

# Aprendizaje Profundo

---

# Índice

---



Aprendizaje  
profundo



Modelos de  
lenguaje



Transfer learning

# Introducción

---

## Datos estructurados

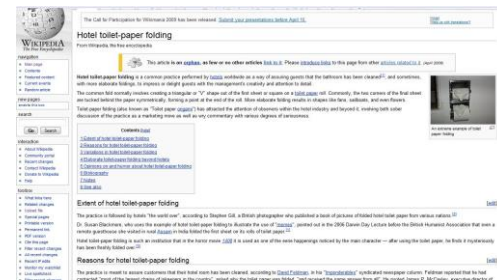
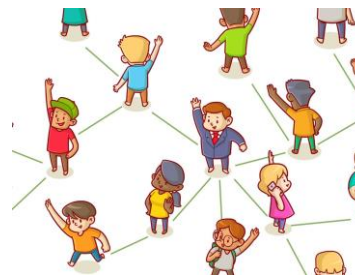


# Introducción

## Datos estructurados

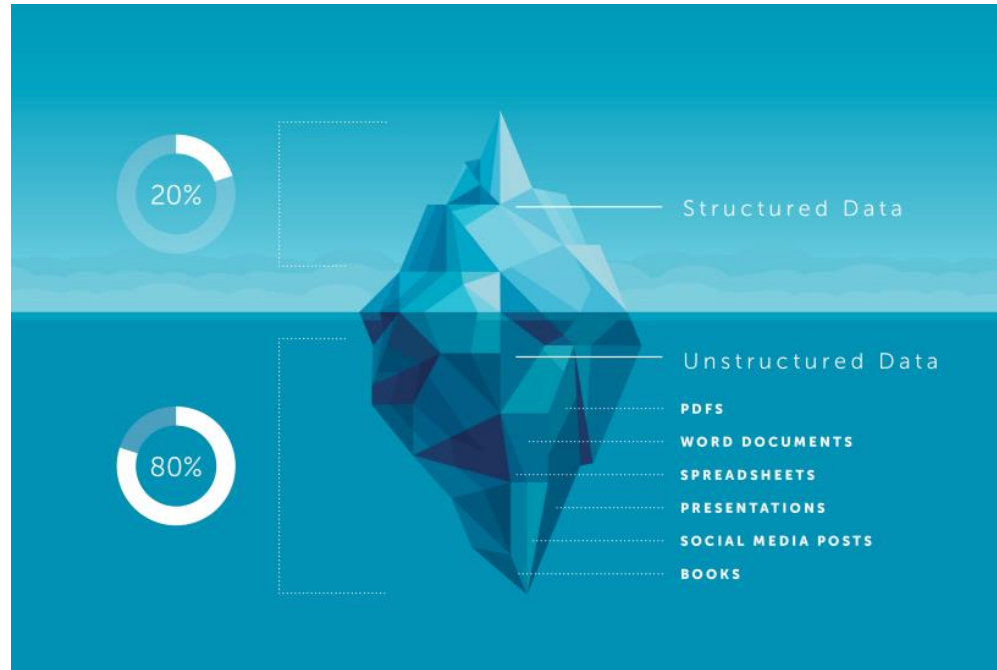


## Datos no estructurados



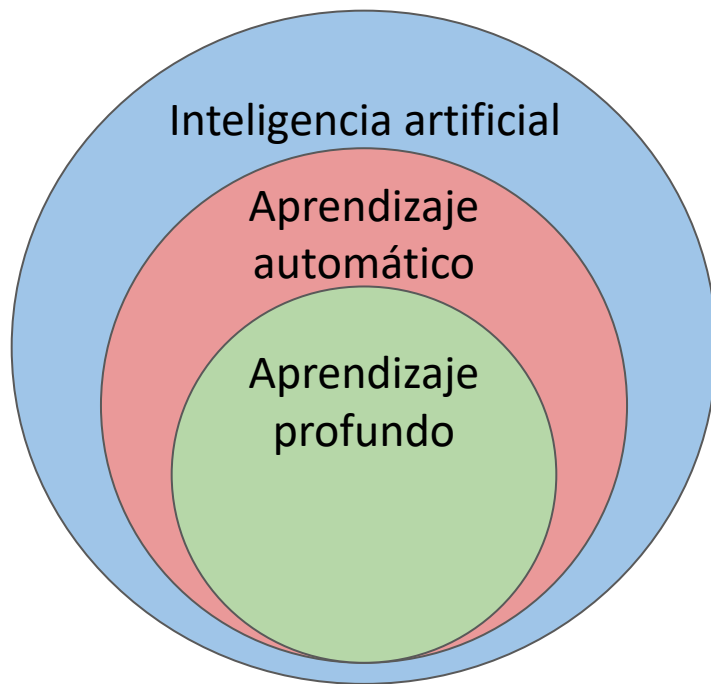
# Introducción

---



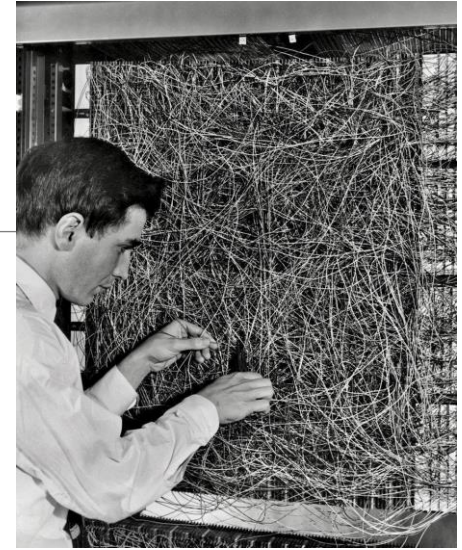
# Aprendizaje profundo

---



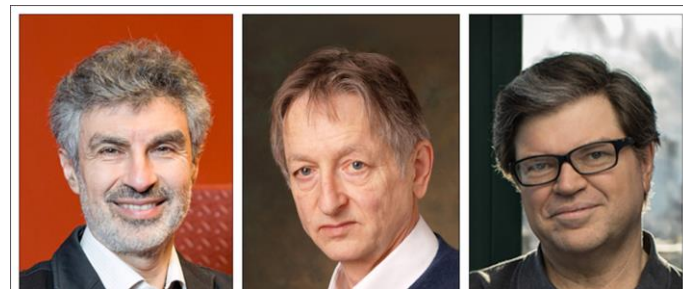
# Redes neuronales

- 1943: McCulloch & Pitts
  - Modelo matemático de una neurona
- 1957: Rosenblatt
  - Perceptrón
- 1974-80: Primer invierno
  - Perceptrón no modela funciones simples
  - Se soluciona con múltiples capas
- 1986: Parallel distributed processing y back propagation



# Redes neuronales

- 1987-93: Segundo invierno
  - Dos capas para aproximador universal
  - No es útil en la práctica (demasiado grandes y lentas)
- 2006: Hinton & Salakhutdinov
  - Deep learning
- 2019: Turing award



Yoshua Bengio, [Geoffrey Hinton](#), [Yann Lecun](#)



