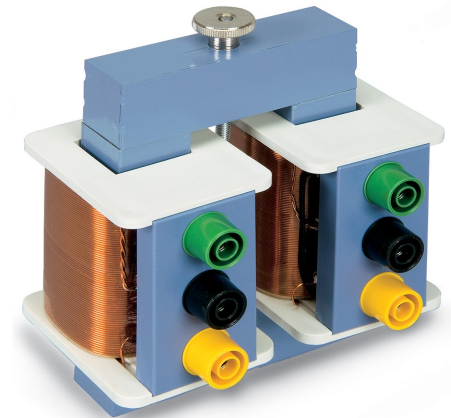
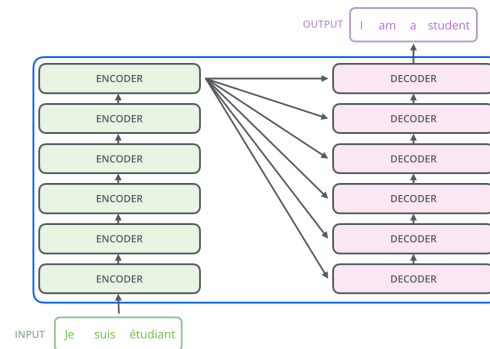


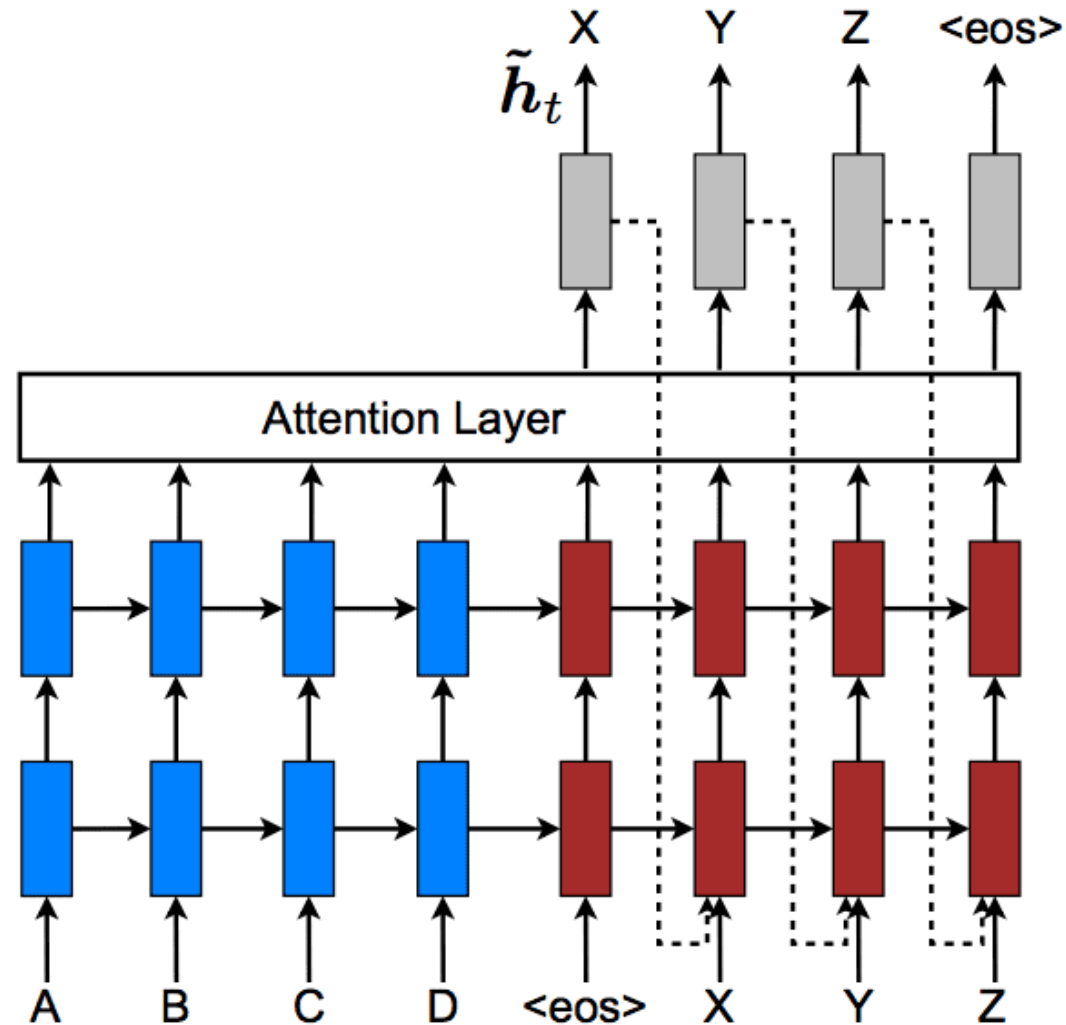


# Transformers

Julio Weissman



# Antes de los transformers



# El artículo que cambió todo

---

## 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

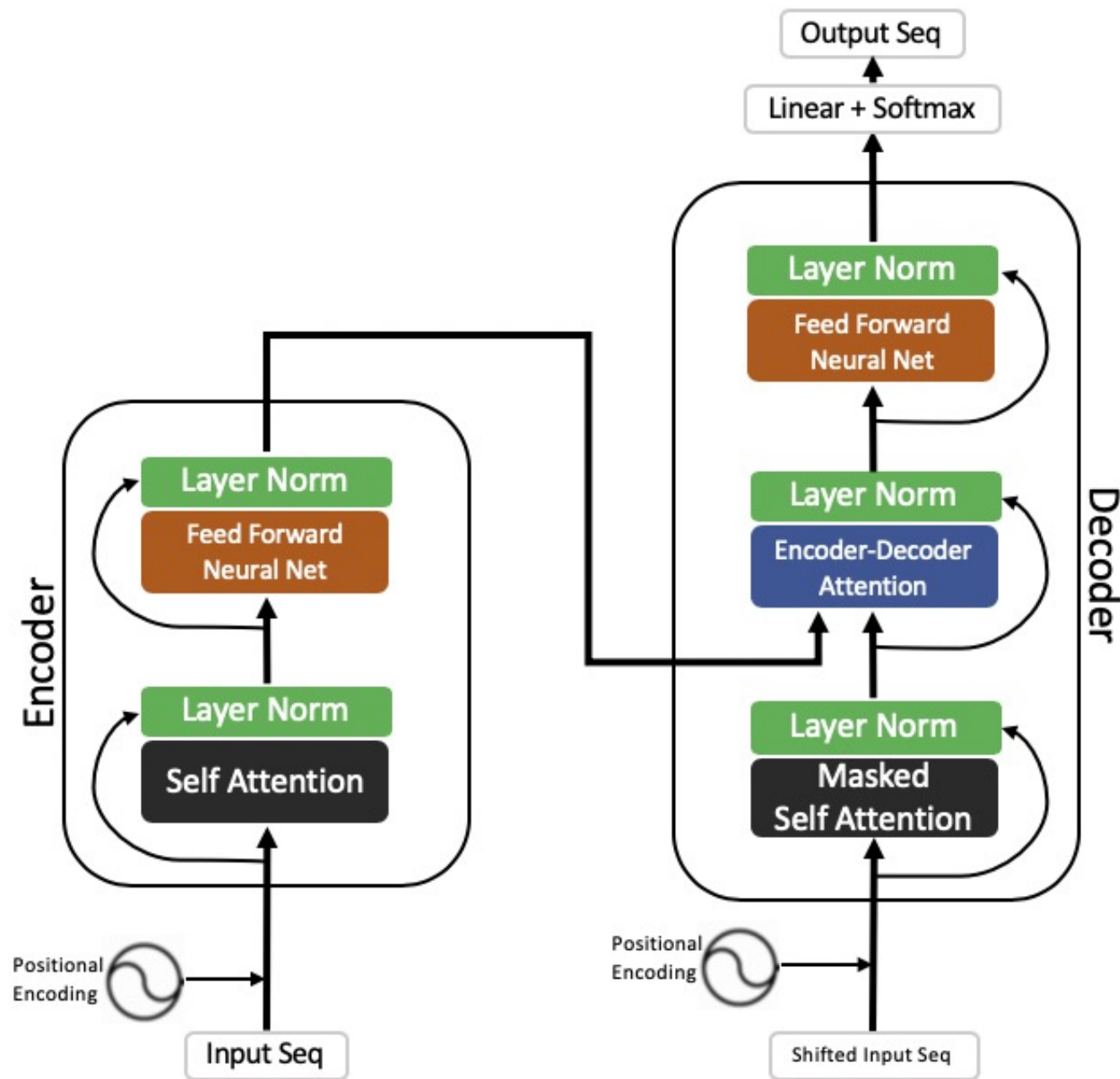
**Llion Jones\***  
Google Research  
llion@google.com

