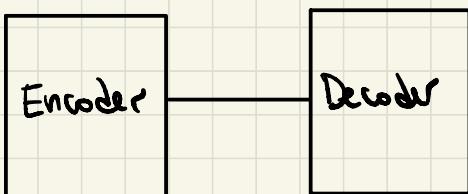


Transformers (Clase 10)

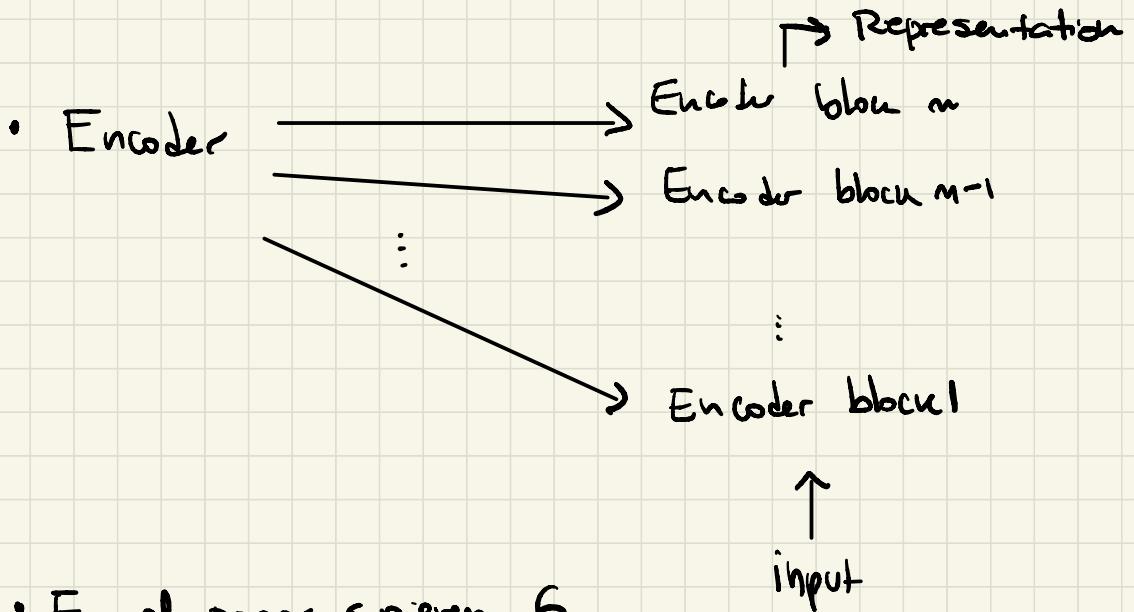
Tutorial Valerio Velardo.

1. Contexto
2. Intuición + arquitectura
3. Encoder
4. Self - attention
5. multi - head attention
6. Positional Encoding
7. Feed forward
8. Add / Norm component
9. Encoder Step by Step.

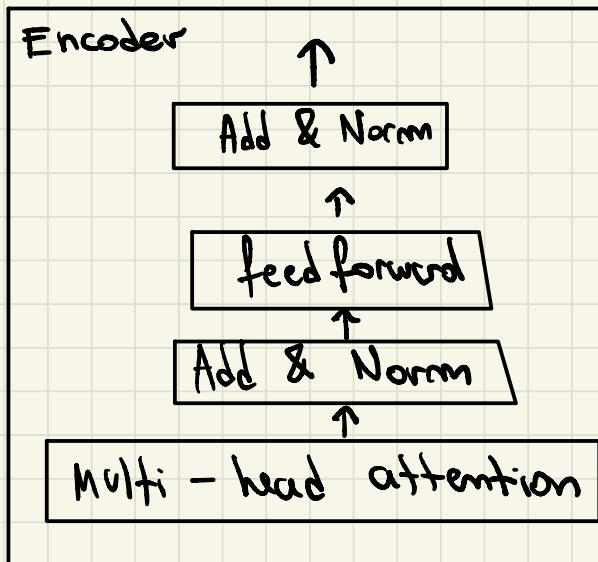
- RNN - LSTM : No capturan relaciones a largo plazo en secuencias.
- Generative Pretrained Transformers (GPT)



↑
input
"¡Ricke ten rodolf"
↓
output
"Yo soy Rodolf"



- En el paper supieren 6 encoders bloks.
- Cada encoder apredra' diferents aspectos de la informacion.



Self - Attention

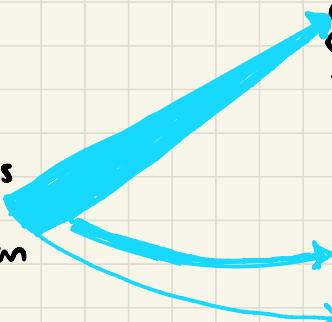
Motivación:

Mis gatos son muy felices, ellos juegan con mis cordones.

- Una palabra apunta/se relaciona con otra
- ¿Cómo lograr que un modelo entienda ese tipo de relaciones / asociaciones entre palabras?

Mis
gatos
son
muy
felices
ellos
juegan
con
mis
cordones

Mis
gatos
son
muy
felices
ellos
juegan
con
mis
cordones



Matrices involucradas en el problema :

- Matriz de embeddings (I)

Con dimensión número de palabras x dimensión del embedding

~ Cada fila es un vector que representa una palabra.

Ejemplo :

feature1 feature2

Tu	→	0,1	2,5
estas		0,7	0,7
new!		0,2	0,1

- Query , Key , Value

Q

Tu	→	0,1	2,5
estas		0,7	0,7
new!		0,2	0,1

K

Tu	→	0,3	1,1
estas		2,7	3
new!		4	2

V

Tu	→	1	1
estas		0,3	0,7
new!		0,2	2

$$I \cdot W_Q = Q$$

↓
Pesos que aprendemos al entrenar

$$I \cdot W_K = K$$

$$I \cdot W_V = V$$

- Las 7 matrices que hay hasta el momento, permiten relacionar una palabra a todas las otras.

$$Z(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_n}} \right) V \quad (1)$$

