Transformers y atención como bloque universal

Self-attention (Q,K,V), multi-head, conexiones residuales + LayerNorm, positional encodings y variantes encoder/decoder.

Temario

- Self-attention: consultas (Q), claves (K) y valores (V); atención token<>token y complejidad $O(T^2)$.
- Multi-head attention, normalización (LayerNorm) y conexiones residuales (skip).
- Arquitectura encoder/decoder y variantes (encoder-only, decoder-only para LLM).
- Positional encodings y variantes (seno/coseno, RoPE, ALiBi, etc.).

Atención encoder-decoder

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

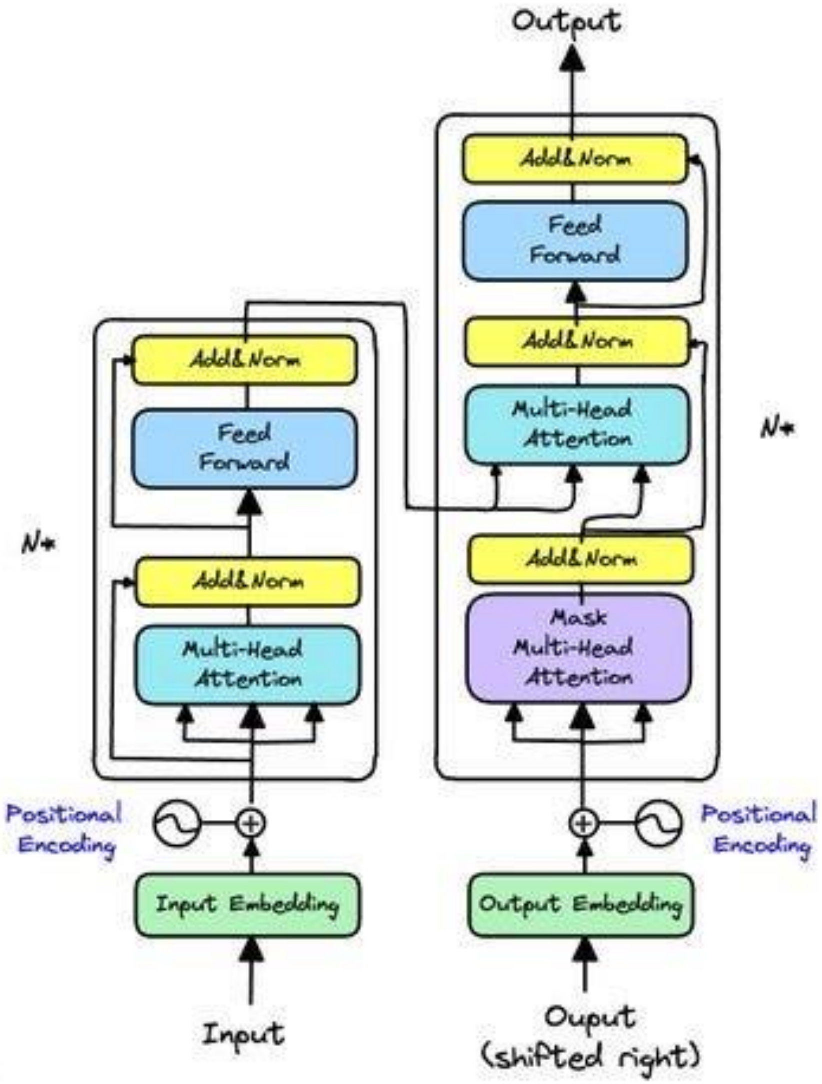
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

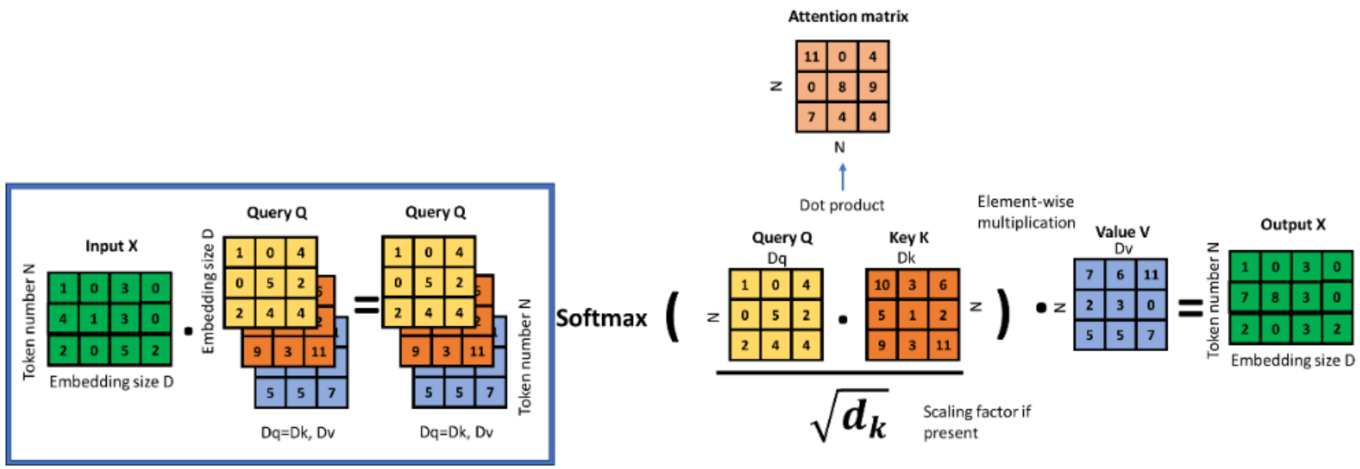
*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.



