IA local para PyMEs · Módulo 13

Especialización de modelos

Cuándo NO hacer fine-tuning (casi siempre), la escalera completa, y el corpus como activo del cliente.

Qué se construye en este módulo

Cuándo NO hacer fine-tuning: la escalera de decisión

Recorrer la escalera prompt → contexto → RAG → herramientas → destilación → LoRA

Construcción de corpus

Curar 500 ejemplos con formato de entrenamiento

Destilación y LoRA/QLoRA paso a paso

Entrenar un adaptador en B o C

Qué se construye en este módulo

El corpus como activo y línea de facturación

Redactar la cláusula de propiedad del corpus

13.1 · Cuándo NO hacer fine-tuning: la escalera de decisión

Cuándo NO hacer fine-tuning: la escalera de decisión

La pregunta «¿hacemos fine-tuning?» tiene una respuesta por defecto: **no todavía**. El fine-tuning enseña **formas** (estilo, formato, vocabulario), no **hechos**; cuesta datos, tiempo y una infraestructura de evaluación que casi ninguna PyME tiene el primer año; y se vuelve a hacer cada vez que cambia el modelo base. Antes de subir ese escalón hay cinco más baratos.

En una tabla

Perfil	Escalones disponibles
A	1–4 en local; 5–6 en la nube o en un equipo prestado
B	1–6 (QLoRA de modelos pequeños/medianos)
C	1–6 con modelos medianos

Cuándo NO usar esto

Cuándo NO hacer fine-tuning: la escalera de decisión

- No haga fine-tuning para «que sepa de la empresa»: eso es RAG. El modelo afinado sigue sin saber la factura de ayer.
- No haga fine-tuning con menos de 500 ejemplos revisados: aprende el ruido.

Qué se degrada en cada perfil

A · sin GPU

1–4 en local; 5–6 en la nube o en un equipo prestado

B · 8–16 GB VRAM

1–6 (QLoRA de modelos pequeños/medianos)

C · 24 GB+

1–6 con modelos medianos

13.2 · Construcción de corpus

Construcción de corpus

Un corpus de especialización son ejemplos **entrada → salida deseada**, revisados por alguien que sabe. Quinientos ejemplos buenos valen más que cinco mil regulares: el modelo aprende los errores con la misma facilidad que los aciertos.

En una tabla

Fuente	Calidad
Salidas del sistema **corregidas y aprobadas** por el cliente (10.4)	alta: es ex quiere
Documentos históricos del cliente (actas viejas) emparejados con su insumo	alta si el in
Generados por un modelo grande (13.3, destilación) y **revisados**	media–alta
Generados sin revisar	baja

El comando, copiable

```
{"messages":[{"role":"system","content":"Redacte el acta con las 14 secciones de El
    Tornillo."},
             {"role":"user","content":"<transcripción resumida>"},
             {"role":"assistant","content":"<acta aprobada por el gerente>"}]}
```

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

Cuándo NO usar esto

Construcción de corpus

- No genere el corpus con el mismo modelo que va a afinar sin revisar: aprende a imitarse.
- No incluya ejemplos con datos sensibles: el adaptador los puede reproducir.

13.3 · Destilación y LoRA/QLoRA paso a paso

Destilación y LoRA/QLoRA paso a paso

Destilación: un modelo grande (que corre en C, o prestado por horas) produce las salidas de los 500 ejemplos; se revisan; con ellas se adapta un modelo pequeño que sí corre en A o B. **LoRA/QLoRA:** en vez de reentrenar los miles de millones de pesos, se entrena un **adaptador** pequeño (decenas de MB) que se suma al modelo base. QLoRA lo hace con el base cuantizado en 4 bits, y por eso cabe en el perfil B.

En una tabla

Perfil	Qué pasa
A	No se entrena en local. Se entrena en B/C prestado o en la nube **con corpus anonimizado**; se infiere en A con el adaptador
B	QLoRA de modelos $\leq$ 8B: horas
C	QLoRA de modelos $\leq$ 14B: minutos–hora; LoRA en 16 bits de pequeños

El comando, copiable

```
cd ~/labs/el-tornillo && mkdir -p labs/13-03 && cd labs/13-03
python3 -m venv .venv && source .venv/bin/activate
pip install -q "transformers" "peft" "datasets" "accelerate" "bitsandbytes"   # CUDA;
     en Apple Silicon: MLX-LM
```

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

Cuándo NO usar esto

Destilación y LoRA/QLoRA paso a paso

- No afine sin `antes.json`: no sabrá si mejoró.
- No entrene con el set de prueba ni con los evals: se engaña a sí mismo.
- No afine cada mes «con lo nuevo»: cada adaptador es una versión que hay que mantener (11.3).

Qué se degrada en cada perfil

A · sin GPU

No se entrena en local. Se entrena en B/C prestado o en la nube **con corpus anonimizado**; se infiere en A con el adaptador

B · 8–16 GB VRAM

QLoRA de modelos ≤ 8B: horas

C · 24 GB+

QLoRA de modelos ≤ 14B: minutos–hora; LoRA en 16 bits de pequeños

13.4 · El corpus como activo y línea de facturación

El corpus como activo y línea de facturación

Los modelos cambian cada seis meses. El adaptador se reentrena. Lo que **no** se pierde es el corpus: 500 ejemplos revisados de cómo esta empresa quiere sus actas, sus extracciones, sus respuestas. Eso es un **activo del cliente**, con valor propio, y hay que tratarlo como tal en el contrato y en la factura.

En una tabla

Pieza	De quién	Por qué
Documentos originales	del cliente	siempre
Corpus curado (los pares entrada→salida)	**del cliente**	está hech terio
Método de curación, scripts, plantillas	del implementador (licenciados al cliente)	es el proc
Adaptador entrenado	del cliente (derivado de su corpus)	se entreg rrida de e
Modelo base	de quien lo publicó, según su licencia	**verificar** cial y der

Cuándo NO usar esto

El corpus como activo y línea de facturación

- No reutilice el corpus de un cliente en otro «porque son parecidos». Es suyo, y la ley (2.3) y el contrato lo dicen.
- No entregue un adaptador sin su corpus y su corrida de evals: es un binario que nadie podrá reproducir.

Entregables del módulo

- la decisión por tarea.
- `corpus/{train,val,test}.jsonl` (en el repo **privado** del cliente) + `corpus.md`.
- `adaptador/`, `antes.json`, `despues.json`, el tablero comparado, y en `labs/13-03/entrenamiento.md`: base, hiperparámetros, tiempo, hardware, hash del corpus.
- la cláusula y la tabla.

Cuándo NO hacer fine-tuning (casi siempre), la escalera completa, y el corpus como activo del cliente.