Deep Learning para Business Analytics

Día 11 · Tema 4 — LLMs y context engineering

Próximo token, contexto y coste.

Eduardo C. Garrido-Merchán

Profesor Propio Adjunto, Métodos Cuantitativos, ICADE

ecgarrido@comillas.edu

Madrid, Día 11

ejemplos del día

Lo que vamos a cubrir hoy.

1. Cómo funciona un LLM por dentro

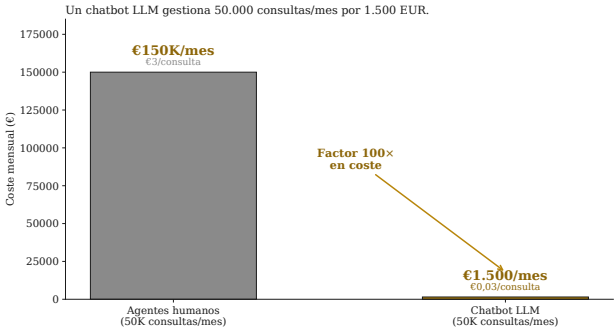
2. Context engineering

3. Coste y elección de modelo

4. Casos y elección

Al final del día: sabes calcular el coste de una sesión de un LLM con RAG, eliges entre cloud y local con argumentos, y dominas las cinco piezas del contexto.

Un chatbot LLM gestiona 50.000 consultas/mes por €1.500 💰.



Factor 100× en coste 💰. Un call center con agentes humanos a €3/consulta cuesta ~€150K/mes para 50K consultas mensuales. Un LLM a €0,03/consulta: €1.500/mes.

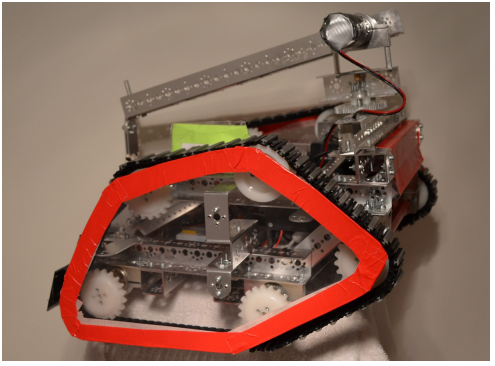
Pero no todo es coste. El 5 % que el bot falla puede costar reputación. La métrica crítica es la tasa de escalado a humano.

Caso BBVA. Combinan LLM (primer filtro) + agente humano (casos complejos). Reducción del 70 % en coste con satisfacción mantenida.

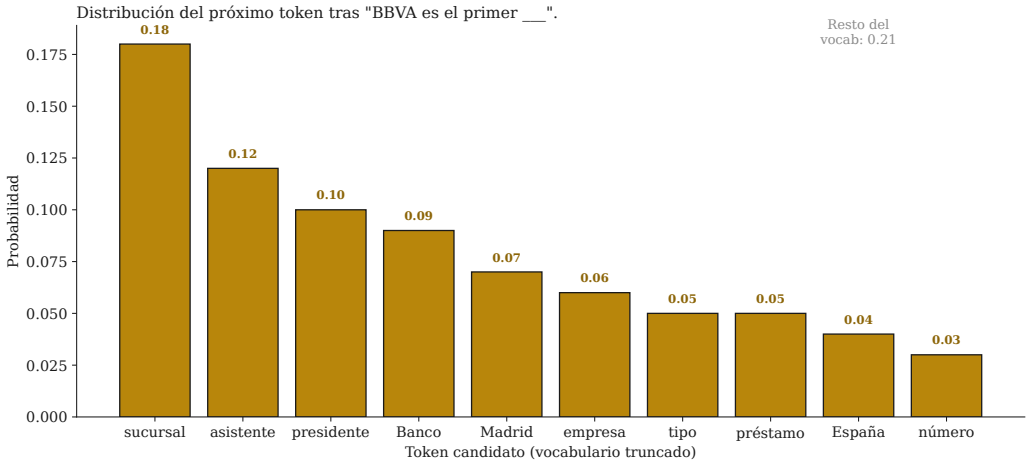
Idea clave. El ROI de un chatbot LLM se mide en coste × calidad × tasa de escalado.

