IA local para PyMEs · Módulo 6

RAG que funciona

La calidad de recuperación pesa más que el tamaño del modelo. Permisos antes de recuperar.

Qué se construye en este módulo

Ingesta y chunking

Cortar por estructura, no por tamaño

Embeddings y recuperación

Elegir embedding por idioma y dominio

Reranking y filtrado por permisos ANTES de recuperar

Poner el permiso como filtro del índice

Qué se construye en este módulo

Diagnóstico de un RAG malo

Localizar la falla: ingesta, recuperación o generación

6.1 · Ingesta y chunking

Ingesta y chunking

Un RAG es tan bueno como sus fichas. Si los documentos se recortan por tamaño («cada 500 caracteres»), una tabla queda partida en dos, un artículo de contrato pierde su número, y una factura queda sin su encabezado. Después ningún modelo lo arregla.

En una tabla

Tipo de documento	Unidad de corte	Tamaño típico	Solapamiento
Contrato	cláusula / artículo numerado	200–600 tokens	0 (las cláusulas unidades)
Factura	**una ficha por factura** (encabezado + líneas + totales)	300–900	0
Acta	punto del orden del día	150–500	1 frase
Manual / política	encabezado de sección (##)	300–800	10 %
Correo	un correo	100–800	0
Texto sin estructura	párrafos, agrupados hasta el tope	400–600	10–15 %

El comando, copiable

```
cd ~/labs/el-tornillo && mkdir -p labs/06-01 && cd labs/06-01 && pip install -q
    pymupdf
```

Si no corre desde cero con un comando, no entra al curso.

Cuándo NO usar esto

Ingesta y chunking

- No use chunking fijo «porque es lo que trae la biblioteca». Es la causa más común de RAG malo.
- No ingiera todo el archivo histórico el primer día: 50 documentos representativos, medir, luego el resto.

6.2 · Embeddings y recuperación

Embeddings y recuperación

El modelo de embeddings decide qué fichas aparecen. Es pequeño (cientos de MB), corre en CPU con soltura, y **se cambia con menos dolor que el modelo de lenguaje** —siempre que se reconstruya el índice.

En una tabla

Criterio	Por qué	Có
Español de verdad, no «multilingüe» de etiqueta	los documentos son en español con jerga contable	rec
Dimensión 384–1 024	más dimensiones ≠ mejor; más memoria en el índice	tam
Longitud máxima $\geq$ 512 tokens	fichas de contrato son largas	fich
Licencia permisiva	va a producción en un cliente	la l
Dos alternativas por tamaño	pequeño (~100 MB) para A; medio (~500 MB) para B/C	me

El comando, copiable

```
CREATE EXTENSION IF NOT EXISTS vector;
CREATE TABLE fichas (id text PRIMARY KEY, texto text, origen text, seccion text,
  coleccion text, permisos text[], fecha_doc date, emb vector(768));
CREATE INDEX ON fichas USING hnsw (emb vector_cosine_ops);
CREATE INDEX ON fichas USING gin (permisos);
```

Si no corre desde cero con un comando, no entra al curso.

Cuándo NO usar esto

Embeddings y recuperación

- No use embeddings para búsqueda exacta (número de factura, cédula): eso es `WHERE numero = …`.
- No mida recall con consultas inventadas por usted: las escribe el cliente, con sus palabras.
- No cambie de modelo de embeddings sin reindexar todo: los vectores no son compatibles.

6.3 · Reranking y filtrado por permisos ANTES de recuperar

Reranking y filtrado por permisos ANTES de recuperar

El error grave: recuperar las 8 mejores fichas y **después** quitar las que el usuario no puede ver. Dos problemas: (1) al vendedor le pueden quedar 2 fichas de 8, y el modelo responde mal; (2) peor, si el filtro falla, **la nómina ya está en el contexto**. El permiso se aplica en la consulta:

En una tabla

Perfil	Qué pasa
A	Reranking en CPU: +0,5–1,5 s por consulta. Aceptable en lote, justo en vivo
B / C	+100–300 ms; se puede rerankear 30–50 candidatos

El comando, copiable

```
-- el vendedor solo puede ver 'ventas' y 'publico'
SELECT id, texto, seccion, origen FROM fichas
WHERE permisos && %s::text[]                  -- grupos del usuario (de la cabecera
    del SSO, 3.2)
  AND coleccion = ANY(%s)                     -- colección pedida
ORDER BY emb <=> %s::vector LIMIT 20;
```

Si no corre desde cero con un comando, no entra al curso.

Cuándo NO usar esto

Reranking y filtrado por permisos ANTES de recuperar

- No rerankee 5 candidatos: no hay nada que reordenar. Recupere 20–50, rerankee, entregue 5.
- No use el reranker como filtro de permisos «porque puntúa bajo lo que no corresponde»: el permiso es determinista, el reranker no.

6.4 · Diagnóstico de un RAG malo

Diagnóstico de un RAG malo

Cuando la respuesta es mala, el instinto es cambiar el modelo. En la mayoría de los casos el modelo es la última causa. El diagnóstico va **de atrás hacia adelante**:

En una tabla

Etapa	Pregunta	Cómo se comprueba	Falla típica
1 · Ingesta	¿la información está en alguna ficha, completa?	buscar a mano en `fichas.json`	ficha partida; PDF si capa de texto ignora do; tabla destruida
2 · Recuperación	¿la ficha correcta aparece en los k candidatos?	recall@k (6.2)	embedding malo par español; consult demasiado corta filtro de permisos ma puesto
3 · Reranking	¿queda entre las 5 entregadas?	comparar antes/después (6.3)	umbral demasiad alto; reranker no mu tilingüe
4 · Contexto	¿cabe en la ventana, en buen orden?	presupuesto (5.1)	ficha correcta cortad por el tope; pregunt al principio
5 · Generación	con la ficha correcta en contexto, ¿respon-	pasar la ficha a mano	modelo demasiad cuantizado; instruc

Cuándo NO usar esto

Diagnóstico de un RAG malo

- No arranque por la etapa 5. Nunca.
- No cambie dos cosas a la vez: no sabrá cuál arregló (o rompió).

Entregables del módulo

- `ingerir.py`, `fichas.json`, y `labs/06-01/revision.md` con las diez fichas revisadas.
- `consultas.json` (40, escritas con el cliente), y la tabla recall@k de dos modelos.
- tabla antes/después, y `labs/06-03/aislamiento.md` con las 20 pruebas.
- `labs/06-04/diagnostico.md` con los 5.

La calidad de recuperación pesa más que el tamaño del modelo.