
# *NLP*

## Celdas con Attention & Transformers

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

Esp. Ing. Hernán Contigiani
hernan4790@gmail.com

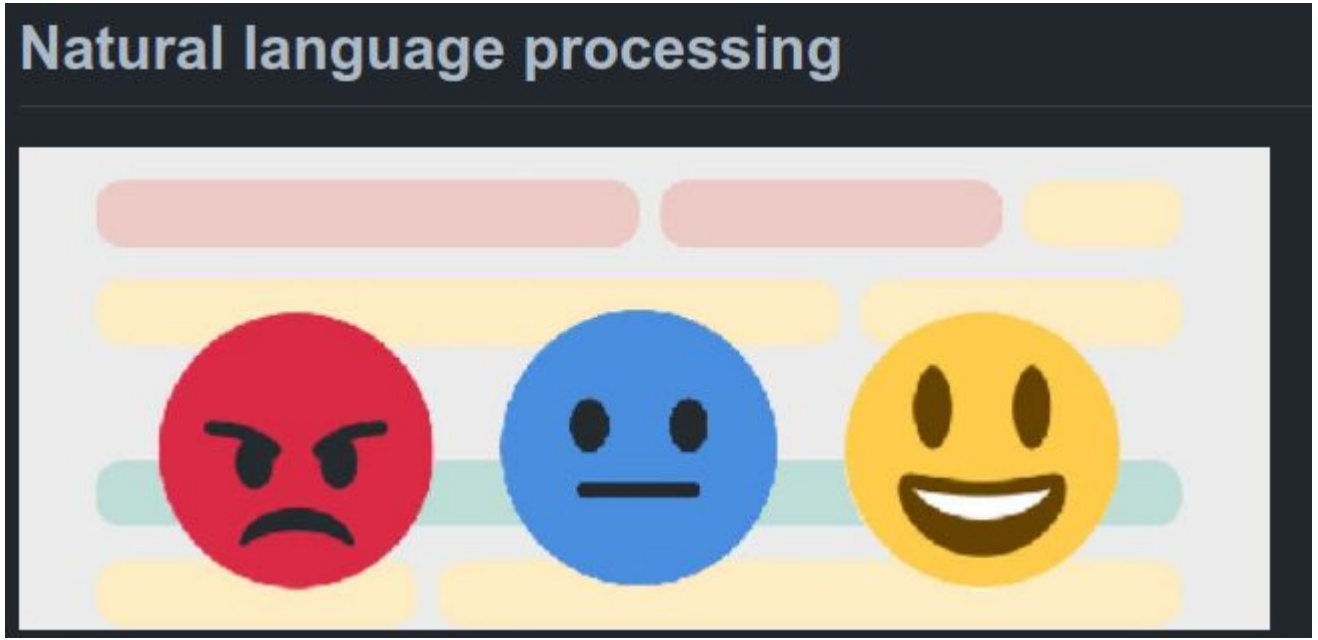
# Programa de la materia

**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

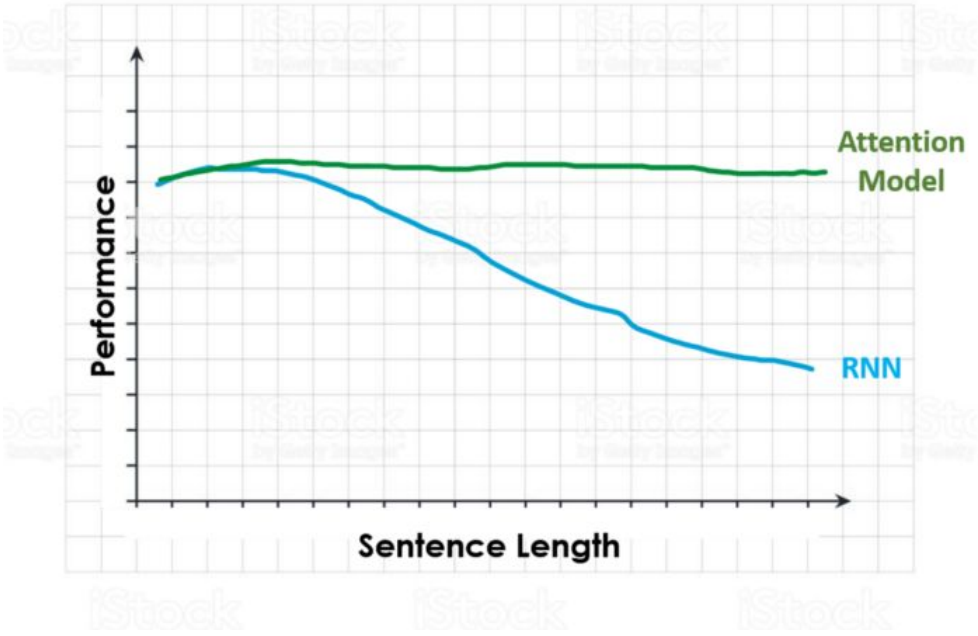
ELMo
2017

Transformers
2017/2018

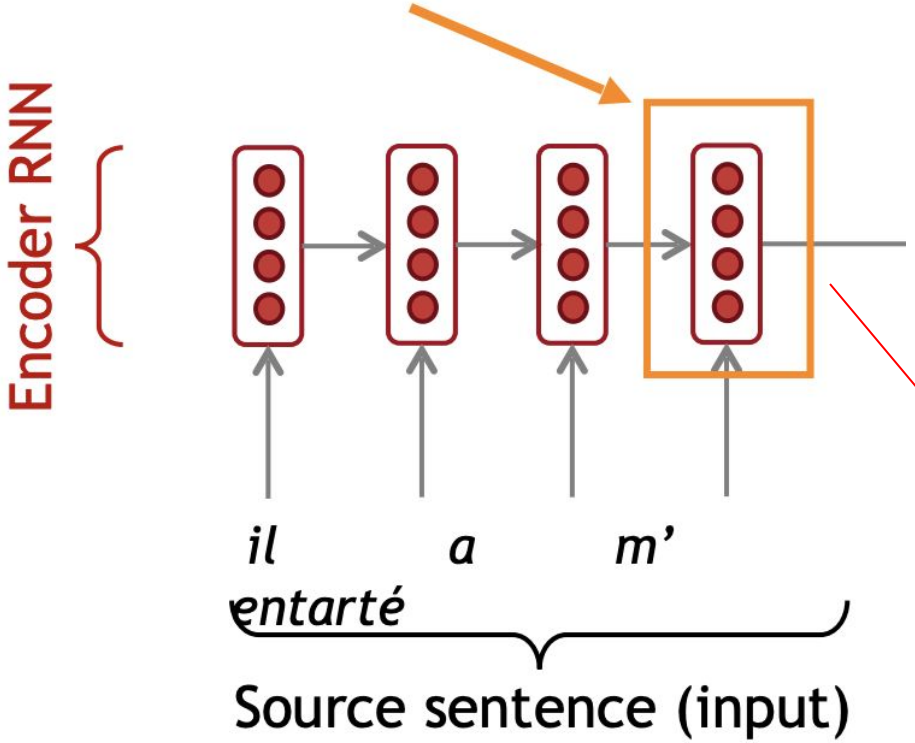
BERT
2019

# 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



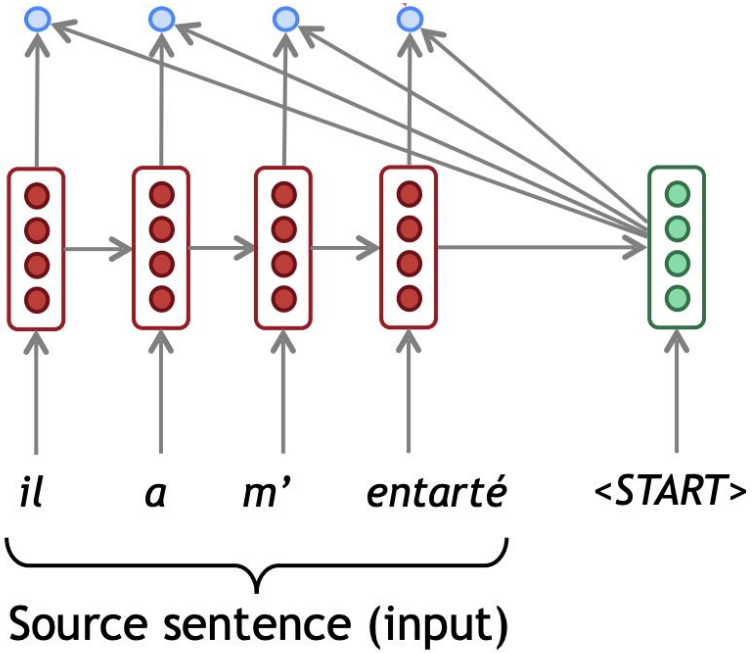

Toda la información de la secuencia de entrada queda condensada en este hidden layer

