

NLP

Celdas con Attention & Transformers

Msc. Rodrigo Cardenas Szigety
rodrigo.cardenas.sz@gmail.com

Programa de la materia



Clase 1: Introducción a NLP, Vectorización de documentos.

Clase 2: Preprocesamiento de texto, librerías de NLP y Rule-Based Bots.

Clase 3: Word Embeddings, CBOW y SkipGRAM, representación de oraciones.

Clase 4: Redes recurrentes (RNN), problemas de secuencia y estimación de próxima palabra.

Clase 5: Redes LSTM, análisis de sentimientos.

Clase 6: Modelos Seq2Seq, traductores y bots conversacionales.

Clase 7: Celdas con Attention. Transformers, BERT & ELMo, fine tuning.

Clase 8: Cierre del curso, NLP hoy y futuro, deploy.

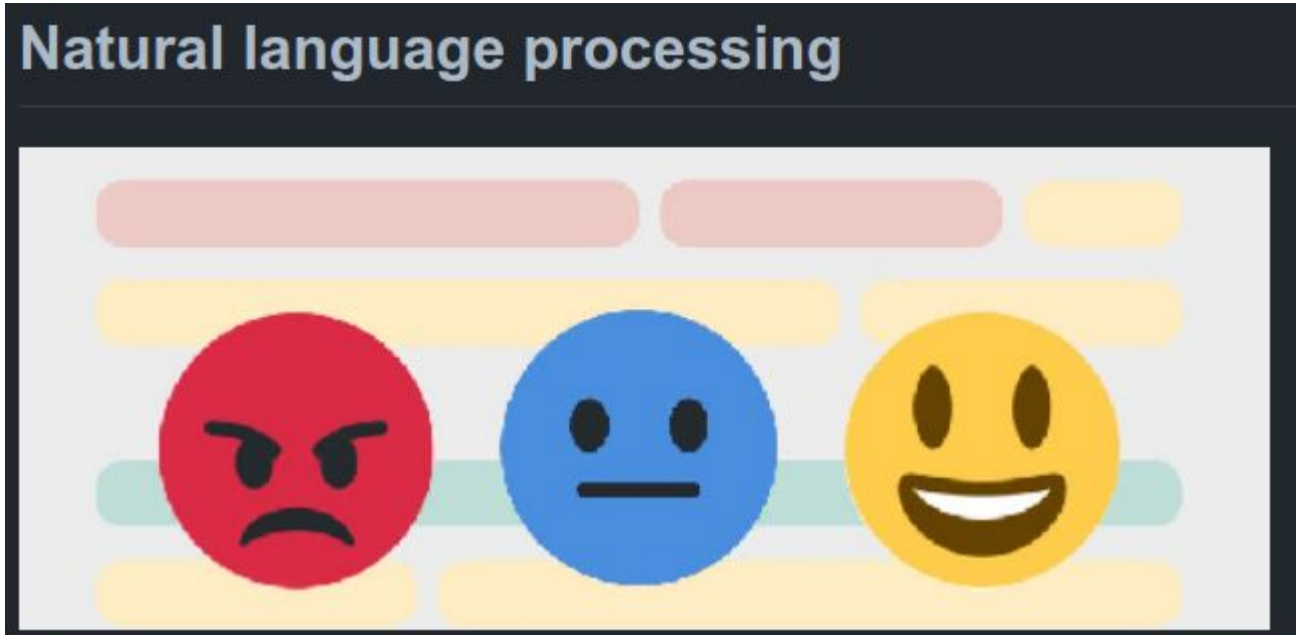
*Unidades con desafíos a presentar al finalizar el curso.

*Último desafío y cierre del contenido práctico del curso.

Desafio final...



Tienen que lucir en su github todo el trabajo que hicieron en esta materia (cada uno de los desafíos) como si fuera un portfolio.



LINK EJEMPLO

Cronología del nuevo "estado del arte"



Attention
2014/2015



ELMo
2017

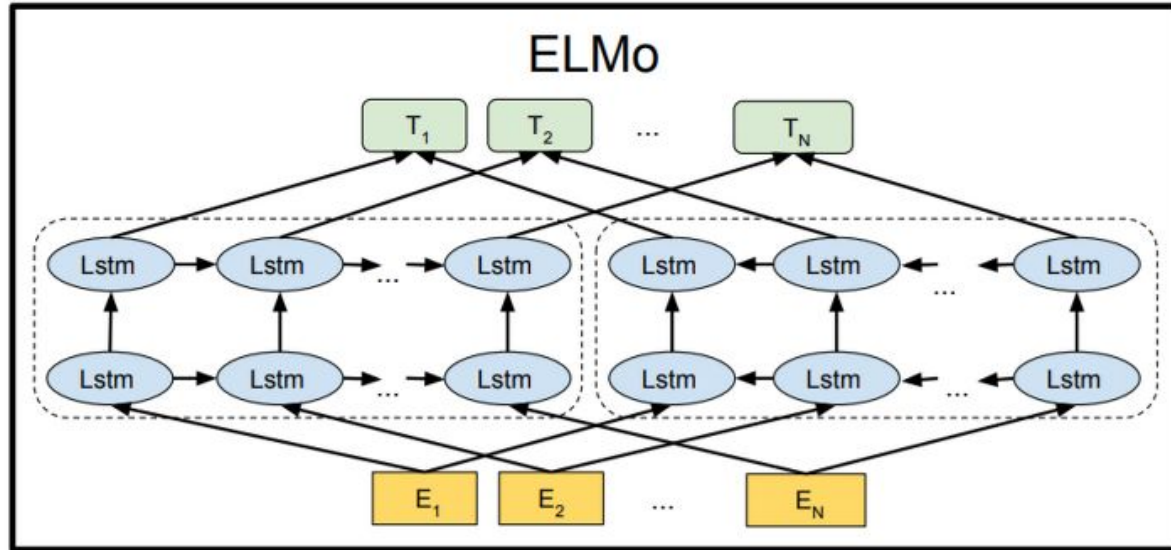


Transformers
2017/2018



BERT
2019

ELMo: Deep Contextual Word Embeddings (2017)



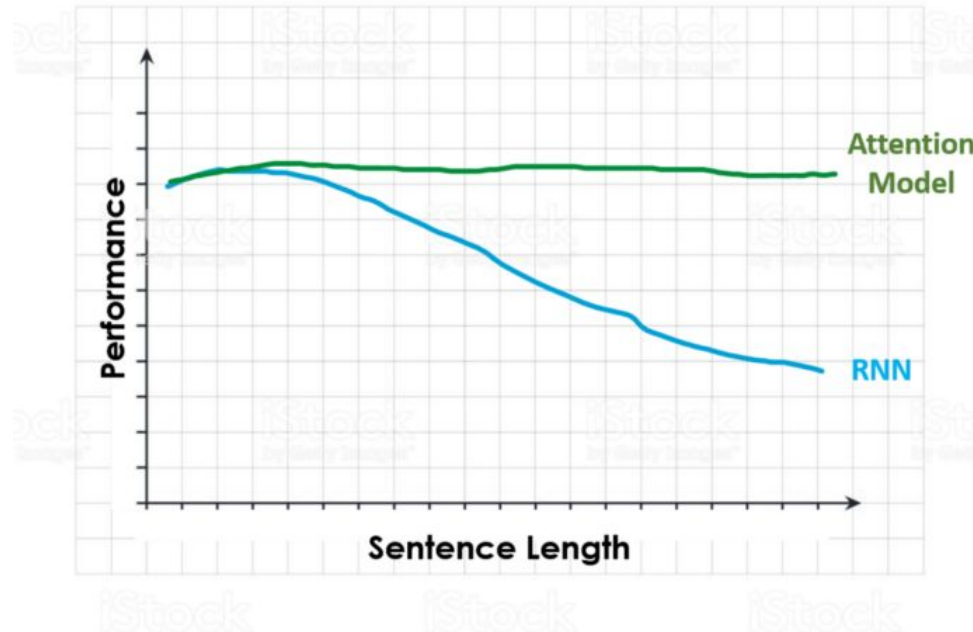
En definitiva los embeddings que se utilizan de cada palabra se forman del contexto (las hidden layers), no son únicos como sucede en GloVe o Fasttext. Entrenado en caracteres utilizando el One Billion Words Dataset

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
ELMo	Chico Ruiz made a spectacular <u>play</u> on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent <u>play</u> .
	Olivia De Havilland signed to do a Broadway <u>play</u> for Garson {...}	{...} they were actors who had been handed fat roles in a successful <u>play</u> , and had talent enough to fill the roles competently , with nice understatement .

