Qué es RAG y cómo funciona: chuleta con código y 5 reglas

> RAG es **buscar en tus documentos los trozos que importan para una pregunta y pegárselos a la IA**, porque el modelo no ha leído tus documentos y no le caben enteros. Funciona en tres pasos: trocear, convertir cada trozo en números (embeddings) y buscar los más cercanos. Si la búsqueda falla, la respuesta también: **usa búsqueda híbrida (significado + palabras clave), trozos con contexto, reordena candidatos, prueba con preguntas reales y pide que cite la fuente.**

Los 3 pasos de un RAG

Paso	Qué hace	Cuándo tocarlo
1. Trocear Ejemplo real: trozos de hasta 1.200 caracteres con 150 de solape.	Corta cada documento en trozos que se buscan por separado. El solape hace que una idea partida por la mitad siga entera en uno de los dos.	Si las respuestas salen cortadas o sin contexto, prueba trozos más grandes o más solape; si recupera paja, trozos más pequeños.
2. Convertir en números Un modelo de embeddings da una lista de números por trozo (384 en el ejemplo).	Textos que dicen cosas parecidas quedan cerca en un «mapa de significados», aunque no compartan palabras.	Si tus documentos son técnicos o están en varios idiomas, elige un modelo de embeddings que los entienda: es lo que más cambia la calidad.
3. Buscar La pregunta también se convierte en números y se cogen los trozos más cercanos (5 en el ejemplo).	Esos trozos se pegan en el prompt junto a la pregunta. El modelo contesta con lo que le llega, y nada más.	Siempre que la respuesta buena exista pero el modelo no la use: el fallo casi nunca está en el modelo, está en lo que le llega.

Dónde falla: la prueba real con el asistente de Frogames

Asistente de dudas de Frogames: 221 cursos, 220.906 trozos. Curso «Ingeniería de LLM» (2.884 trozos). Puesto en el que quedó la clase que responde, con cada forma de buscar. La IA solo lee los 5 primeros.

Pregunta	Clase que responde	Embeddings · BM25 · híbrida
¿Cómo puedo ver en un dibujo cómo se agrupan los vectores de mis textos?	Visualización de incrustaciones (no comparte palabras clave con la pregunta)	**2** · 6 · 5
¿Cuándo uso RAG y cuándo ajuste fino?	Prompting, RAG y ajuste fino: cuándo utilizar cada enfoque	37 · **1** · **1**
¿Qué diferencia hay entre RAG y hacer fine-tuning a un modelo?	La misma	16 · **2** · 6

El código de la búsqueda híbrida, línea a línea

Script completo y listo para ejecutar: busqueda_hibrida.py. Se lanza con `uv run --with sentence-transformers --with rank-bm25 python busqueda_hibrida.py`.

Qué escribes	Qué hace	Cuándo usarlo
`modelo.encode(TROZOS, normalize_embeddings=True)`	Convierte cada trozo en su lista de números, normalizada a longitud 1.	Una vez, al indexar. Guarda el resultado: recalcularlo en cada pregunta es lo más lento.
`np.argsort(-(emb @ q))`	Ordena los trozos por similitud con la pregunta (producto escalar = coseno con vectores normalizados).	Para preguntas escritas con otras palabras: es la búsqueda por significado.
`BM25Okapi([palabras(t) for t in TROZOS])`	Índice de palabras clave: puntúa cuántas veces aparece cada palabra, lo rara que es y la longitud del trozo.	Para siglas, nombres propios, códigos de error o términos técnicos.
`puntos[i] += 1 / (RRF_K + puesto)`	Fusión RRF: cada trozo suma puntos según su puesto en cada lista; con `RRF_K = 60`, ir 1.º o 3.º pesa parecido.	Para mezclar los dos rankings sin tener que comparar puntuaciones de escalas distintas.

Qué escribes	Qué hace	Cuándo usarlo
`hibrida[:K]`	Los `K` mejores trozos tras la fusión: los que van al prompt.	Al final. Si vas a reordenar con otro modelo, pide más (por ejemplo 20) y quédate con 5 después.