# Attention 2014/2015

LINK

Attention scores

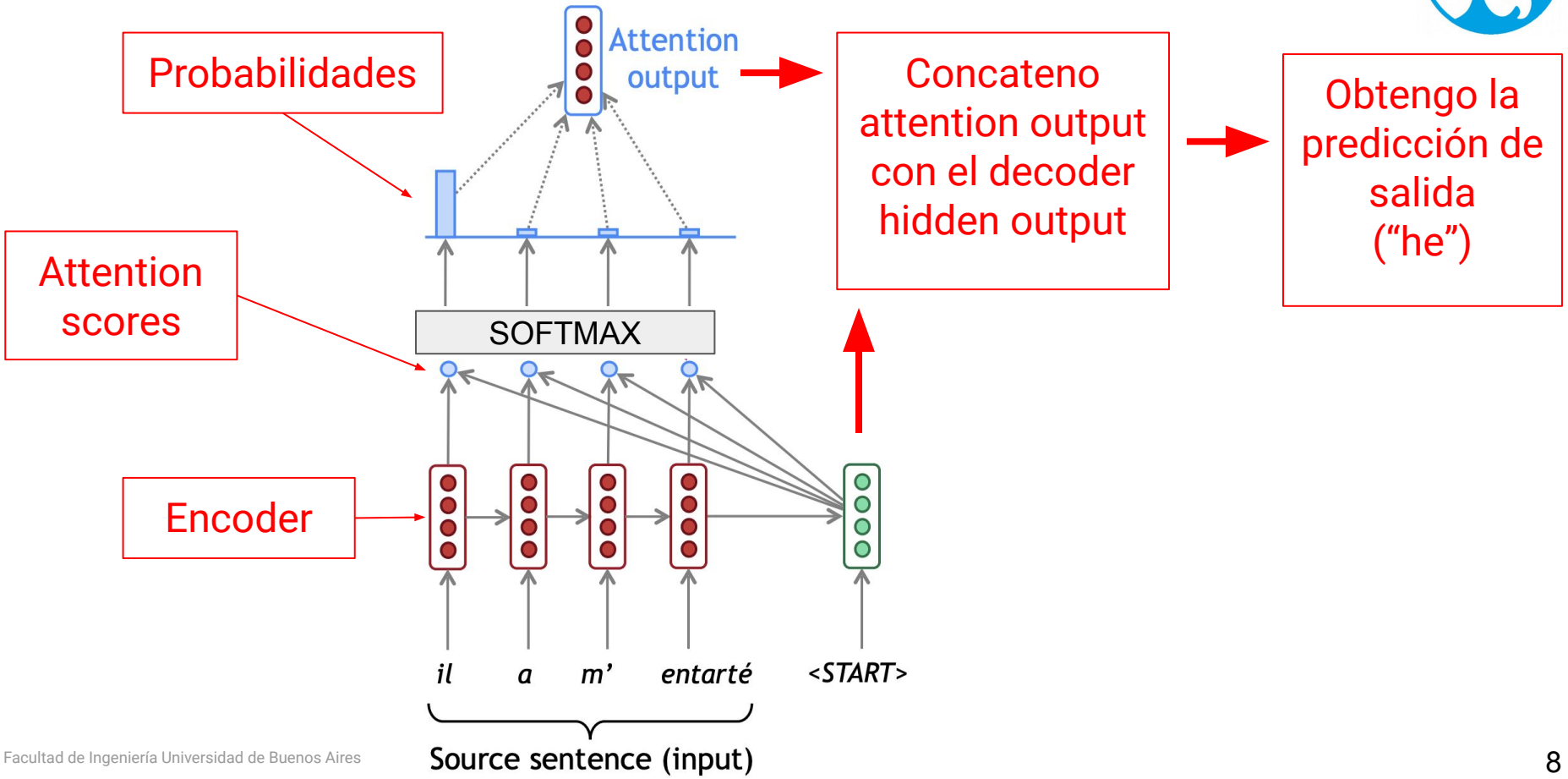
Encoder




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)