Paso 1 : QK^T (self - attention)

$$QK^T = \begin{matrix} Tu & \text{estas} & \text{aqui} \\ \text{estas} & \begin{bmatrix} 1.3 & 0.8 \\ 0.7 & 3.5 \end{bmatrix} & q_1 \\ \text{aqui} & \begin{bmatrix} 1.9 & 0.1 \end{bmatrix} & q_2 \\ & & q_3 \end{matrix} \quad \begin{matrix} Tu & \text{estas} & \text{aqui} \\ \text{estas} & \begin{bmatrix} 0.6 & 0.8 & 2.5 \\ 2.4 & 1.7 & 0.3 \end{bmatrix} & k_1 \\ \text{aqui} & k_2 & k_3 \end{matrix}$$

$$QK^T = \begin{bmatrix} \langle q_1, k_1 \rangle & \langle q_1, k_2 \rangle & \langle q_1, k_3 \rangle \\ \langle q_2, k_1 \rangle & \langle q_2, k_2 \rangle & \langle q_2, k_3 \rangle \\ \langle q_3, k_1 \rangle & \langle q_3, k_2 \rangle & \langle q_3, k_3 \rangle \end{bmatrix} \quad \begin{cases} \text{Productos} \\ \text{Internos!} \end{cases}$$

$$QK^T = \begin{bmatrix} \text{tu} & \text{estuv} & \text{agr.} \\ \text{tú} & 2.4 & 3.49 \\ \text{estuv} & 8.82 & 2.6 \\ \text{agr.} & 1.39 & 1.69 & 4.78 \end{bmatrix}$$

} Matriz de
Similaridades

→ "similaridad"
entre dos palabras.

Intuición de estas matrices

- Q es el objeto en la secuencia que estamos procesando
 - ¿Qué otras palabras son relevantes en relación a este objeto? → ¿Qué otras palabras son relevantes?
- K son todos los objetos en la secuencia a los que la query puede poner atención
 - puede relacionar consigo misma.
- d_K : dimensión del embedding en K .
(Número de columnas en K). En el ejemplo \mathbb{R}^2 .

- Permite escalar los valores para obtener gradientes estables.

$$QK^T = \frac{1}{\sqrt{2}} \times \begin{bmatrix} 2.7 & 2.4 & 3.49 \\ 8.82 & 6.51 & 2.8 \\ 1.38 & 1.69 & 4.78 \end{bmatrix} = \begin{bmatrix} \frac{2.7}{\sqrt{2}} & \frac{2.4}{\sqrt{2}} & \frac{3.49}{\sqrt{2}} \\ \frac{8.82}{\sqrt{2}} & \frac{6.51}{\sqrt{2}} & \frac{2.8}{\sqrt{2}} \\ \frac{1.38}{\sqrt{2}} & \frac{1.69}{\sqrt{2}} & \frac{4.78}{\sqrt{2}} \end{bmatrix}$$

- Luego, aplicar softmax permite obtener una probabilidad para cada palabra.

y_0 estoy así;

$$\text{softmax}\left(\frac{\theta_k h^T}{\|h\|_2}\right) =$$

$$\begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \begin{array}{l} y_0 \\ \text{(estoy)} \\ \text{así} \end{array}$$

- * La suma de las filas da 1.
- * La matriz entrega información sobre la relevancia de diferentes partes de la secuencia consigo misma.

- Softmax $\left(\frac{QK^T}{\sqrt{d_h}} \right) V = Z$

$$\begin{bmatrix} 0,7 & 0,2 & 0,1 \\ 0,2 & 0,6 & 0,2 \\ 0,4 & 0,1 & 0,5 \end{bmatrix}$$

Relevancia

$$\begin{bmatrix} 0,9 & 1 \\ 1,2 & 2,6 \\ 1,7 & 0,2 \end{bmatrix} V_1 \\ V_2 \\ V_3$$

contenido de la sec.

$$\begin{bmatrix} 0,69 & 1,28 \\ 1,14 & 1,92 \\ 1,13 & 0,78 \end{bmatrix} \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \text{Matriz de Atención}$$

Relevancia + Contenido

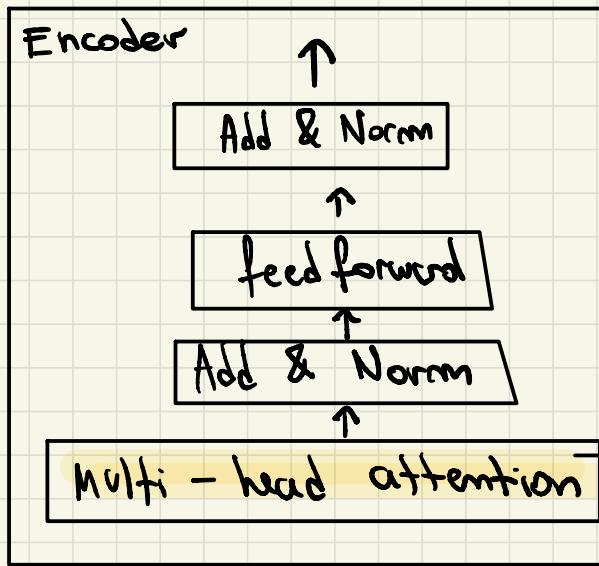
$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \rightarrow \begin{array}{l} \text{atención palabra 1} \\ \text{" " " 2} \\ \text{" " " 3} \end{array}$$

$$\vec{z}_1 = 0,7 V_1 + 0,2 V_2 + 0,1 V_3$$

contenidos

pesos o relevancia

- Este proceso de cálculo de atención se realiza m -veces de forma paralela (m cabezales de atención)
- Cálcula Q, K, V, z m -veces.



$$\begin{aligned}
 z_1 &= \text{soft} \cdot (Q_1 K_1^T) V_1 \\
 z_2 &= \text{soft} \cdot (Q_2 K_2^T) V_2 \\
 &\vdots \\
 z_m &= \text{soft} \cdot (Q_m K_m^T) V_m
 \end{aligned}$$

$\underbrace{\hspace{10em}}$

Concatenamos

$$z = \text{concat}(z_1, z_2, \dots, z_m) W_0$$

- Los diferentes cabezales pueden colocar atención en diferentes aspectos de las secuencias.

- La gran diferencia con los modelos recurrentes LSTM, es que en el caso LSTM el modelo se "alimenta" de las palabras de forma secuencial. En el caso de transformers se alimenta la secuencia de forma paralela.

- ~ Más rápido el tiempo de entrenamiento
- ~ Mantiene mejor las relaciones a largo plazo.

- ¿Cómo el modelo codifica la posición de las palabras? → Position Embedding.

$$\tilde{I} = \underbrace{I}_{\text{Contenido Semántico}} + \underbrace{P}_{\substack{\text{Embedding} \\ \text{de Posición}}} \quad \left. \right\} \begin{array}{l} \text{Antes de} \\ \text{entrar en} \\ \text{el encoder!} \end{array}$$

- El modelo es capaz de entender la mezcla de I y P .

