Deep Learning para Business Analytics

Día 07 · Tema 2 — Atención, puente al Transformer

Por qué la recurrencia ya no es suficiente.

Eduardo C. Garrido-Merchán

Profesor Propio Adjunto, Métodos Cuantitativos, ICADE

ecgarrido@comillas.edu

Madrid, Día 07

ejemplos del día

Lo que vamos a cubrir hoy.

1. Las RNN y su límite

2. La idea de la atención

3. RNN vs atención, comparativa

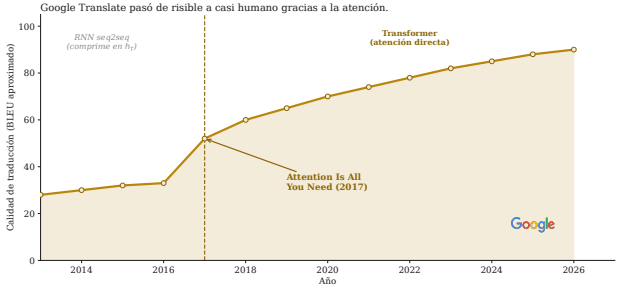
4. Hoja de ruta

Al final del día: entiendes la 🧠 atención conceptual y haces un ejemplo a mano. El Día 09 te enseña el ⚡ Transformer completo.

Parte 1 / 4

Las RNN y su límite

💬🚀 Google Translate pasó de risible a casi humano en 2017 gracias a la atención.



Antes de la atención. La RNN comprimía una frase de 50 palabras en un solo vector h_T. Imagina resumir *El Quijote* en un post-it.

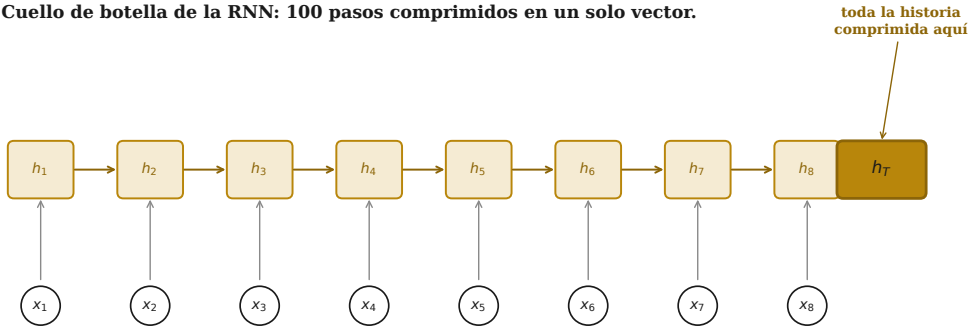
El punto de inflexión. En 2017, "Attention Is All You Need" permitió al decoder mirar directamente cada posición del encoder. La calidad de traducción se disparó.

Hoy. Google Translate, DeepL y todos los traductores modernos usan atención. Sin ella, los modelos de lenguaje actuales no existirían.

Idea clave. La atención resolvió el cuello de botella que impedía traducir frases largas.

⚠️ Cuello de botella: toda la historia cabe en un solo vector.

Cuello de botella de la RNN: 100 pasos comprimidos en un solo vector.



Por qué duele el bottleneck.

Información comprimida. 🔒 Con $h_T \in \mathbb{R}^{64}$ hay que resumir 100 pasos de historia en 64 números. Para tareas con dependencias largas (traducción de un párrafo, predicción de demanda a 12 meses), esos 64 números no llegan.

Paralelismo limitado. ⏱️ Para calcular h_{100} hay que pasar por $h_1, h_2, \ldots, h_{99}$ en orden. La GPU se queda esperando.

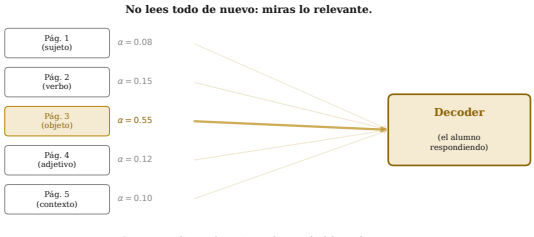
Olvido en la retropropagación. ⚠️ El gradiente de $\partial \mathcal{L} / \partial w$ se calcula multiplicando matrices a lo largo de toda la cadena temporal; para secuencias largas se desvanece o explota.

Idea clave. La atención resuelve los tres problemas al mismo tiempo.

Parte 2 / 4

La idea de la atención

🔍📖 Atención = mirar atrás en tus apuntes cuando respondes una pregunta.



Los pesos α_j dicen cuánto mirar cada posición del encoder.

La analogía. Estás en un examen (decoder). No relees todo el temario: vuelves a las páginas concretas de tus apuntes (encoder) que contienen la respuesta.