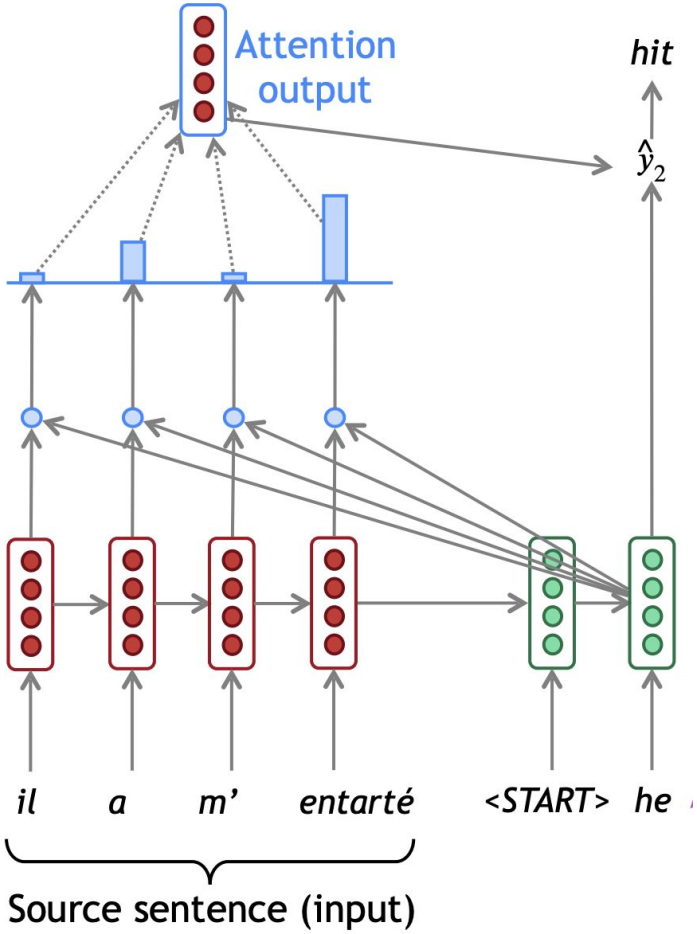
# Attention output



# Attention output

LINK

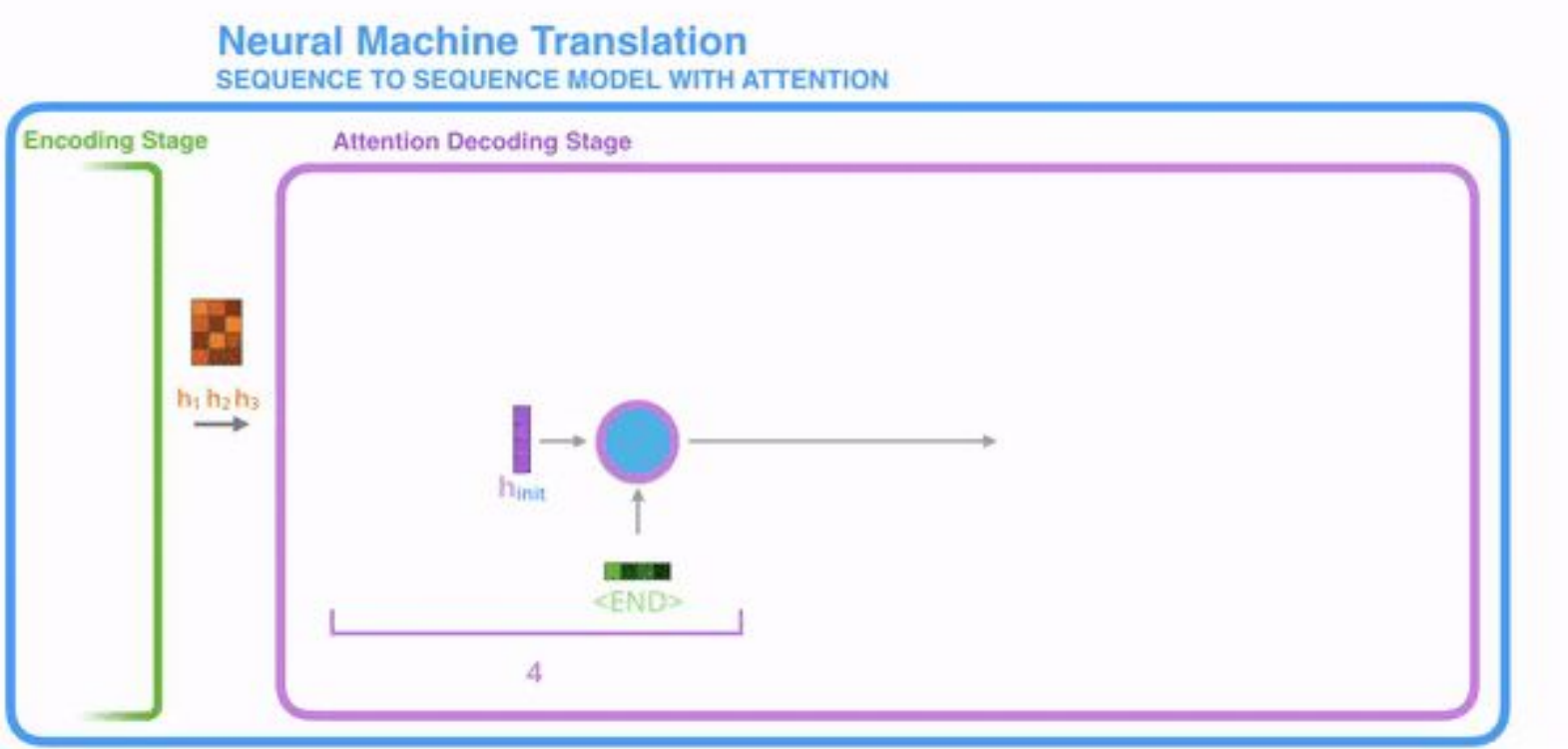
# Attention Seq2Seq



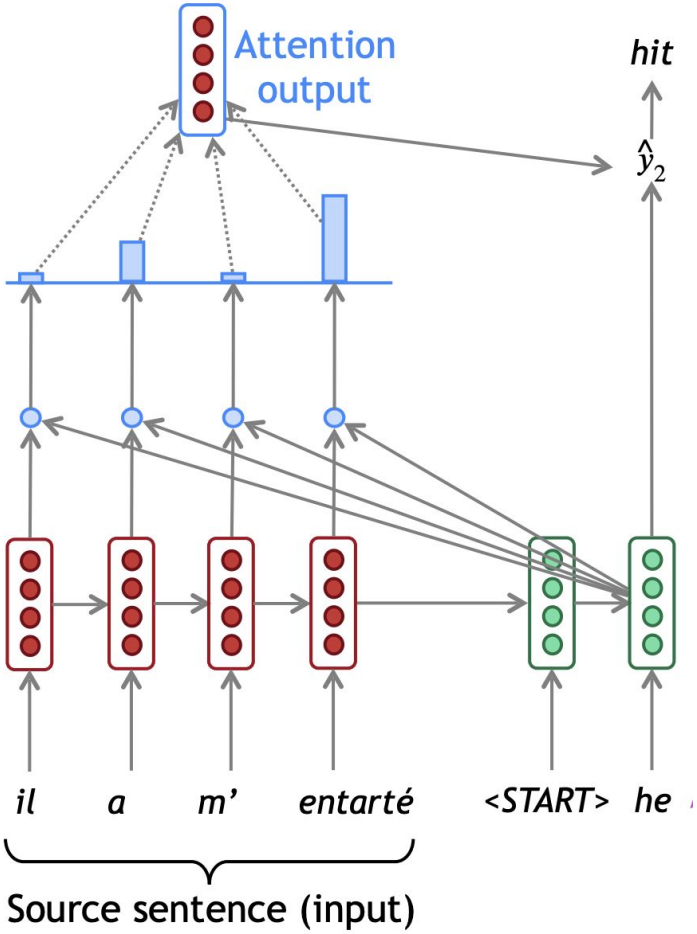

Repito el procedimiento para cada etapa del decoder

# Attention Seq2Seq

LINK



# 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.

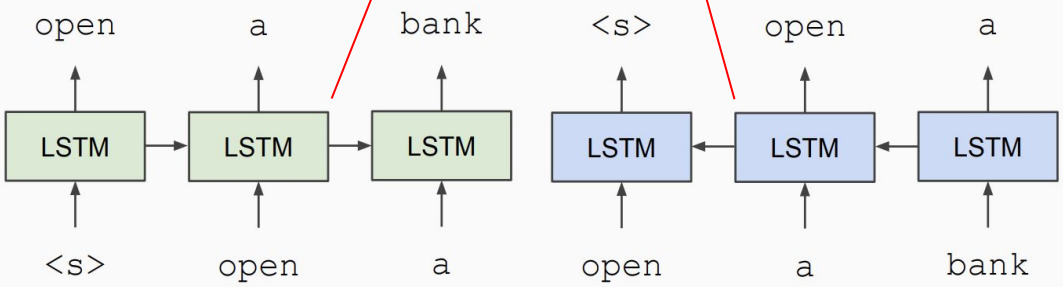
**Desventajas:**

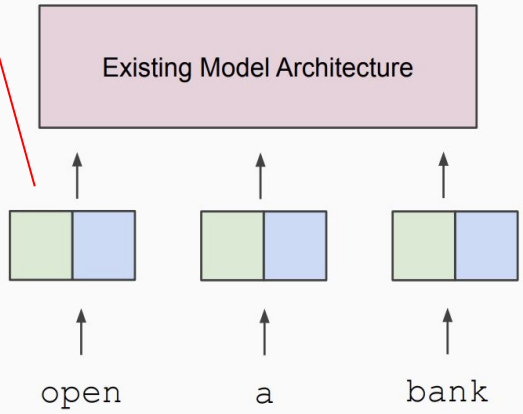
- Se agregan más operaciones no paralelizables al proceso.

# ELMo: Deep Contextual Word Embeddings (2017)

Entrenaron las redes LSTM en ambas direcciones (BRNN)

Durante la inferencia concatenan los embeddings

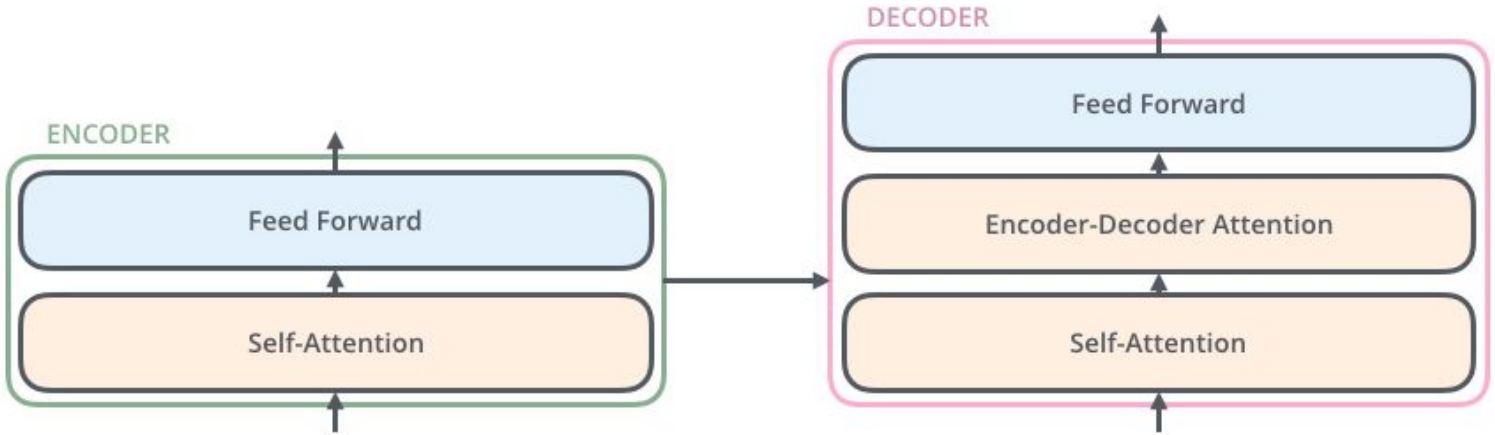


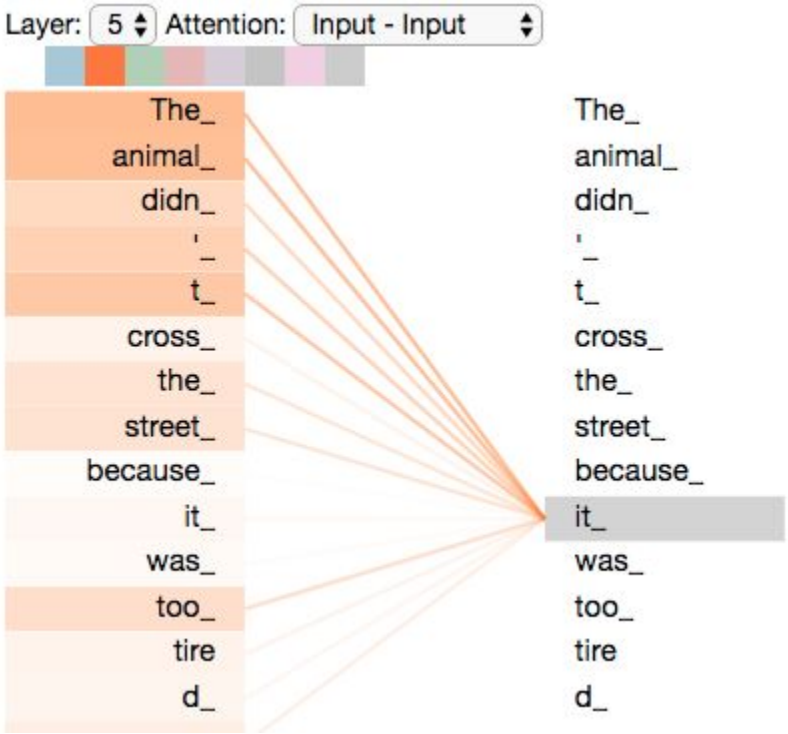

*En definitiva los embeddings que se utilizan de cada palabra se forman del contexto (las hidden layers), no son únicos como sucede en Glove o Fasttext*