Calculo del Embedding de Posición

$$P(pos, 2i) = \sin \left(\frac{pos}{10000^{2i/d_{\text{model}}}} \right) \quad (2)$$

↓
Posición de las palabras } índice en las filas → Posición del embedding

$$P(pos, 2i+1) = \cos \left(\frac{pos}{10000^{2i/d_{\text{model}}}} \right) \quad (3)$$

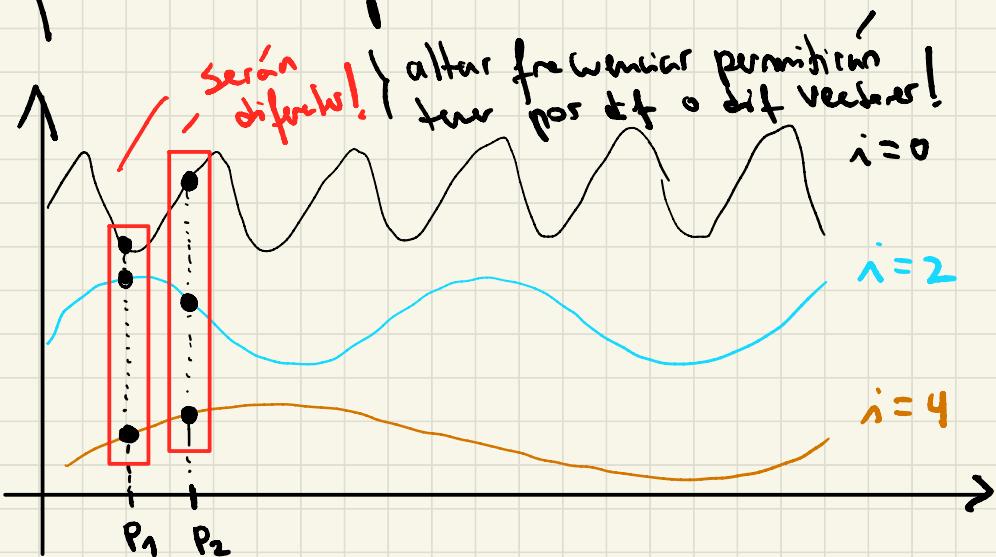
(2) Para columnas Pares.

(3) " " " Impares.

Ejemplo :

Los	$\sin\left(\frac{0}{10000^{2 \cdot 0/2}}\right)$	$\cos\left(\frac{0}{10000^{2 \cdot 1/2}}\right)$	$\sin\left(\frac{0}{10000^{2 \cdot 2/2}}\right)$
gatos	$\sin\left(\frac{L}{10000^{2 \cdot 0/2}}\right)$	$\cos\left(\frac{L}{10000^{2 \cdot 1/2}}\right)$	$\sin\left(\frac{1}{10000^{2 \cdot 2/2}}\right)$
son	$\sin\left(\frac{2}{10000^{2 \cdot 0/2}}\right)$	$\cos\left(\frac{2}{10000^{2 \cdot 1/2}}\right)$	$\sin\left(\frac{2}{10000^{2 \cdot 2/2}}\right)$
lindos	$\sin\left(\frac{3}{10000^{2 \cdot 0/2}}\right)$	$\cos\left(\frac{3}{10000^{2 \cdot 1/2}}\right)$	$\sin\left(\frac{3}{10000^{2 \cdot 2/2}}\right)$

- Estamos eligiendo diferentes frecuencias para cada posición.

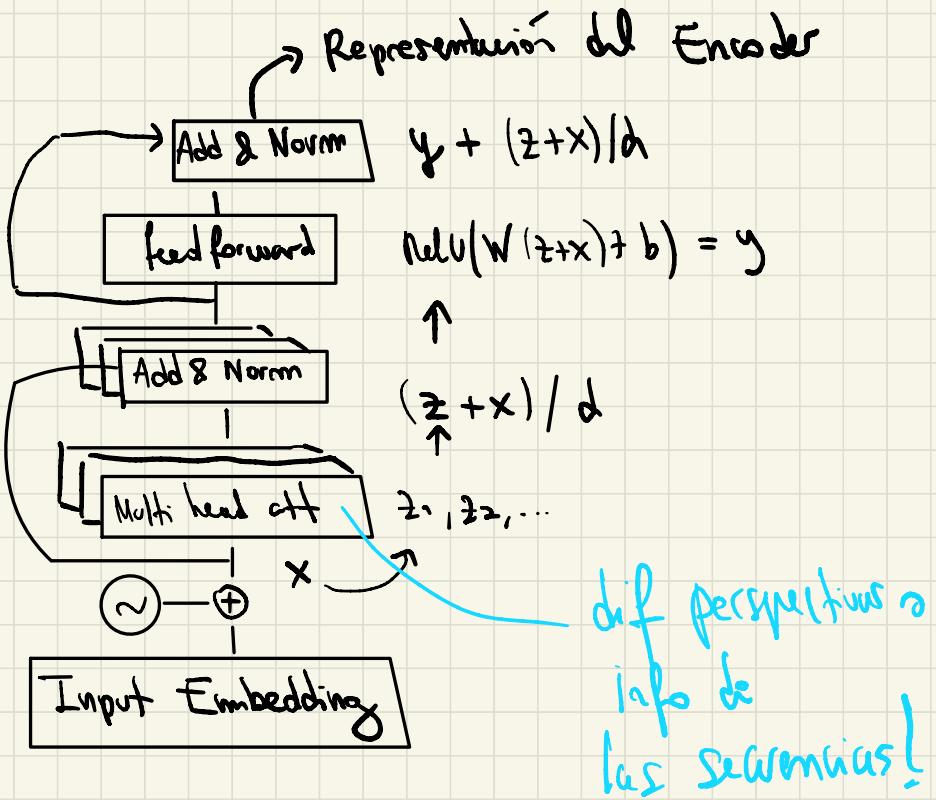
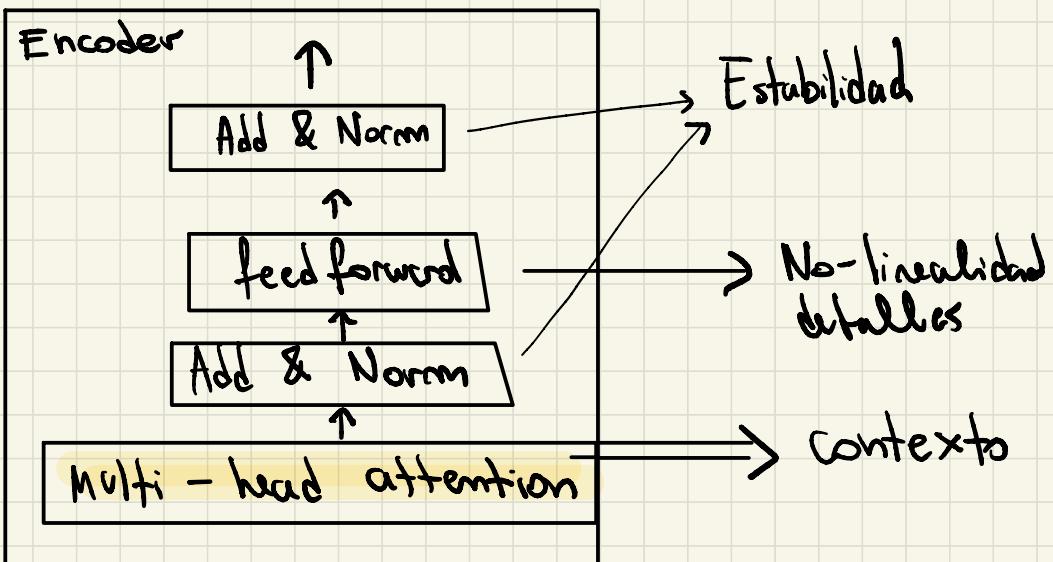