# ¿Qué diferencia hay?

## Método automático tradicional

---



Extractor de  
descriptores

Entrenamiento  
de modelo

# ¿Qué diferencia hay?

## Método tradicional



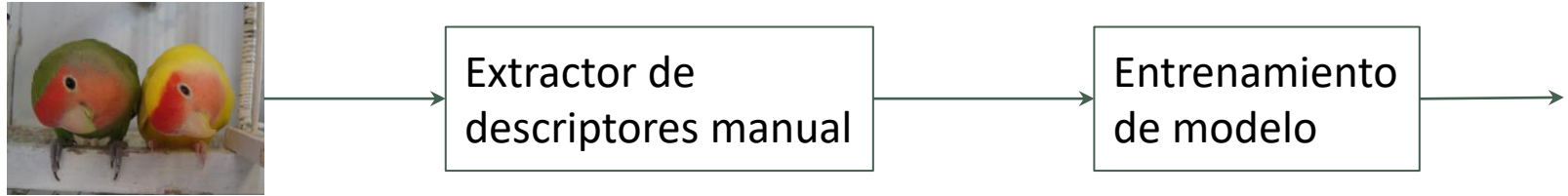
Extractor de  
descriptores

Entrenamiento  
de modelo

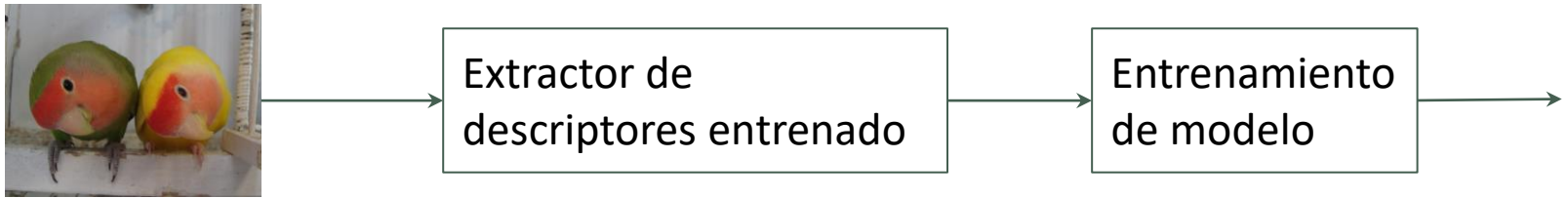


# ¿Qué diferencia hay?

## Método tradicional



## Aprendizaje profundo - aprendizaje de la representación



# Representación jerárquica

---

- Representación jerárquica con niveles incrementales de abstracción
- Cada nivel transforma los descriptores

# ¿Qué modelos son profundos?

---

Redes que aprenden una representación jerárquica de los datos

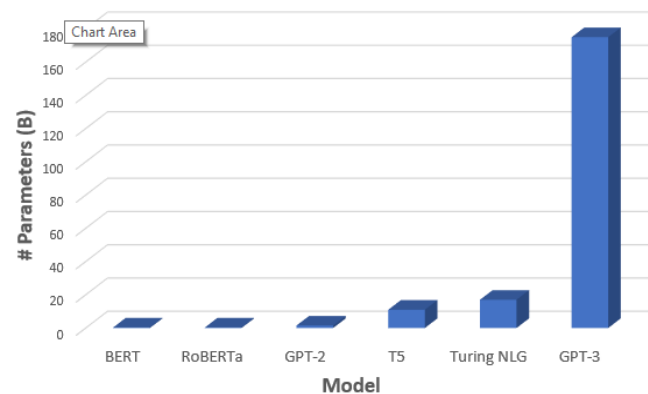
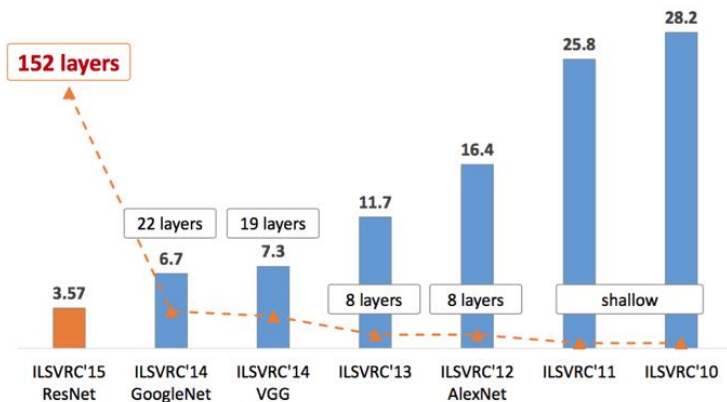
# ¿Qué ha cambiado?