Mecanismo self-attention



$$Attention\,(Q, K, V) = softmax\left(\frac{Q \cdot k^T}{\sqrt{d_k}}\right) \cdot V$$

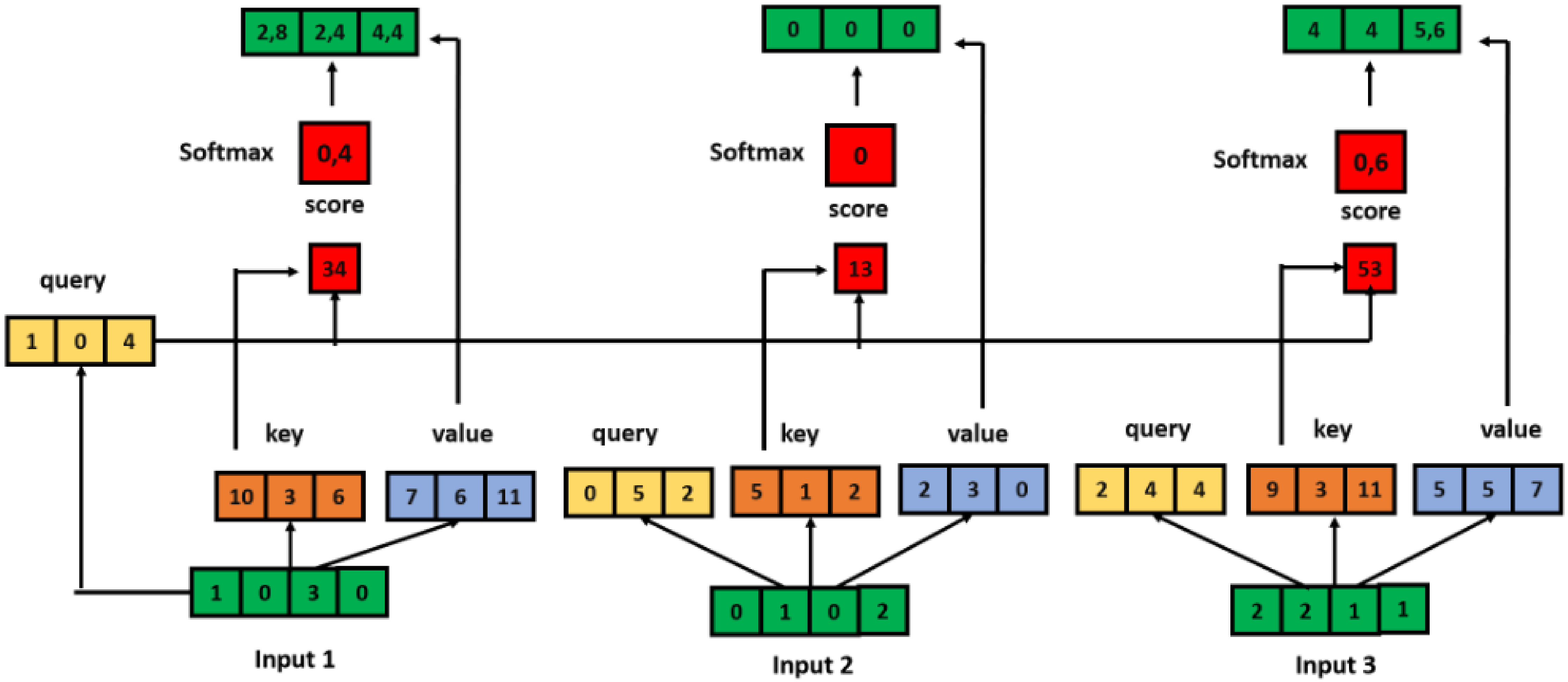
Softmax como distribución de atención

$$y = \frac{e^{x_i}}{\sum_{i=1}^{i=n} e^{x_i}}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = softmax(x) = f\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right) = \begin{bmatrix} \frac{e^{x_1}}{e^{x_1} + e^{x_2} + e^{x_3}} \\ \frac{e^{x_2}}{e^{x_1} + e^{x_2} + e^{x_3}} \\ \frac{e^{x_3}}{e^{x_1} + e^{x_2} + e^{x_3}} \end{bmatrix}$$

$$python : y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = softmax(x) = f\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right) = \frac{[x_1 \quad x_2 \quad x_3]}{e^{x_1} + e^{x_2} + e^{x_3}} = \frac{e^x}{sum(e^x)}$$