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

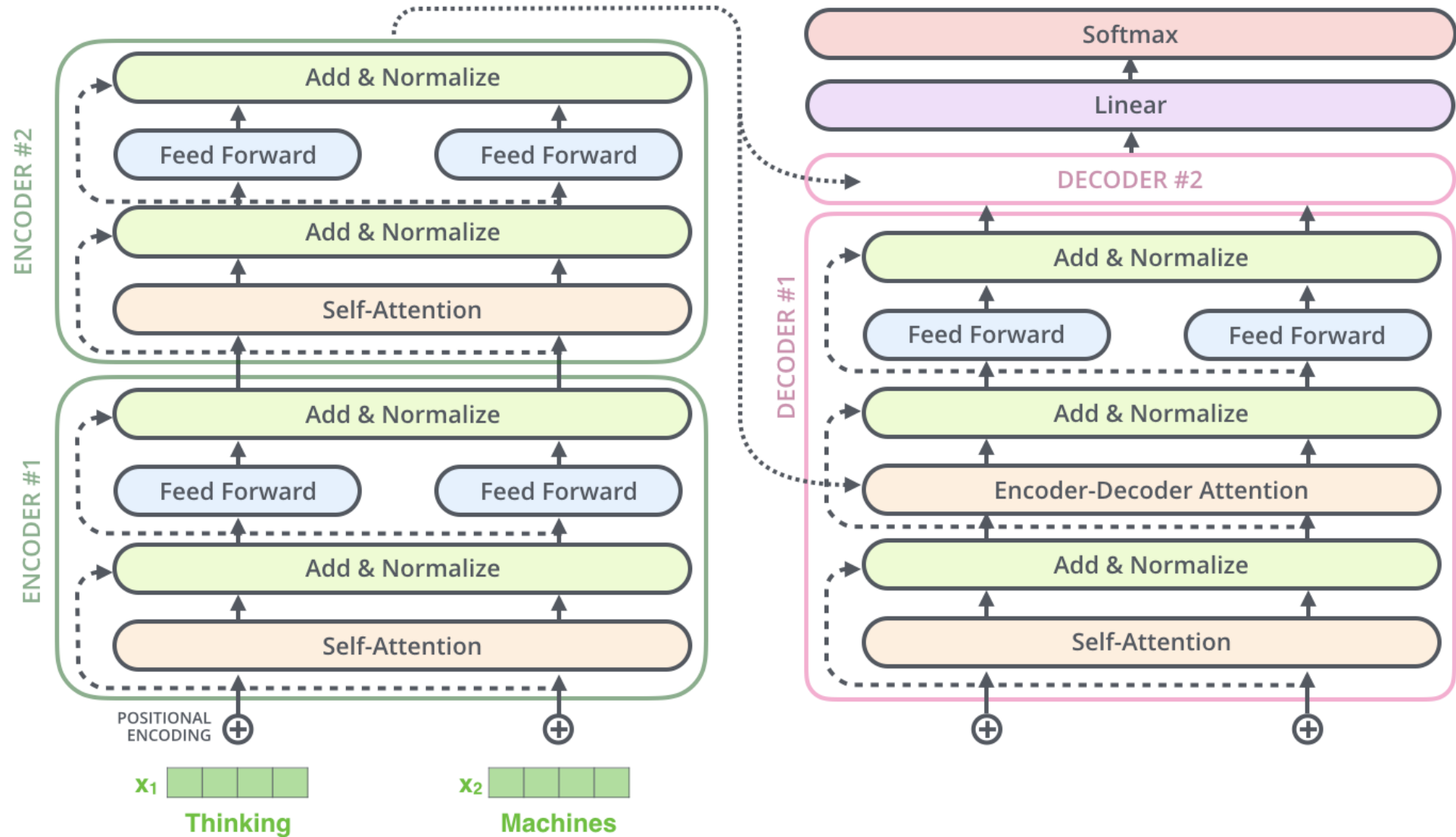
**Łukasz Kaiser\***  
Google Brain  
lukaszkaiser@google.com

**Illia Polosukhin\* ‡**  
illia.polosukhin@gmail.com

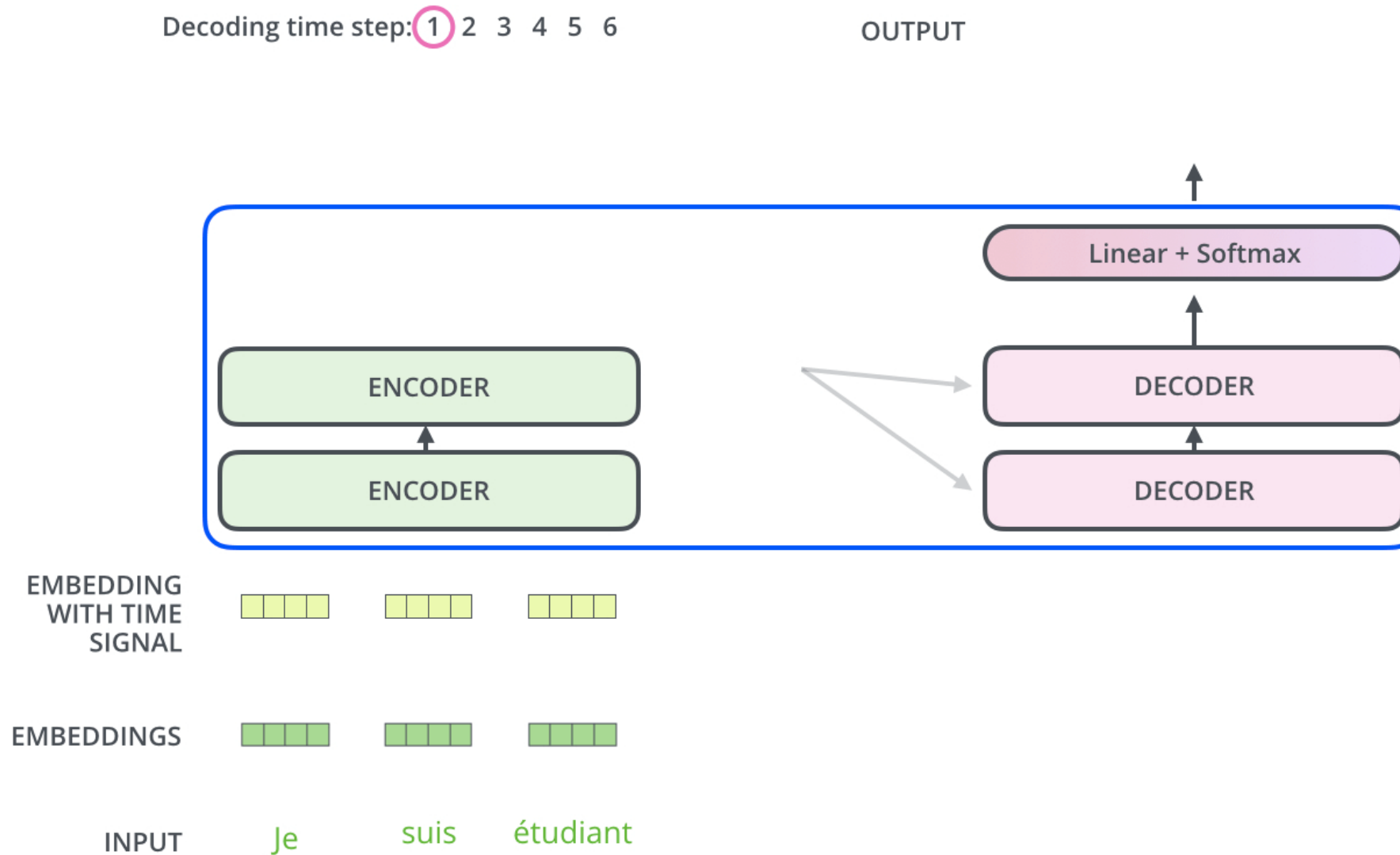
# Modelo Propuesto (Transformer)