# Transformers 2017/2018 LINK LINK

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



# Deseamos reforzar el análisis semántico

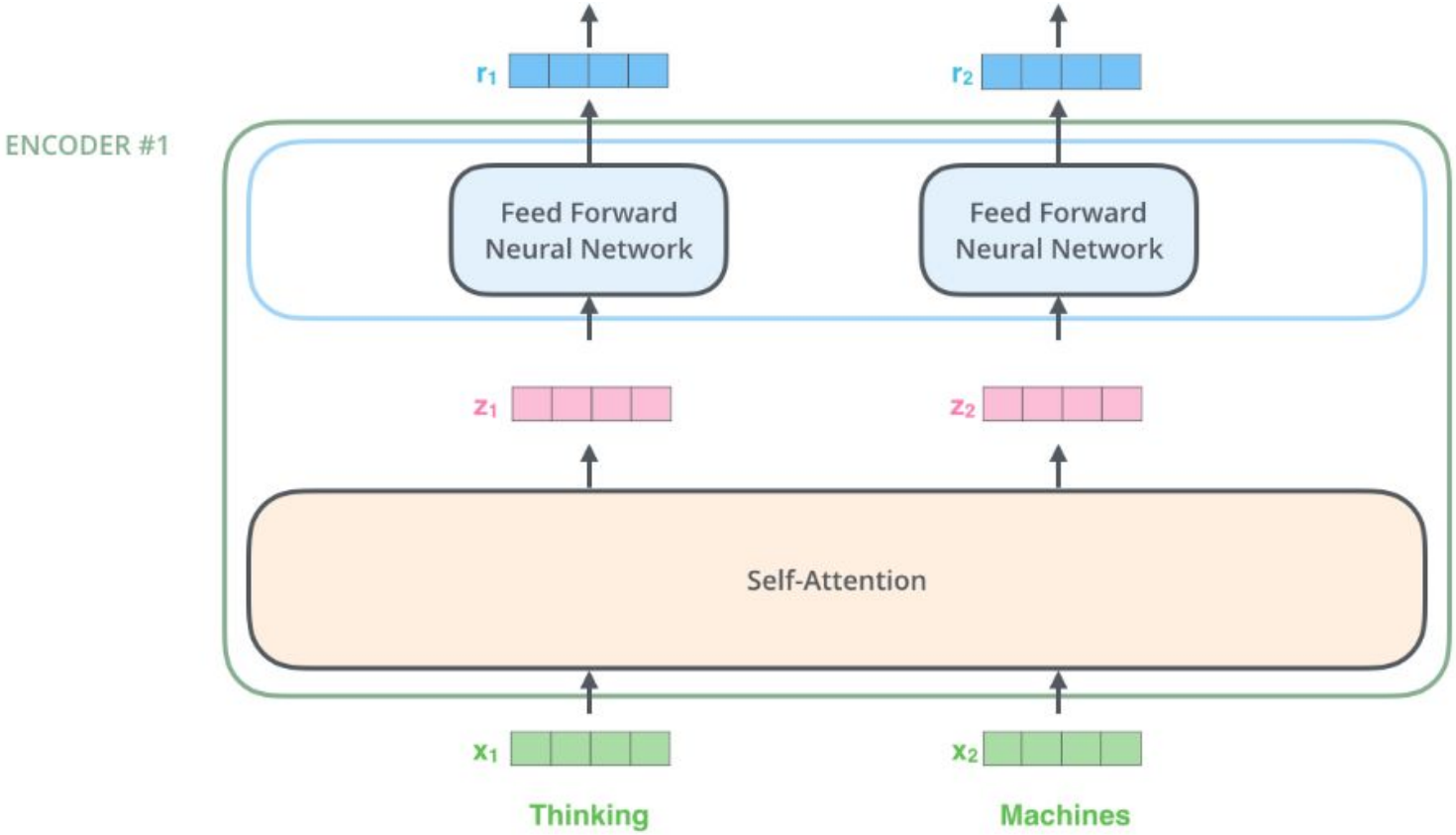



¿"It" se refiere a "animal" o "street"?

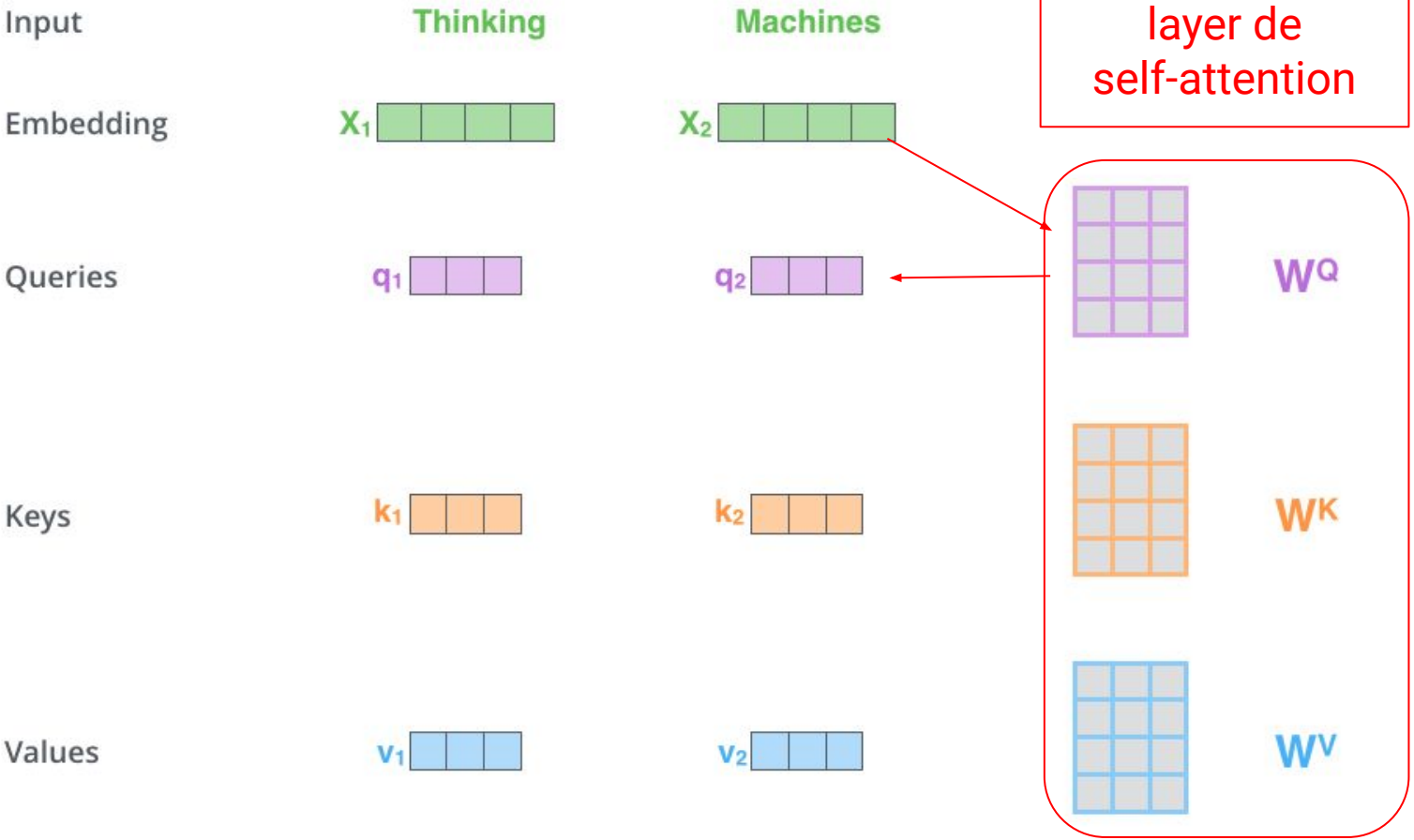
"The animal didn't cross the street because it was too tired"

# Self-attention - corazón del transformer

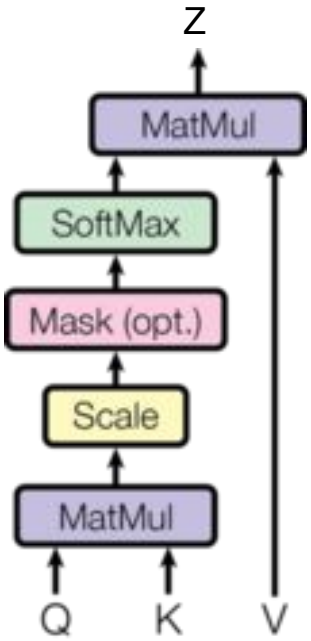
*Transforma todas las palabras de entrada en vectores (es posible paralelizar).*