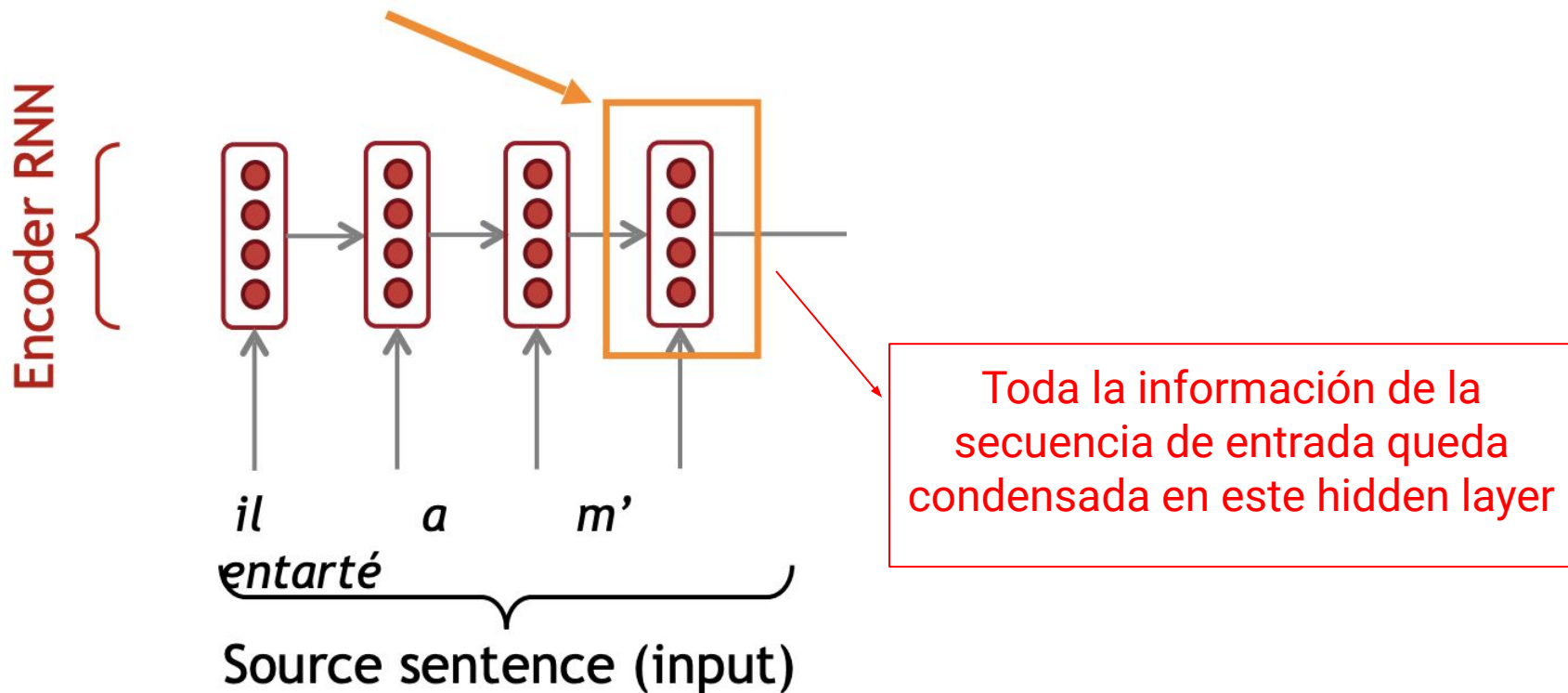
Limitaciones de las RNN/LSTM

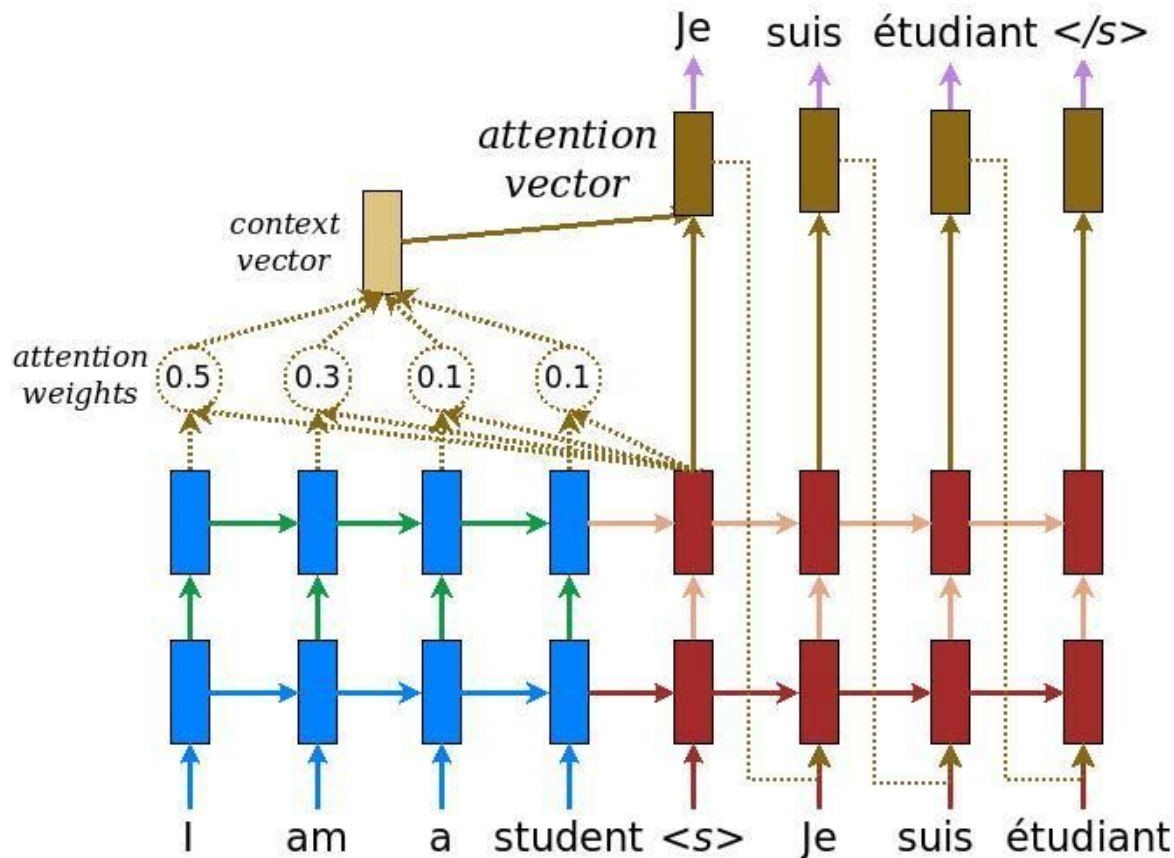


"Una celda RNN o LSTM pierde performance con secuencias de texto largos, ya que el contexto (hidden state) de la celda se degrada".



Limitaciones de las RNN/LSTM





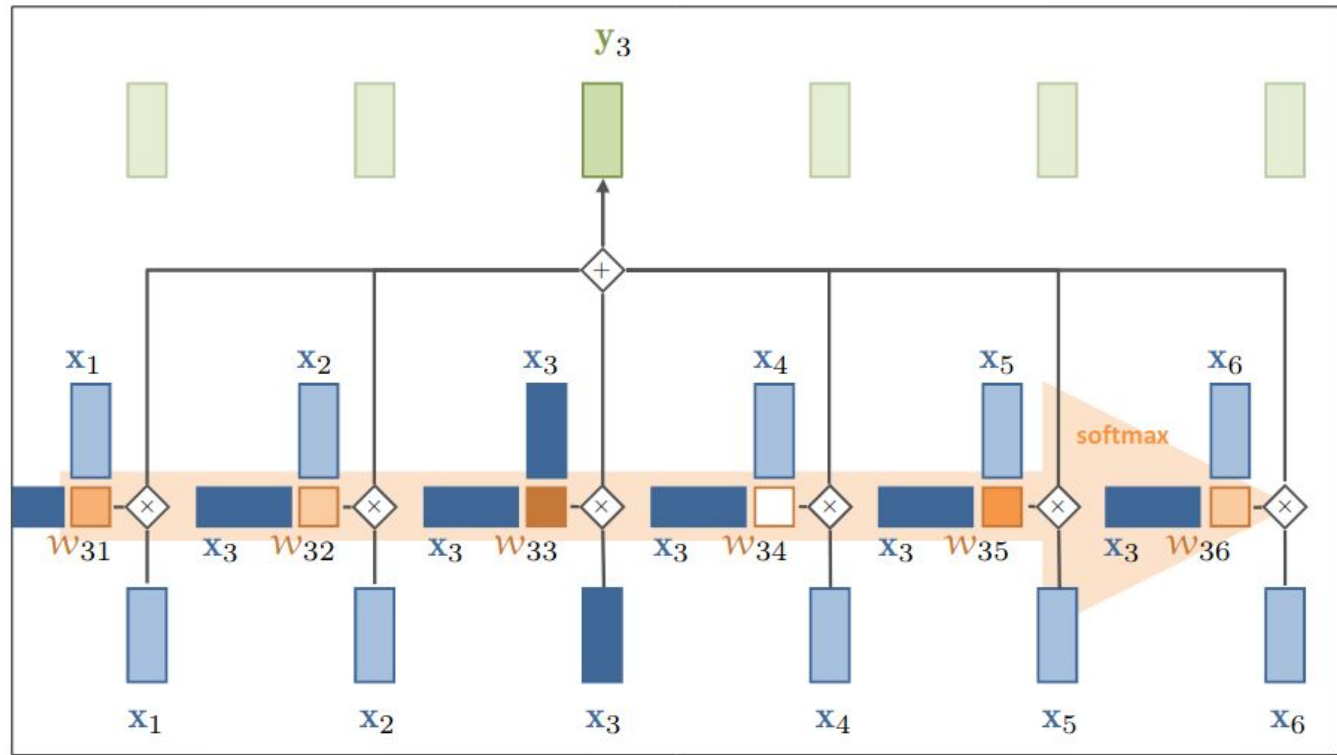
Ventajas:

- El decoder se focaliza en las palabras más relevantes de la secuencia de entrada para cada etapa.
- Agrega explicabilidad a las predicciones del decoder, porque podemos observar las palabras relevantes en cada step.

Desventajas:

- Se agregan más operaciones no paralelizables al proceso.

Self-attention



Idea: Usar las mismas (Self) salidas del encoder para computar los scores de atención.

No tiene que “esperar” al decoder. ¡Ahora es más paralelizable!

¡Mejora la señal de gradiente!

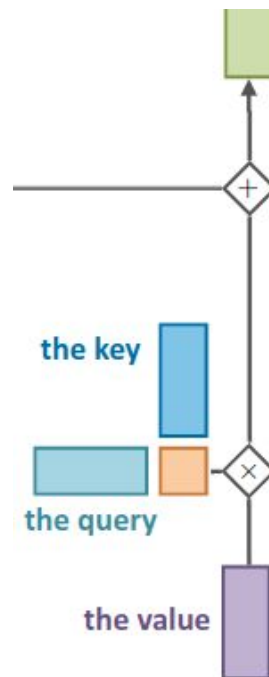
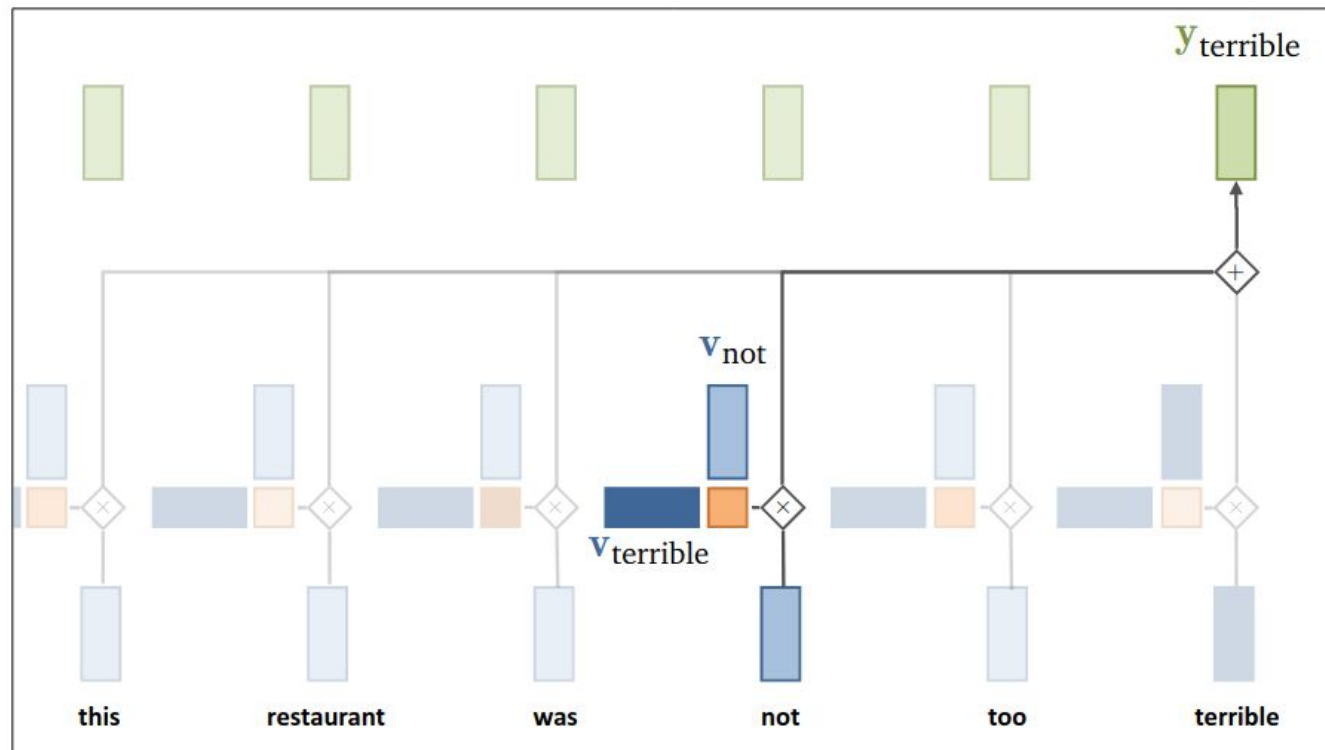
¡Funciona para cualquier tamaño de secuencia!