Feedforward Part

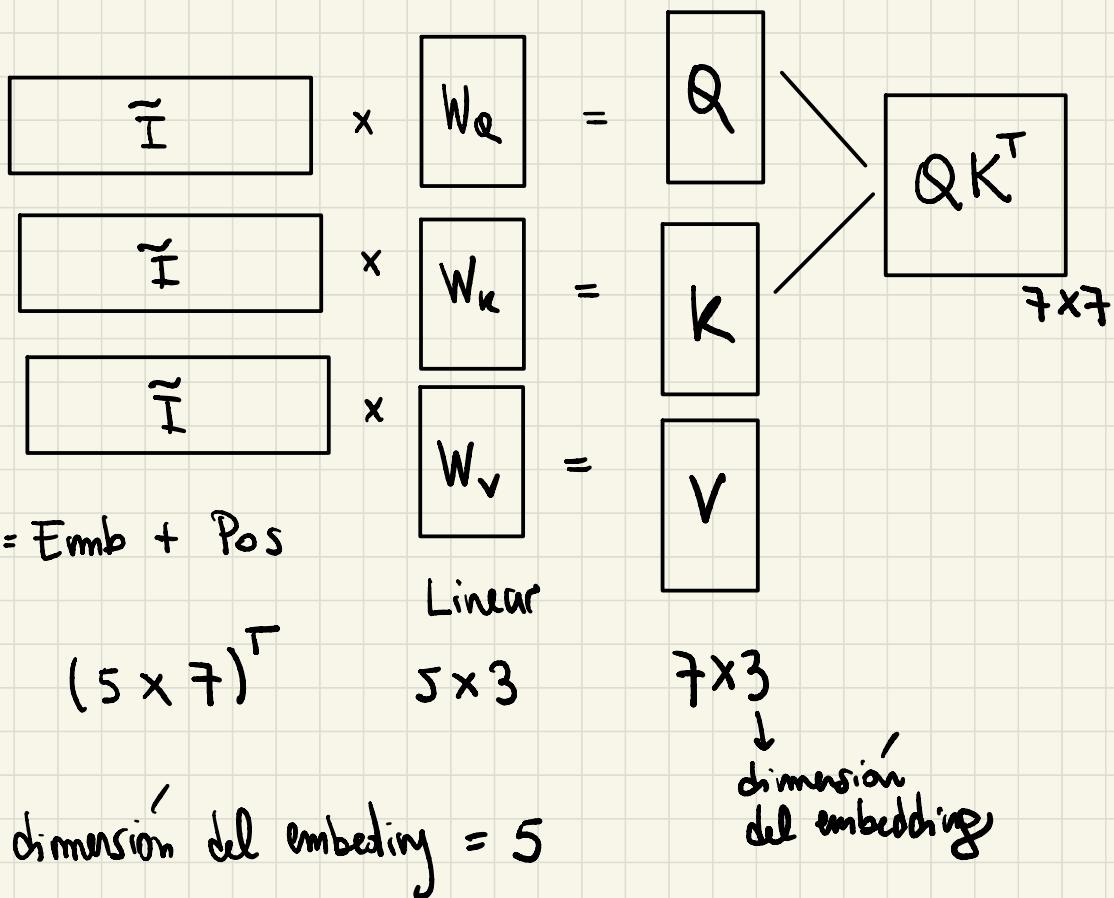
- 2 fully connected
- ~ Procesa cada posición por separado
- ReLU (Activación)

Add & Norms

- ~ Conecta la entrada de una capa de atención con su salida (Residuos feedback)
- ~ Conecta la entrada del módulo feed forward a la salida.
- ~ Esto ayuda a evitar los gradientes que desvanece a la hora de entregar.
- ~ La normalización : $\text{mean} = 0$
 $\text{std} = 1$
 - Acelera la convergencia
 - Estabilidad



Ejemplo : Solo para clarificar las dimensiones

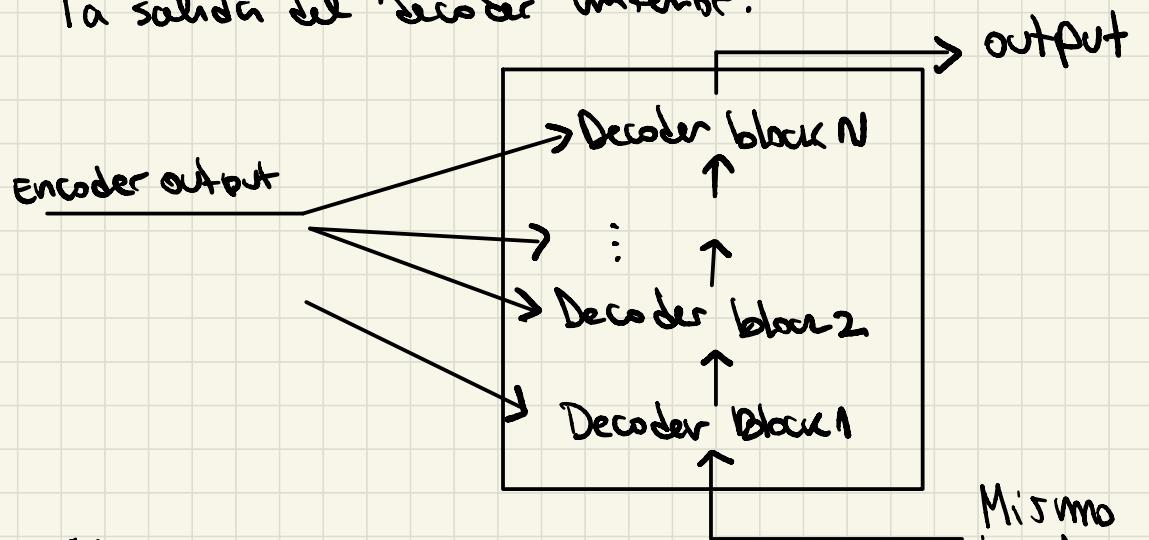


- Existen diferentes tipos de atención :

- ~ Encoder | Decoder
- ~ Casual
- ~ Bi directional

Decoder en los transformers

- Cada decoder recibirá el vector representado por el encoder + la salida del decoder anterior.