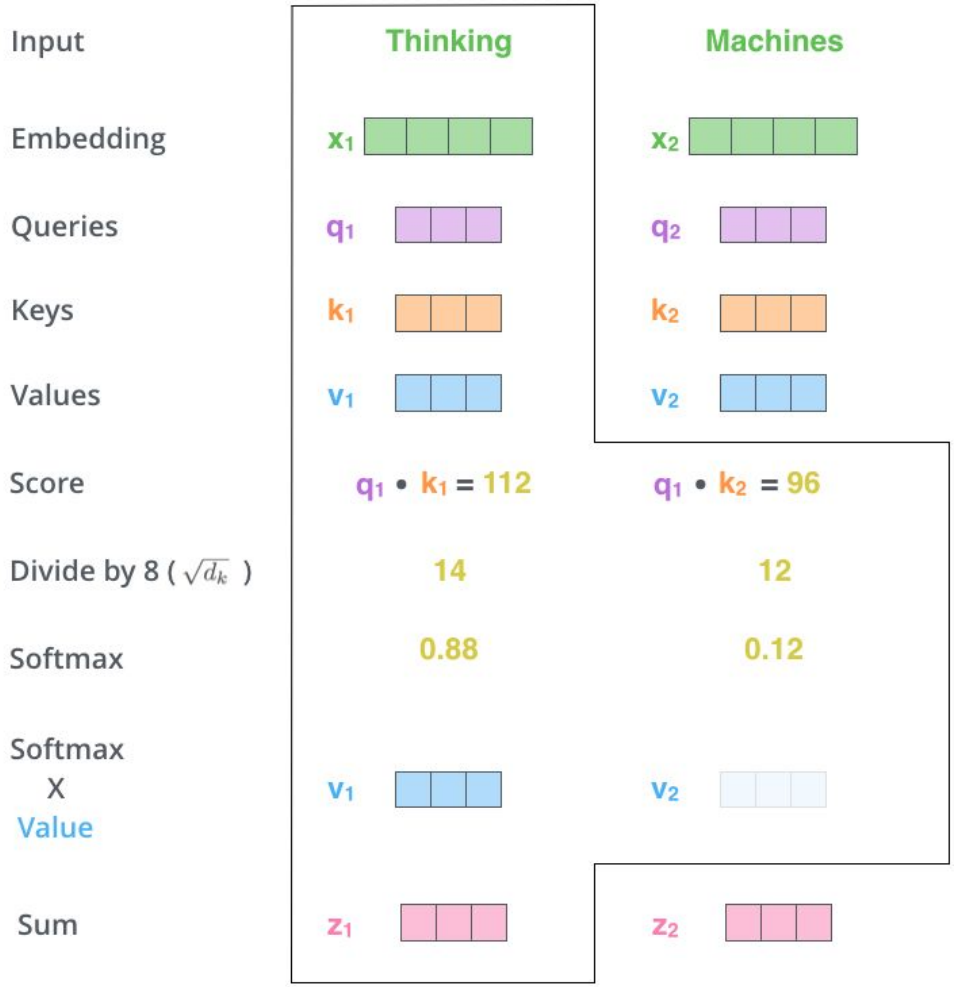
# Self-attention



# Self-attention

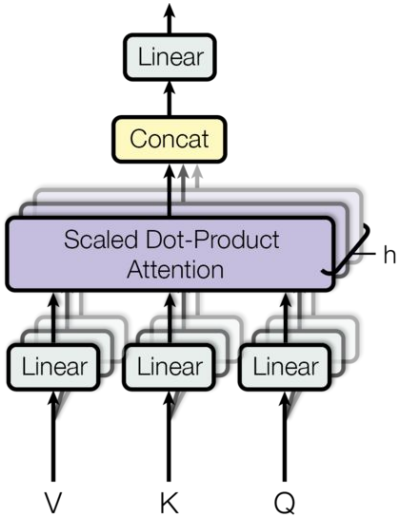



LINK PAPER (pytorch)



# Multi head Self-attention

## “The Beast With Many Heads”




1) This is our input sentence*

2) We embed each word*

Thinking Machines

X

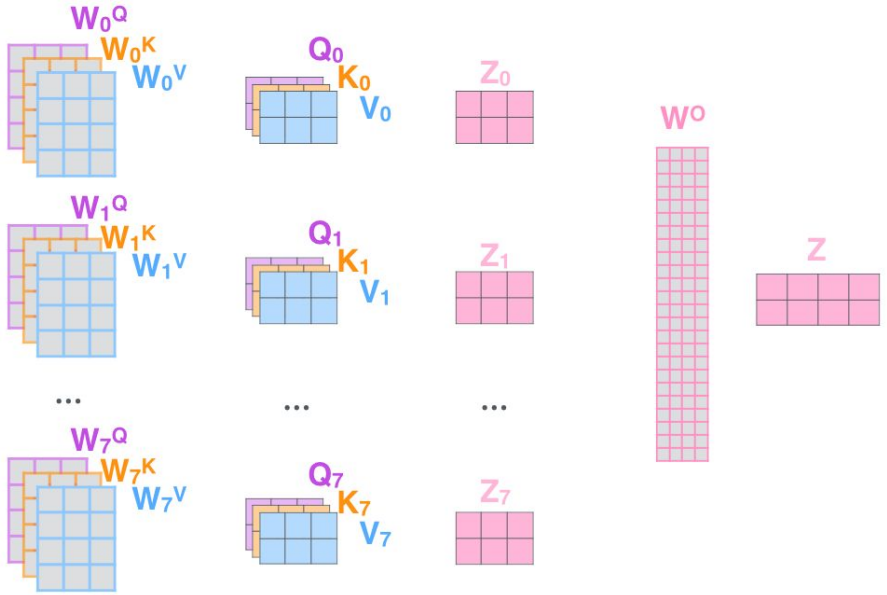
* In all encoders other than #0, we don’t need embedding. We start directly with the output of the encoder right below this one

R

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

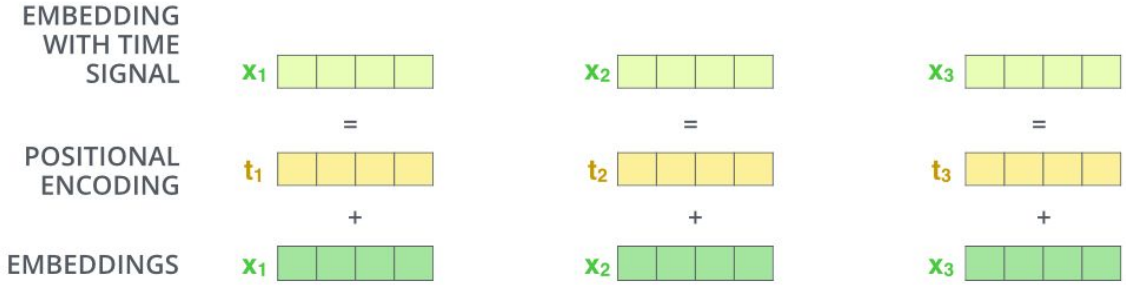


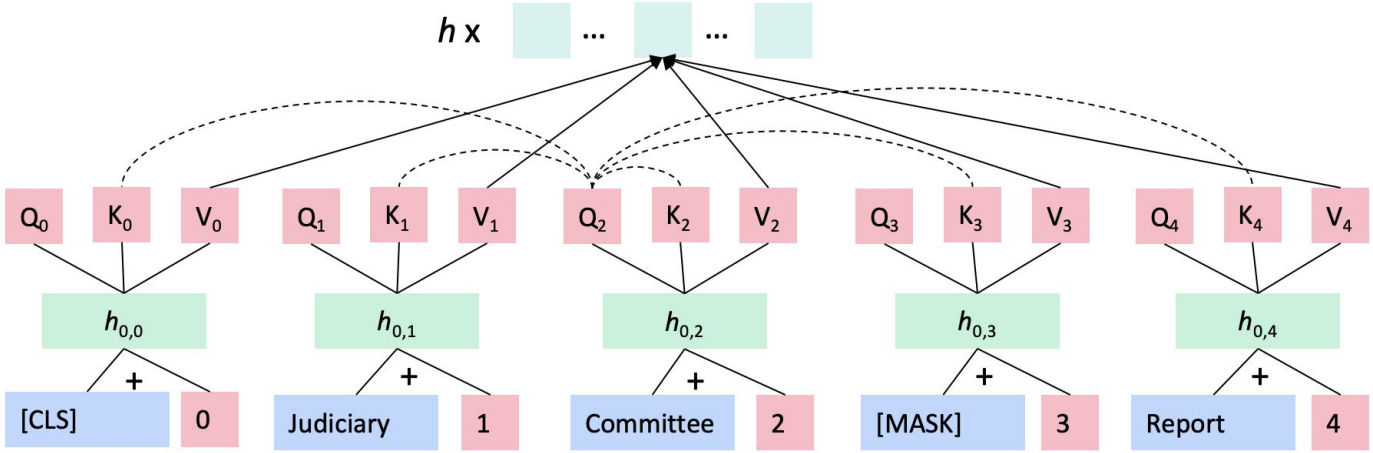

*En cada stack de self-attention obtenemos "N" vectores representados con diferentes matrices*