$$y_i = \sum_j w_{ij} x_j$$

$$w'_{ij} = x_i^T x_j$$

$$w_{ij} = \frac{\exp w'_{ij}}{\sum_j \exp w'_{ij}}$$

Attention e information retrieval



El mecanismo de atención es como un diccionario
"suave"

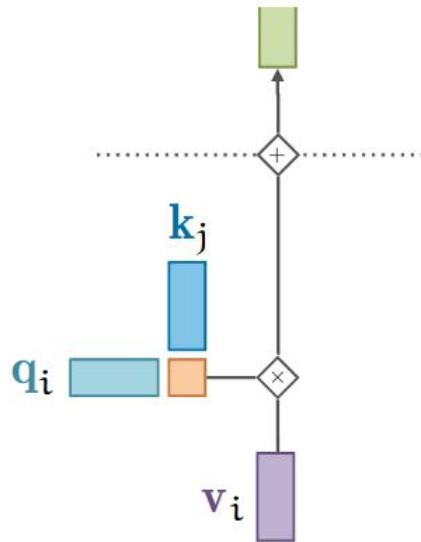
Queries, keys & values



$$\mathbf{k}_i = \mathbf{K}\mathbf{x}_i + \mathbf{b}_k$$

$$\mathbf{q}_i = \mathbf{Q}\mathbf{x}_i + \mathbf{b}_q$$

$$\mathbf{v}_i = \mathbf{V}\mathbf{x}_i + \mathbf{b}_v$$



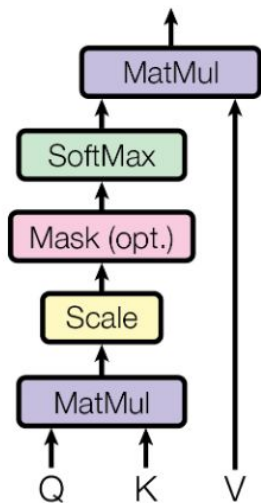
Las transformaciones lineales le dotan más flexibilidad al comportamiento de la capa.

K, Q y V serán los parámetros a aprender la capa de attention. También reducen dimensionalidad.

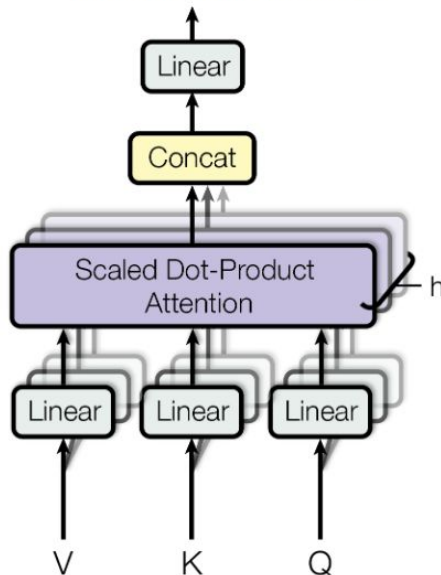
Armando el Transformer



Scaled Dot-Product Attention



Multi-Head Attention



Multi-head es similar a tener varios filtros convolucionales en redes convolucionales. Cada módulo de atención puede aprender a “atender” diferentes relaciones entre las secuencias.

Transformers 2017/2018

[LINK](#)

[LINK](#)

[LINK PAPER](#)
[\(pytorch\)](#)



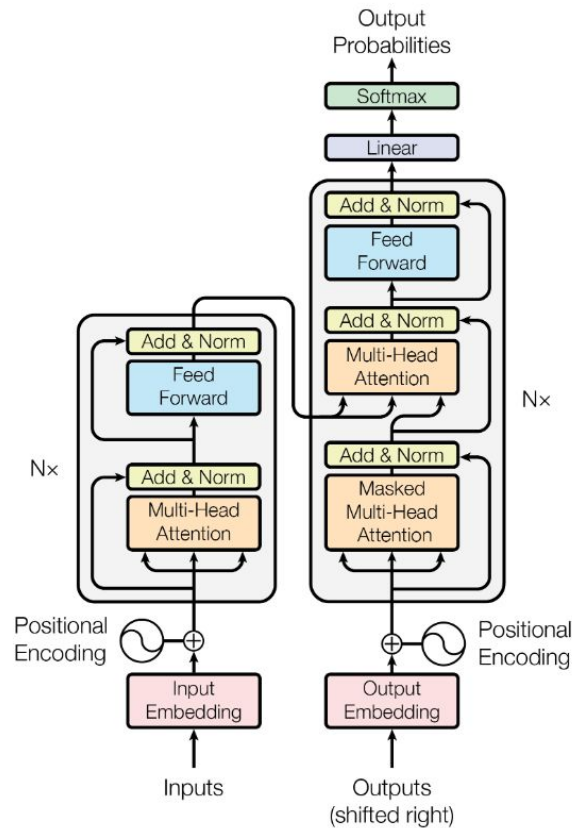
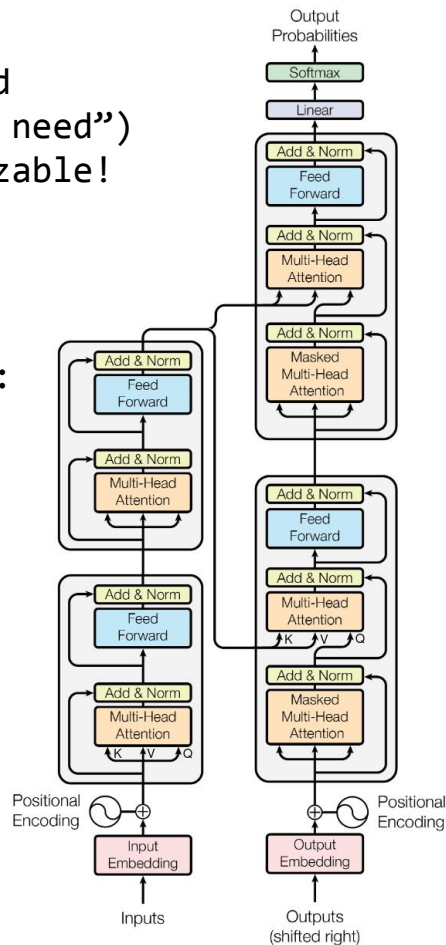
Todo attention y feed forward (“is all you need”) ¡totalmente paralelizable!

¡Muchos parámetros!

Detalles adicionales:

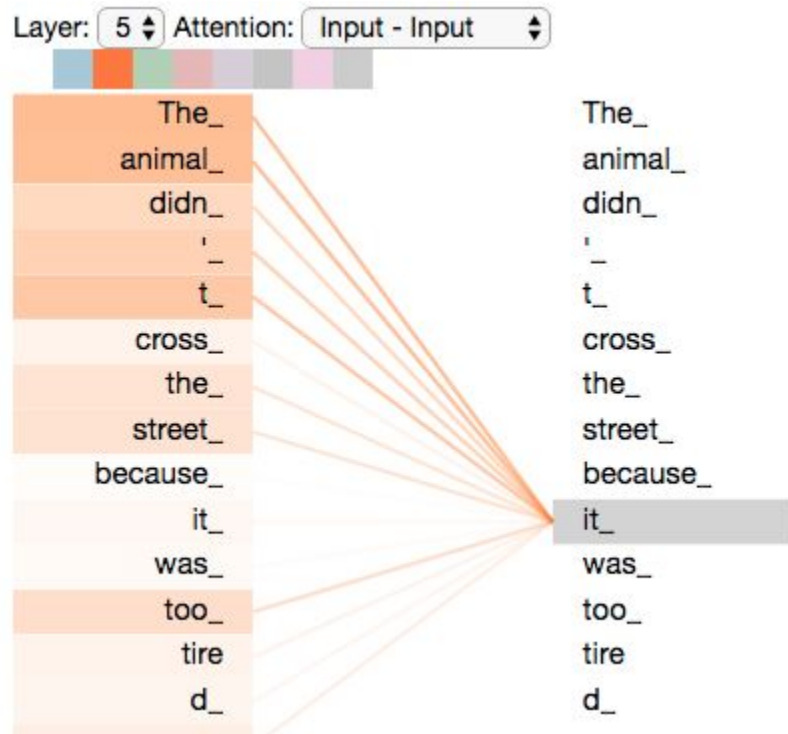
- enmascaramiento
- encoding posicional
- layer normalization

Ejemplo con dos bloques transformers en encoder y decoder



Arquitectura general

Deseamos reforzar el análisis semántico



¿"It" se refiere a "animal" o "street"?