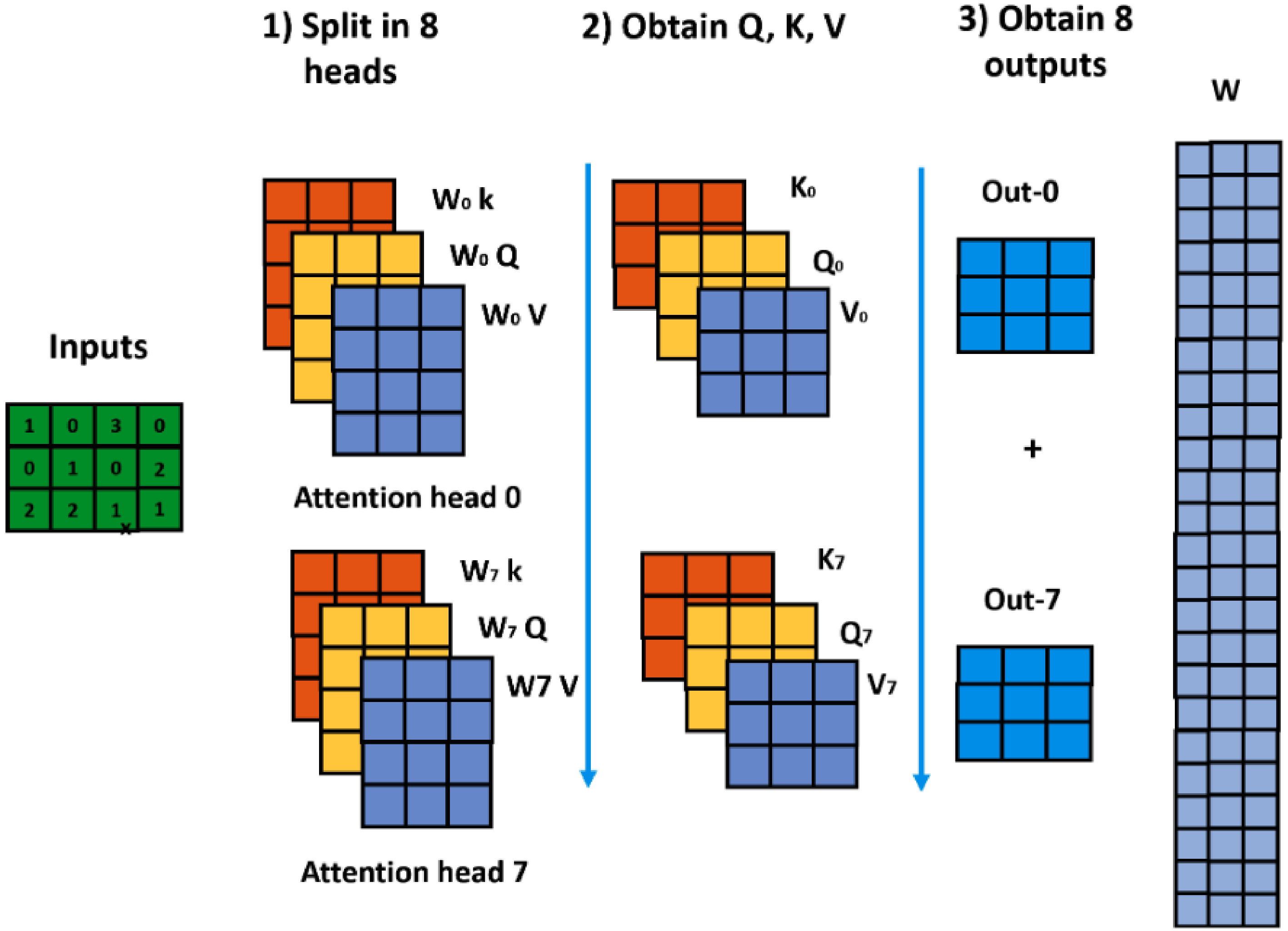
Self-attention token->token



Proyecciones Q, K, V = X·W

$$\boldsymbol{Q} = \boldsymbol{X} \cdot \boldsymbol{W}^{\mathrm{Q}}, \ \boldsymbol{K} = \boldsymbol{X} \cdot \boldsymbol{W}^{V}, \ \boldsymbol{V} = \boldsymbol{X} \cdot \boldsymbol{W}^{V}$$