---

- Más capas y más neuronas
- Para que funcione:
  - Más datos
  - Más potencia (GPUs, TPUs)
  - Variantes y mejoras de los algoritmos
  - Software libre

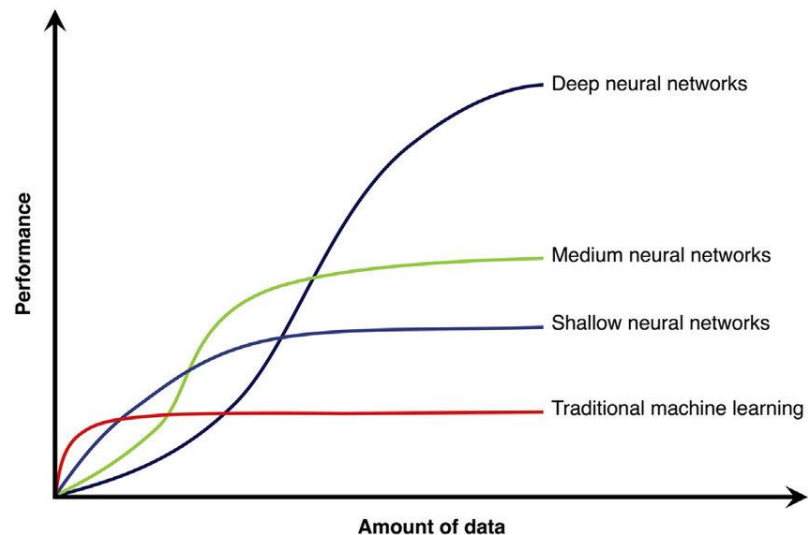
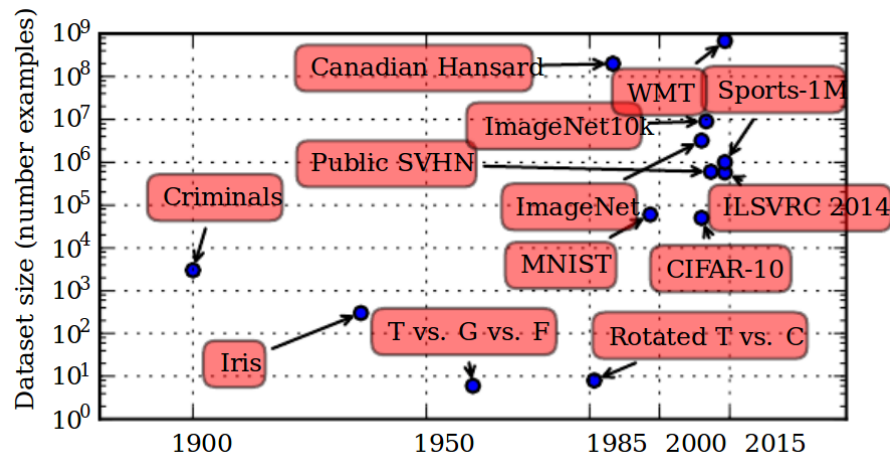
# ¿Qué ha cambiado?