"The animal didn't cross the street because it was too tired"

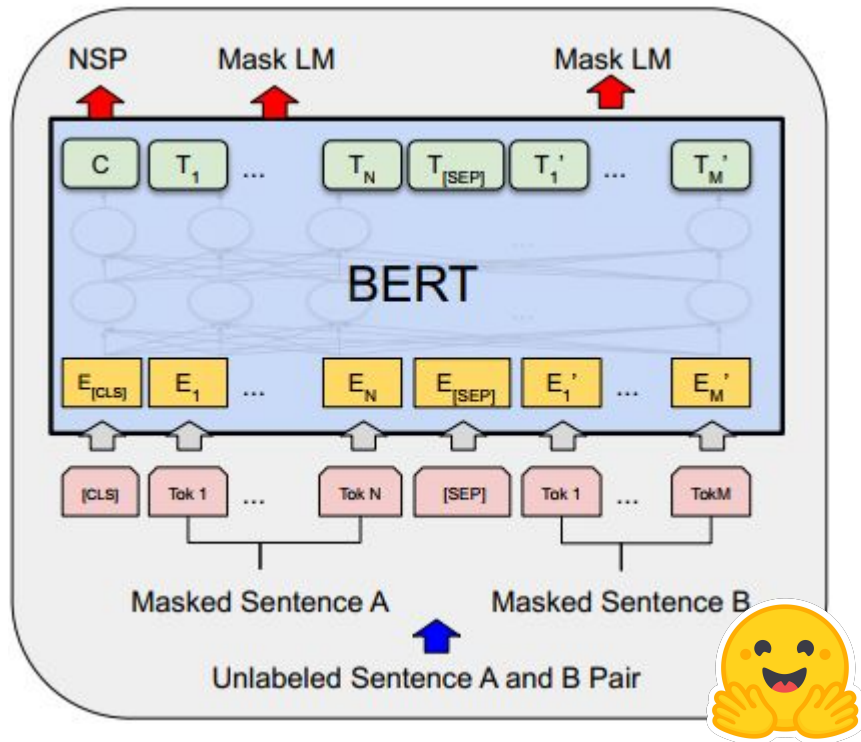
BERT (2019)

“Pre-training of Deep Bidirectional Transformers”

[LINK](#)

[LINK](#)

[LINK](#)



[Creado por Hugging Face/Google](#)

Su arquitectura se basa en stack de layers de transformers (encoders)

Se entrenó observando dos secuencias de entrada, su misión era determinar si la segunda secuencia estaba relacionada o no con la primera.

De forma aleatoria en cada inferencia se ocultó una palabra de cada secuencia (masked)

Utiliza un tokenizador especial (WordPiece)

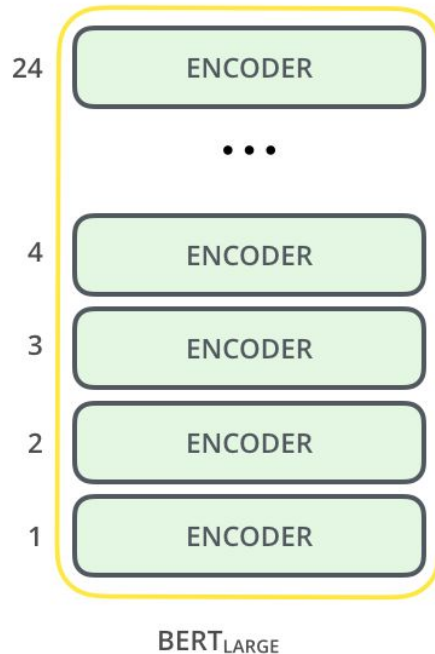
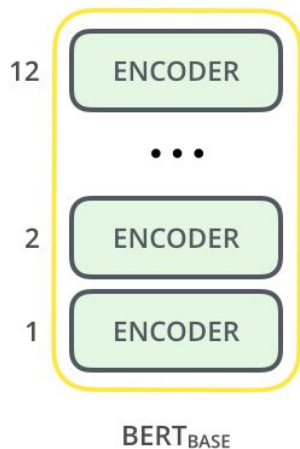
[CLS] Hermoso día el [MASK] hoy [SEP]₁₅

BERT base y large



Bert base tiene 12 encoders layer

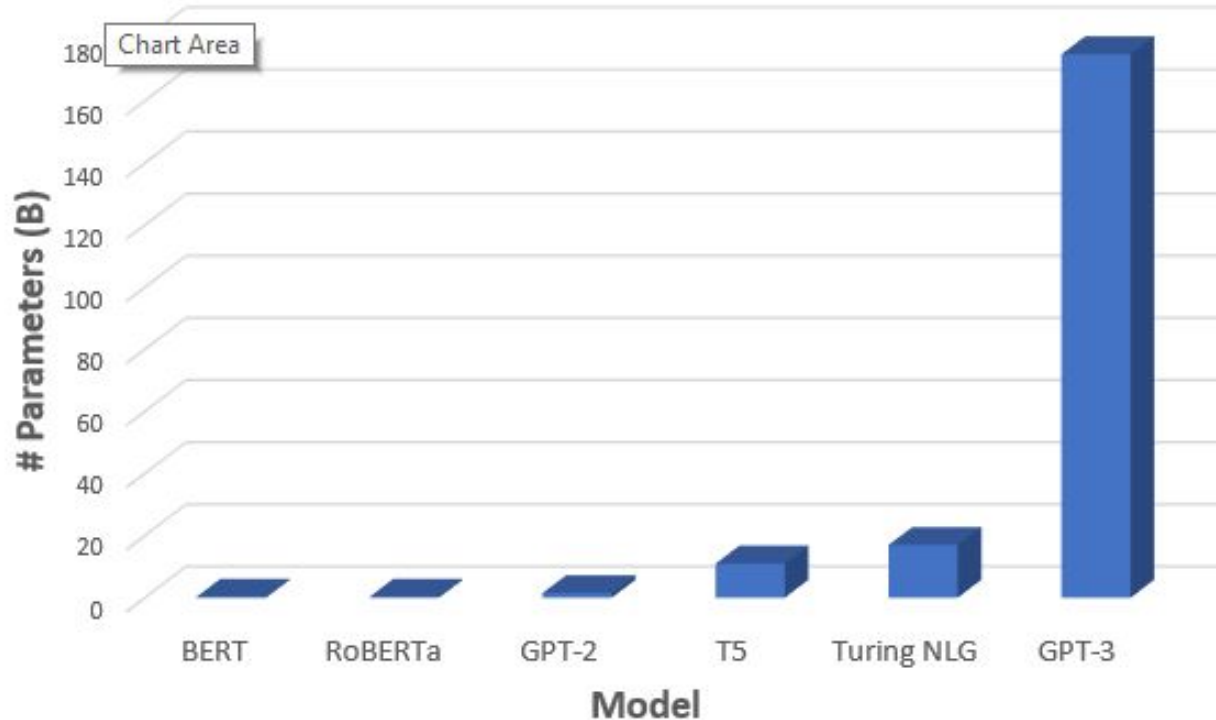
Bert large tiene 24 encoders layer



BERT vs los demás



Bert (base) tiene 110 millones de parámetros
~54 horas de entrenamiento en Google TPUs



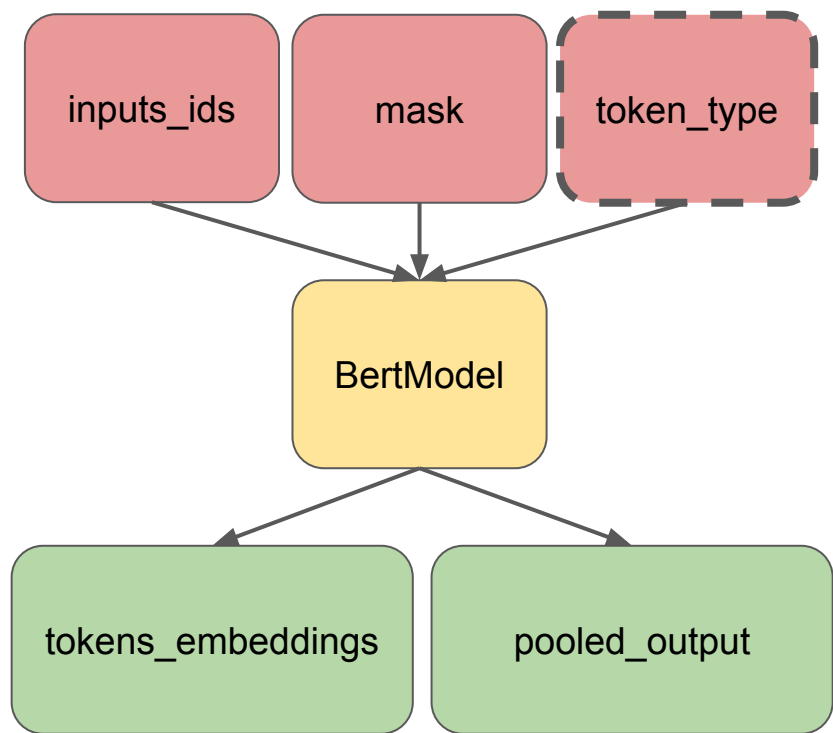
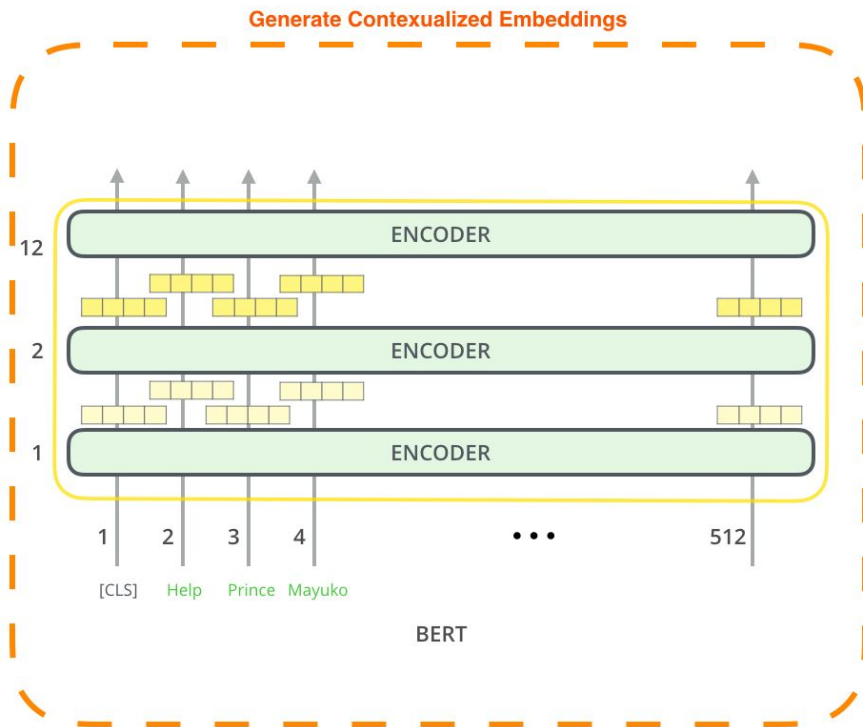