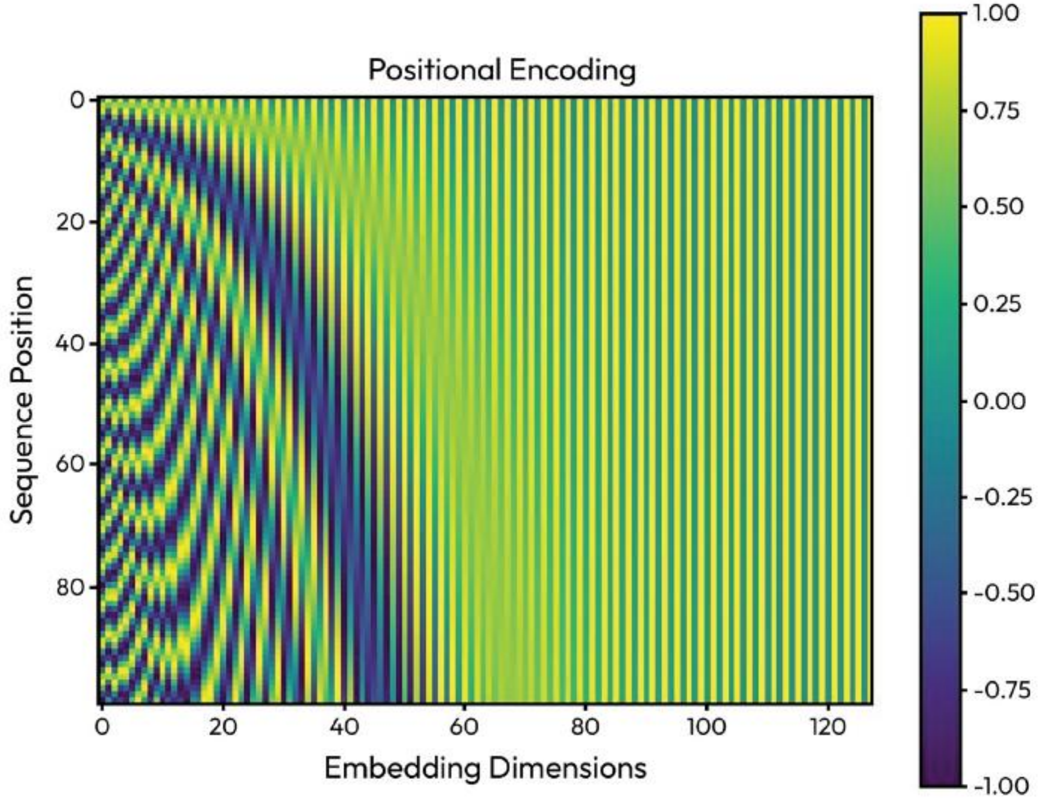
Multi-head self-attention



Costo computacional $O(T^2)$

$$time = \mathcal{O}(T^2 + d) \qquad space = \mathcal{O}(T^2 + Td)$$

Codificación posicional seno/coseno



$$PE_{(pos,2i)} = \sin\left(pos/1000^{2i/d}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(pos/1000^{2i/d}\right)$$

