LO ESENCIAL · **Módulo 13 · Especialización de modelos**

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

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

Objetivos

- Recorrer la escalera prompt contexto RAG herramientas destilación LoRA
- Documentar la decisión para el asistente

Casi nunca

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.

La escalera

Escalón	Qué se intenta	Cuándo basta	Costo
1 · **Instrucciones**	mejores instrucciones, ejemplos en el prompt	tareas de formato y tono	horas
2 · **Contexto**	poner los documentos correctos, presupuestados (módulo 5)	el modelo se equivoca porque no tiene la información	días
3 · **RAG**	recuperación medida, permisos, reranking (módulo 6)	la información existe pero no cabe	días–semanas
4 · **Herramientas**	dar acceso a la base, la calculadora, el validador (5.3)	el modelo «calcula» mal o inventa datos que están en un sistema	días
5 · **Destilación**	un modelo grande genera ejemplos; uno pequeño aprende de ellos (13.3)	la tarea es estable, el volumen alto, y el modelo grande no cabe en el perfil	semanas
6 · **LoRA / QLoRA**	adaptador entrenado sobre corpus curado (13.3)	el estilo o el formato del cliente **no** se logra con 1–4, y hay $\geq$ 500 ejemplos buenos	semanas + mantenimiento
7 · Fine-tuning completo	reentrenar todos los pesos	prácticamente nunca en una PyME	—

Se sube un escalón **solo con el set de evals en la mano** (módulo 12): si el escalón anterior llegó al umbral, se para ahí.

Señales de que sí toca 6

- El formato de salida del cliente es raro y consistente (un acta con 14 secciones fijas), y el modelo lo pierde en 1 de cada 5 aunque el prompt lo describa.
- El vocabulario del dominio confunde al modelo (nombres de piezas de ferretería que parecen otra cosa), y hay cientos de ejemplos etiquetados.
- Se necesita un modelo **pequeño** (perfil A/B) que haga lo que hoy solo hace uno grande, y la tarea es una sola y estable.

> PROCEDIMIENTO · **Laboratorio 13.1 · la decisión documentada**
>
> Para cada una de las tres tareas del asistente, `labs/13-01/decision.md`: en qué escalón está, qué nota tiene en evals, qué escalón sigue y **qué evidencia** haría falta para subir. Para al menos una, la respuesta debe ser «se queda en el escalón N».
> **Entregable:** la decisión por tarea. Commit: «Capa 13.1: escalera».

Qué se degrada en cada perfil

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.

PRÁCTICA · **Práctica**

1. «El modelo no sabe nuestros precios.» ¿Escalón?
2. «El modelo escribe las actas con secciones en otro orden.» ¿Escalón? ¿Qué evidencia pide?

Referencias

1. Hu, E. J. et al. (2021). *LoRA: Low-Rank Adaptation of Large Language Models.* arXiv:2106.09685.
2. Dettmers, T. et al. (2023). *QLoRA.* arXiv:2305.14314.
3. Módulo 12 — sin evals no hay escalera.

13.2 · Construcción de corpus

> Objetivos
>
> - Curar 500 ejemplos con formato de entrenamiento
> - Separar entrenamiento, validación y prueba

El corpus es trabajo, no descarga

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.

Formato

```
{"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>"}]}
```

Una línea por ejemplo. El `system` es **el mismo** que usará el asistente en producción: si cambia, el adaptador deja de servir.

De dónde salen los ejemplos

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

Curación

1. **Deduplicar** (exactos y casi exactos).
2. **Revisar el 100 %** si son < 1 000; una muestra del 20 % si son más, con dos revisores.
3. **Equilibrar**: si el 80 % de las actas son de comité y el 20 % de proveedores, el corpus debe reflejarlo —o el adaptador se especializa en comités.
4. **Separar**: 80 % entrenamiento, 10 % validación, 10 % prueba. El de prueba **no se mira** hasta el final, y no se solapa con el set de evals (12.1).
5. **Anonimizar** según 2.3, sin alterar la estructura.

> PROCEDIMIENTO · **Laboratorio 13.2**
>
> Construya 500 ejemplos para **una** de las tres tareas (la que la decisión 13.1 dejó en escalón 6). Documente en `labs/13-02/corpus.md`: fuentes, cuántos se descartaron y por qué, equilibrio, partición, y quién revisó.
> **Entregable:** `corpus/{train,val,test}.jsonl` (en el repo **privado** del cliente) + `corpus.md`. Commit: «Capa 13.2: corpus».
> Nada: curar es trabajo humano.

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.

PRÁCTICA · **Práctica**

1. Tiene 380 actas aprobadas y 40 de proveedores. ¿Cómo equilibra sin inventar?
2. ¿Qué pasa si el `system` del corpus tiene 200 tokens y el de producción 1 200?

Referencias

1. Zhou, C. et al. (2023). *LIMA: Less Is More for Alignment*. arXiv:2305.11206 — la evidencia de que la calidad pesa más que la cantidad.
2. Módulo 2.3 — anonimización.

13.3 · Destilación y LoRA/QLoRA paso a paso

> Objetivos
>
> - Entrenar un adaptador en B o C
> - Evaluar antes y después con el mismo set

Destilar y adaptar

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.