## Más capas y más neuronas



# ¿Qué ha cambiado?

## Mas datos





# ¿Qué ha cambiado?

---

## GPUs:

- Resultados más rápidos
- Bajo consumo
- Bajo coste

# ¿Qué ha cambiado?

---

- EfficientNet: 128 núcleos TPUv3
- GPT-3: 285000 núcleos CPU y 10000 GPUs

# ¿Qué ha cambiado?

---

... pero veremos que con una GPU (gratuita) podemos crear buenos modelos

# ¿Qué ha cambiado?

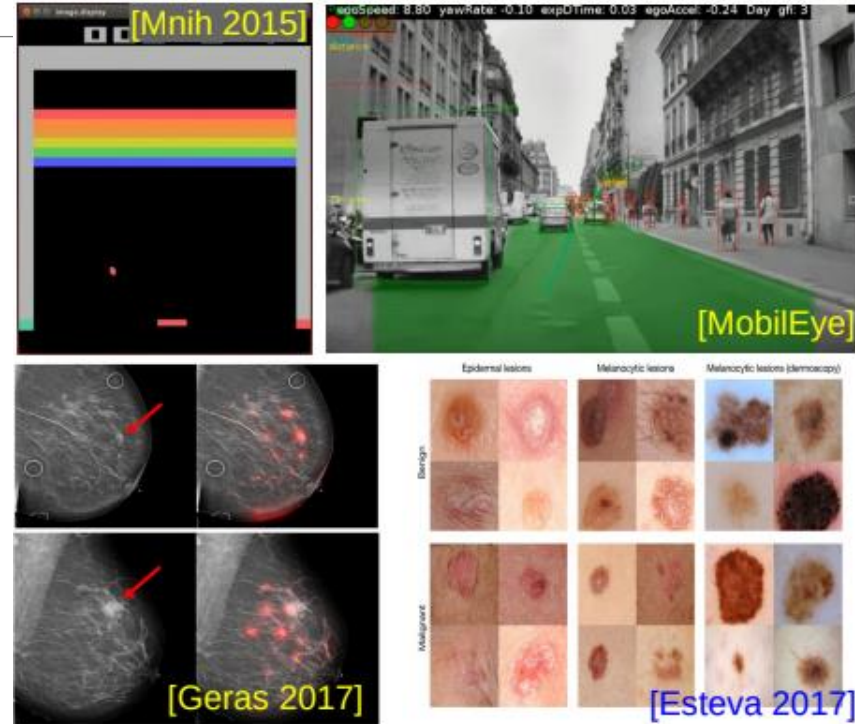
---

## Software libre:

- Tensorflow y Pytorch
- Keras, FastAI, Pytorch lightning
- HuggingFace
- Gran cantidad de repositorios en GitHub

# ¿Dónde funciona el aprendizaje profundo?

- Visión por computador
- Procesado lenguaje natural
- Medicina
- Biología
- Generación de imágenes
- Sistemas de recomendación
- Robótica
- ...



# Índice

---

1. Aprendizaje profundo
- 2. Modelos de lenguaje**
3. Transfer learning

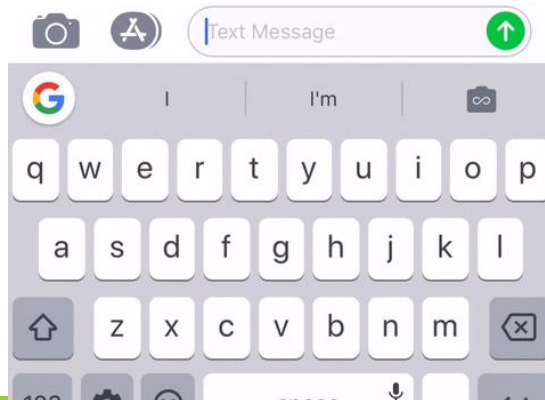
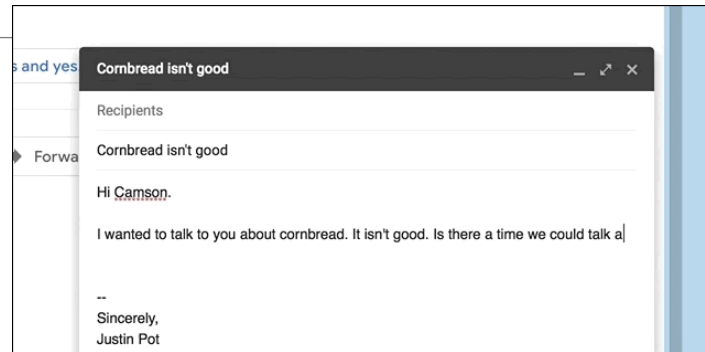
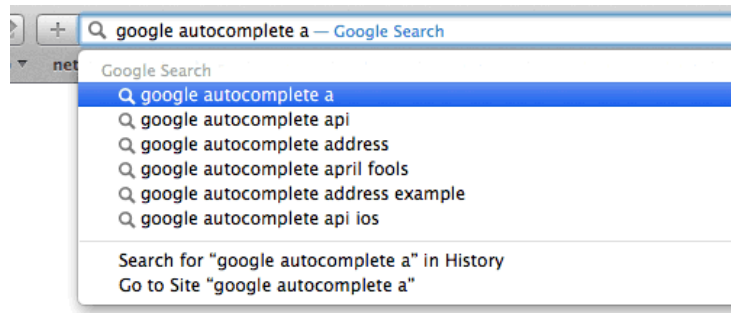
# Modelo de lenguaje