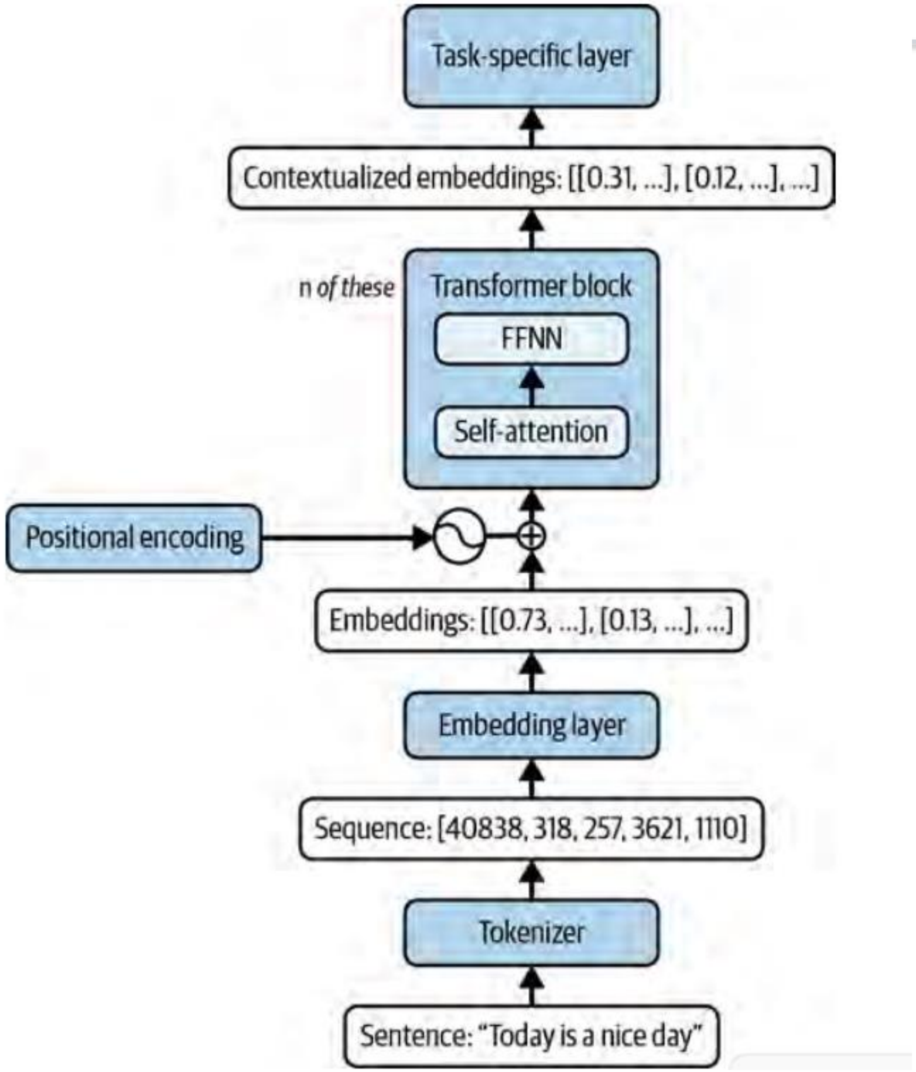
Positional encodings modernos

RoPE, ALiBi y embeddings posicionales aprendidos

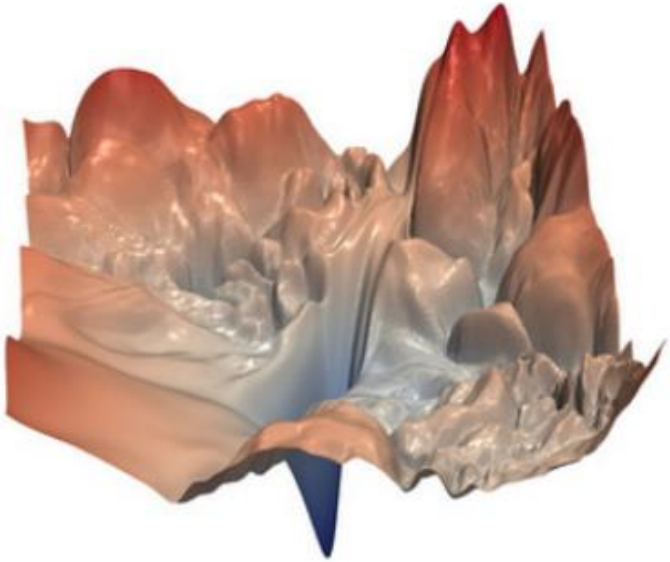
- Objetivo: inyectar información de orden sin cambiar el bloque de atención (mismo Q,K,V).
  - RoPE (Rotary): rota Q y K según posición -> atención depende de distancias relativas; muy común en LLM.
  - ALiBi: sesgo lineal en logits de atención por distancia -> favorece relaciones locales y escala a contextos largos.
  - Embeddings posicionales aprendidos: vector por posición (absoluto) -> simple/efectivo, pero extrapola peor fuera del rango entrenado.

Regla práctica: si esperas extrapolar a contextos más largos, RoPE/ALiBi suelen ser más robustos que embeddings absolutos aprendidos.

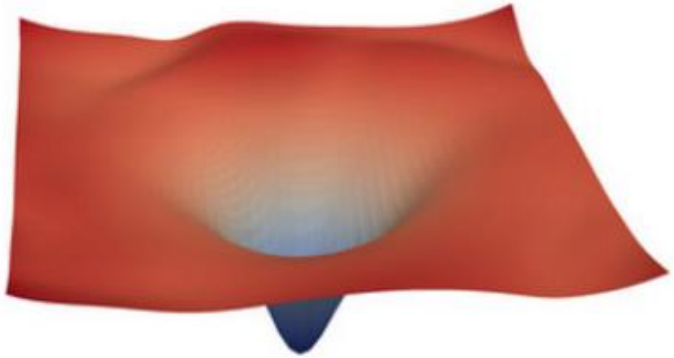
Bloque Transformer basado en modelo de lenguaje



Conexiones residuales (skip)



Loss landscape without residual



Loss landscape with residual

LayerNorm: normalizar y re-escalar

$$\mu = \frac{1}{d}\sum_{i=1}^{d} x_i \qquad \sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2}$$

$$\hat{x} = \frac{(x - \mu)}{\sigma}$$

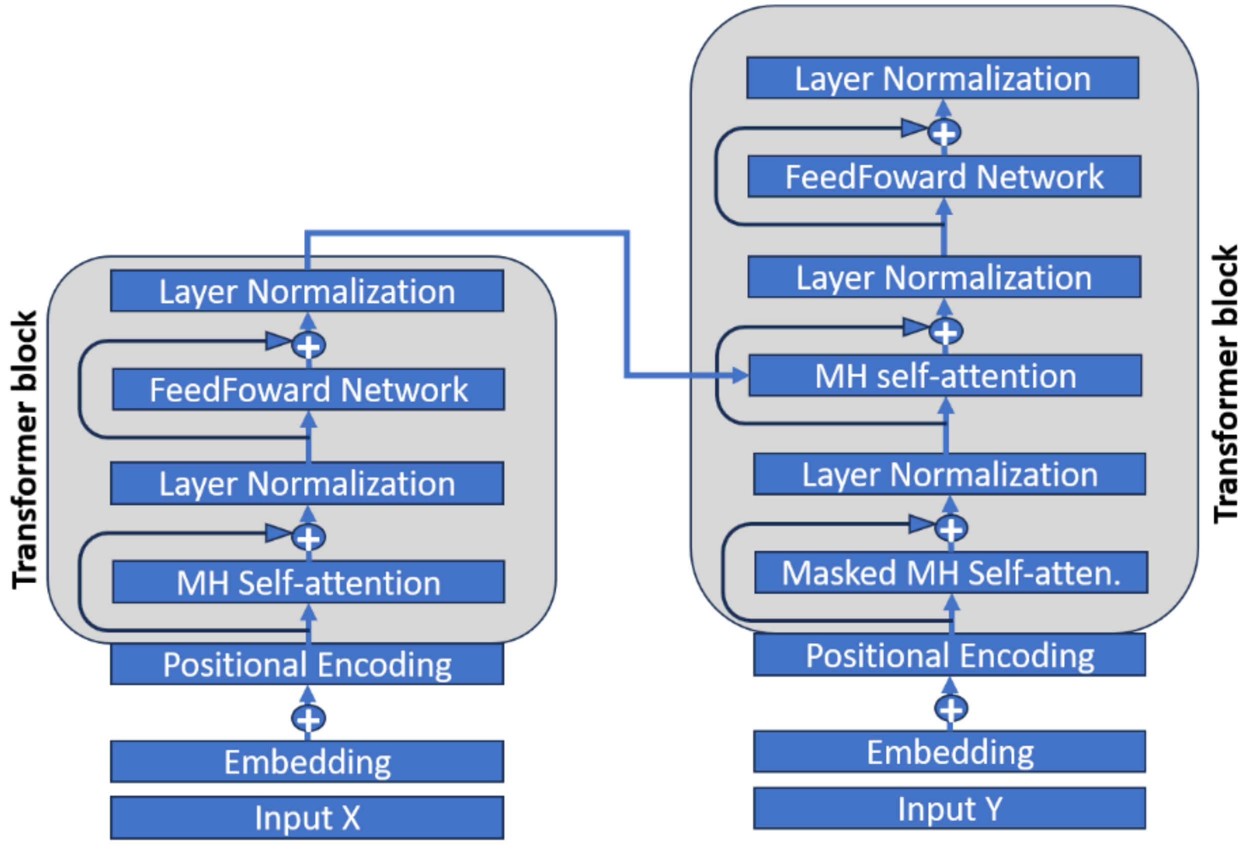
$$LayerNormalization = \gamma\hat{x} + \beta$$