- Es un proceso autoregresivo.
- La entrada al primer decoder permite a este bloque tener información del contexto y la posición.
 $\tilde{I} \approx \text{Emb} + \text{Pos}$
- El objeto nuevo dentro del decoder es llamado "Mask Multi head Attention".

- Un problema al entrenar transformers es en relación a información "futura" de las secuencias.

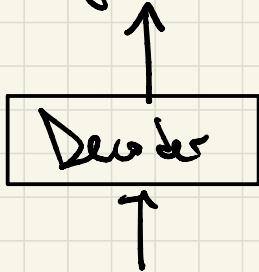
- ~ El mecanismo de self attention relaciona una palabra a las otras
- ~ El decodificador genera la salida de una palabra a la vez
- ~ Sin embargo, el decodificador conoce info del futuro (Hace trampa) pues conoce una frase completa
(Esto en entrenamiento o predicción)
- ~ Lo último, afecta la capacidad del modelo de predecir información y generalizar.

- SOS : Start of a Sequence
"Inicio de una frase".

- Se utiliza como referencia para comenzar a generar tokens de salida.

Training / Inference Discrepancy

Me gustan los



SOS Me gustan } info en inferencia

SOS Me gustan los gato } info en training

Es decir la
solución al momento
de entrenar.

$$z_i(\theta_{ri}, K_i, V_i) = \text{Softmax} \left(\frac{\Theta_{ri} K_i^T}{\sqrt{d_k}} \right) V_i$$

	SOS	me	gustan	los	gatos
sos	1.3	0.8	1.3	2.8	2.3
me	2.4	2.8	2.3	6.8	1.9
gustan	1.6	7.4	1.6	0.3	0.5
los	2.1	1.2	9.3	5.2	0.2
gatos	4.3	3.8	6.3	1.8	2.3

* Queremos eliminar las relaciones futuras en la medida que las palabras aparecen.

$$z_i(\theta_{ri}, K_i, V_i) = \text{Softmax} \left(\frac{\Theta_{ri} K_i^T}{\sqrt{d_k}} + M \right) V_i$$

↓
Mask
Matrix

$$Z = \text{concat}(z_0, z_1, \dots) W_0$$

Ahora tenemos 2 entradas :

target sentence

i) Q : Se obtiene a través de la entrada enmascarada

$$RW_Q = Q$$

ii) K, V : Se obtienen de la representación del vector

source sentence

$$RW_K = K$$

$$RW_V = V$$

$$Z = \begin{bmatrix} | & \text{like} & \text{cats} \\ \text{sos} & \begin{bmatrix} 0.7 & 0.2 & 0.1 \end{bmatrix} \\ \text{me} & \begin{bmatrix} 0.6 & 0.3 & 0.1 \end{bmatrix} \\ \text{gustan} & \begin{bmatrix} 0.1 & 0.8 & 0.1 \end{bmatrix} \\ \text{los} & \begin{bmatrix} 0.1 & 0.3 & 0.6 \end{bmatrix} \\ \text{gatos} & \begin{bmatrix} 0.1 & 0.1 & 0.8 \end{bmatrix} \end{bmatrix}$$

$$\text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right)$$

$$V = \begin{bmatrix} | & V_1 & V_2 & V_3 & V_4 & V_5 \\ \text{like} & \begin{bmatrix} 0.4 & 1.0 \end{bmatrix} \\ \text{cats} & \begin{bmatrix} 1.2 & 2.8 \end{bmatrix} \\ & \begin{bmatrix} 1.7 & 0.2 \end{bmatrix} \end{bmatrix}$$

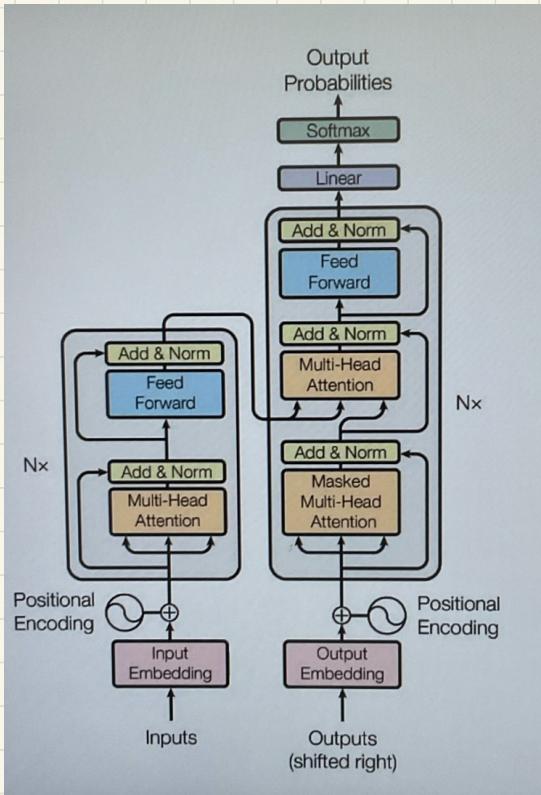
el valor
de atención
es alto!

$$z_3 = 0,2 \downarrow v_1 + 0,6 \downarrow v_2 + 0,1 \downarrow v_3$$

gustan I like cats

Estamos representando una palabra objetivo como una combinación lineal de la secuencia de origen.

- ~ Masked Multi-head Attention : autoregresivo
- ~ Multi Head Attention : Combinar target + Source
- ~ Linear layer : genera logits con el tamaño del vocabulario
- ~ Softmax : Selecciona la palabra con mayor probabilidad.



Secuencia
de entrada

Secuencia
Objetivo

- } Softmax: Samplo una palabra
- } Linear : logits o vocabulario
- } Complejidad / No linealidad
- } Representación del encoder + target
- } Métricas de atención enmascaradas + info de la Secuencia objetivo

Para entrenar :

- ~ Minimizamos la diferencia entre las distribuciones de probabilidad de lo predicho y lo esperado.
- ~ Cross-Entropy.
- ~ Optimizador : Adam.
- ~ Dropout para evitar overfitting.