> PROCEDIMIENTO · **Laboratorio 13.3 · paso a paso** `[VOLÁTIL — verificado: 2026-09]`
>
> ```
> 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

# entrenar.py — QLoRA mínimo, neutral de modelo: el base se pasa por argumento
import sys, torch
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForCausalLM, BitsAndBytesConfig,
    TrainingArguments
from peft import LoraConfig, get_peft_model, prepare_model_for_kbit_training
from trl import SFTTrainer
base = sys.argv[1]                                          # <modelo-base>  [VOLÁTIL]
tok = AutoTokenizer.from_pretrained(base); tok.pad_token = tok.eos_token
model = AutoModelForCausalLM.from_pretrained(base, device_map="auto",
          quantization_config=BitsAndBytesConfig(load_in_4bit=True, bnb_4bit_compute_dtype
    =torch.bfloat16))
model = prepare_model_for_kbit_training(model)
model = get_peft_model(model, LoraConfig(r=16, lora_alpha=32, lora_dropout=0.05, task_type
    ="CAUSAL_LM",
          target_modules=["q_proj","k_proj","v_proj","o_proj"]))
ds = load_dataset("json", data_files={"train":"../13-02/corpus/train.jsonl","val":"
    ../13-02/corpus/val.jsonl"})
tr = SFTTrainer(model=model, tokenizer=tok, train_dataset=ds["train"], eval_dataset=ds["
    val"],
       args=TrainingArguments(output_dir="salida", num_train_epochs=2,
    per_device_train_batch_size=2,
            gradient_accumulation_steps=8, learning_rate=2e-4, bf16=True, logging_steps=10,

            eval_strategy="steps", eval_steps=50, save_strategy="epoch"))
tr.train(); model.save_pretrained("adaptador")

python3 entrenar.py <modelo-base>          # B: ~13 h para 500 ejemplos con un ~78B; C:
    minutos
# fusionar y exportar a GGUF para el runtime (o cargar el adaptador directamente si el
    runtime lo soporta)
```

Evaluar antes y después — obligatorio

```
python3 ../12-01/evals.py --set ../12-01/evals/ --modelo base     --salida antes.json
python3 ../12-01/evals.py --set ../12-01/evals/ --modelo adaptado --salida despues.json
python3 ../12-02/tablero.py antes.json despues.json
```

Lo que debe ver: la tarea afinada **sube**, y **ninguna otra baja**. Si otra baja (el adaptador «olvidó» algo), el adaptador se usa solo para esa tarea (razón 3 de 10.2) o se descarta.

Entregable: `adaptador/`, `antes.json`, `despues.json`, el tablero comparado, y en `labs/13-`

`03/entrenamiento.md`: base, hiperparámetros, tiempo, hardware, hash del corpus. Commit: «Capa 13.3: adaptador evaluado».

Qué se degrada en cada perfil

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

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).

PRÁCTICA · **Práctica**

1. El adaptador sube actas de 78 % a 94 % y baja extracción de facturas de 96 % a 93 %. ¿Qué hace?
2. ¿Por qué el `system` del entrenamiento y el de producción deben ser idénticos?

Referencias

1. Hu, E. J. et al. (2021). LoRA — arXiv:2106.09685. · Dettmers et al. (2023). QLoRA — arXiv:2305.14314.

2. Hugging Face PEFT — https://huggingface.co/docs/peft · TRL — https://huggingface.co/-docs/trl `[VOLÁTIL — verificado: 2026-09]`

3. Hinton, G. et al. (2015). *Distilling the Knowledge in a Neural Network.* arXiv:1503.02531.

13.4 · El corpus como activo y línea de facturación

> Objetivos
>
> - Redactar la cláusula de propiedad del corpus
> - Ponerle precio a la curación

Lo que queda cuando cambia el modelo

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.

Propiedad

Pieza	De quién	Por qué
Documentos originales	del cliente	siempre
Corpus curado (los pares entrada→salida)	**del cliente**	está hecho con sus datos y s
Método de curación, scripts, plantillas	del implementador (licenciados al cliente)	es el producto reutilizable
Adaptador entrenado	del cliente (derivado de su corpus)	se entrega con su hash y s evals
Modelo base	de quien lo publicó, según su licencia	**verificar** que permite uso co rivados

Cláusula tipo, para el contrato (14.3): *«El corpus de ejemplos y los adaptadores derivados son propiedad de EL CLIENTE. EL IMPLEMENTADOR conserva la propiedad de sus herramientas y métodos, y no usará el corpus para terceros.»*

Línea de facturación

Concepto	Unidad	Nota
Curación de corpus	por 100 ejemplos revisados	trabajo humano; se factura
Entrenamiento y evaluación de adaptador	por corrida	incluye antes/después
Mantenimiento del corpus	mensual	incorporar salidas aproba (13.2)

El corpus **crece con el uso** (cada aprobación en la cola de 10.4 es un ejemplo candidato): el mantenimiento mensual es lo que convierte un proyecto en un servicio.

PROCEDIMIENTO · **Laboratorio 13.4**

`labs/13-04/clausula.md`: la cláusula de propiedad adaptada a la ferretería, la tabla de facturación con precios en COP, y la verificación de la licencia del modelo base elegido (con enlace y fecha).
Entregable: la cláusula y la tabla. Commit: «Capa 13: el corpus es del cliente».
Nada.

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.

PRÁCTICA · **Práctica**

1. Un modelo base tiene licencia que prohíbe uso comercial. ¿Qué hace con el adaptador?
2. Ponga precio a curar 500 ejemplos con una revisora contable a tarifa local.

Referencias

1. Licencias de modelos abiertos: leer **el texto** de cada una (Apache-2.0, MIT, licencias propias con cláusulas de uso) antes de entrenar.
2. Módulo 14.3 — el modelo de precios.