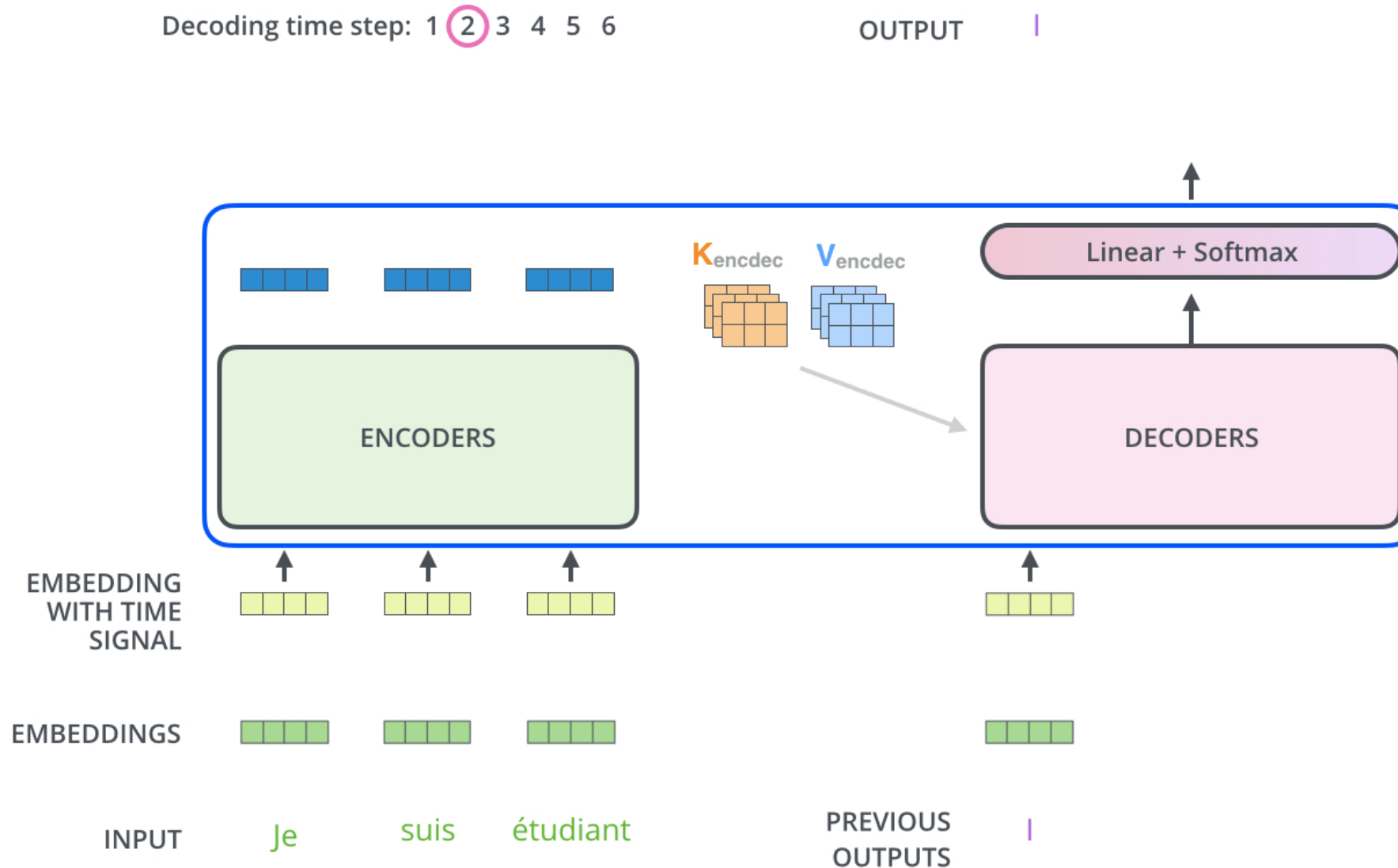
# ¿Cómo funciona todo?



