Día 11 · Tema 4 — Cómo funciona un LLM, sampling y context engineering

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

1. Un LLM es un predictor de próximo token

Dado un prefijo $x_{1:t}$ (la conversación hasta el paso t), un LLM produce la distribución sobre el siguiente token:

$$p_\theta(x_{t+1} \mid x_{1:t}) \;=\; \mathrm{softmax}\big(W^O h_t\big), \tag{1}$$

donde $h_t \in \mathbb{R}^d$ es el último estado oculto del Transformer y $W^O \in \mathbb{R}^{|\mathcal{V}|\times d}$ es la matriz de salida sobre el vocabulario $\mathcal{V}$ (típicamente $|\mathcal{V}| \sim 50\,000$ *subwords*). El entrenamiento maximiza

$$\mathcal{L}(\theta) \;=\; \sum_{t=1}^{T-1} \log p_\theta(x_{t+1} \mid x_{1:t}), \tag{2}$$

sobre todo internet, $O(10^{13})$ tokens. La generación es **autoregresiva**: se muestrea $\hat{x}_{t+1} \sim p_\theta(\cdot|x_{1:t})$, se concatena al contexto, se repite.

2. Sampling: temperatura, top-k, top-p

La distribución cruda no se usa directamente; se modula:

$$p_\tau(x \mid x_{1:t}) \;=\; \mathrm{softmax}\!\left(\frac{\mathrm{logits}(x)}{\tau}\right). \tag{3}$$

τ temperatura; $\tau \to 0$ es greedy (determinista), $\tau \to \infty$ es uniforme
logits el vector $W^O h_t$, antes del softmax

top-k: truncar a los k tokens con mayor probabilidad y renormalizar. **top-p (nucleus)**: truncar al menor conjunto acumulando probabilidad $\geq p$. Por defecto los APIs usan $\tau = 0{,}7$, top-$p = 0{,}9$ y dejan top-k libre.

Regla de configuración: código $\to \tau \leq 0{,}2$; análisis estructurado $\to \tau = 0{,}3$; redacción creativa $\to \tau = 0{,}9$.

3. Anatomía del contexto

Lo que el LLM ve en cada llamada se reparte entre cinco piezas:

1. **Instrucciones de sistema**: rol, formato de salida, restricciones.
2. **Memoria persistente**: lo que ya se sabe de sesiones previas (archivos como `CLAUDE.md`, embeddings de perfil).
3. **Recuperado (RAG)**: documentos relevantes pasados a contexto.
4. **Histórico de conversación**: turnos previos del usuario y el modelo.
5. **Pregunta actual**: lo que el usuario acaba de escribir.

Dado el presupuesto de tokens B, el problema es de *asignación*:

$$\max \; \mathbb{E}[\text{calidad}] \quad \text{s. a.} \quad \sum_{i=1}^{5} t_i \leq B, \tag{4}$$

con t_i los tokens dedicados a cada pieza. A esto se le llama *context engineering* y es la habilidad central del Bloque II.

4. Coste de una llamada cloud

Recordando la ecuación de coste del Día 01,

$$C \;=\; \frac{N_{\text{in}}}{10^6}\, p_{\text{in}} + \frac{N_{\text{out}}}{10^6}\, p_{\text{out}}. \tag{5}$$

Con Claude Sonnet 4.7 ($p_{\text{in}} = 3$ \$, $p_{\text{out}} = 15$ \$): una sesión RAG con $N_{\text{in}} = 8000$ y $N_{\text{out}} = 600$ cuesta $C = 0{,}024 + 0{,}009 = 0{,}033$ \$. Mil sesiones al día: 33 \$. Un mes: ~\$1000.

4.1. Prompt caching

El proveedor permite *cachear* prefijos que se repiten (system prompt, ejemplos few-shot, documentación pegada). El caché tiene un precio menor ($\approx 0{,}30 \times p_{\text{in}}$) y devuelve el resultado sin recomputar. Llave: la cache key son los primeros N tokens del prompt. Mover lo inmutable al inicio del contexto es lo que activa el cache.

5. Long context y memoria persistente

Modelos modernos admiten $\geq$ 200K tokens (Claude 3.5/4) o $\geq$ 1M (Gemini 1.5 Pro). El coste sigue siendo lineal en N_{in}, así que un contexto largo no es gratis. La *memoria persistente* consiste en guardar en disco lo que el modelo ya ha aprendido sobre el usuario y recuperarlo selectivamente, en vez de cargar toda la historia.

6. Elección de modelo

Una matriz simplificada (precios y latencia de 2026):

Modelo	$\$_{\text{in}}$/M	$\$_{\text{out}}$/M	**Caso**
Claude Sonnet 4.7	3.0	15.0	Razonamiento y código en producción.
GPT-5 (estándar)	2.5	10.0	Texto comercial, RAG general.
Mistral Large	2.0	6.0	Europa, datos sensibles.
Gemini 1.5 Pro	1.25	5.0	Contexto $\geq$ 1M tokens.
Qwen 2.5 local	0	0	Borrador, gratis tras descarga.

Idea clave. La regla práctica: **borrador en local** (gratis), **producción en cloud** (calidad). Quien paga el cloud lo justifica con el coste evitado en errores que el modelo barato hubiera dejado pasar.