---

Los modelos de lenguaje estiman la probabilidad de distintas unidades lingüísticas

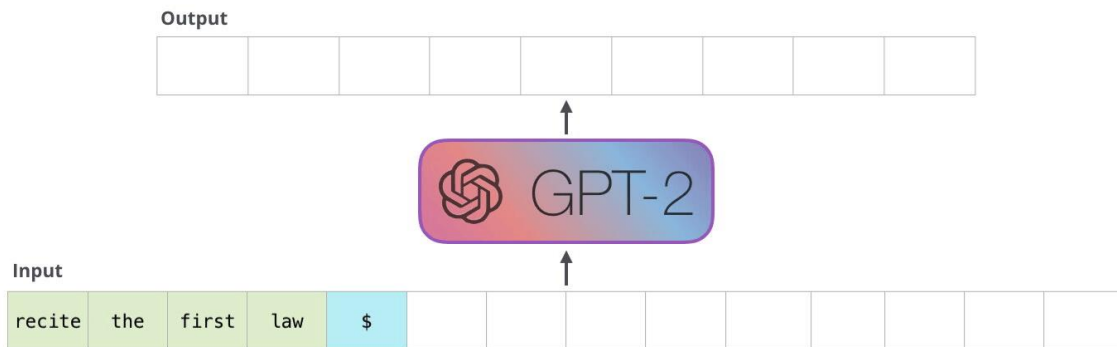
Un modelo que dada parte de una frase es capaz de predecir la siguiente palabra

# Modelo de lenguaje





# Modelos de lenguaje



# Modelos de lenguaje

---

Intentan predecir la siguiente palabra a partir del contexto

Para cenar voy a hacer \_\_\_\_\_

$P(\text{fajitas} \mid \text{Para cenar voy a hacer}) > P(\text{cemento} \mid \text{Para cenar voy a hacer})$

# Modelos de lenguaje

Texto

Tokenizaci  
ón

Modelo  
lenguaje

Probabilidades

Muestra

Para cenar voy

[1135,2337,1222]

...	...
a	0.1
en	0.05
hacer	0.0001
fajitas	0.0001
cemento	0.0001
croquetas	0.0001
...	...

a

# Modelos de lenguaje

Texto

Tokenizaci  
ón

Modelo  
lenguaje

Probabilidad

Muestra

Para cenar voy a

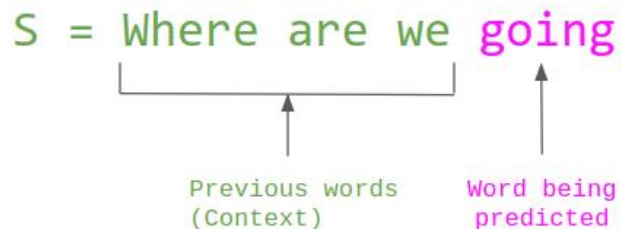
[1135,2337,1222,84  
36]

...	s	...
a		0.0001
en		0.0001
hacer		0.3
fajitas		0.0001
cemento		0.0001
croquetas		0.0001
...		...

hacer

# Modelos de lenguaje

## Estiman probabilidad de que una frase ocurra



$$P(S) = P(\text{Where}) \times P(\text{are} \mid \text{Where}) \times P(\text{we} \mid \text{Where are}) \times P(\text{going} \mid \text{Where are we})$$

# Perplexity (perplejidad)

---

Una medida para saber cómo de bien un modelo de probabilidad predice una muestra

La perplejidad nos dice si un conjunto de frases parecen escritas por personas y no por una máquina

Frases escritas por personas tienen una perplejidad baja, mientras que frases aleatorias tienen una perplejidad alta

# Perplejidad

---

Menor perplejidad = mejor modelo

Valores típicos entre 10 y 60

# Modelos de lenguaje

---

- Bert
- ULMFit
- ELMo
- T5
- XLNet
- GPT-2, GPT-3, GPT-4
- ...