Las 5 reglas para un RAG que encuentre lo que buscas

Qué haces	Qué consigues	Cuándo usarlo
1. Búsqueda híbrida. Embeddings + BM25, fusionados.	Cada método cubre los fallos del otro: en la prueba, la clase del puesto 37 pasó al 1.	Siempre que tus documentos tengan siglas, nombres de producto o jerga técnica.
2. Trozos con contexto. Que cada trozo lleve el título de su documento o sección.	Un trozo suelto deja de estar huérfano: se sabe de qué va y de dónde sale.	Cuando un trozo, leído solo, no dice de qué tema o documento es.
3. Más candidatos y reordenar. Recupera de más y reordena con otro modelo.	Anthropic midió fallos de recuperación del 5,7 % al 1,9 % combinando contexto, BM25 y reordenación.	Cuando la respuesta buena aparece en los primeros 20, pero no en los primeros 5.
4. Prueba con preguntas reales. Mira en qué puesto queda la respuesta buena.	Ves el fallo antes que tus usuarios. Así salió el puesto 37.	Antes de lanzar, y cada vez que cambias el troceado o el modelo de embeddings.
5. Que cite la fuente. El trozo y el documento de donde sale la respuesta.	Puedes comprobar cada respuesta en un clic, y detectas cuando contesta sin la fuente buena.	Siempre que la respuesta se vaya a usar para algo que importa.

Errores típicos

- **Culpar al modelo cuando falla la búsqueda.** Si el trozo bueno no llega al prompt, ningún modelo lo va a usar. Mira primero qué trozos recibió.
- **Fiarte solo de los embeddings.** Captan el sentido general, pero con siglas como RAG pueden hundir la respuesta buena fuera de los primeros puestos.
- **Pasar solo 5 trozos sin mirar el resto.** Lo que queda en el puesto 6 no existe para la IA. Si dudas, recupera más y reordena.
- **Trocear sin solape.** Una idea partida entre dos trozos se pierde en los dos. Un solape pequeño lo evita.

- **No probar con preguntas reales.** Una demo con las preguntas que tú ya sabes que funcionan no te dice nada. Usa las de tus usuarios.

Preguntas frecuentes

¿Qué es RAG en inteligencia artificial?

RAG (generación aumentada por recuperación) es buscar en tus documentos los trozos que importan para una pregunta y pegárselos al modelo junto a la pregunta, para que conteste con ellos. Lo propusieron Lewis y otros investigadores de Facebook AI Research en 2020. Sirve porque el modelo no ha leído tus documentos y casi nunca caben enteros en su contexto.

¿Qué es un embedding?

Una lista de números que representa el significado de un texto. Textos que dicen cosas parecidas dan listas parecidas, así que se pueden comparar: con vectores normalizados, la similitud del coseno es el producto escalar. El modelo del asistente de esta chuleta usa 384 números por trozo.

¿Por qué mi RAG no encuentra la respuesta si está en los documentos?

Casi siempre porque la búsqueda no la pone entre los primeros trozos, y el modelo solo lee esos. La búsqueda por embeddings capta bien el sentido general pero patina con siglas y términos técnicos: en la prueba de esta chuleta, la clase que respondía quedó en el puesto 37 de 2.884. Con búsqueda por palabras clave o híbrida salió la primera.

¿Qué es la búsqueda híbrida?

Buscar a la vez por significado (embeddings) y por palabras clave (BM25) y mezclar los dos rankings, por ejemplo con RRF: cada trozo suma 1/(60 + su puesto) en cada lista. Cada método acierta donde el otro falla, así que juntos fallan menos. No es perfecta: en la prueba, con otra forma de preguntar, la clase buena quedó sexta.

¿De qué tamaño hago los trozos?

No hay un tamaño universal: hay que probar con tus documentos y tus preguntas. Como referencia, el asistente de esta chuleta usa trozos de hasta 1.200 caracteres con 150 de solape, y la búsqueda de archivos de OpenAI usa por defecto 800 tokens con 400 de solape.