Ideas clave :

- ~ El decodificador genera palabra en palabras la salida
- El bloque decodificador se compone de :
 - 1) Multi Head Attention
 - 2) Masked Multi-Head Attention
 - 3) feed forward
- ~ Entrenar minimiza la diferencia entre las probas predichas y esperadas.

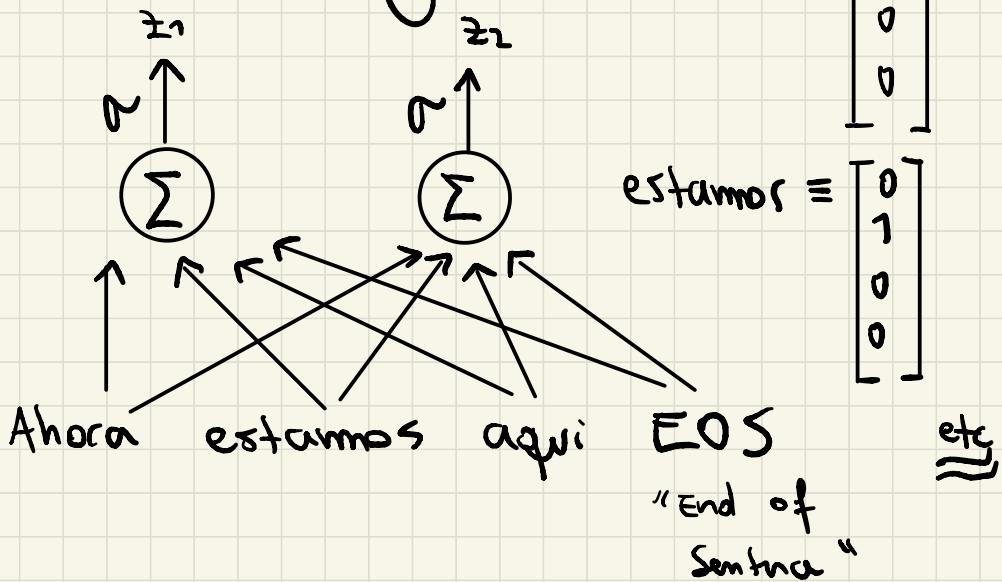
- ~ En el caso de la música, la frase como secuencia de toques.
- ~ Data musical es clave para generar.
- ~ Captura relaciones o frases pero se rompe en dependencia de larga alcance.

Segunda Explicación

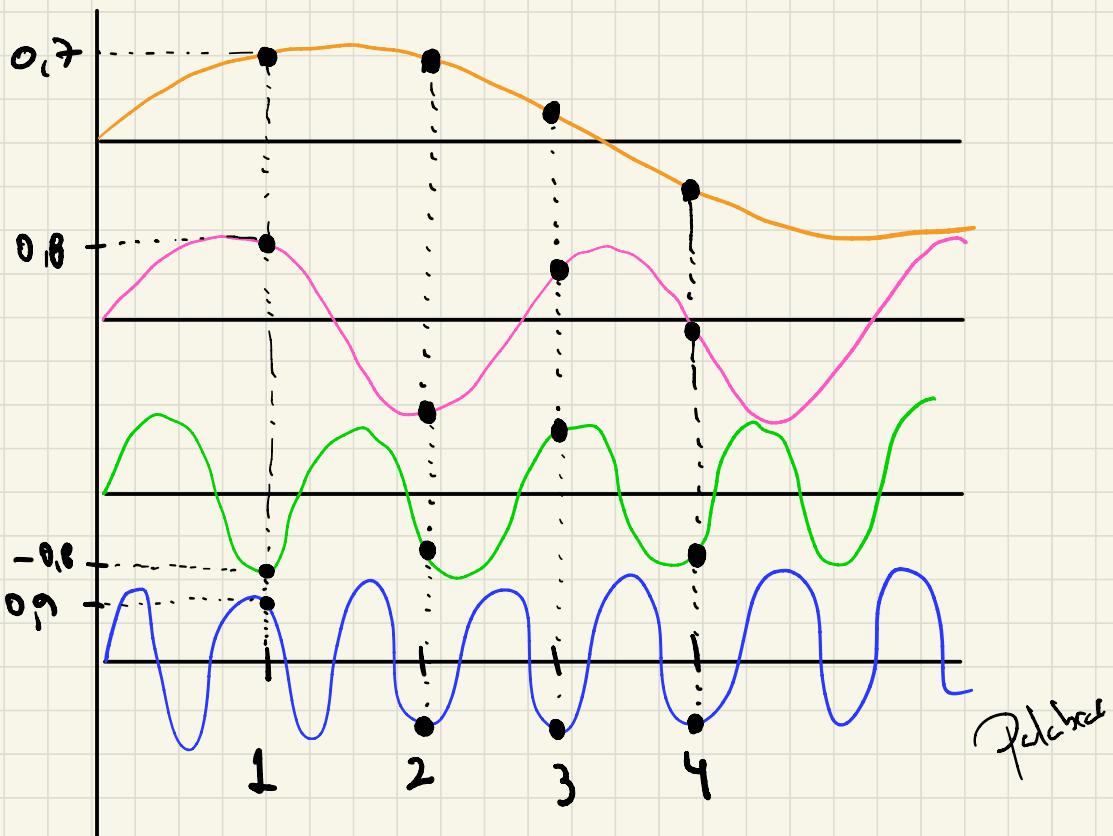
(Mirada alternativa)