# Modelos de lenguaje en español

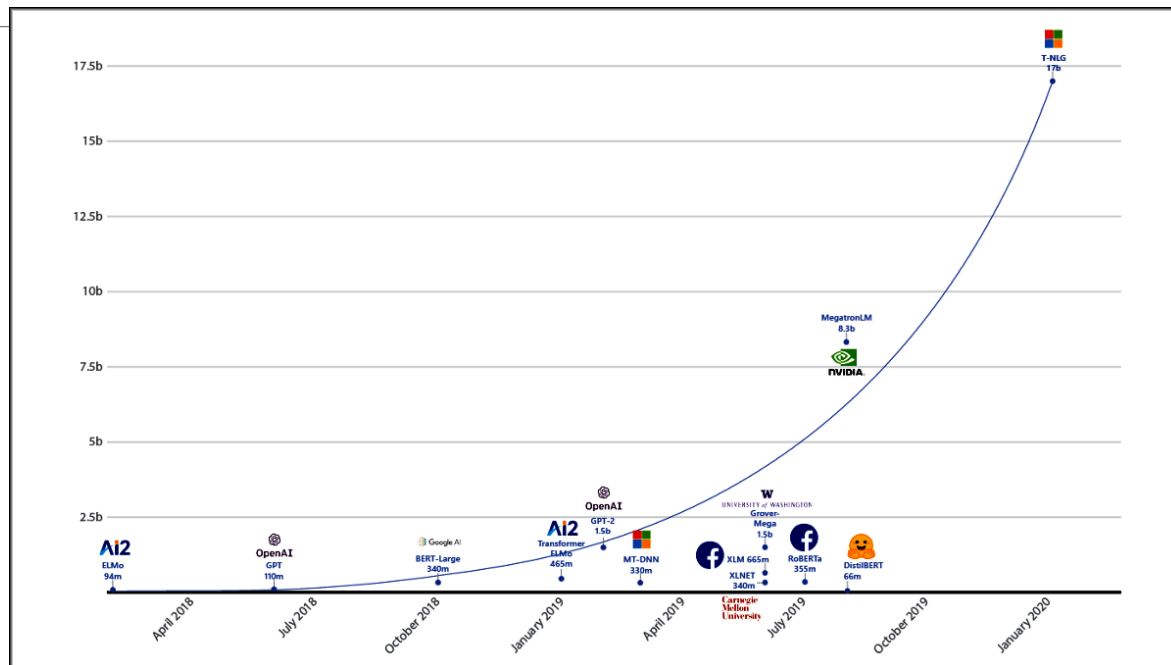
- BETO
- Bertin
- MarIA
- RigoBERTa






	Dataset	BETO	BERTIN	MarIA	RigoBERTa
NER	CANTEMISTNER	89.9%	79.5%	92.3%	93.3% ★
NER	CAPITEL	87.0%	86.5%	87.8% ★	87.4%
NER	CONLL2002	89.6%	90.1% ★	89.9%	89.5%
NER	MEDDOCAN	84.7%	72.2%	84.1%	85.0% ★
NER	MEDDOPROF1	80.5%	71.0%	80.7%	83.1% ★
NER	MEDDOPROF2	81.8%	44.2%	78.5%	86.4% ★
Class	MLDOC	95.4%	94.4%	95.6% ★	95.6% ★
Class	PAWS-X	89.7%	90.1%	88.9%	91.0% ★
NER	PHARMACONER	61.4%	47.1%	57.1%	70.0% ★
QA	SQAC	76.2%	75.0%	86.6%	89.7% ★
QA	SQUADES	75.6%	70.0%	81.8%	85.4% ★
Class	TASS2020	46.1%	46.1%	47.3% ★	46.7%
Class	XNLI	81.7%	79.4%	81.6%	83.4% ★
	<b>TOTALES</b>	<b>76.5%</b>	<b>69.6% ★</b>	<b>77.3% ★</b>	<b>79.8% ★</b>