Link al Colab



LINK

BERT - Embeddings contextualizados





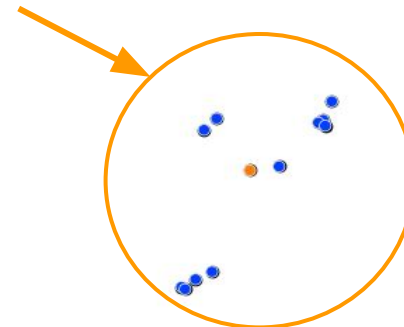
Link al Colab



LINK



Cluster que representa
a todo el texto



Se calcula el centroide
(punto naranja)

Se mide qué sentencias están cerca al
centro y se arma un nuevo texto



Link al Colab



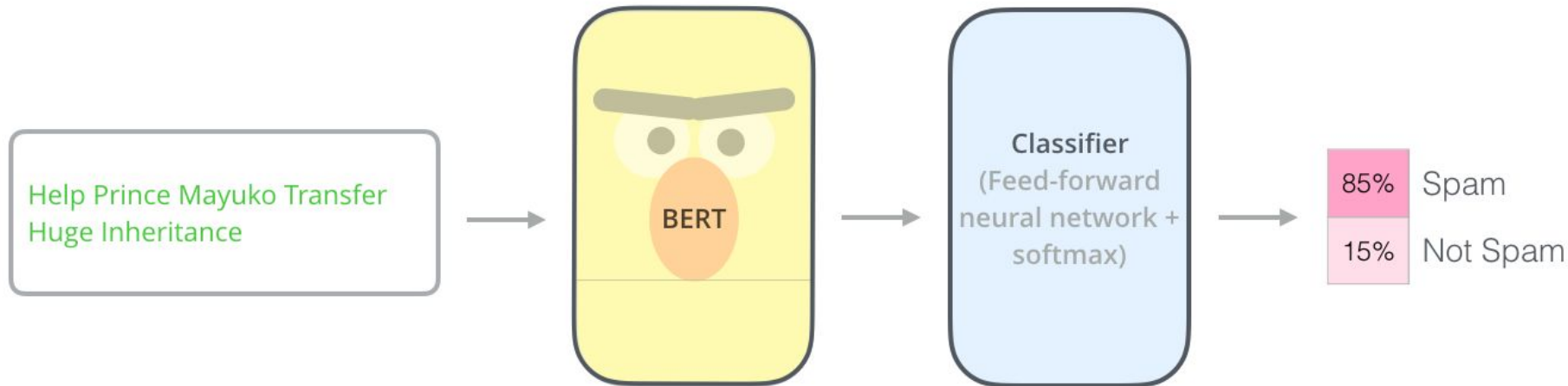
LINK

BERT - Classifier (sentiment analysis)



Input
Features

Output
Prediction



Fine tuning



"En el proceso de transfer learning agregamos a un modelo con pesos pre-entrenados nuestras capas custom por entrenar".

"En el proceso de fine-tuning, ya habiendo realizado un primer entrenamiento, se realiza un ajuste fino de todo el modelo, incluyendo las capas pre-entrenadas".





Link al Colab



[LINK](#)



Link al Colab



[LINK](#)



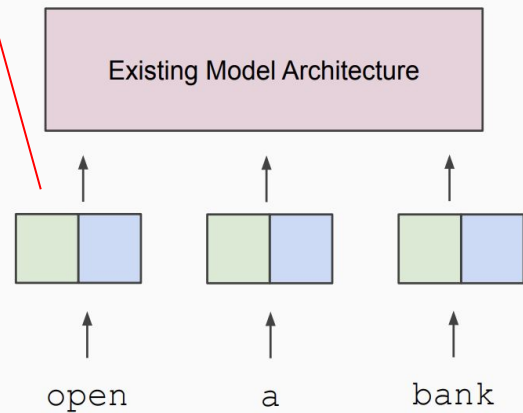
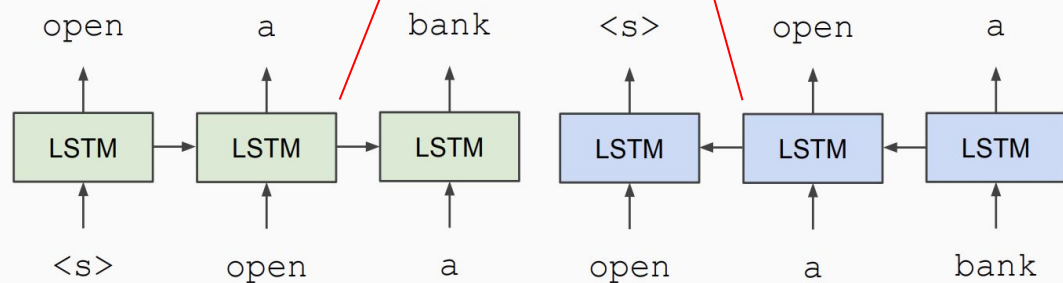
¡Muchas gracias!

ELMo: Deep Contextual Word Embeddings (2017)



Entrenaron las redes
LSTM en ambas
direcciones (BRNN)

Durante la inferencia
concatenan los
embeddings

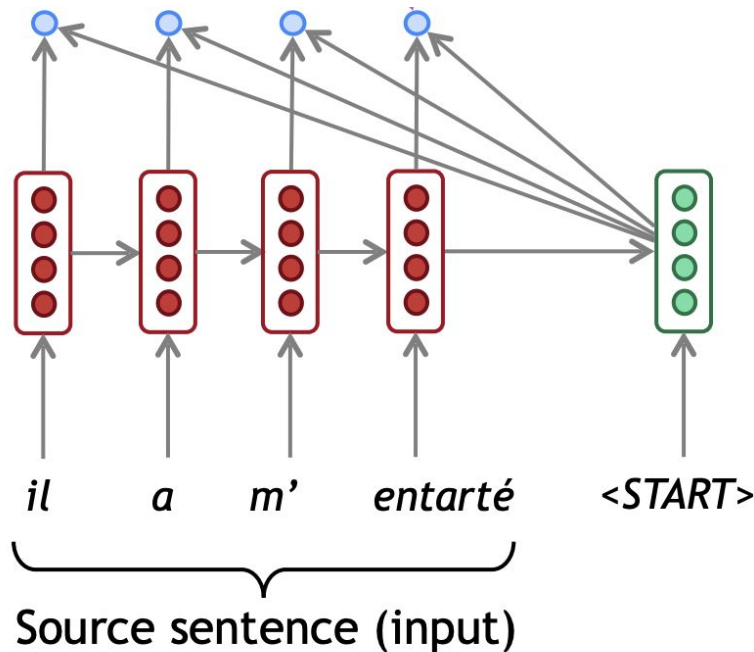


En definitiva los embeddings que se utilizan de cada palabra se forman del contexto (las hidden layers), no son únicos como sucede en Glove o Fasttext



Attention
scores

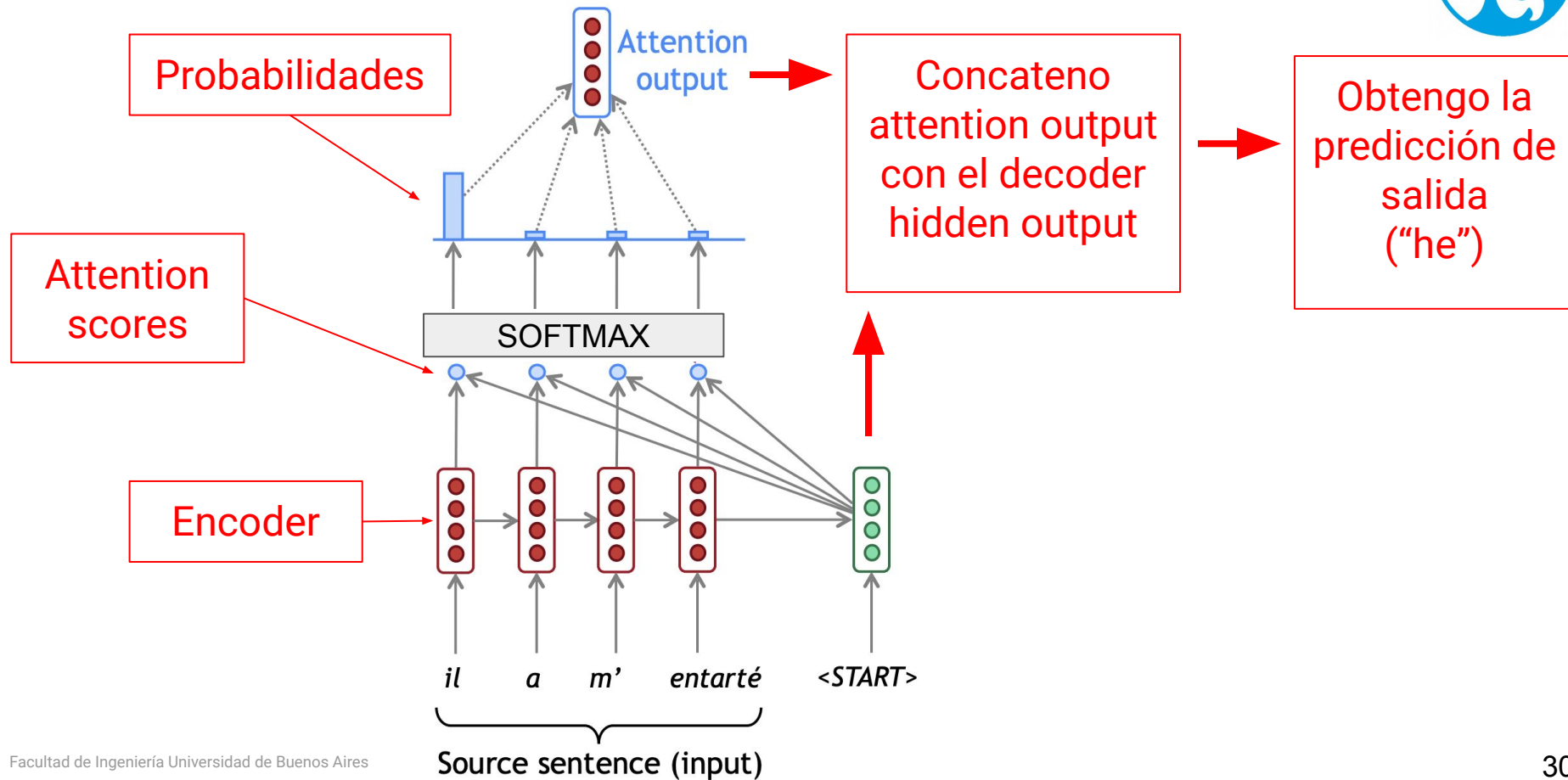
Encoder



En cada instante del decoder, uso el hidden layer del **decoder** para calcular los attention score ponderando las hidden layer del **encoder** con el contexto actual.