Parte 1 / 4

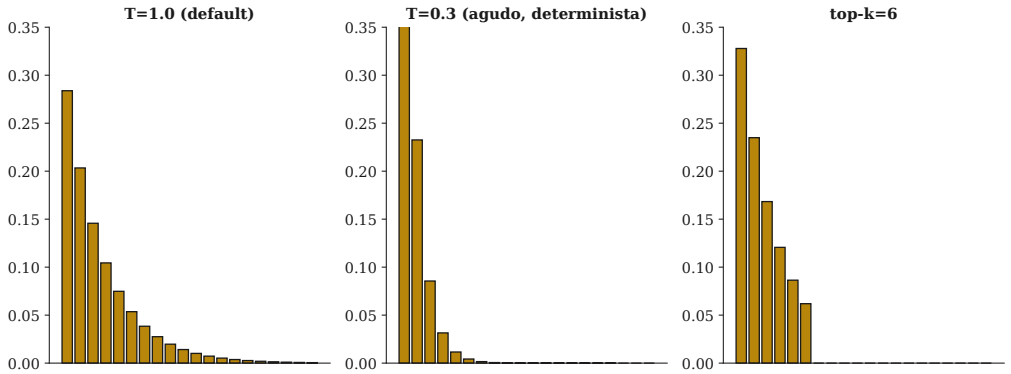
Cómo funciona un LLM por dentro



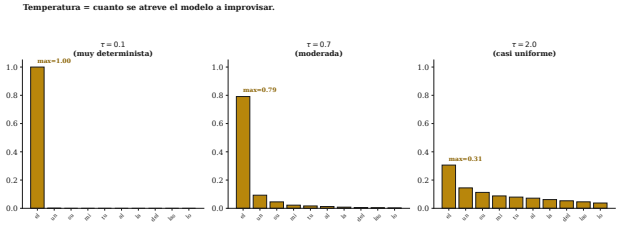
🤖 Un LLM es un predictor de próximo token, miles de millones de veces.



🌡️ Sampling 🎲: temperatura, top-k, top-p en una imagen.



🌡️ Temperatura = cuánto se atreve el modelo a improvisar 🎲.



$\tau \to 0$**: determinista.** Un solo token domina. Ideal para código y extracción de campos.

$\tau = 0{,}7$**: moderada.** Distribución suave. El equilibrio habitual para texto.

$\tau \to \infty$**: uniforme.** Elige casi al azar. Solo para creatividad extrema o red-teaming.

En producción. τ=0,2 para código, τ=0,7 para texto, τ=0,9 para brainstorming. Nunca $\tau > 1{,}5$ en producción real.

Idea clave. La temperatura no cambia el modelo: solo redistribuye la probabilidad sobre los mismos tokens.

Temperatura y top-p paso a paso: de logits a muestreo.

Temperatura y top-p paso a paso.

Paso 1: Logits crudos

el: 2.0	un: 1.0	su: 0.5	mi: -0.5	tu: -1.0

Paso 2: Dividir por $\tau = 0.7$

el: 2.86	un: 1.43	su: 0.71	mi: -0.71	tu: -1.43

Paso 3: Softmax

el: 0.715	un: 0.171	su: 0.084	mi: 0.020	tu: 0.010

Paso 4: Top-p = 0.9 (acumular hasta ≥ 0.9, renormalizar)

el: 0.737	**un: 0.177**	**su: 0.086**	*(mi, tu: eliminados)*

Resultado: de 5 tokens candidatos, top-p=0.9 retiene 3 y descarta los de menor probabilidad.

Paso 1. Logits: (2.0, 1.0, 0.5, -0.5, -1.0).

Paso 2. Dividir por τ=0,7: (2.86, 1.43, 0.71, -0.71, -1.43).

Paso 3. Softmax: “el” = 0.715, “un” = 0.171, “su” = 0.084, “mi” = 0.020, “tu” = 0.010. (Los altos dominan más que con τ=1.)

Paso 4. Top-p = 0.9: acumular desde el mayor. “el” (0.715) + “un” (0.171) = 0.886. Añadir “su” (0.084) $\rightarrow$ 0.970 $\geq$ 0.9. Se retienen 3 tokens, se descartan “mi” y “tu”. Renormalizar.

Idea clave. Top-p es adaptativo: retiene más tokens cuando la distribución es plana y menos cuando está concentrada.

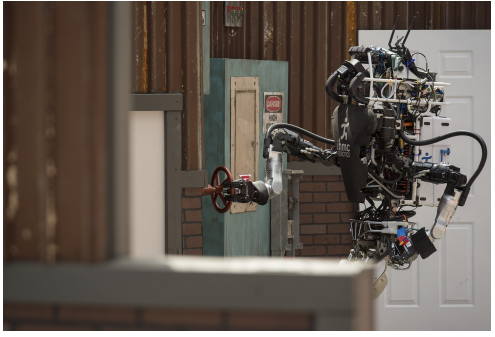
🌡️ Concepto → aplicación: cómo configurar el sampling 🎲.

Aplicación	Temperatura sugerida	Por qué
Extracción de campos en factura	0,0 (greedy)	Resultado reproducible y determinista.
Asistente Q&A corporativo	0,2 – 0,3	Determinista con margen mínimo.
Resumen de informes	0,3 – 0,5	Algo de variabilidad sin alucinar.
Brainstorming de campañas	0,8 – 1,1	Necesitas variedad creativa.
Generación de prompts adversarios	1,2 – 1,5	Para red-teaming explícito (Día 18).