Solo las
secciones

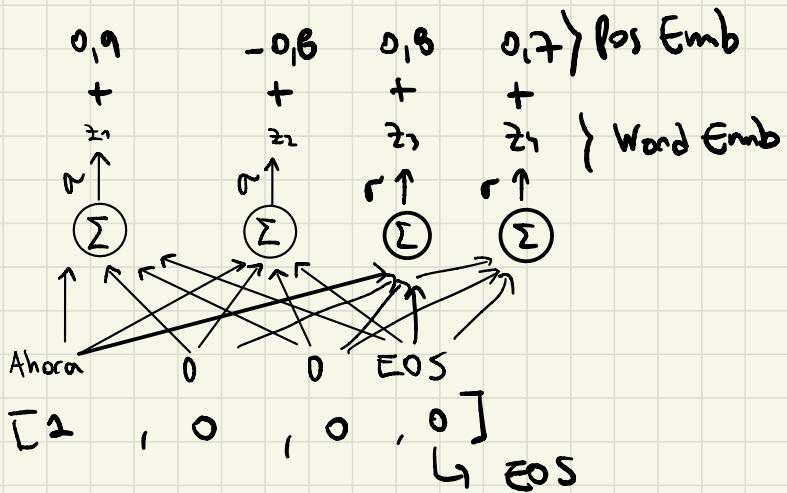
1) Word Embedding



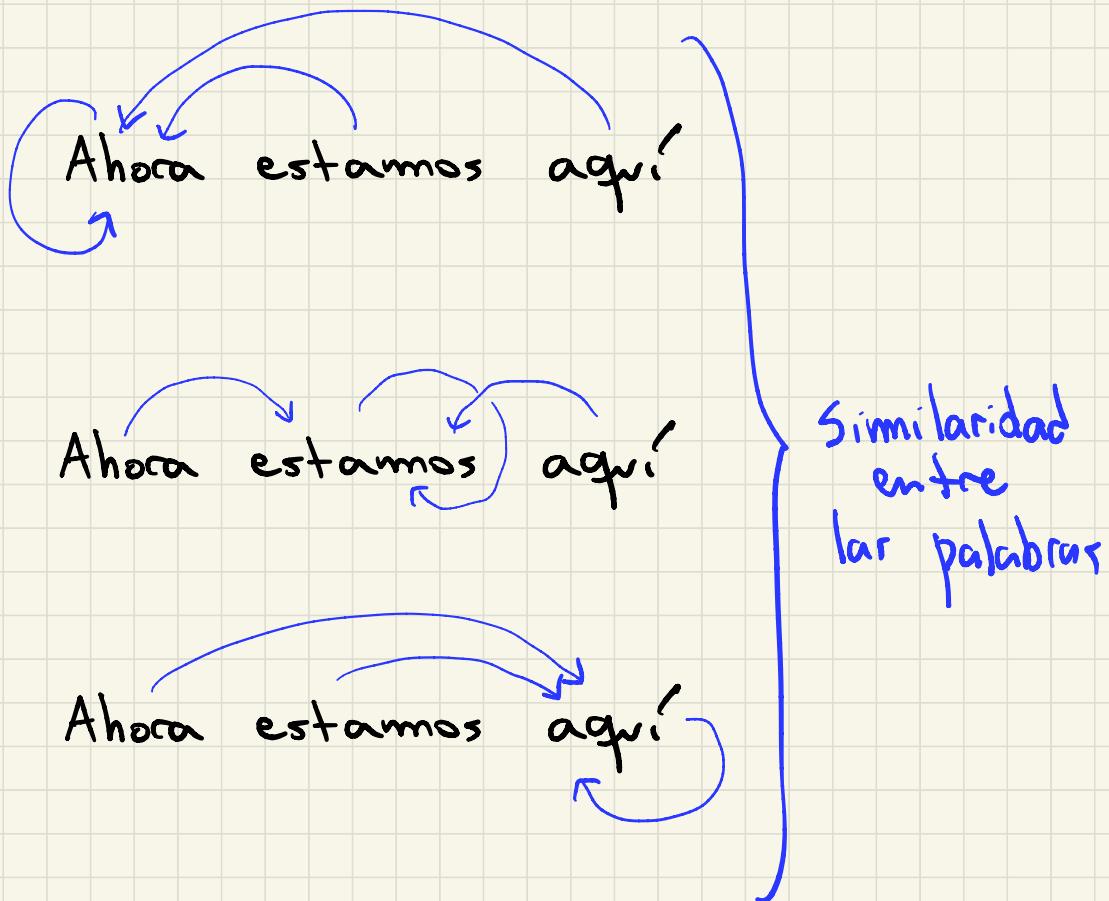
2) Positional Encoding

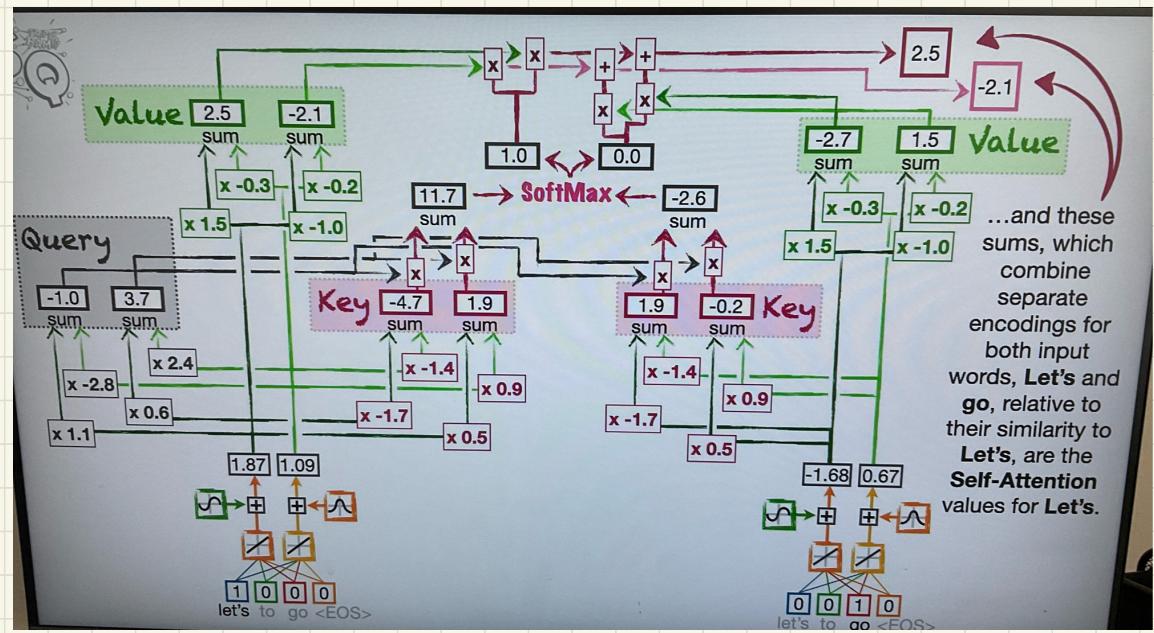
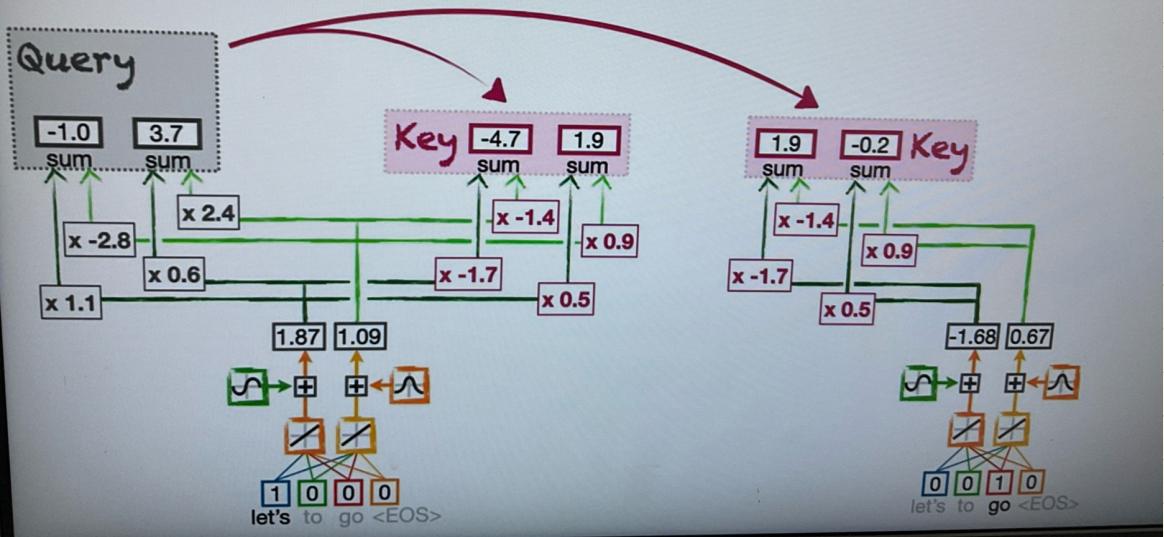