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

# Positional encoding

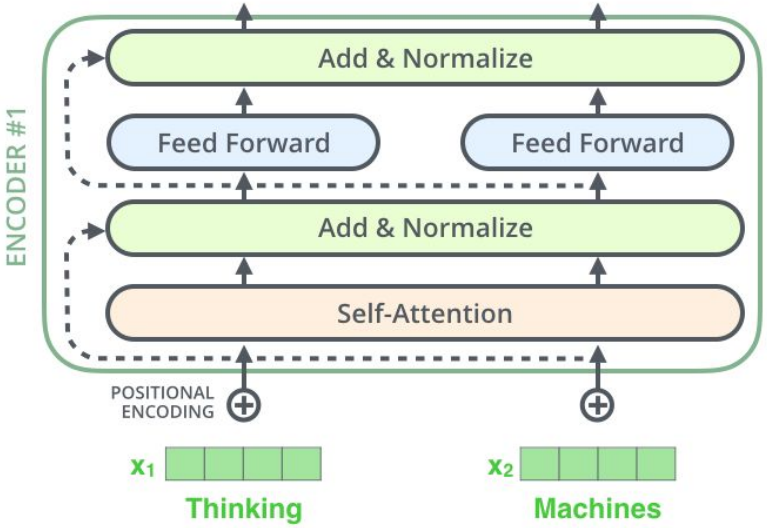
*Al embedding de entrada mezclarlo con un vector relativo a su posición en la secuencia*

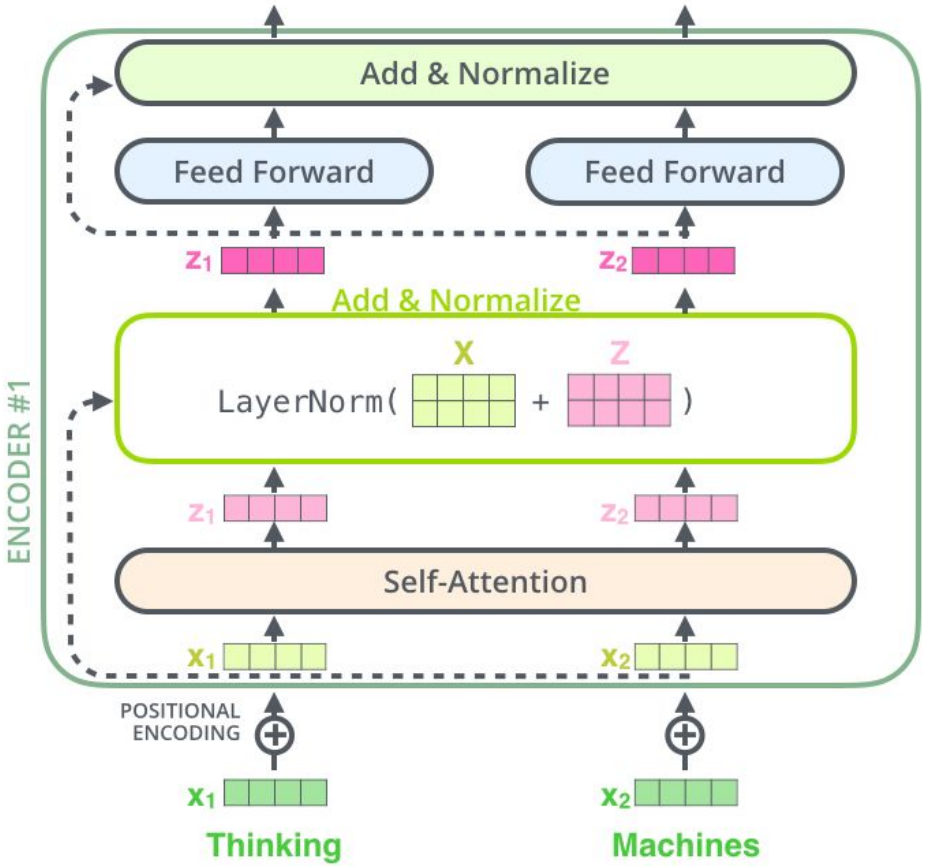


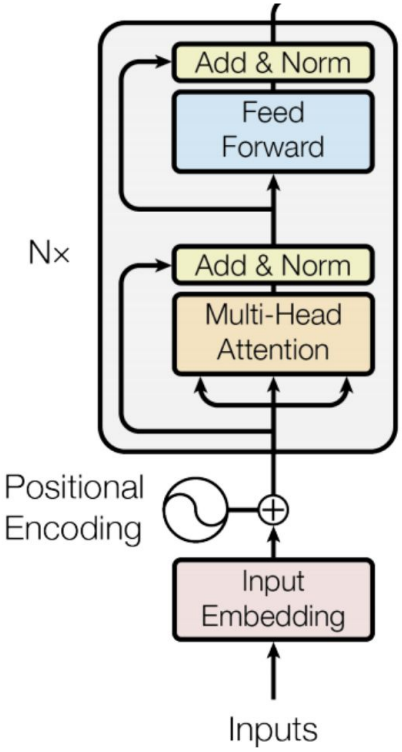
# Residual connection

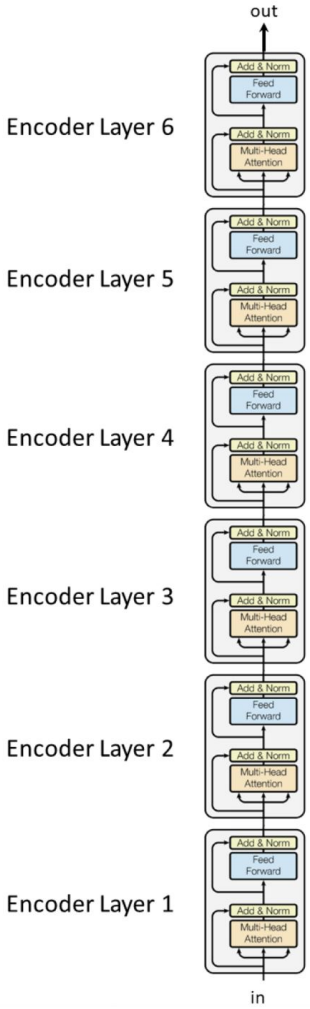
*Permite que parte de la info del embedding de entrada se propague en en la salida de esa layer*