Idea clave. Cualquier respuesta crítica debería ir con temperatura $\leq 0{,}3$.

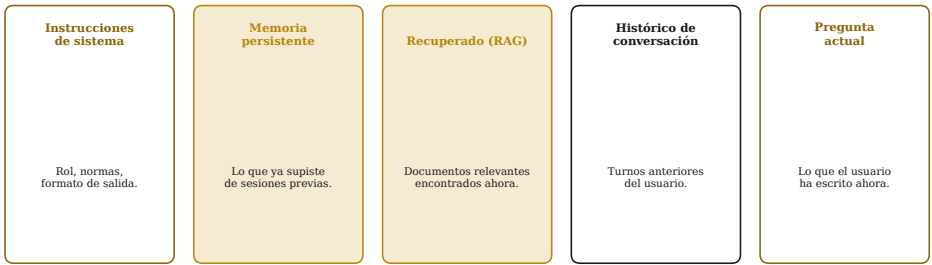
Parte 2 / 4

Context engineering



💬 Anatomía del contexto: cinco piezas a repartir.

Anatomía del contexto que recibe un LLM en cada llamada.



El reparto de tokens entre estas cinco piezas es el "context engineering".

💬 Por qué *context engineering* desplazó a *prompt engineering*.

Prompt engineering (2022-2023). Encontrar la cadena mágica de instrucciones que persuade al modelo 🤖.

Context engineering (2024-). Diseñar qué información llega al modelo y en qué orden: instrucciones, memoria, RAG, histórico, pregunta.

Cambio práctico. El 80 % de la mejora hoy viene de *qué* contexto pasas, no de *cómo* lo escribes. La habilidad técnica más valiosa es montar pipelines de contexto, no escribir prompts ingeniosos.

Idea clave. Esta es la habilidad clave del Tema 5 (agentes) y del Tema 6 (RAG).

💬 Long context, prompt caching 💰, memoria persistente.

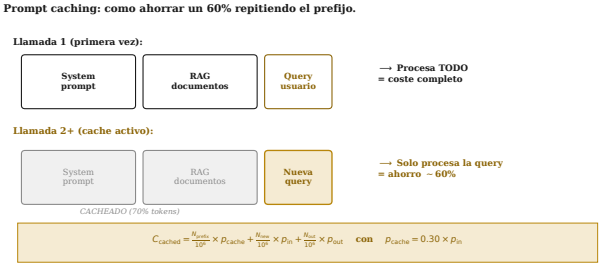
Long context. Modelos como Claude Sonnet manejan 200 K tokens; Gemini 1.5 y derivados llegan a 2 M. Permite cargar un libro entero como contexto.

Prompt caching ⏱️. Si el prefijo (instrucciones + documentos) se repite entre llamadas, el proveedor lo cachea y cobra un 30 % del precio normal. Ahorro brutal en asistentes corporativos con base estable.

Memoria persistente. El asistente recuerda sesiones pasadas escribiendo en un archivo. Lo veremos en el Día 13 con Claude Code.

Idea clave. Estas tres palancas reducen 5–10× el coste real de un asistente en producción.

💰 Prompt caching: cómo ahorrar ~45 % repitiendo el prefijo 🚀.



Idea. Si el prefijo (sistema + RAG) se repite entre llamadas, el proveedor lo cachea y cobra solo el 30 % del precio normal por esos tokens.

Ecuación. $C = \frac{N_{\text{prefix}}}{10^6} \cdot p_{\text{cache}} + \frac{N_{\text{new}}}{10^6} \cdot p_{\text{in}} + \frac{N_{\text{out}}}{10^6} \cdot p_{\text{out}}$ con $p_{\text{cache}} = 0{,}30 \times p_{\text{in}}$.

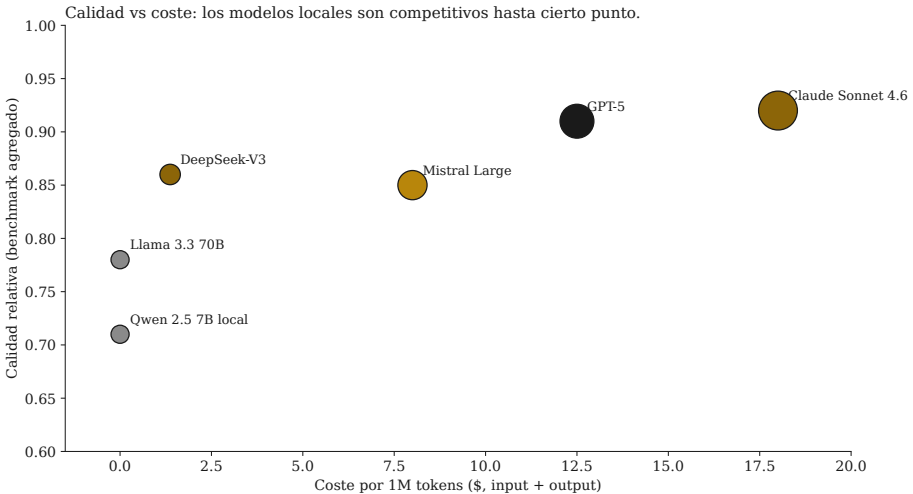
Ejemplo numérico. Prefijo de 7.000 tokens, query nueva de 200 tokens, respuesta de 800 tokens, Sonnet a \$3/M input y \$15/M output. Sin caché: \$0.034. Con caché: \$0.019. Ahorro: 44 %.