Se calcula como el producto escalar de hidden layer del decoder con cada hidden del encoder (un score por hidden layer del encoder)

Attention output



Attention output

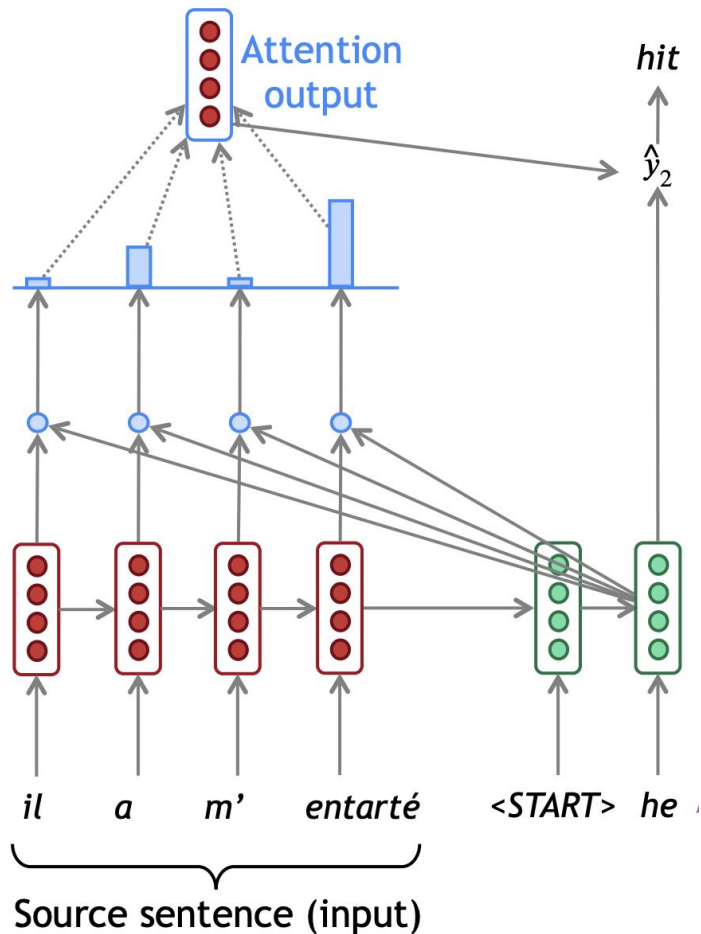
[LINK](#)

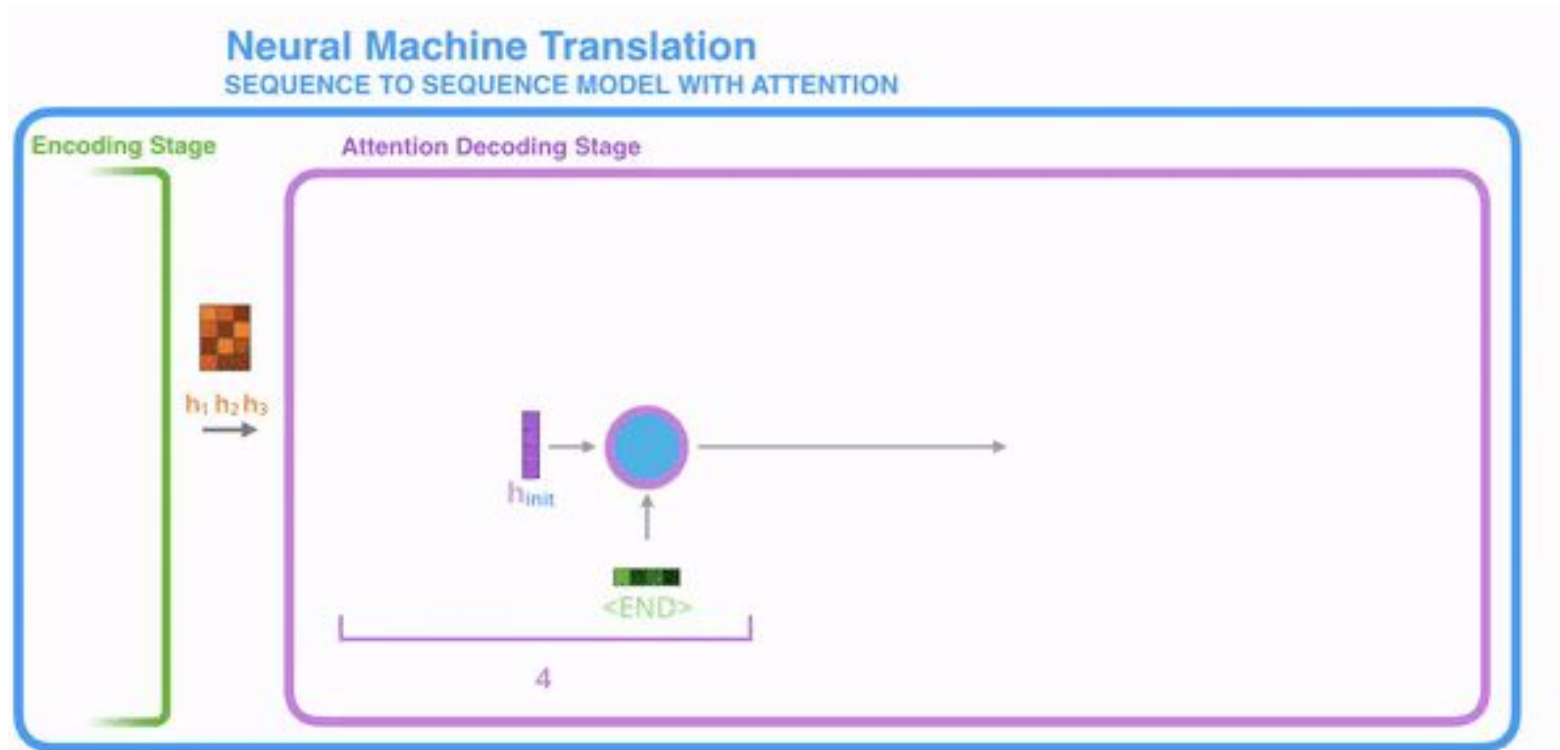


Attention at time step 4



Attention Seq2Seq





Attention Seq2Seq

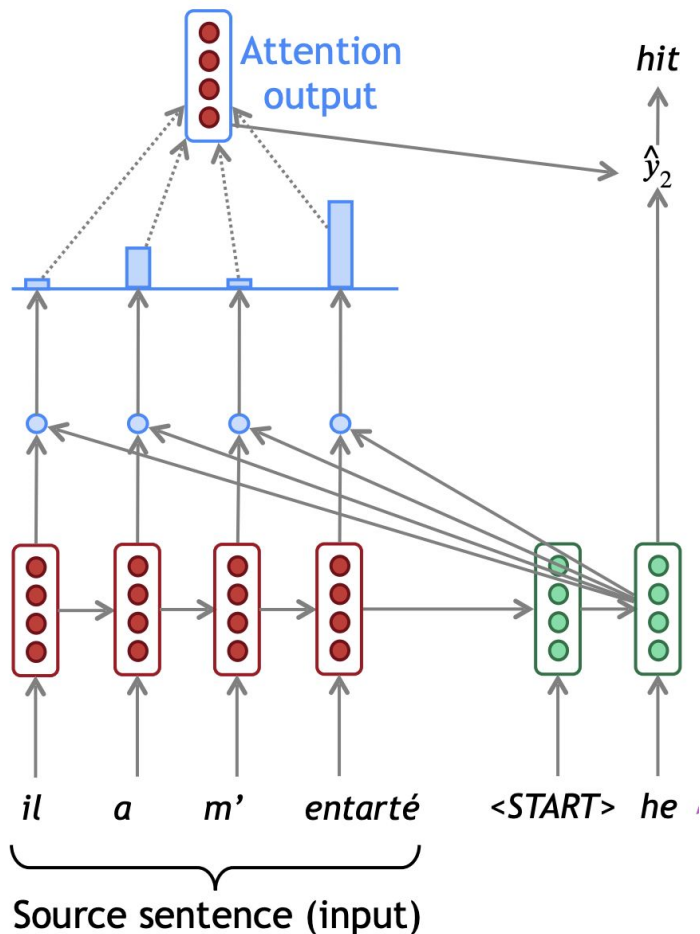


Ventajas:

- El decoder se focaliza en las palabras más relevantes de la secuencia de entrada para cada etapa.
- Agrega explicabilidad a las predicciones del decoder, porque podemos observar las palabras relevantes en cada step.

Desventajas:

- Se agregan más operaciones no paralelizables al proceso.



Self-attention

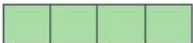


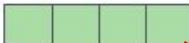
Input

Thinking

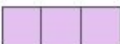
Machines

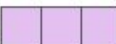
Embedding

x_1 

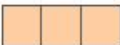
x_2 

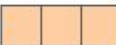
Queries

q_1 

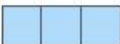
q_2 

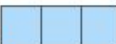
Keys

k_1 

k_2 

Values

v_1 

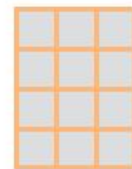
v_2 

Pesos(W) de la
layer de
self-attention



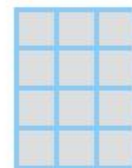
W^Q

Lo que
busca
(query)



W^K

Lo que
representa o
simboliza
(key)

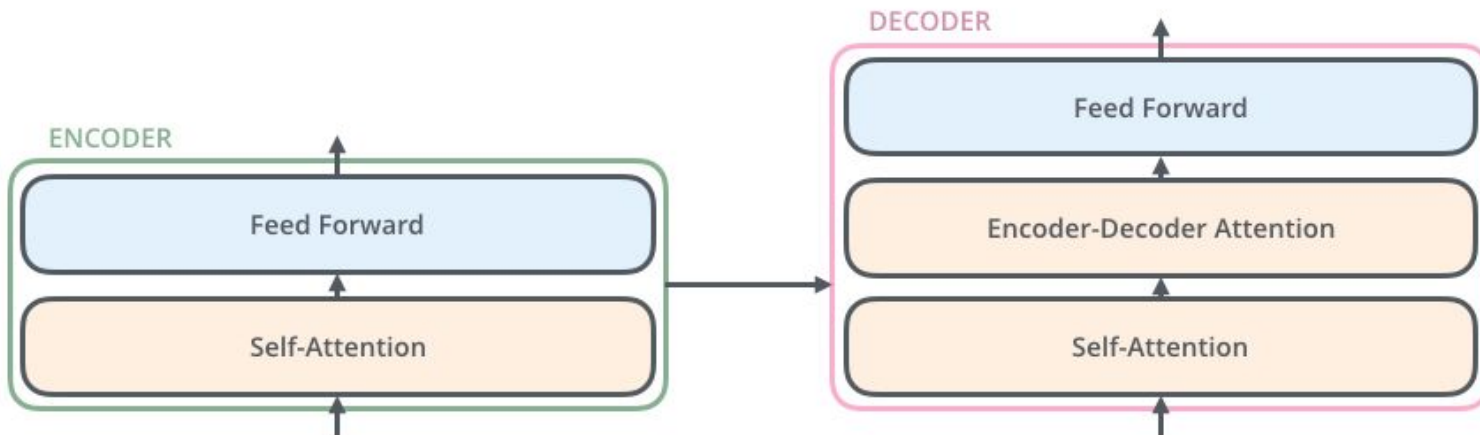


W^V

La matriz
peso de
salida
(value)



