?
5. ¿Por qué groundedness debe mirar afirmaciones y no impresiones?
6. ¿Qué diferencia práctica hay entre precisión de cita y cobertura de cita?
7. ¿Por qué necesitas preguntas no respondibles en el dataset?
8. ¿Qué métrica o check bloquearía una versión que responde sin evidencia?
9. ¿Qué cambiarías si Recall@k es alto pero groundedness es bajo?
10. ¿Qué debería guardar una traza profesional de RAG?
11. ¿Por qué una matriz de experimentos debe cambiar una pieza cada vez?
12. ¿Qué diferencia hay entre `dev`, `regression` y `holdout`?
13. ¿Qué significa que una variante esté dominada?
14. ¿Qué archivos entrega la práctica del capítulo?

En resumen

IDEA	QUÉ TE LLEVAS
RAG se evalúa por capas.	Corpus, retrieval, contexto, respuesta, citas y abstención tienen señales distintas.
Los qrels sostienen la evaluación.	Sin evidencia esperada no puedes medir retrieval de forma seria.
Retrieval se mide antes de generación.	Ahorras coste y localizas el fallo antes de culpar al modelo.
Las fórmulas académicas están en ranking.	Precision@k, Recall@k, MRR y nDCG@k tienen tradición de IR y fuentes reconocibles.
Groundedness es una señal operativa.	En la práctica se audita claim por claim, no como impresión general.
Abstención es parte de calidad.	Responder sin evidencia puede bloquear una versión.
La scorecard debe decidir.	Métricas sin gate no cambian el sistema.
Las variantes se comparan con diseño.	Una matriz de experimentos evita mejorar a ciegas.
La traza es parte de la eval.	Sin versiones, scores, contexto, coste y latencia no hay depuración seria.
El holdout se protege.	Si optimizas contra el examen final, la mejora deja de ser fiable.

Para saber más

Cormack, G. V., Clarke, C. L. A. y Buettcher, S. (2009). Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. *SIGIR*, 758-759. https://doi.org/10.1145/1571941.1572114

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1), 1-26. https://doi.org/10.1214/aos/1176344552

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

Järvelin, K. y Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4), 422-446. https://doi.org/10.1145/582415.582418

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

Lewis, P. y otros (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems 33*, 9459-9474.

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

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2), 153-157. https://doi.org/10.1007/BF02295996

Muennighoff, N. y otros (2023). *MTEB: Massive Text Embedding Benchmark*. https://arxiv.org/abs/2210.07316

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

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. Lewis, P. y otros (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems 33*, 9459-9474. ↩

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

3. Es, S., James, J., Espinosa-Anke, L. y Schockaert, S. (2023). *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. arXiv:2309.15217. Ragas. (2026). *List of available metrics*. https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/. TruLens. (2026). *RAG Triad*. https://www.trulens.org/getting_started/core_concepts/rag_triad/. LangChain. (2026). *Evaluate a RAG application*. https://docs.langchain.com/langsmith/evaluate-rag-tutorial. Arize Phoenix. (2026). *Evaluate RAG*. https://arize.com/docs/phoenix/cookbook/evaluation/evaluate-rag. ↩

4. Järvelin, K. y Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4), 422-446. https://doi.org/10.1145/582415.582418 ↩

5. Cormack, G. V., Clarke, C. L. A. y Buettcher, S. (2009). Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. *SIGIR*, 758-759. https://doi.org/10.1145/1571941.1572114 ↩

6. Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1), 1-26. McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2), 153-157. ↩