Día 16 · Tema 6 — Adaptación de LLMs: prompting, LoRA, RLHF, DPO

Eduardo C. Garrido-Merchán · ecgarrido@comillas.edu Universidad Pontificia Comillas · ICADE

1. Cinco técnicas de adaptación, comparadas

Dado un LLM base f_{θ_0}, hay cinco vías para adaptarlo a una tarea o estilo particular, ordenadas por coste creciente:

Técnica	Modifica θ?	Datos típicos	Coste compute
Prompt engineering	no	0 ejemplos	0$
Few-shot in-context	no	1–20 ejemplos en prompt	0$
RAG	no	corpus indexado	~0.02 $/consulta
LoRA / fine-tuning	adaptadores pequeños	100–10 000 pares (x,y)	1–100 $
RLHF / DPO	pesos completos o adapt.	preferencias humanas	10^2–10^4 $

La regla práctica: subir de nivel sólo cuando el inferior se quede corto.

2. LoRA: adaptación de bajo rango

Para no actualizar todos los pesos $W \in \mathbb{R}^{m\times n}$ del modelo, LoRA inserta una perturbación de rango $r \ll \min(m,n)$:

$$W' \;=\; W \;+\; \Delta W, \qquad \Delta W \;=\; \tfrac{\alpha}{r}\, B\, A, \tag{1}$$

$A \in \mathbb{R}^{r\times n}$ matriz *down-projection*, inicializada gaussiana
$B \in \mathbb{R}^{m\times r}$ matriz *up-projection*, inicializada a cero
r rango (típicamente 4, 8, 16, 64)
α factor de escala (típicamente $\alpha = r$ o $\alpha = 2r$)

Se entrenan sólo A, B, no W (que sigue congelado). El número de parámetros entrenables baja de mn a $r(m+n)$, típicamente un factor ~1000×. Una vez entrenados, BA se puede fusionar con W a la inferencia o mantener separado para conmutar adaptadores.

3. Quantization-aware: QLoRA

QLoRA combina LoRA con cuantización de W a 4 bits (NF4). El modelo base ocupa 4× menos VRAM; los adaptadores se mantienen en FP16. Permite entrenar Llama-3 70B en una sola GPU de 48 GB.

4. RLHF: alineación con preferencias humanas

Si la métrica de calidad no es la pérdida de *next-token* sino la preferencia humana, entramos en RLHF. Tres etapas:

1. **SFT**: fine-tune supervisado en demostraciones humanas.
2. **Reward model** $r_\phi(x, y)$: dado un prompt x y dos respuestas candidatas y_w, y_l (winner / loser), entrenar con Bradley-Terry:
$$\mathcal{L}_{\text{RM}}(\phi) = -\mathbb{E}\big[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))\big]. \tag{2}$$
3. **Policy optimization (PPO)**: maximizar la recompensa con un KL-penalty contra el modelo SFT,
$$\max_\theta \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta}\big[\, r_\phi(x,y) - \beta\, \mathrm{KL}\big(\pi_\theta(\cdot|x) \,\|\, \pi_{\text{SFT}}(\cdot|x)\big)\,\big]. \tag{3}$$

r_ϕ modelo de recompensa, $\mathbb{R}^d \to \mathbb{R}$
y_w, y_l respuestas preferida y no preferida
π_θ política a optimizar (el LLM)
β peso del KL: mayor $\beta \Rightarrow$ más cerca de SFT

RLHF es la receta detrás de InstructGPT, ChatGPT y los LLM *aligned*.

5. DPO: la simplificación de 2023

RLHF con PPO es caro y delicado. **Direct Preference Optimization** (Rafailov et al.) elimina el RM explícito y el RL:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{SFT}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{SFT}}(y_l|x)}\right)\right]. \tag{4}$$

Es supervisión clásica sobre pares de preferencia: las gradientes son estables, no hay reward model que entrenar, y empíricamente alcanza la calidad de PPO con un fraction del coste. DPO es el default 2026.

6. Concepto → aplicación

- **Estilo corporativo** (tono BBVA, jerga interna): prompt engineering + few-shot. Si no basta, LoRA sobre 200 pares.
- **Conocimiento corporativo** (normativa, FAQ): RAG.
- **Tarea muy específica** (extracción de campos de factura en un formato fijo): LoRA con 500–2000 ejemplos.
- **Comportamiento alineado** (no responder a queries hostiles, seguir tono de marca): RLHF/DPO sobre $\sim 10^4$ pares de preferencia.

Idea clave. En 2026 muy pocos equipos hacen RLHF/DPO en producción: lo hacen los laboratorios (Anthropic, OpenAI, Meta) y la empresa consume modelos alineados via API o pesos abiertos.