# ¿Cómo funciona todo?

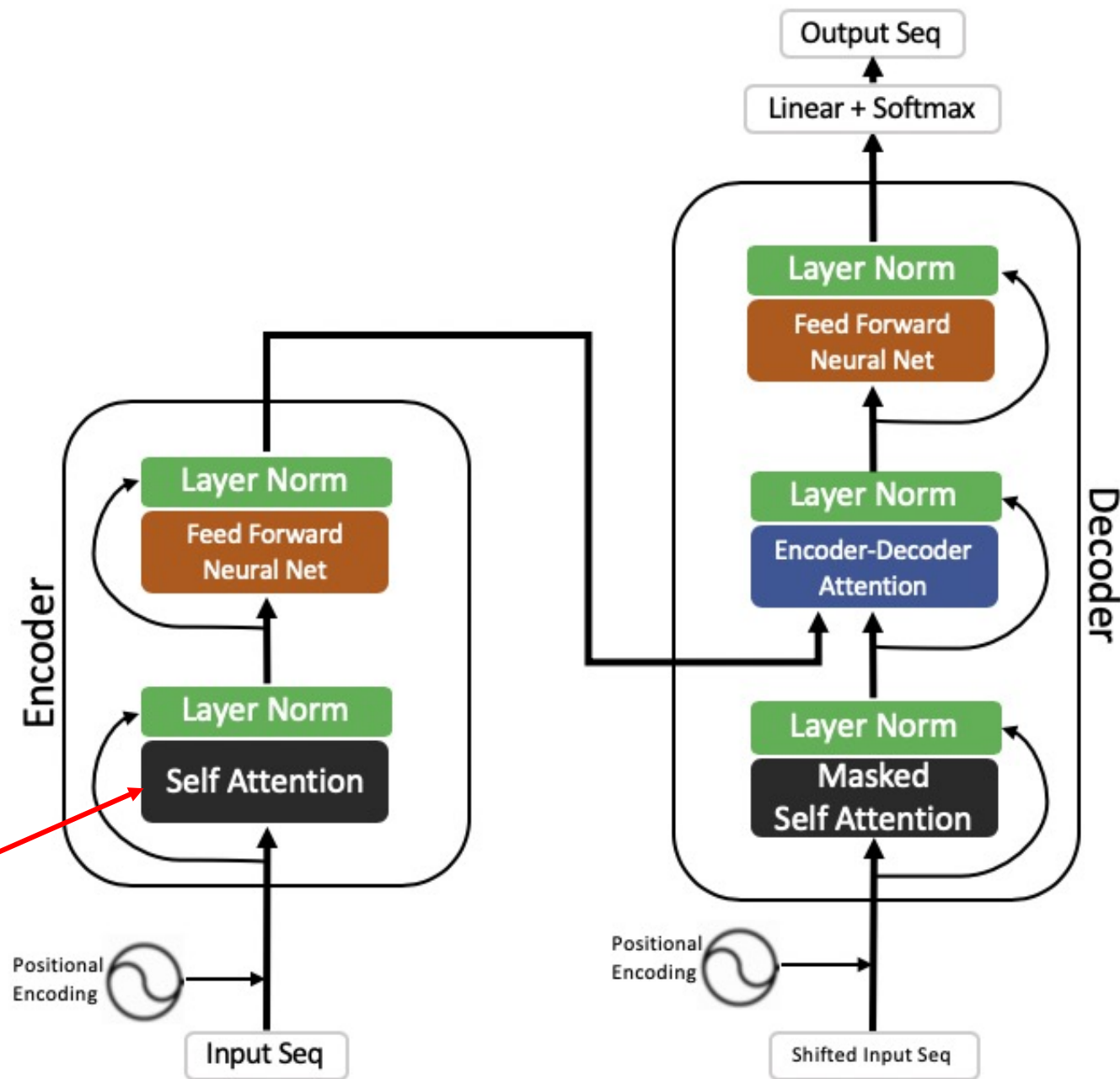


# ¿Cómo funciona todo?



# Modelo Propuesto (Transformer)

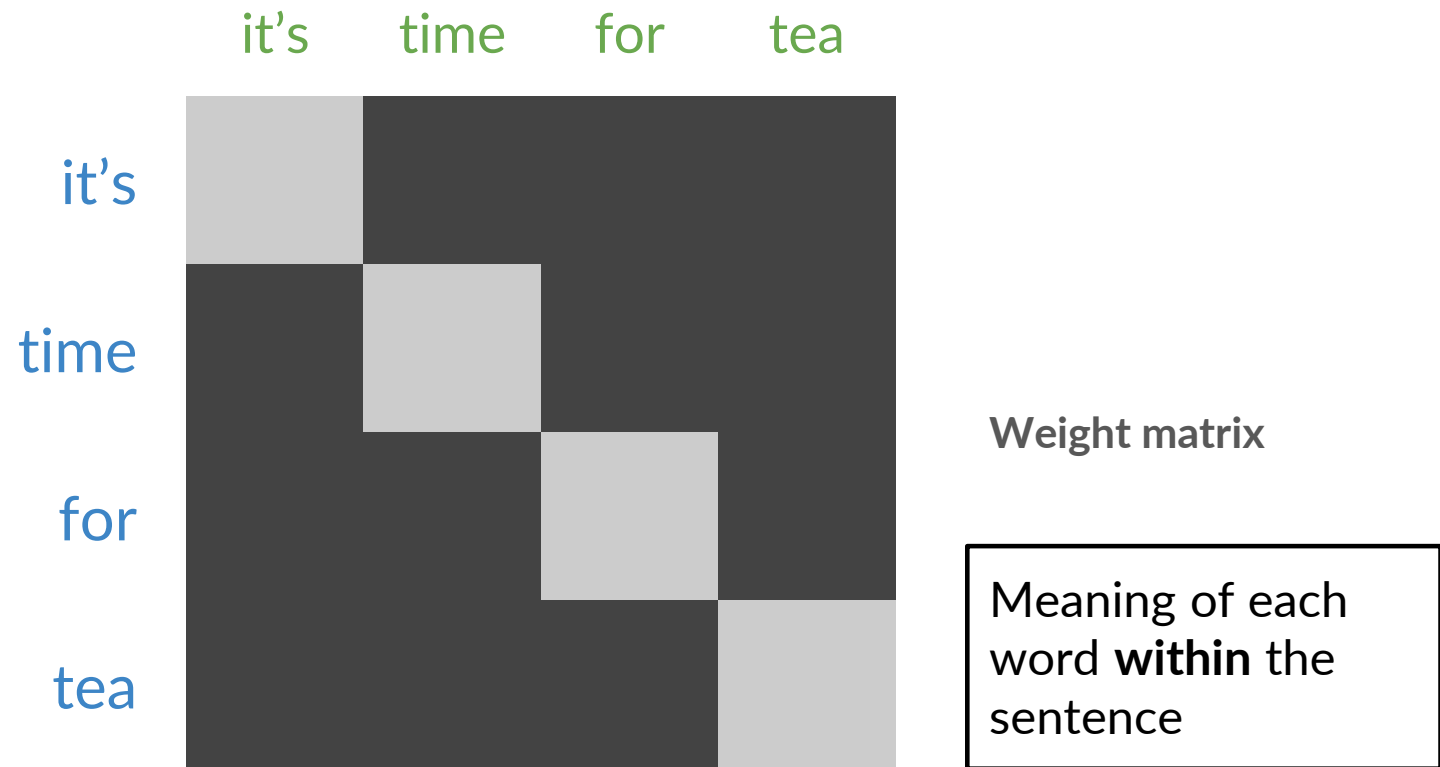
La unidad básica





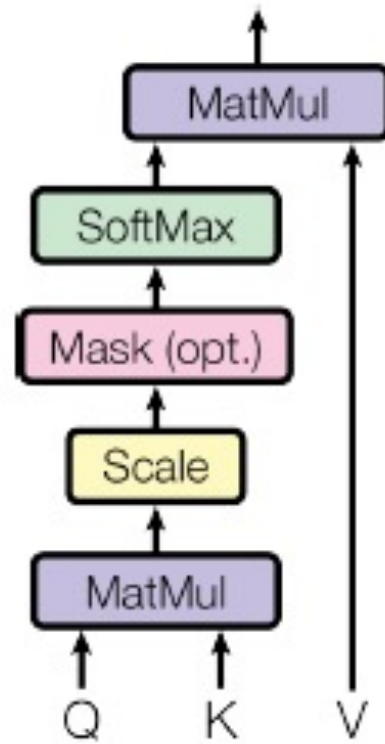
# Autoatención

Queries, keys and values come from the **same sentence**

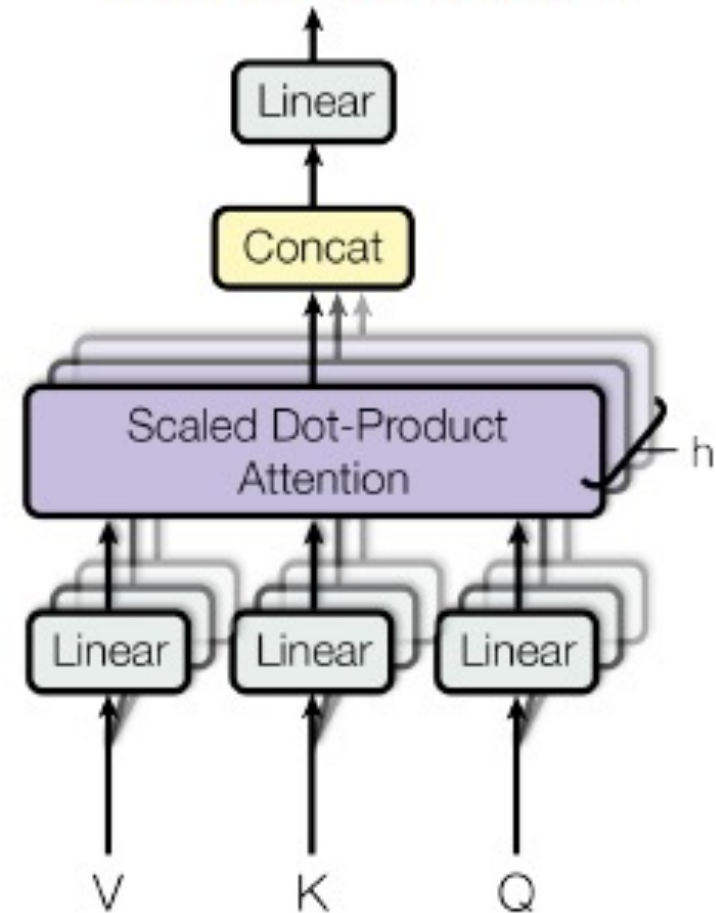


# Attention vs MultiHead Attention

Scaled Dot-Product Attention



Multi-Head Attention

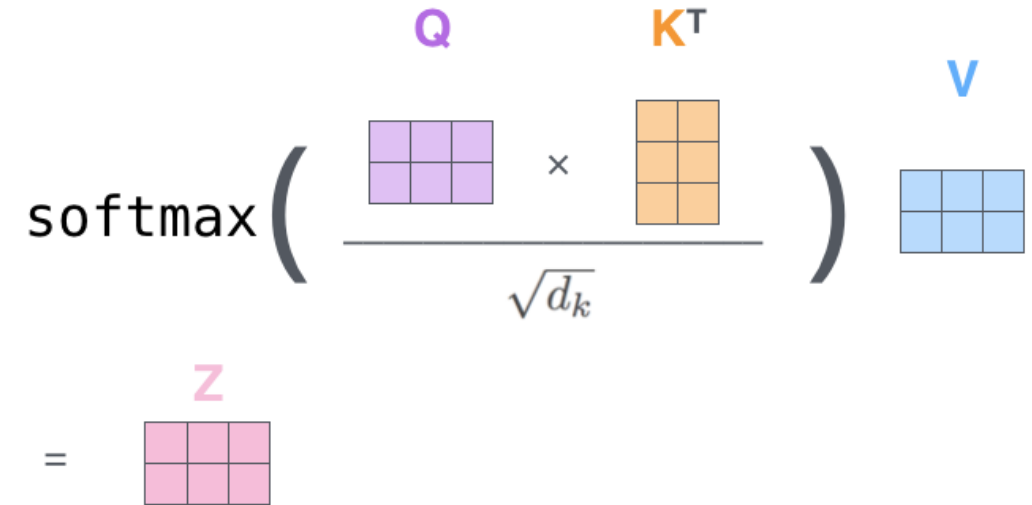


# Self Attention

$$\mathbf{X} \times \mathbf{W}^Q = \mathbf{Q}$$


$$\mathbf{X} \times \mathbf{W}^K = \mathbf{K}$$


$$\mathbf{X} \times \mathbf{W}^V = \mathbf{V}$$

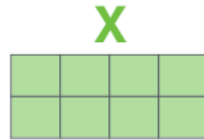

$$\text{softmax}\left(\frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V}$$

$$= \mathbf{Z}$$

# Multi-head Self Attention

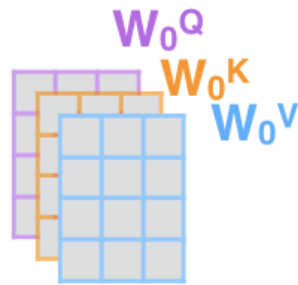
1) This is our input sentence\*

Thinking  
Machines

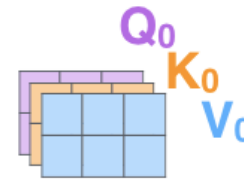
2) We embed each word\*



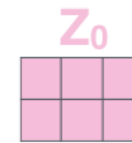
3) Split into 8 heads.  
We multiply  $X$  or  $R$  with weight matrices



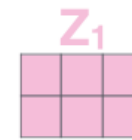
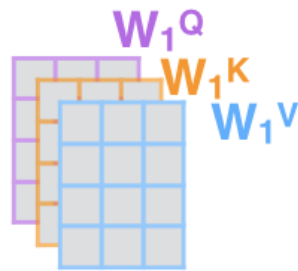
4) Calculate attention using the resulting  $Q/K/V$  matrices



5) Concatenate the resulting  $Z$  matrices, then multiply with weight matrix  $W^O$  to produce the output of the layer



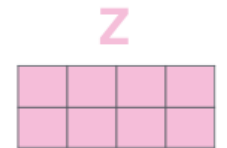
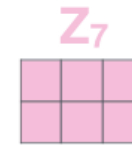
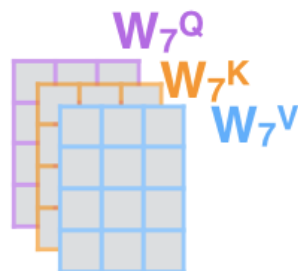
\* In all encoders other than #0, we don't need embedding.  
We start directly with the output of the encoder right below this one



...