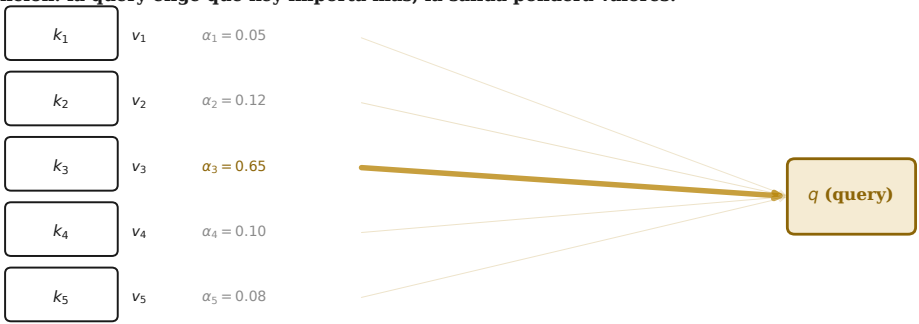
Los pesos α_j. Miden cuánto "mirar" cada posición del encoder. Un peso alto (0.55) significa "esta página es crucial"; uno bajo (0.03) significa "irrelevante".

Ventaja. La RNN obligaba a leer todos los apuntes en orden. La atención permite saltar directamente a lo relevante.

IDEA CLAVE. No lees todo de nuevo: miras lo relevante. Esa es la atención.

🧠 Atención: la query pondera los valores según los keys.

Atención: la query elige qué key importa más; la salida pondera valores.



Atención escalada en una fórmula.

$$\mathrm{Att}(Q, K, V) \;=\; \mathrm{softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right) V.$$

Lectura.

- $Q \in \mathbb{R}^{T \times d}$ — las queries (qué buscamos).
- $K \in \mathbb{R}^{T \times d}$ — los keys (qué hay disponible).
- $V \in \mathbb{R}^{T \times d_v}$ — los values (qué llevarse si encajan).

Coste. ⚠️ La matriz de scores $QK^{\top}$ es $T \times T$. Para $T = 4000$ hay 16 millones de elementos. Esta es la fuente del coste cuadrático del Transformer.

Idea clave. Cada token "mira.ª todos los demás y elige qué le interesa.

🔍 Ejemplo a mano: atención sobre 3 keys con números.

Atención escalada (sin escala por $\sqrt{d}$, para 2 dimensiones): ejemplo a mano.

Query y keys	$q = (1, 0);\ \ K = \{(1, 0), (0.5, 0.5), (-1, 0.5)\}$
Producto $q \cdot K^\top$	$= (1.0,\ 0.5,\ -1.0)$
Softmax	$\alpha = (0.57,\ 0.35,\ 0.08)$
Values	$V = (10, 20, 30)^\top$
Salida $\alpha \cdot V$	$= 15.04$

🎯 Cálculo paso a paso, en texto.

Setup. 🔍 Una query $q = (1, 0)$, tres keys $k_1 = (1, 0)$, $k_2 = (0{,}5, 0{,}5)$, $k_3 = (-1, 0{,}5)$, y tres values $v_1 = 10$, $v_2 = 20$, $v_3 = 30$.

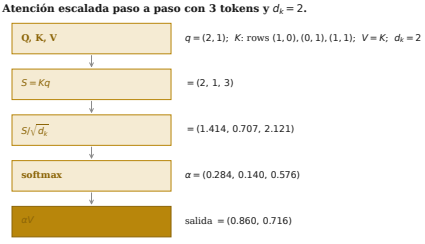
Paso 1 — scores. $q \cdot k_1 = 1$, $q \cdot k_2 = 0{,}5$, $q \cdot k_3 = -1$.

Paso 2 — softmax. $\alpha_1 = e^1/(e^1 + e^{0{,}5} + e^{-1}) \approx 0{,}57$; $\alpha_2 \approx 0{,}35$; $\alpha_3 \approx 0{,}08$.

Paso 3 — salida. $\hat{y} = 0{,}57 \cdot 10 + 0{,}35 \cdot 20 + 0{,}08 \cdot 30 =$ **15,1**.

Idea clave. La query "se parece"más a k_1, así que el resultado tira hacia v_1. Si la query cambia, los pesos cambian.

🔍 Ejemplo a mano: atención escalada paso a paso con 3 tokens.



Setup. $q = (2, 1)$, tres keys $k_1 = (1, 0)$, $k_2 = (0, 1)$, $k_3 = (1, 1)$, values $V = K$, $d_k = 2$.

Paso 1 — scores. $Kq = (2, 1, 3)$.