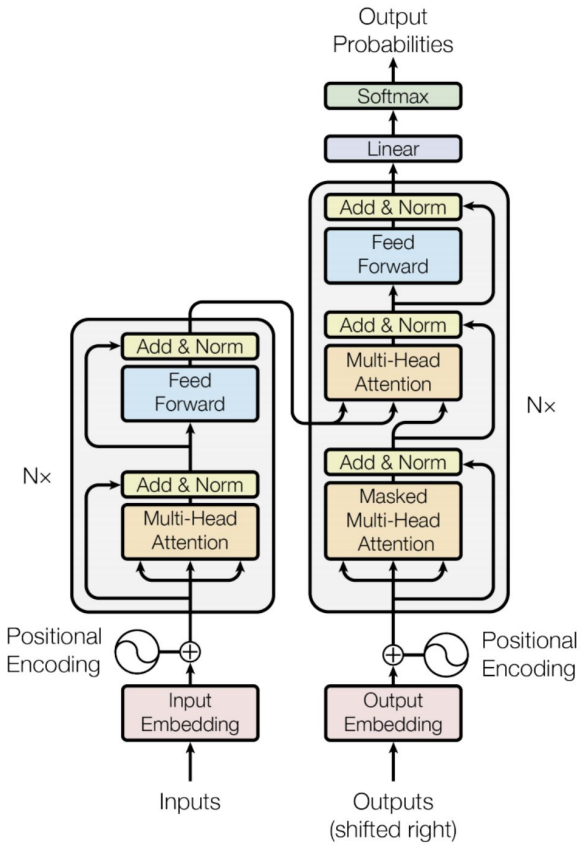



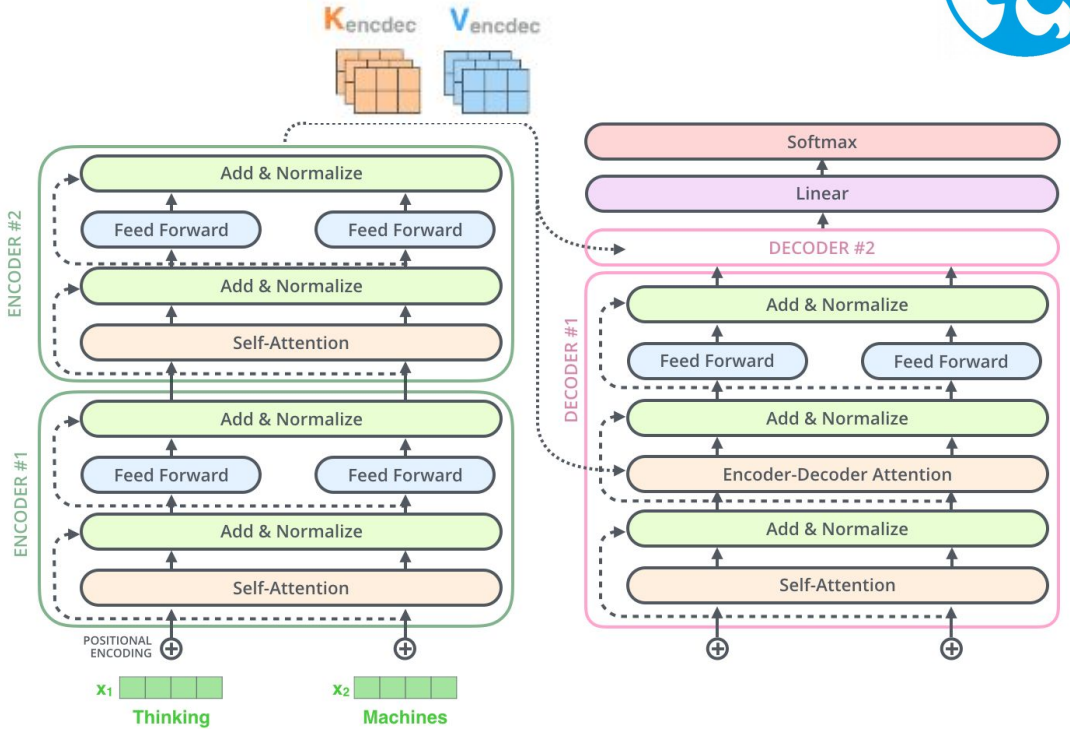
# Transformer stack Encoders



# Transformer Encoder & Decoder



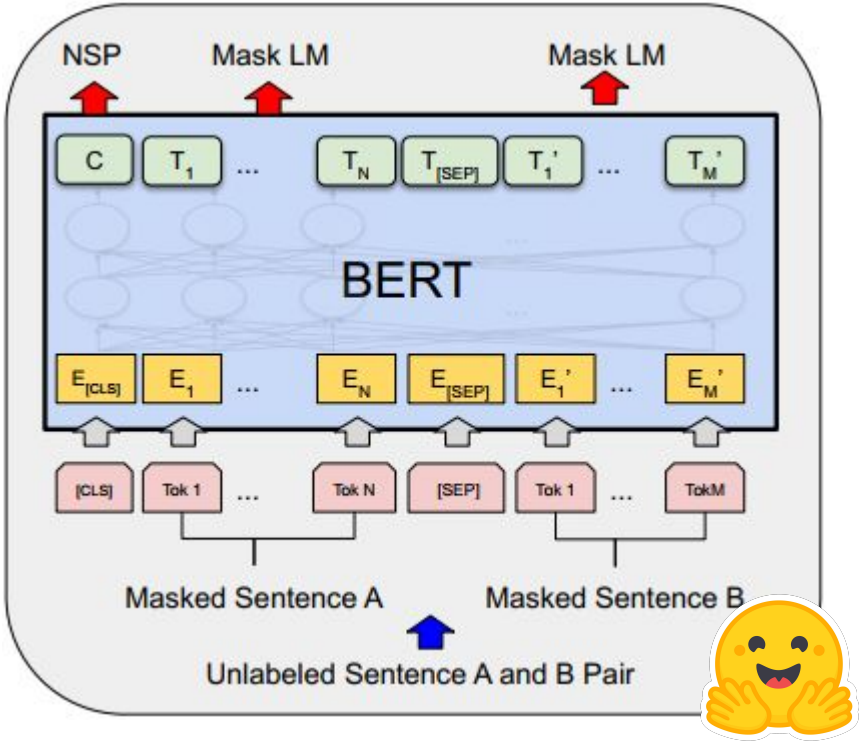




*El Decoder toma los Key Value del Encoder ponderado por attention score (idem attention Seq2Seq)*

# BERT (2019)

LINK LINK LINK

## “Pre-training of Deep Bidirectional Transformers”




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 nó con la primera.

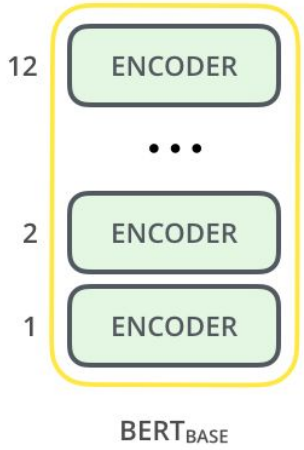
De forma aleatoria en cada inferencia se ocultó una palabra de cada secuencia (masked)

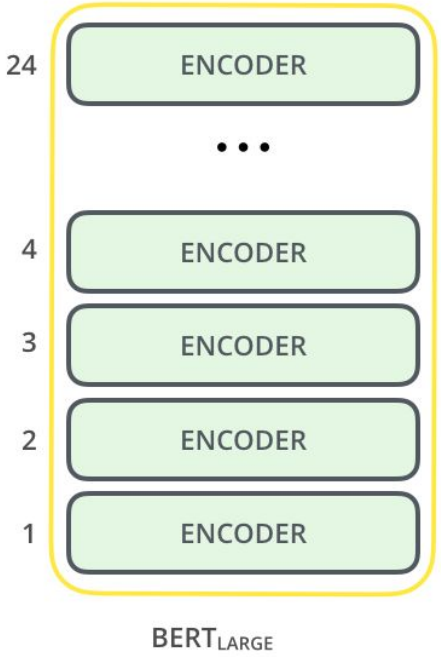
[CLS] Hermoso día el [MASK] hoy [SEP]