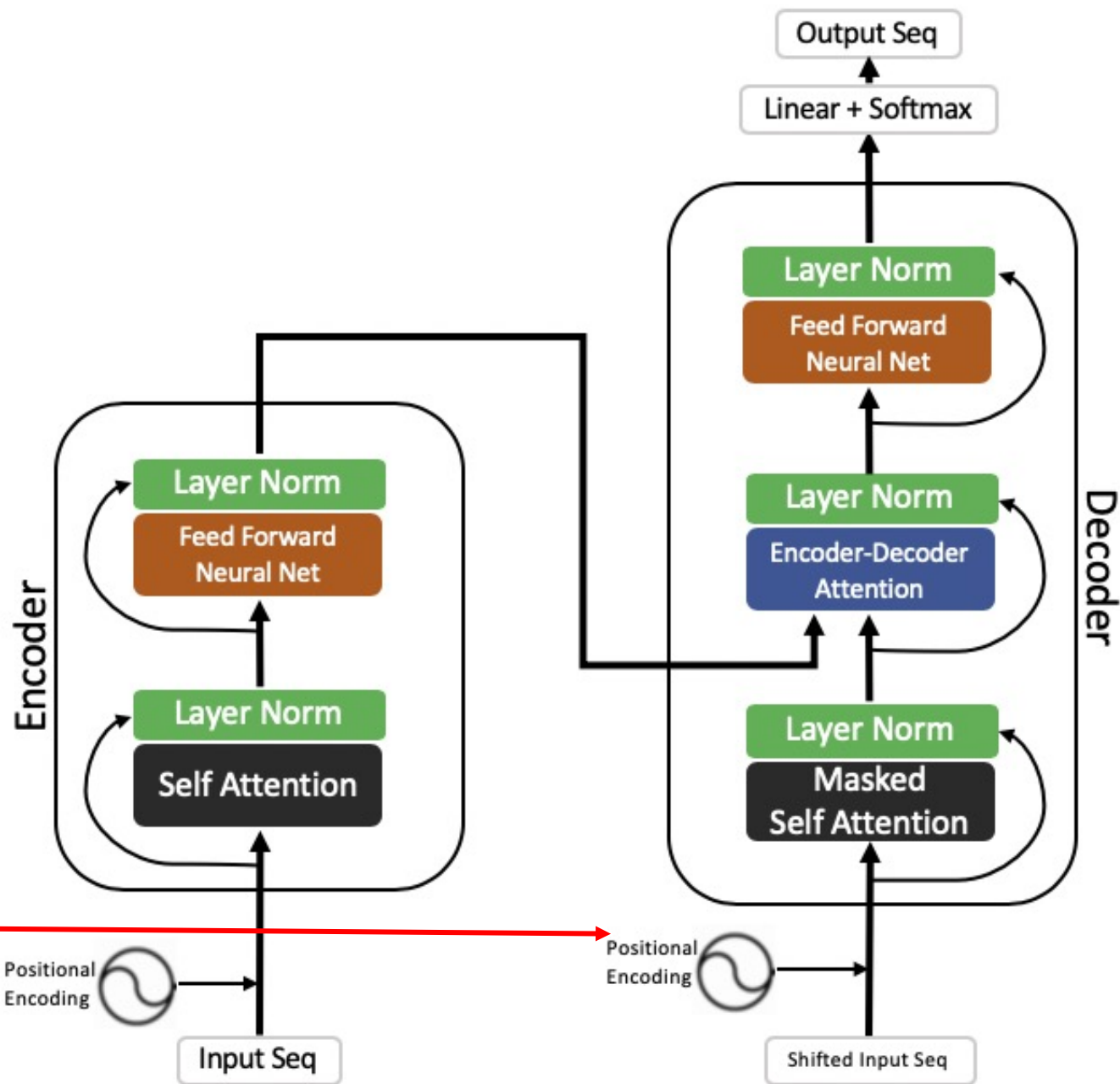
...

...