Idea clave. En un asistente con base de contexto estable, el caching recorta casi a la mitad el coste.

Parte 3 / 4

Coste y elección de modelo

💰 Calidad vs coste por modelo, vista de pájaro.



💰 Ejemplo a mano: cuánto cuesta una sesión RAG 🤖 en Sonnet.

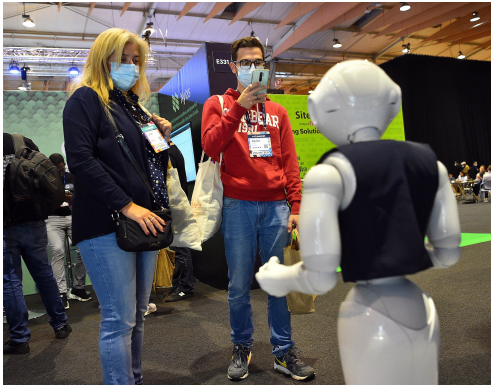
Cálculo de coste a mano: una sesión RAG empresarial con Claude Sonnet.

Componente	Tokens estimados	Coste $/M (Sonnet)	$ por sesión
Instrucciones de sistema	2 000	3,00 (input)	0,006
RAG: 5 chunks de 1 000 tokens	5 000	3,00 (input)	0,015
Histórico conversación 4 turnos	3 000	3,00 (input)	0,009
Pregunta usuario	200	3,00 (input)	0,001
Respuesta del modelo	800	15,00 (output)	0,012
Total por sesión	**11 000**	**—**	**0,043 $**

Si haces 50.000 sesiones/mes ⇒ $2.150/mes. Compara con un Llama 3.3 self-hosted: $0/sesión + GPUs amortizadas.

Parte 4 / 4

Casos y elección



🤖 Concepto → aplicación: qué LLM elijo y para qué.

Caso	Modelo recomendado	Coste estimado	Latencia
Asistente Q&A para 50K empleados	Claude Sonnet + caching	1 000–3 000 $/mes	< 2 s
Generación de borradores marketing	GPT-5 o Mistral Large	200–500 $/mes	< 3 s
OCR de documentos masivos	Pipeline VLM local (Qwen-VL)	0 $/petición	5–15 s
Asistente de código interno	Claude Code + Sonnet 4.6	50 $/mes/dev	< 5 s
Resumen masivo (millones de docs)	DeepSeek-V3 (MoE)	< 100 $/M tokens	10–30 s

Idea clave. La elección de modelo es siempre coste × latencia × calidad × privacidad.

Lo que el alumno se lleva a casa de este día.

Concepto 1. 🤖 Un LLM es un predictor de próximo token; la "creatividad" es sampling 🎲 sobre una distribución.

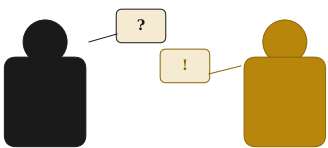
Concepto 2. 💬 El contexto se divide en cinco piezas: instrucciones, memoria, RAG, histórico y pregunta. Su orquestación es *context engineering*.

Concepto 3. 💰 El coste por petición se calcula en tokens × $/M tokens. Con caching y modelos locales, baja un orden de magnitud.

Para la práctica. Probar Sonnet vs Qwen 2.5 local con el mismo prompt y reportar coste, latencia ⏱️ y calidad.

Próximo día. Día 12 (Tema 4.2): generación multimodal de imagen, audio y video.

Pausa para debatir: ¿Está sobrevalorada la habilidad de “redactar prompts”?



La pregunta. Si el 80 % del valor viene de qué contexto pasas, ¿por qué seguimos hablando de prompt engineering?

Posición A. Sí, es un mito; lo importante es el pipeline.

Posición B. No, sigue importando: el orden y el formato del prompt influyen en la salida.

Reglas. 5 minutos, dos turnos de palabra por bando, móviles boca abajo. Yo modero, no participo.

Hasta el Día 12.

Imagen, audio, video, código.

Eduardo C. Garrido-Merchán

ecgarrido@comillas.edu