# BERT base y large

Bert base tiene 12 encoders layer

Bert large tiene 24 encoders layer

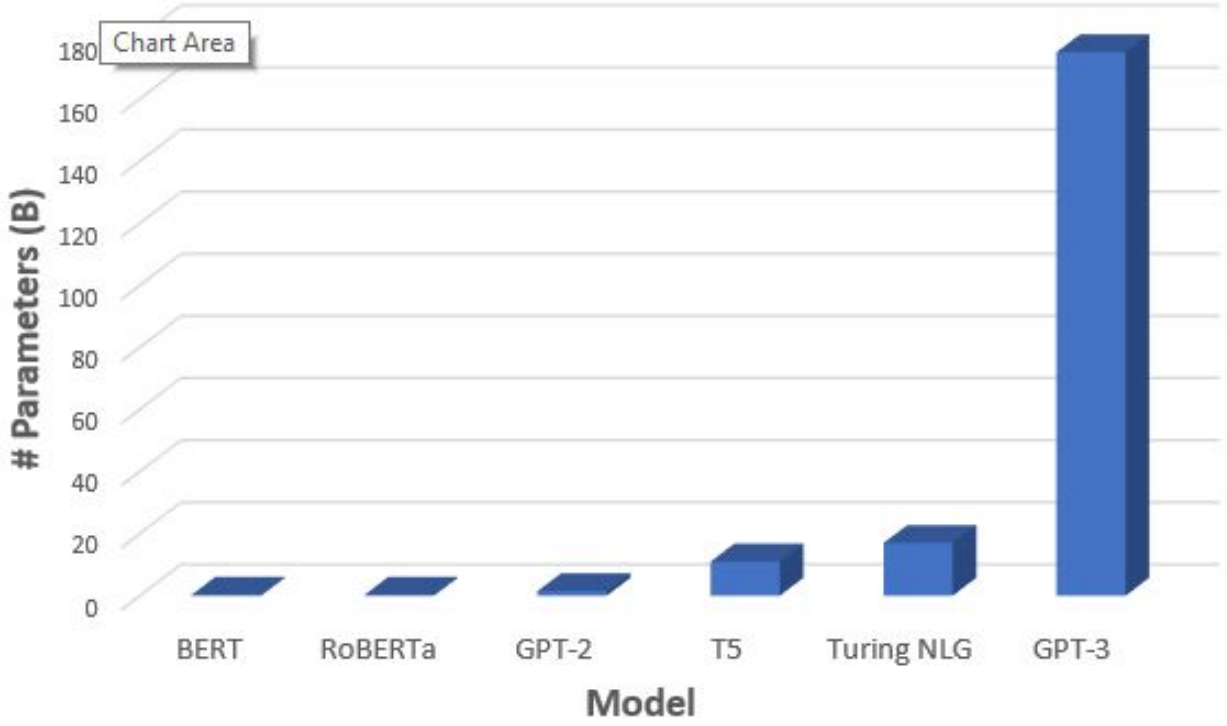


# BERT vs los demás

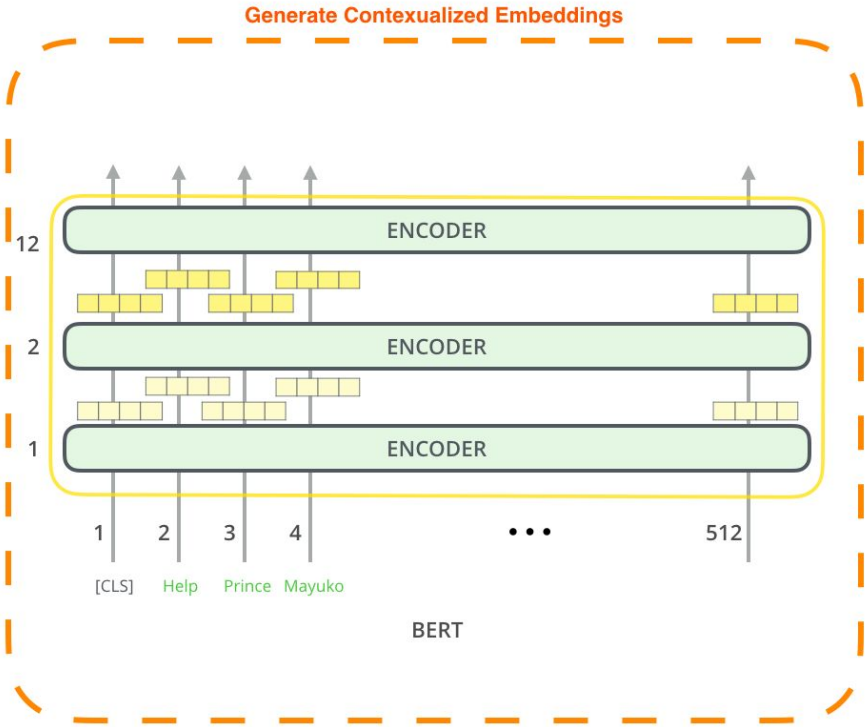
Bert (base) tiene 110 millones de parámetros

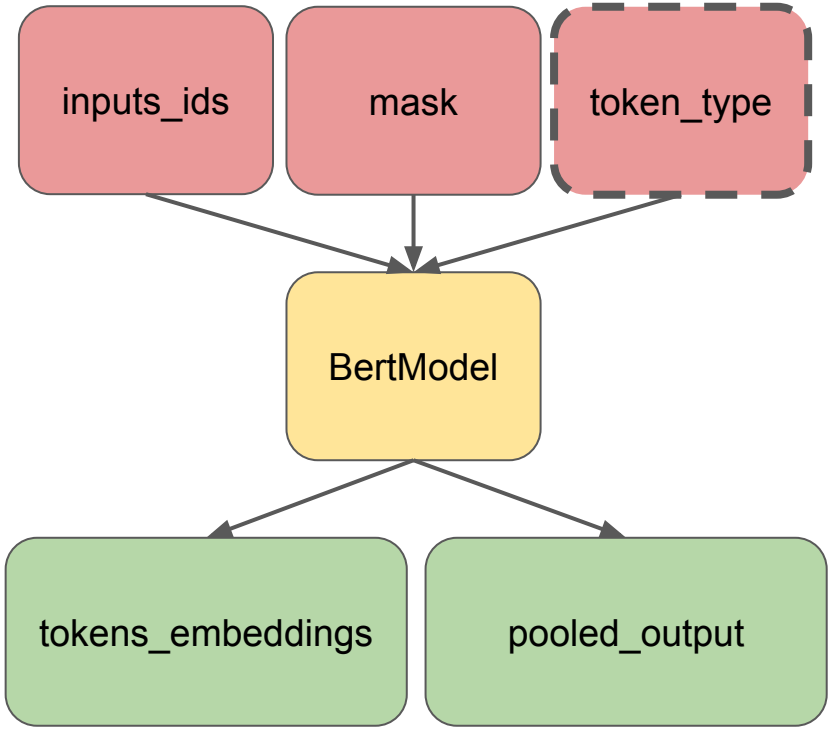
~54 horas de entrenamiento en Google TPUs



# Conociendo a BETO (BERT español)

# BERT - Embeddings contextualizados



# BERT - Embeddings contextuales (simil ELMo)

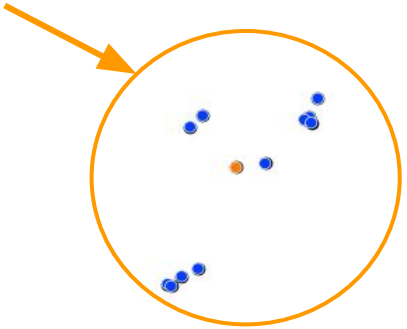
Link al Colab

LINK

# BERT - Summary Text (resumen texto)

LINK

Cluster que representa
a todo el texto



**Se calcula el centroide
(punto naranja)**

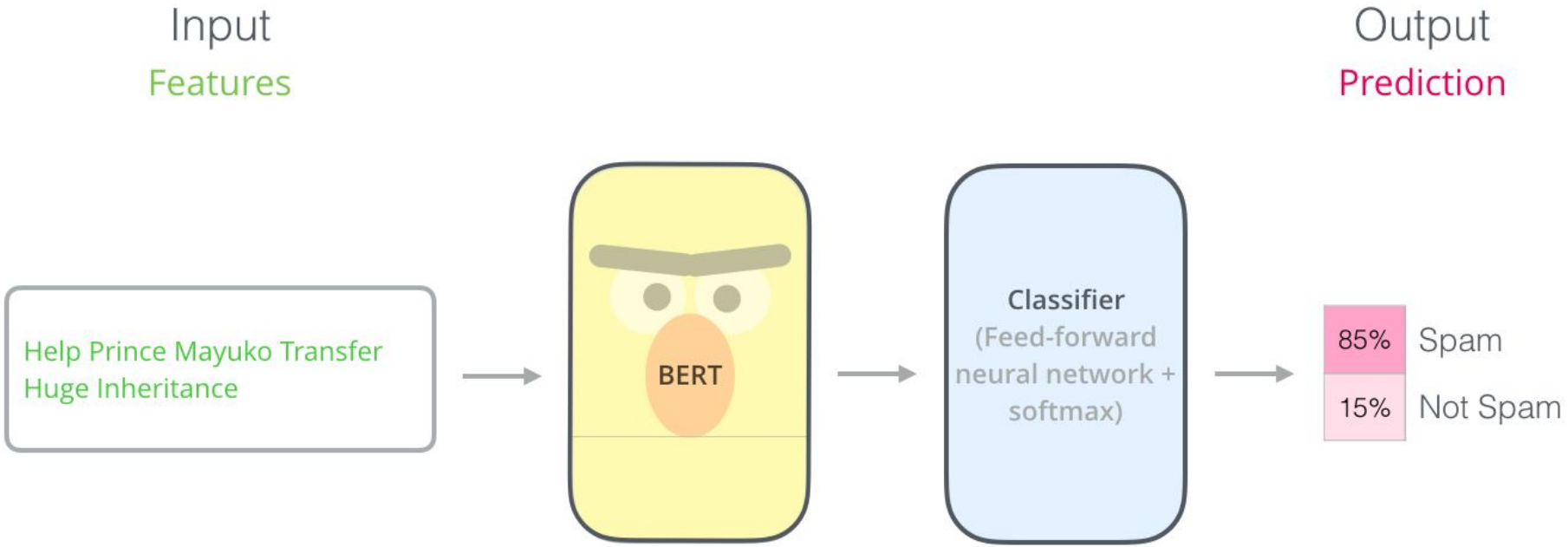
**Se mide qué sentencias están cerca al
centro y se arma un nuevo texto**

# BERT - Resumen de texto

Link al Colab

LINK

# BERT - Classifier (sentiment analysis)



# Fine tuning

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

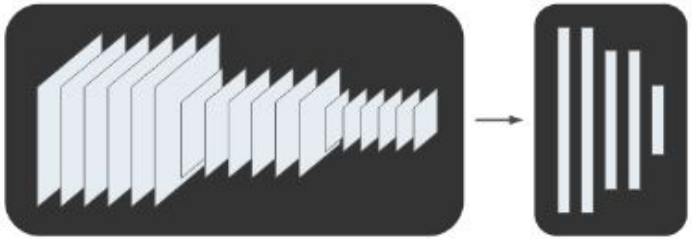
*“En el proceso de fine-tuning, ya habiendo realizado un primer entrenamiento, se realiza un ajuste fino de todo el modelo, incluyendo las capas pre-entrenadas”.*

Entrenamiento grueso

Pesos de BERT

Nuevas capas de clasificación

Entrenamiento fino



Entrenamiento punta a punta

***Menor learning rate***

# BERT - Sentiment analysis (críticas de Apps)

# BERT - Sentiment analysis (IMDB dataset)

Link al Colab

LINK

# ¡Muchas gracias!