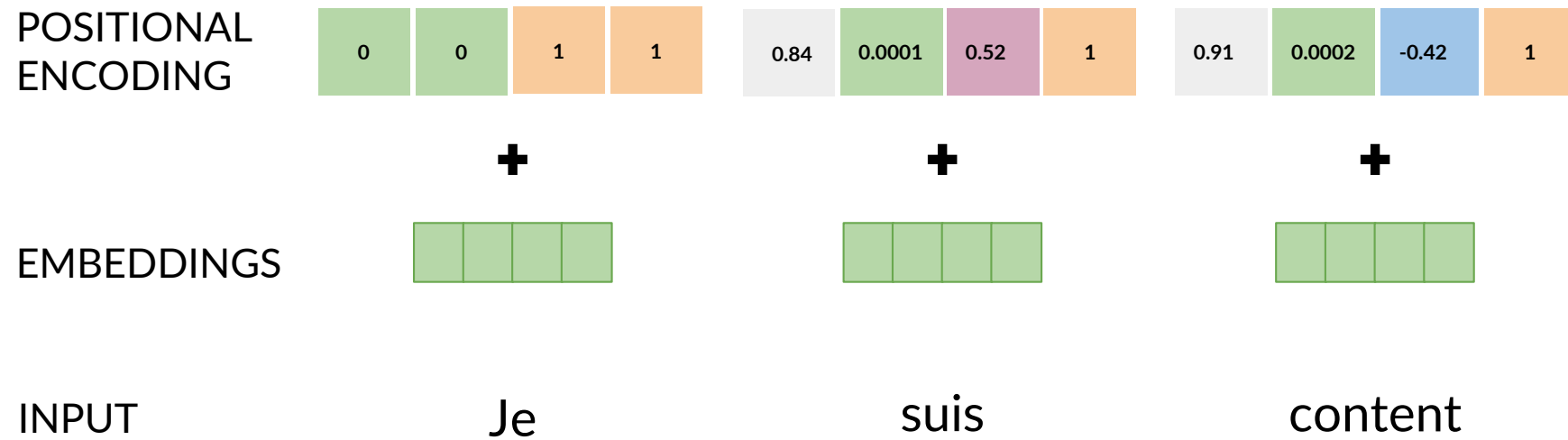


# Modelo Propuesto (Transformer)

Mantener el orden  
de entrada



# Positional encoding

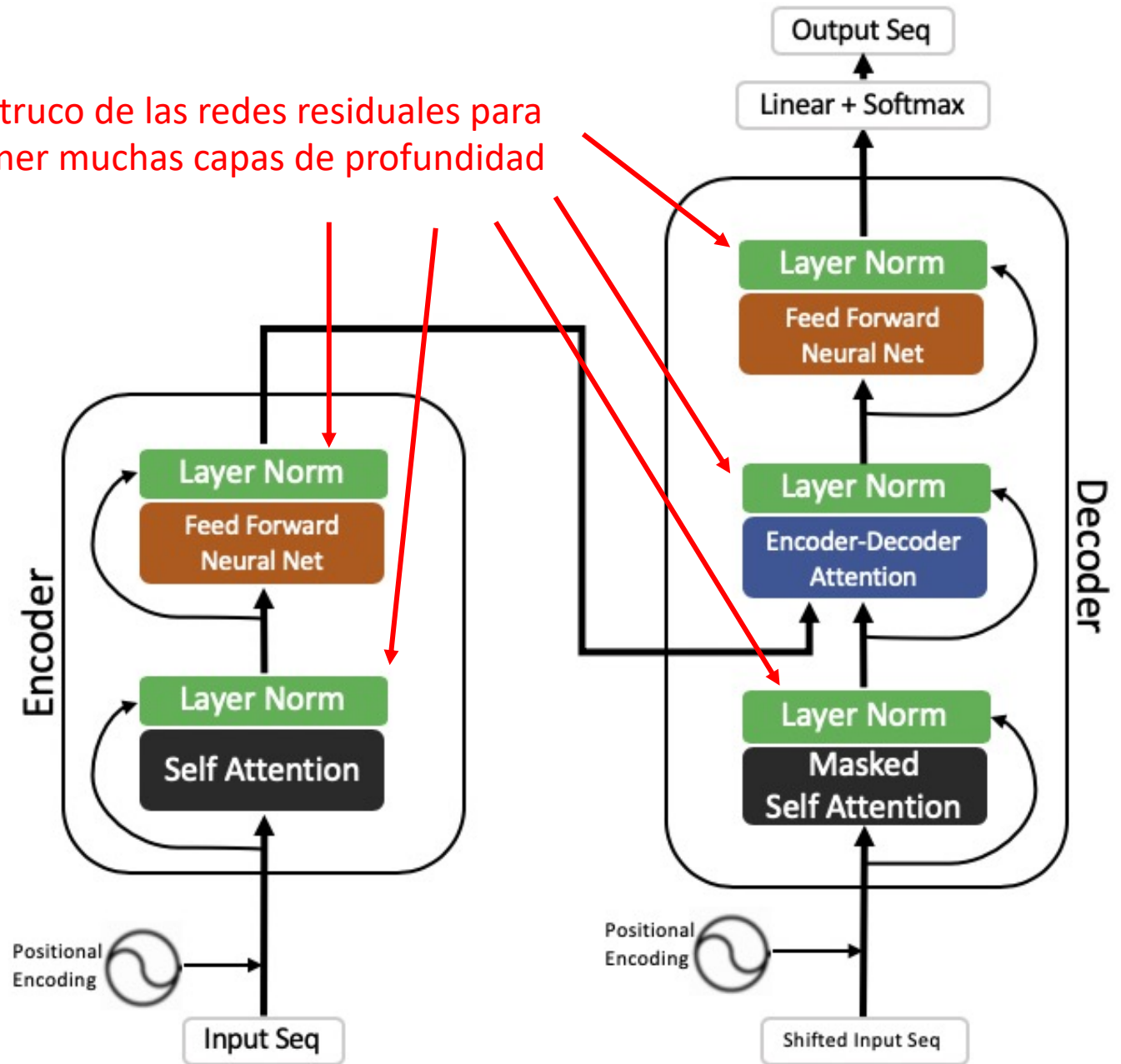


RNNs vs Transformer: Positional Encoding

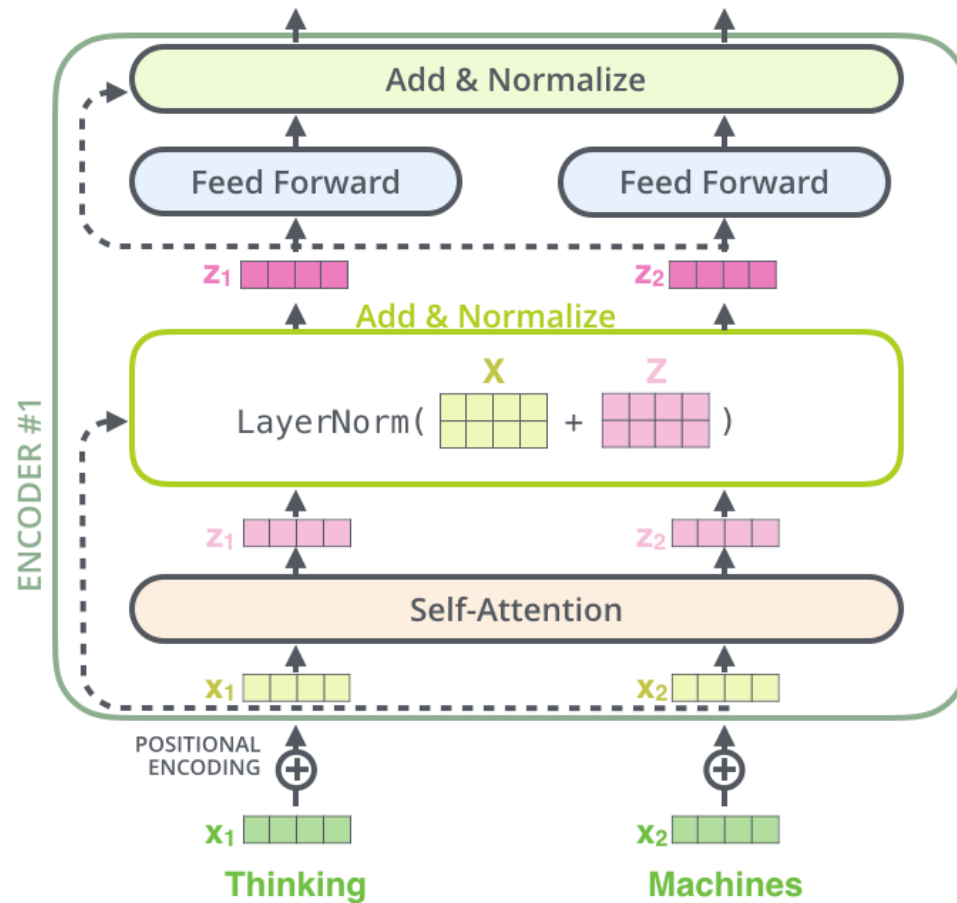
# Modelo Propuesto (Transformer)