Pre-Norm en Transformers modernos

$$\boldsymbol{H} = LayerNorm(\boldsymbol{X} + MultiHeadSelfAttention(\boldsymbol{X}))$$

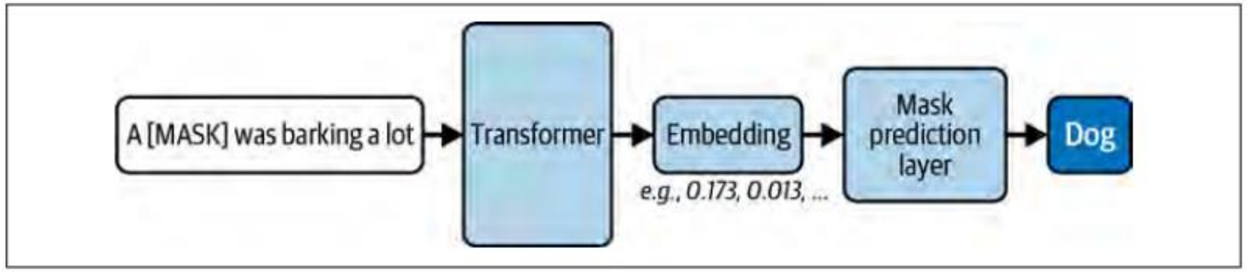
$$\boldsymbol{H} = LayerNorm(\boldsymbol{H} + FFN(\boldsymbol{H}))$$

Encoder vs Decoder (self, masked, cross-attention)

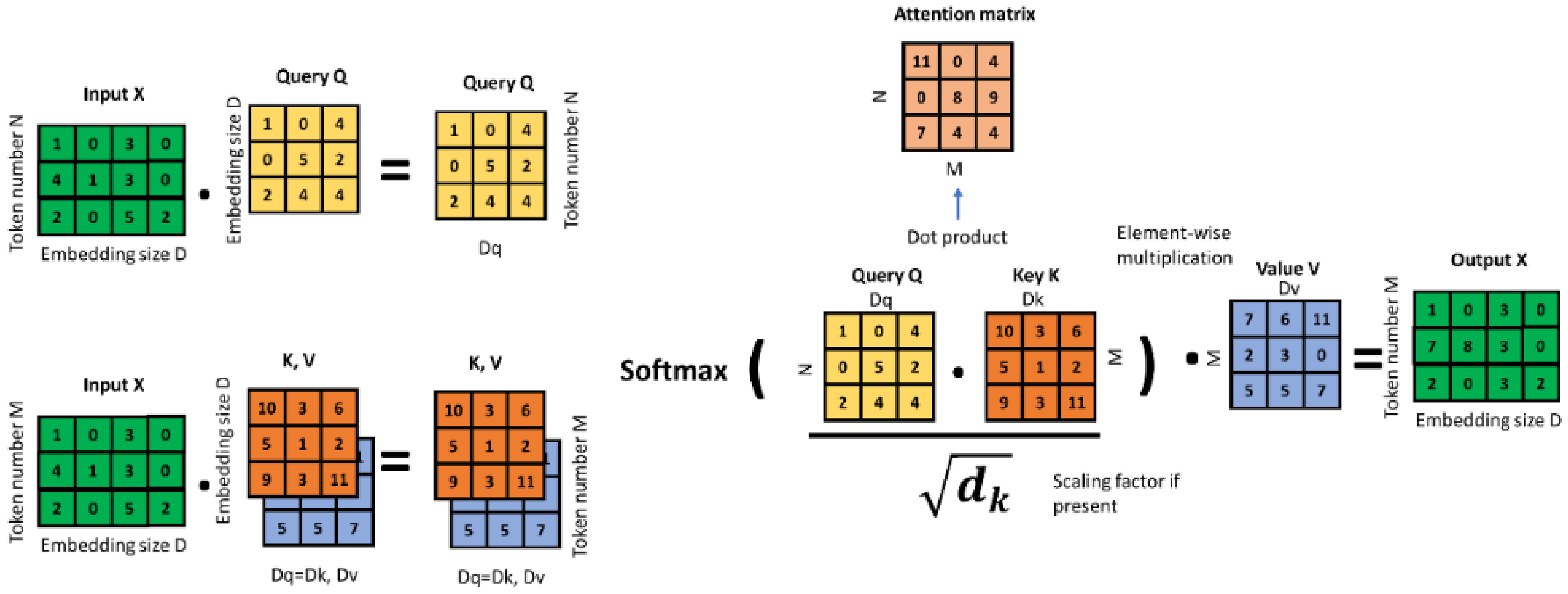


Variantes: encoder-only (MLM) vs decoder-only (LLM)

- Encoder-only: predicción de [MASK] (BERT).
- Decoder-only: modelo autoregresivo (next-token) con máscara causal.
- Encoder/decoder: cross-attention para condicionamiento (traducción, T5, etc.).