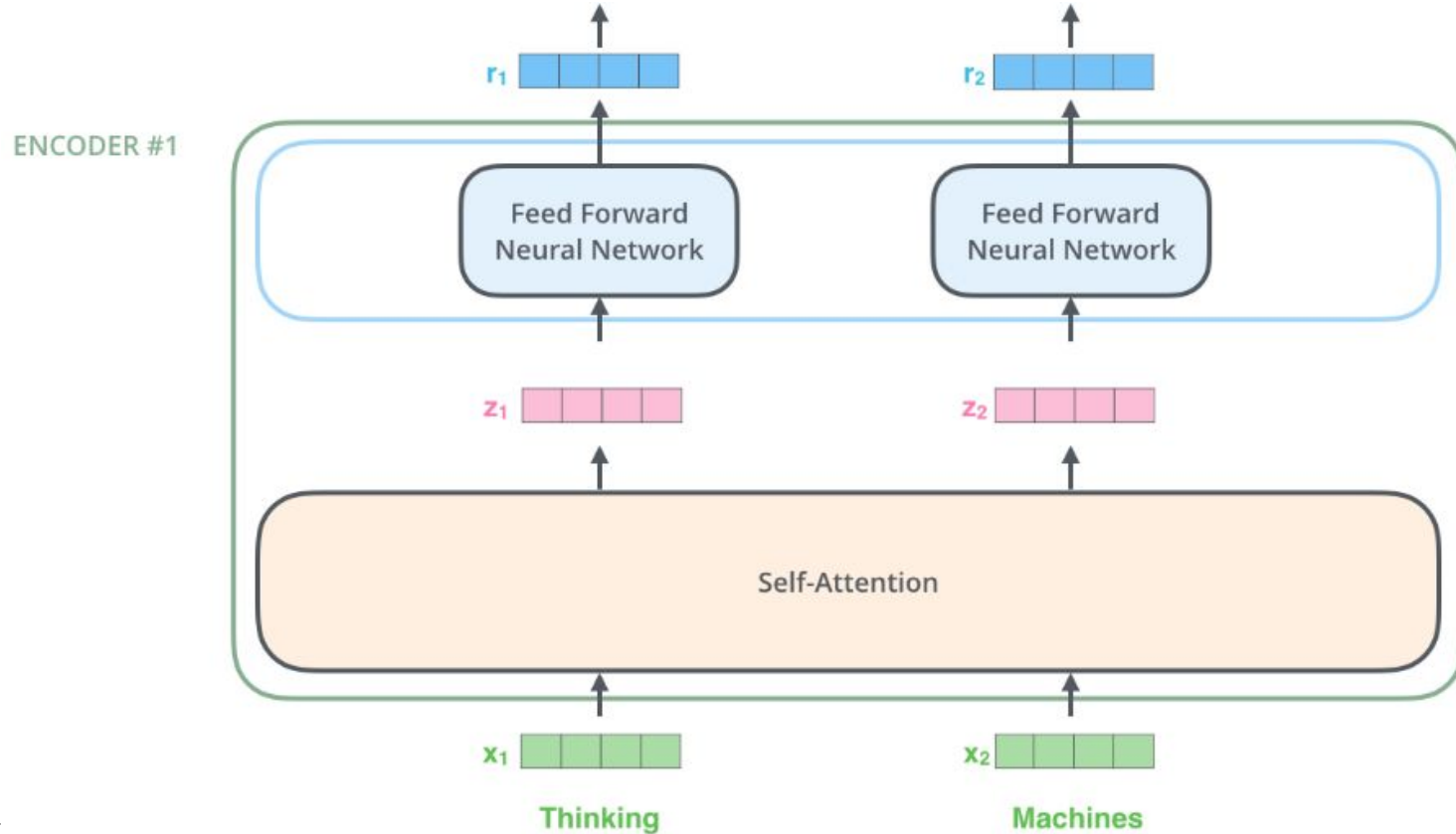
"Son el reemplazo de las celdas RNN/LSTM basadas en utilizar attention para capturar la importancia del contexto de cada palabra".



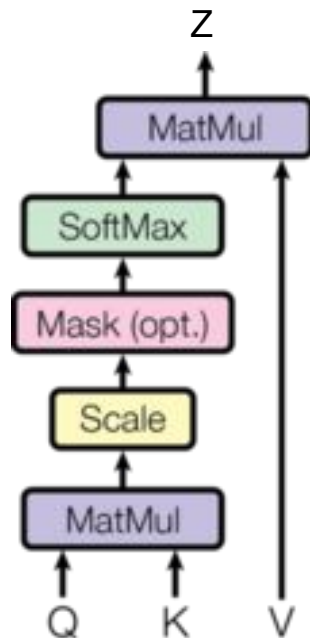
Self-attention - corazón del transformer



Transforma todas las palabras de entrada en vectores (es posible paralelizar).



Self-attention



[LINK PAPER](#)
(pytorch)

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 ($\sqrt{d_k}$)

Softmax

Softmax

X
Value

Sum

Thinking

Machines

x_1

x_2

q_1

q_2

k_1

k_2

v_1

v_2

$q_1 \cdot k_1 = 112$

$q_1 \cdot k_2 = 96$

14

12

0.88

0.12

v_1

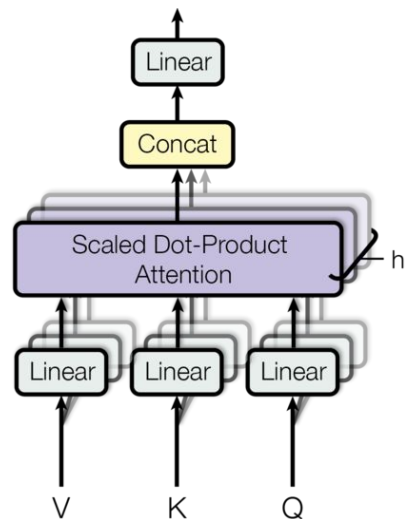
v_2

z_1

z_2

Multi head Self-attention

“The Beast With Many Heads”



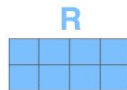
1) This is our input sentence*

Thinking Machines

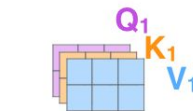
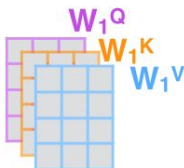
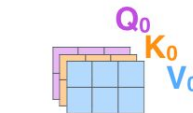
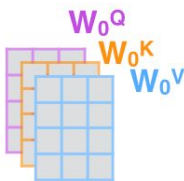
2) We embed each word*



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



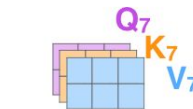
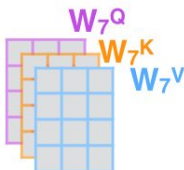
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



...

...

...



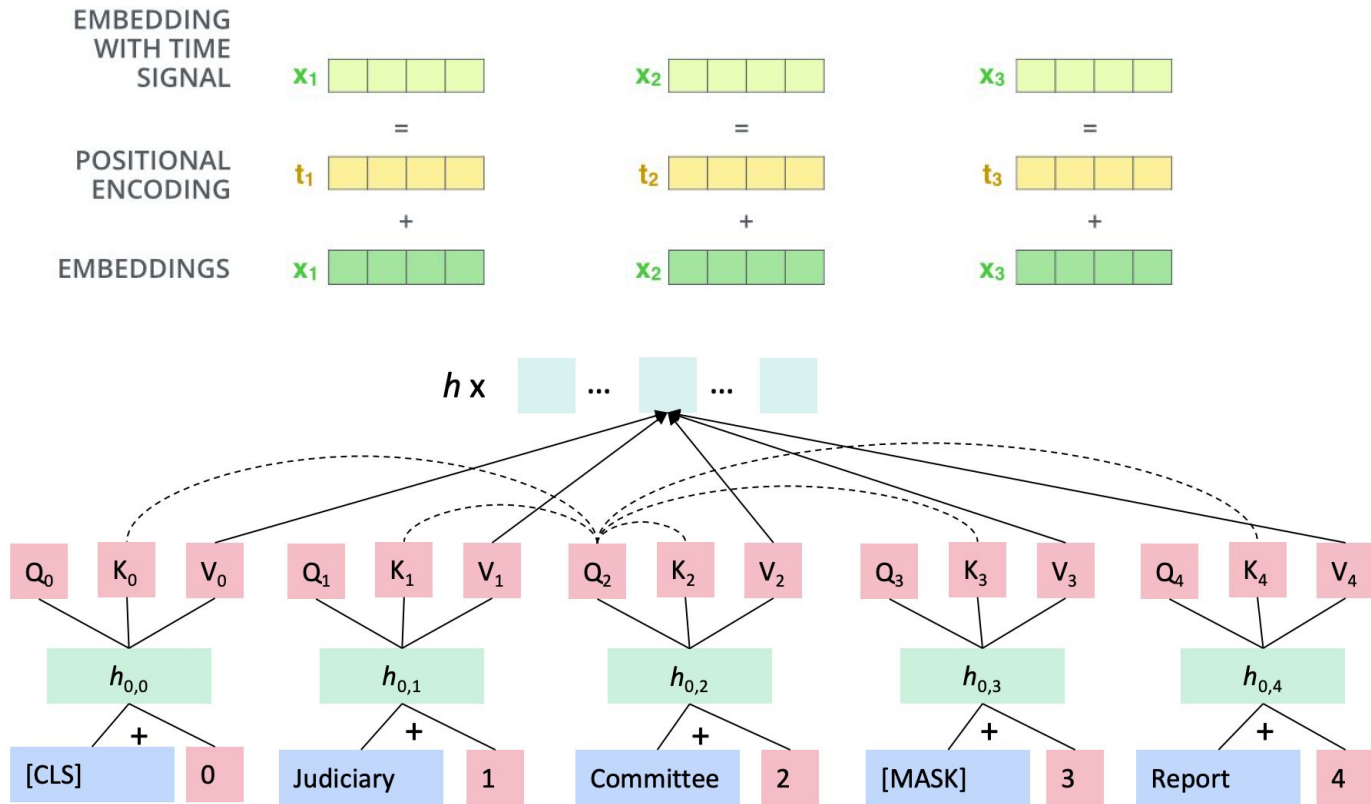
En cada stack de self-attention obtenemos "N" vectores representados con diferentes matrices