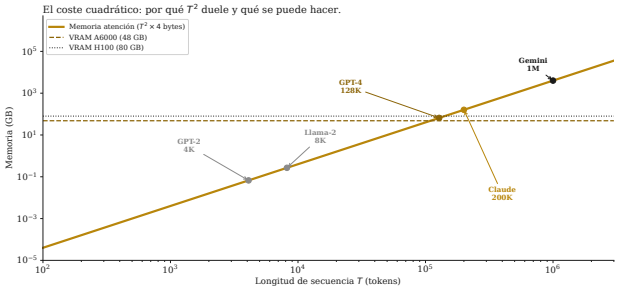
Paso 2 — escalar. $S/\sqrt{2} = (1{,}414, 0{,}707, 2{,}121)$.

Paso 3 — softmax. $\alpha = (0{,}284, 0{,}140, 0{,}576)$. El token 3 domina porque k_3 es más afín a q.

Paso 4 — salida. $\alpha V = (0{,}860, 0{,}716)$.

Idea clave. Verifica: los pesos suman 1 y la salida tira hacia $v_3 = (1, 1)$.

⚠️💰 El coste cuadrático: por qué T^2 duele y qué se puede hacer.



La ecuación. La matriz de atención $QK^\top$ ocupa $T^2 \times 4$ bytes en float32. Para $T = 128\,000$ (GPT-4) eso son ~ 61 GB solo en scores.

Dónde duele. Una A6000 (48 GB) se queda sin memoria a $\sim$ 110K tokens. Para contextos más largos (Claude 200K, Gemini 1M) se necesitan trucos.

Soluciones. KV-cache (guarda keys/values ya calculados), Flash Attention (recomputa en lugar de almacenar), Paged Attention (vLLM), o modelos sub-cuadráticos como Mamba (Día 10).

Idea clave. El Transformer paga calidad con memoria $O(T^2)$. Los trucos de ingeniería lo hacen viable.

Parte 3 / 4

RNN vs atención, comparativa

⚖️ Comparativa formal entre familias.

	Recurrente (RNN/LSTM/GRU)		Atencional (Transformer)
Recorre	secuencialmente $t = 1...T$	**Recorre**	todas las posiciones a la vez
Memoria	h_t comprime el pasado	**Memoria**	matrices Q, K, V explícitas
Paralelismo	limitado por la cadena temporal	**Paralelismo**	total dentro de la secuencia
Coste	$\mathcal{O}(T)$ tiempo, $\mathcal{O}(d^2)$ memoria	**Coste**	$\mathcal{O}(T^2 d)$ tiempo, $\mathcal{O}(T^2)$ memoria
Brilla en	secuencias cortas, online	**Brilla en**	secuencias largas con contexto rico

🚀 Concepto → aplicación: cuándo recurrente y cuándo atencional.

Caso	Mejor opción	Por qué	Coste relativo
📊 Forecast 12 sem. hist. 156 sem.	LSTM/GRU	Memoria suficiente, dataset moderado.	Bajo
🌍 Traducción de un párrafo	⚡ Transformer	Dependencias largas, contexto rico.	Medio
💬 Asistente convers. 16K tokens	Transformer + KV cache	Posiciones todas a la vez.	Alto
🎤 Audio streaming (latencia ms)	RNN/GRU o SSM (Día 10)	Estado fijo, decisión online.	Bajo
📄 Informes 500 págs.	Transformer + RAG	Búsqueda + síntesis.	Muy alto

Idea clave. Recurrente no es obsoleto; sigue ganando en latencia y stream.

Parte 4 / 4

Hoja de ruta

🎯 De aquí al final del Bloque I.

De aquí al final del Bloque I: hoja de ruta de los próximos cuatro días.

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

Concepto 1. 🔒 La RNN aprieta toda la historia en un solo h_T; la 🧠 atención mira directamente a todos los pasos.

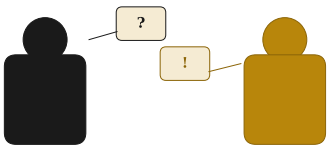
Concepto 2. ⚡ Atención = $\text{softmax}(QK^\top/\sqrt{d})V$. Es el bloque base del Transformer.

Concepto 3. ⚠️ El coste es cuadrático en la longitud de la secuencia (Día 10 ofrece alternativas).

Para la práctica. 🔧 Implementar la atención a mano con numpy en 20 líneas y verificarla contra `torch.nn.functional.scaled_dot_product_attention`.

Próximo día. 📖 Día 08 (Tema 3.1): embeddings semánticos.

Pausa para debatir: Si la atención es $O(T^2)$, ¿por qué ganó?



La pregunta. ⚡ El Transformer es caro en memoria. ¿Es una victoria perpetua o un accidente histórico?

Posición A. 👑 Perpetua: ningún reemplazo ha igualado calidad punta.

Posición B. 🚀 Accidente: Mamba y MoE ya lo desplazan en contexto largo (Día 10).

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

Hasta el Día 08.

Vectores que significan.

Eduardo C. Garrido-Merchán

ecgarrido@comillas.edu