Cross-attention Q(source) <> K,V(contexto)



Máscara causal (no mirar el futuro)

	<START>	TO	BE	OR	NOT	TO	BE
<START>	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
TO		$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
BE			$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
OR				$-\infty$	$-\infty$	$-\infty$	$-\infty$
NOT					$-\infty$	$-\infty$	$-\infty$
TO						$-\infty$	$-\infty$
BE							$-\infty$

Modelo de lenguaje autoregresivo

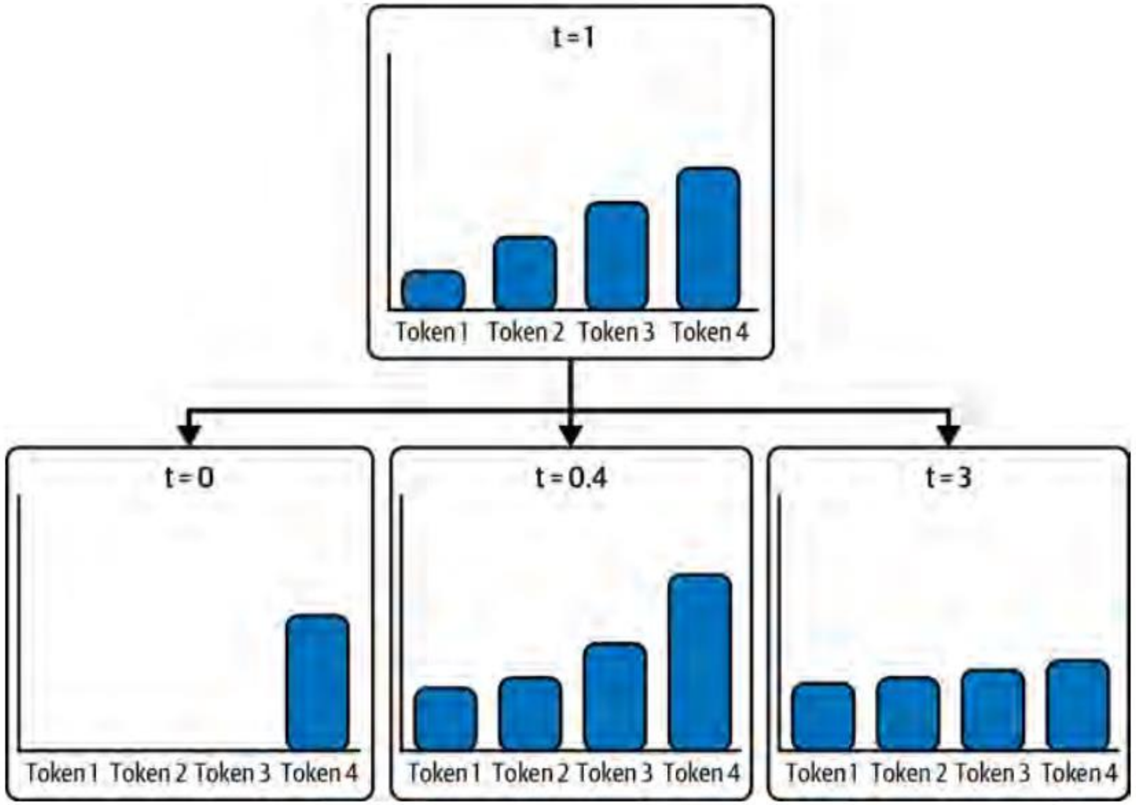
$$P(w|h) = P(w_n|w_{1:n-1}) = \prod_{i=1}^{n} P(w_i|w_{1:i-1})$$

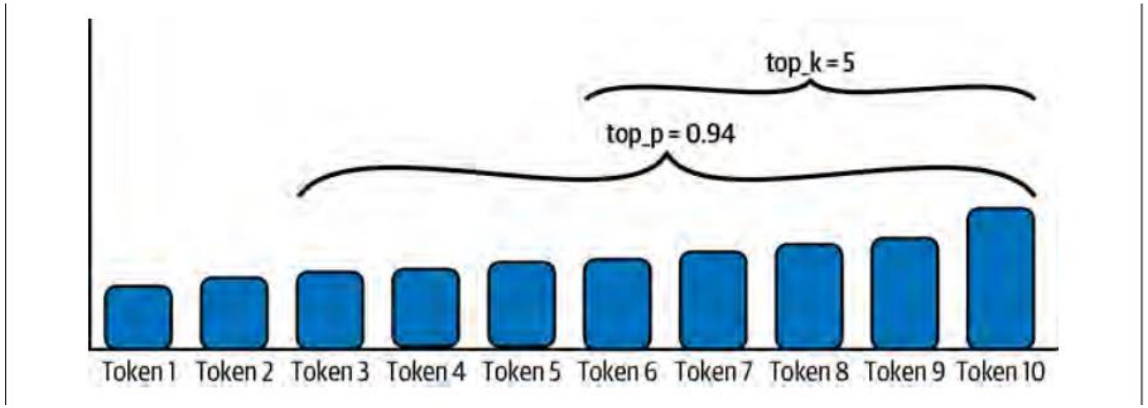
Pérdida de entropía cruzada (next-token)