Supongamos:

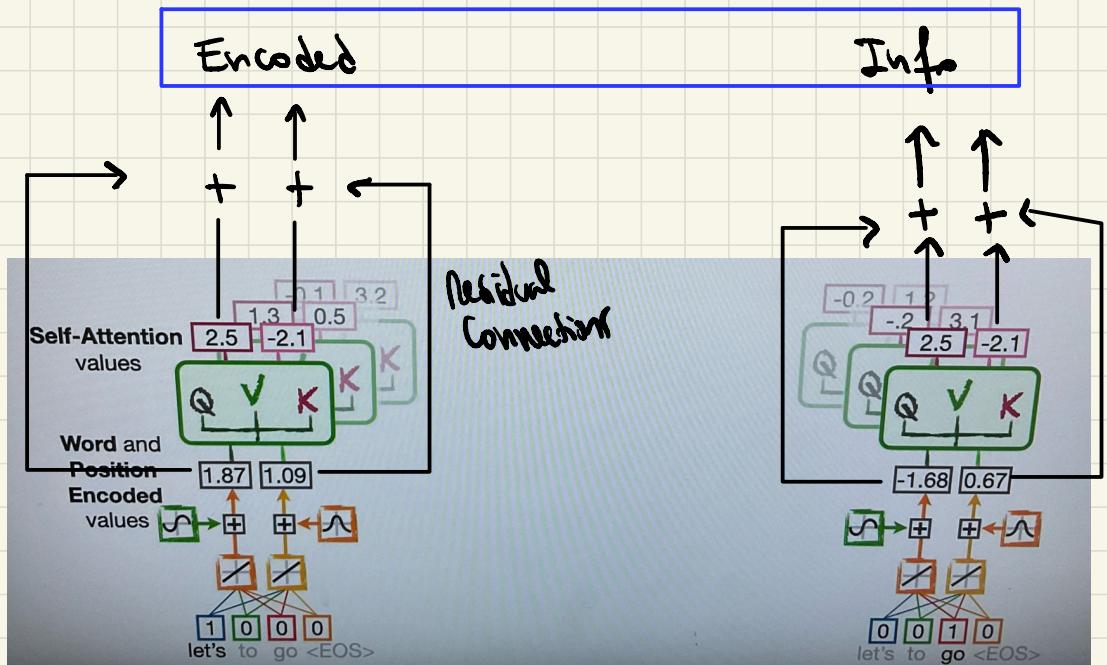


3) Medida de las relaciones entre las diferentes palabras :
Auto atención. (Self - Attention).



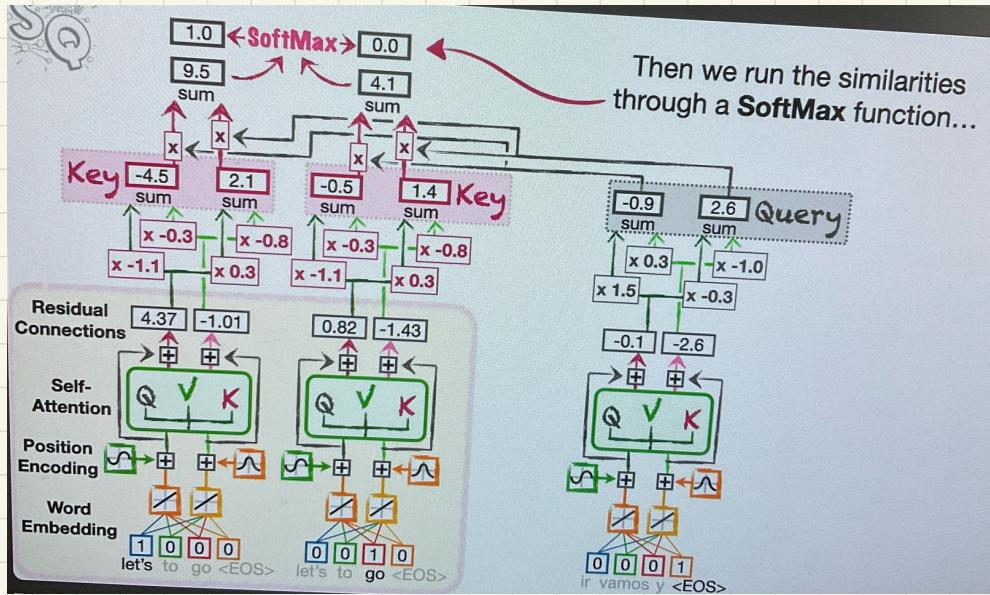


Query weights for 'misnos' : $Q_r = \sum w_a$
 Key weights " " " " : $K = \sum w_k$
 Value weights " " " " : $V = \sum w_v$



- Para decodificar la información se entrega como entrada al bloque decoder la frase objetivo!
- Embeddings \Rightarrow Positional encoding (lo mismo).
- Creamos una copia del módulo de atención por palabra \Rightarrow puedes calcular todo al mismo tiempo.

- Las matrices W_Q , W_K , W_V son diferentes a las del encoder.



★ Keys del encoder control every del decoder
↳ los valores del encoder.