Lo mismo que representan los "N" filtros por layer de Convolución en visión

Positional encoding



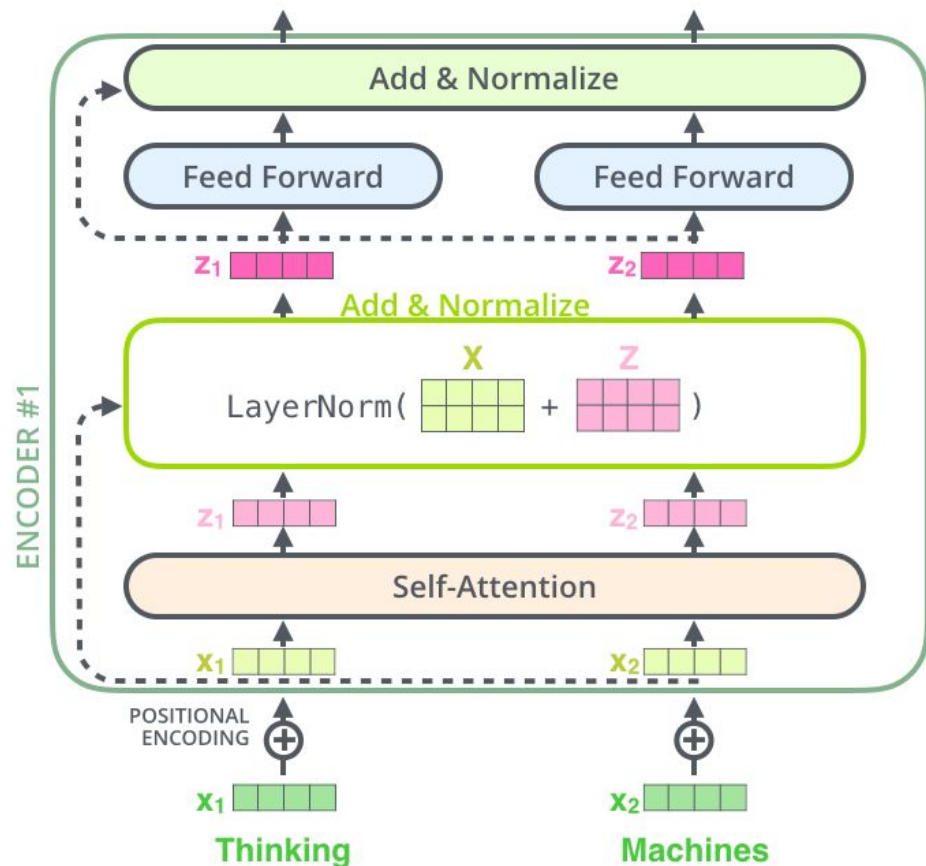
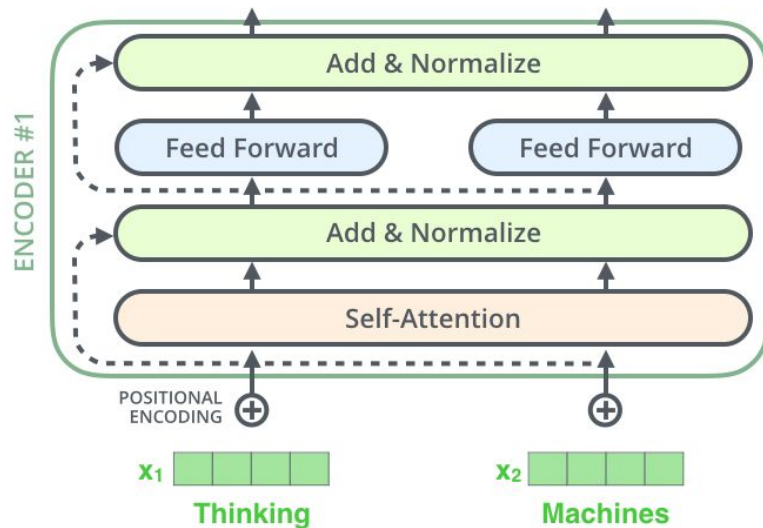
*Al embedding
de entrada
mezclarlo
con un
vector
relativo a
su posición
en la
secuencia*



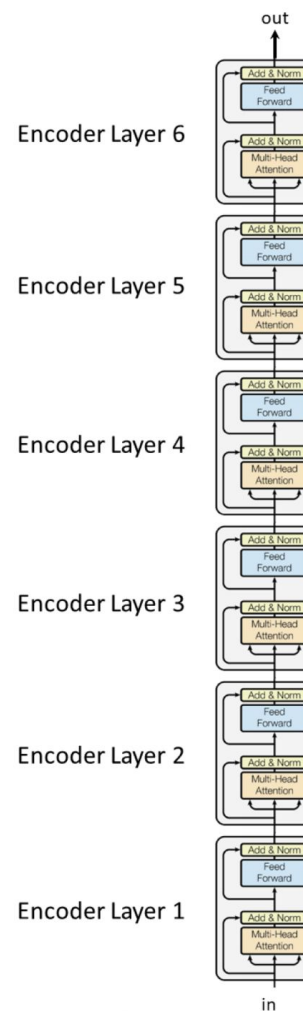
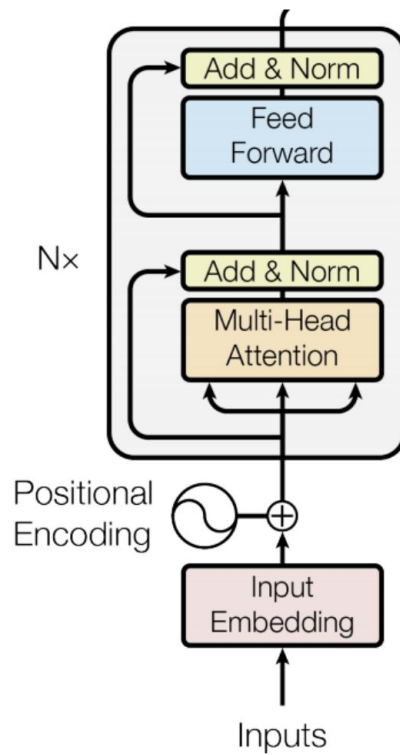
Residual connection



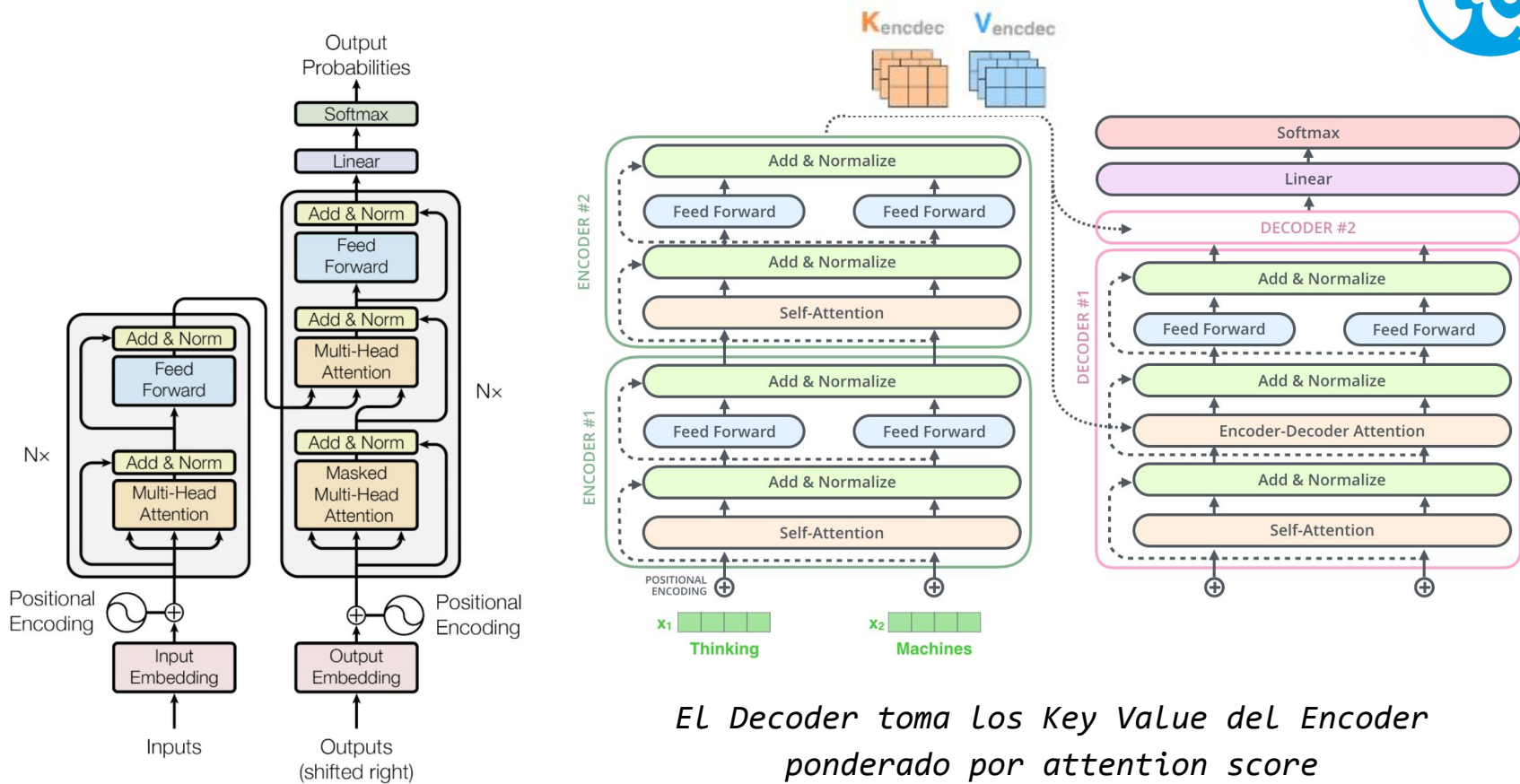
Permite que parte de la info del embedding de entrada se propague en en la salida de esa layer



Transformer stack Encoders



Transformer Encoder & Decoder



El Decoder toma los Key Value del Encoder ponderado por attention score (idem attention Seq2Seq)