$$L_{CE} = -\sum_{w \in V} y_t[w] \log \hat{y}_t[w]$$

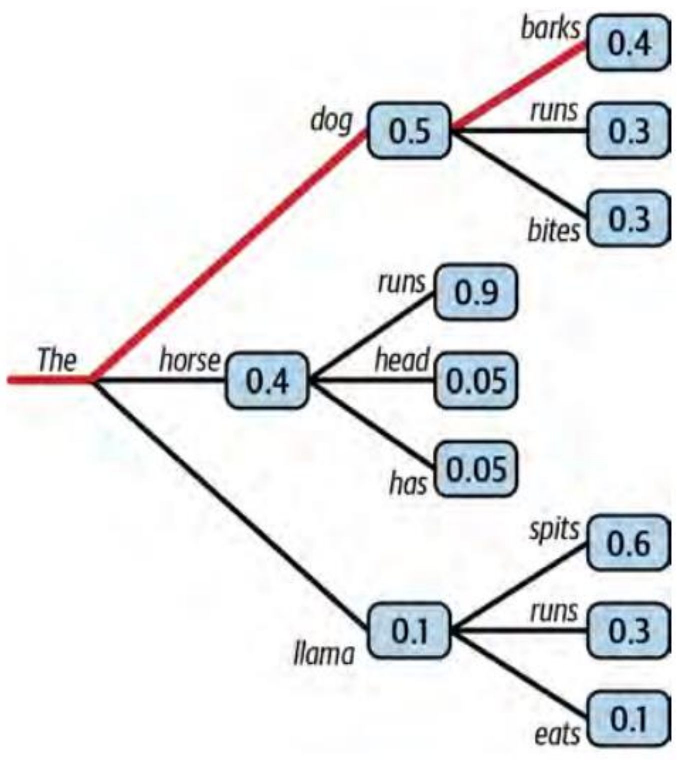
Decoding: temperatura, top-k y top-p

- Temperatura (T): controla entropía (más alto = más diversidad).
- Top-k: restringe a los k tokens más probables.
- Top-p (nucleus): restringe al menor conjunto con probabilidad acumulada ≥ p.

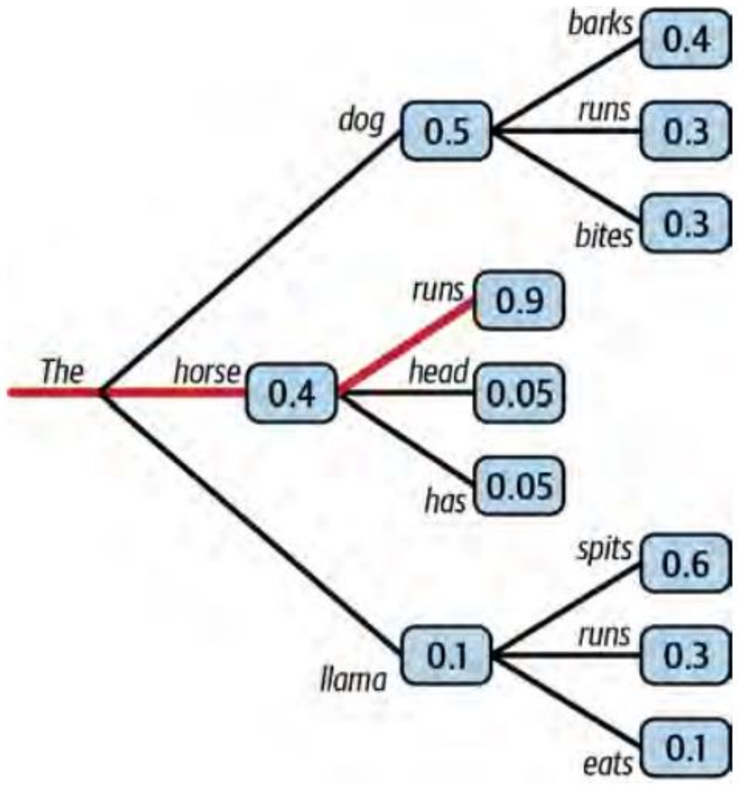




Decoding: Greedy y beam search



Decodificación greedy



Decodificación beam-search

Entrenamiento y Optimización

- AdamW (weight decay desacoplado).
- LR: warmup corto (1-5%) -> cosine decay.
- Gradient clipping (norm ≤ 1.0) para evitar spikes.
- Label smoothing ($\varepsilon \approx 0.1$) mejora calibración.
- Mixed precision (BF16/FP16) para acelerar y ahorrar memoria.

Preentrenamiento y ajuste fino de modelos Transformer