Deep Residual Learning for  
Image Recognition

El truco de las redes residuales para  
tener muchas capas de profundidad



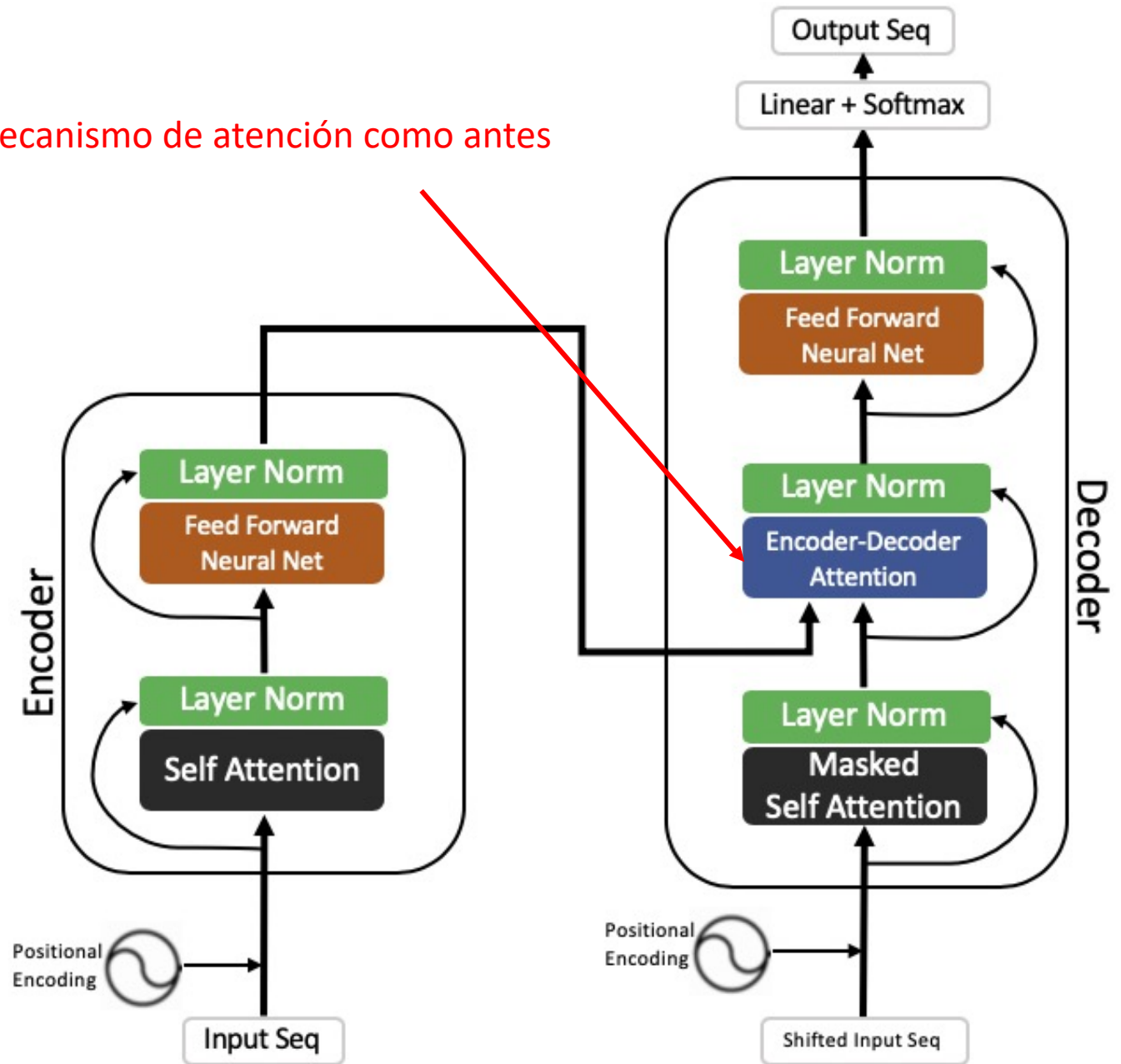
# Add & Normalize



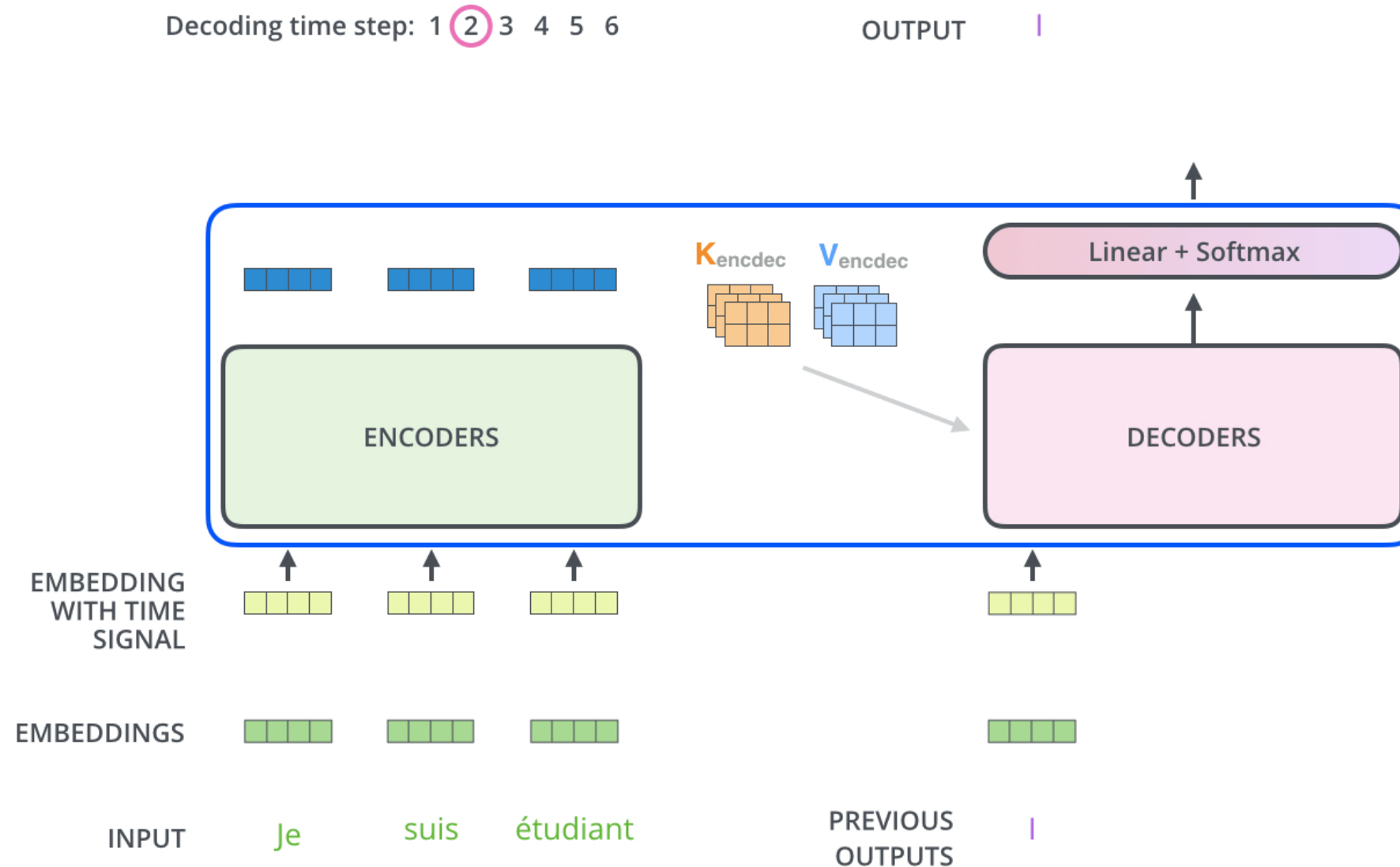


# Modelo Propuesto (Transformer)

Mecanismo de atención como antes

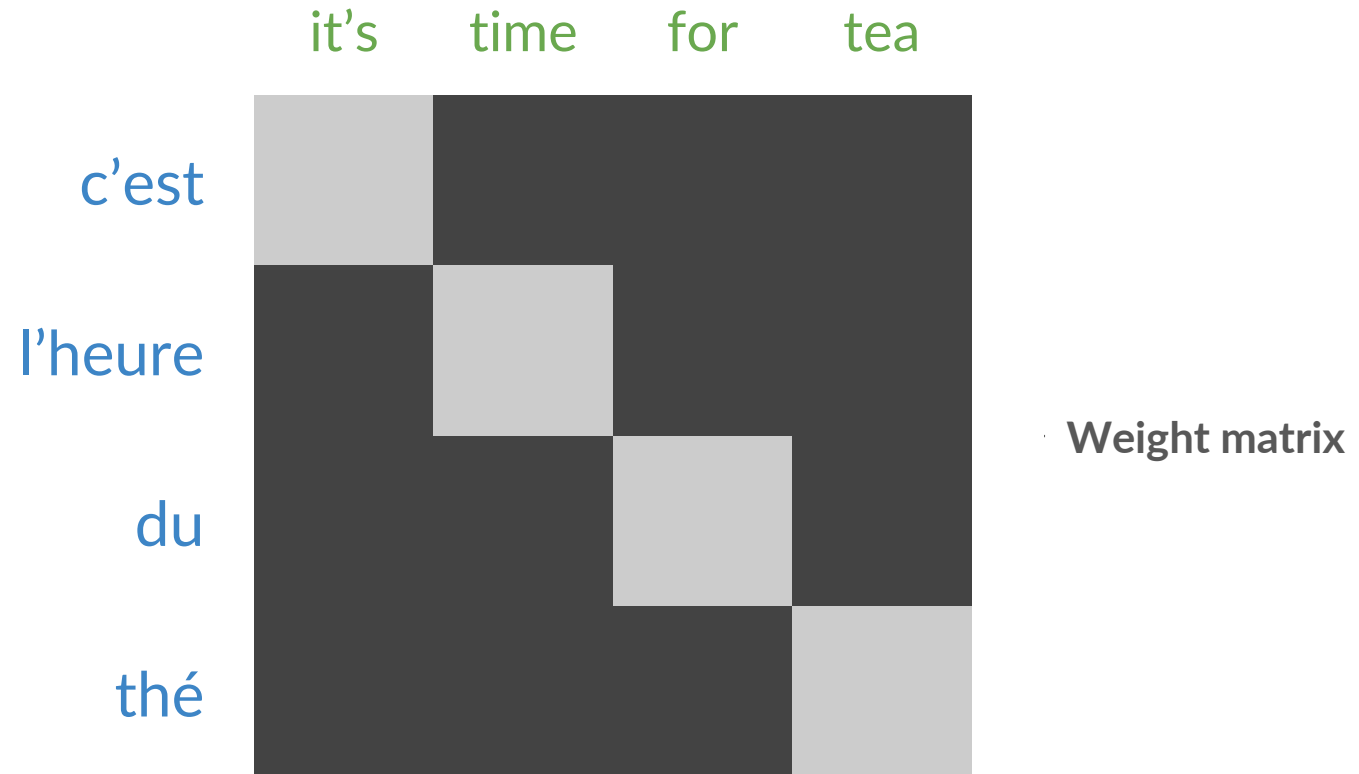


# Encode-Decoder Self-Attention



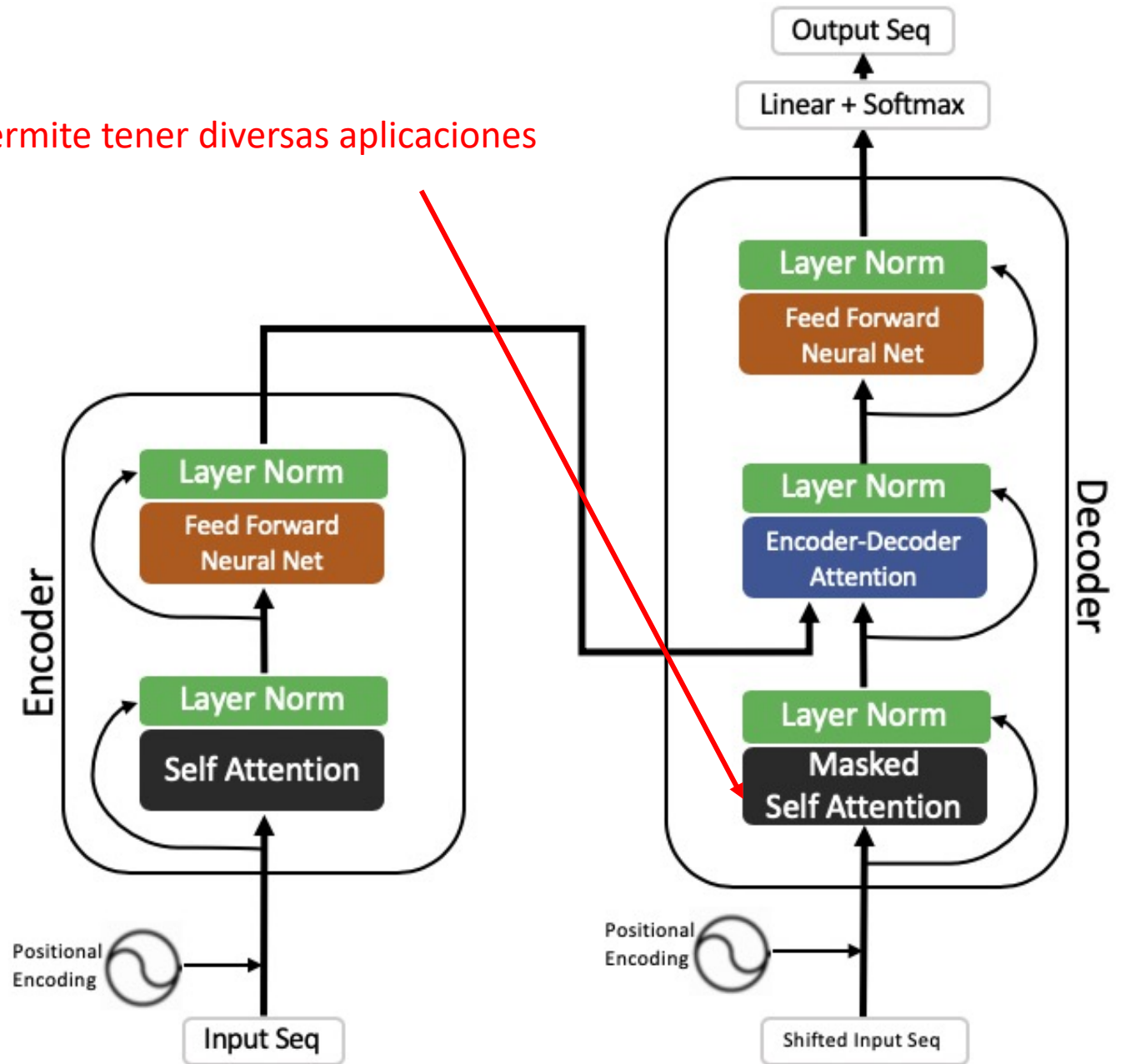
# Encoder-Decoder Attention

Queries from one sentence, **keys** and **values** from another



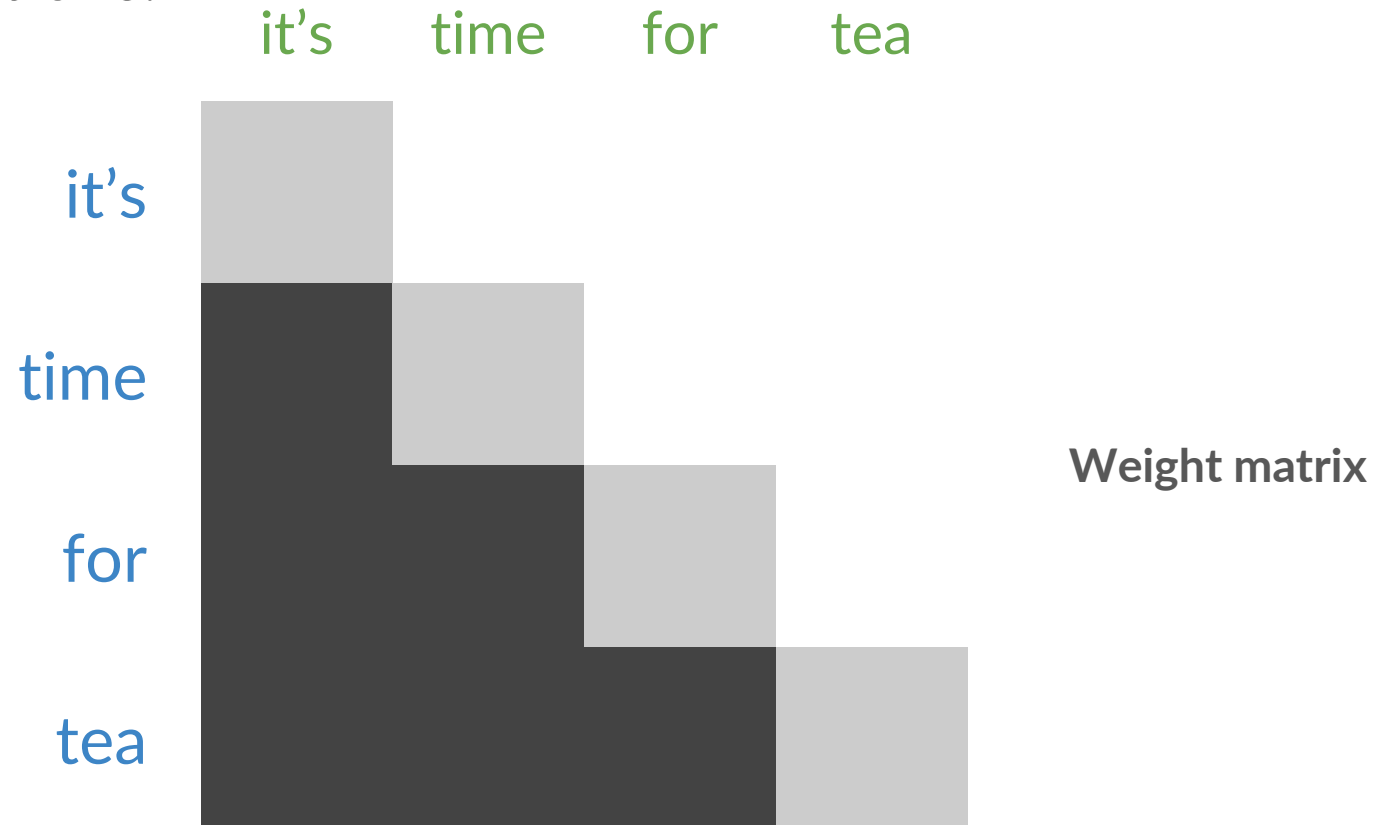
# Modelo Propuesto (Transformer)

Permite tener diversas aplicaciones




# Masked Self-Attention

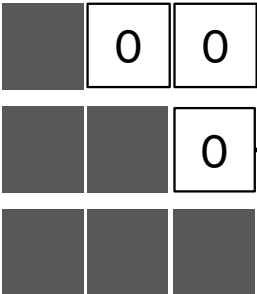
Queries, keys and values come from the **same sentence**. Queries don't attend to future positions.



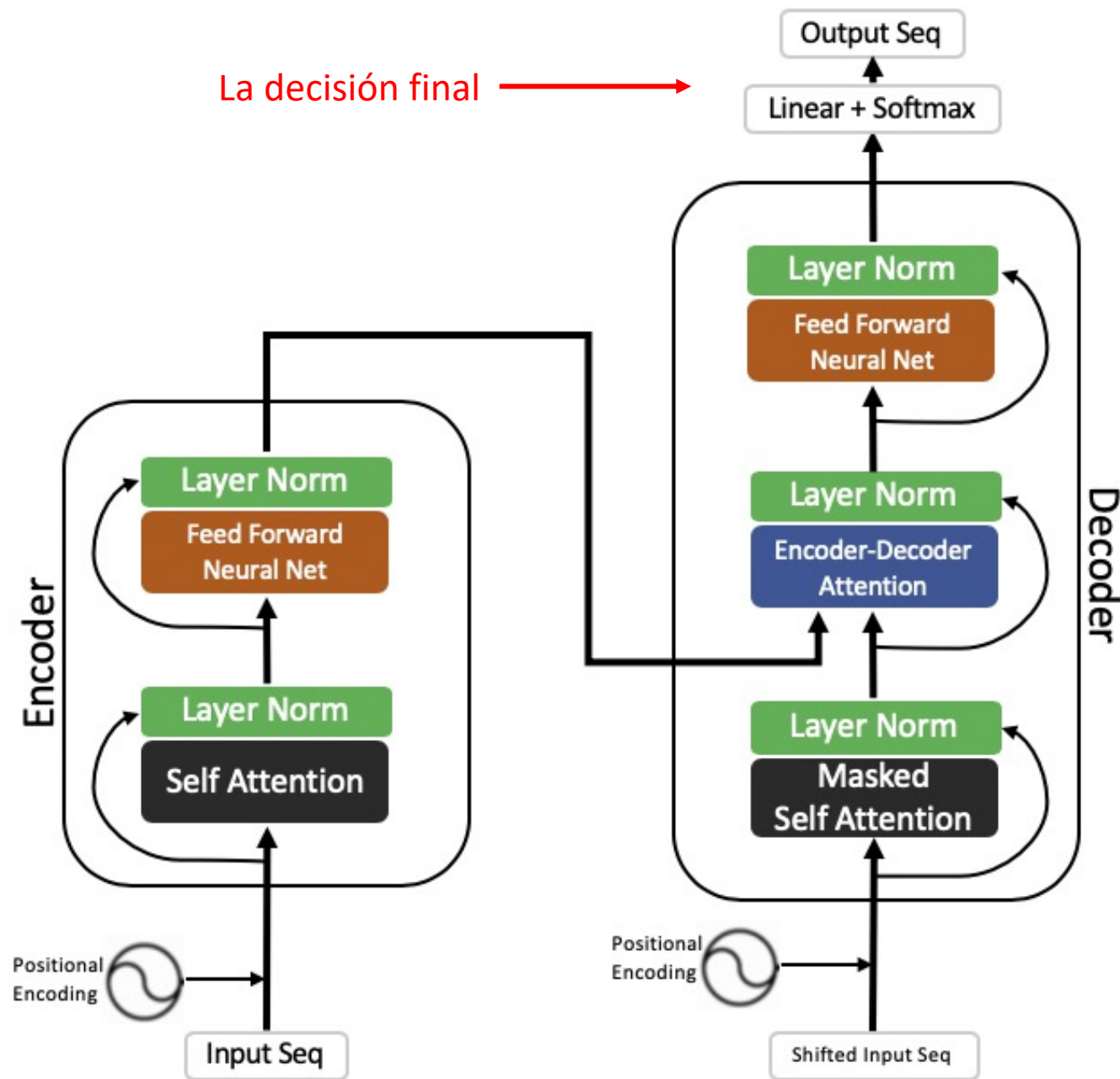
# ¿Y cómo se hace el Masked Self-Attention?

 → Minus infinity

$$\text{softmax} \left( \frac{\begin{matrix} Q & K^T \\ \begin{matrix} \text{blue grid} & \text{green grid} \end{matrix} \end{matrix}}{\sqrt{d_k}} + \begin{matrix} \begin{matrix} 0 & \text{pink} & \text{pink} \\ 0 & 0 & \text{pink} \\ 0 & 0 & 0 \end{matrix} \end{matrix} \right) \begin{matrix} V \\ \text{orange grid} \end{matrix}$$

 → Weights assigned to future positions are equal to 0

# Modelo Propuesto (Transformer)



# Capa final de los decoders

Which word in our vocabulary  
is associated with this index?

Get the index of the cell  
with the highest value  
(argmax)

am

5

log\_probs



Softmax

logits



Linear

Decoder stack output