# Tamaño



# Datasets

---

Dataset	# Tokens (Billions)
Total	499
Common Crawl (filtered by quality)	410
WebText2	19
Books1	12
Books2	55
Wikipedia	3

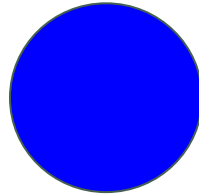
---

## Procesamiento de secuencias

A solid green horizontal bar at the bottom of the slide.

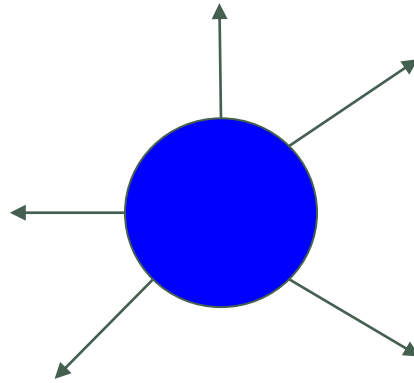
¿Hacia dónde va la bola?

---



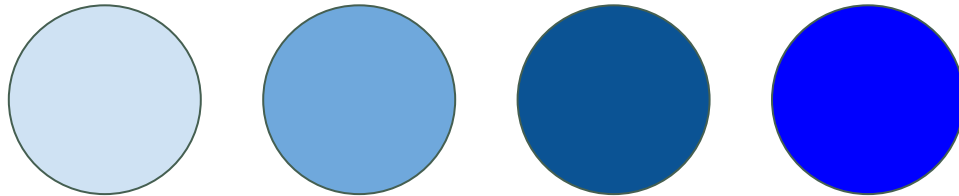
# ¿Hacia dónde va la bola?

---



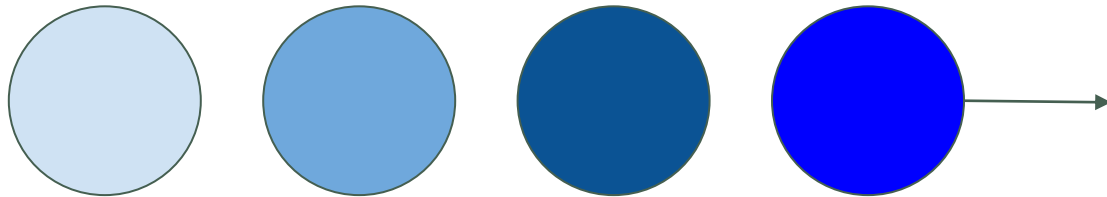
# ¿Hacia dónde va la bola?

---



# ¿Hacia dónde va la bola?

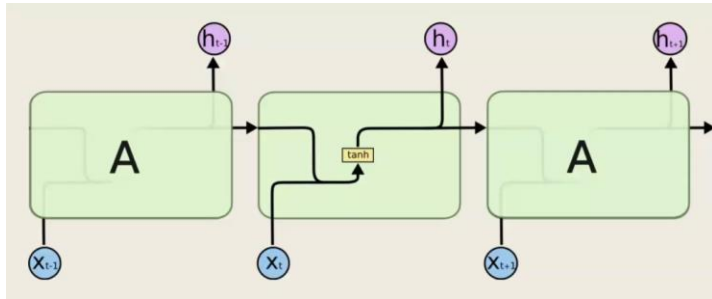
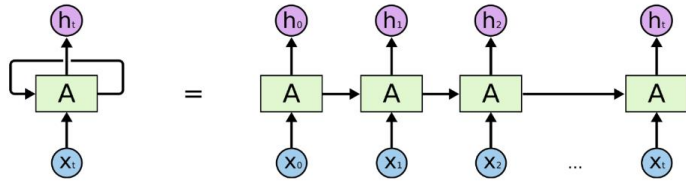
---





- 
- RNN
  - LSTM
  - Transformers
- 

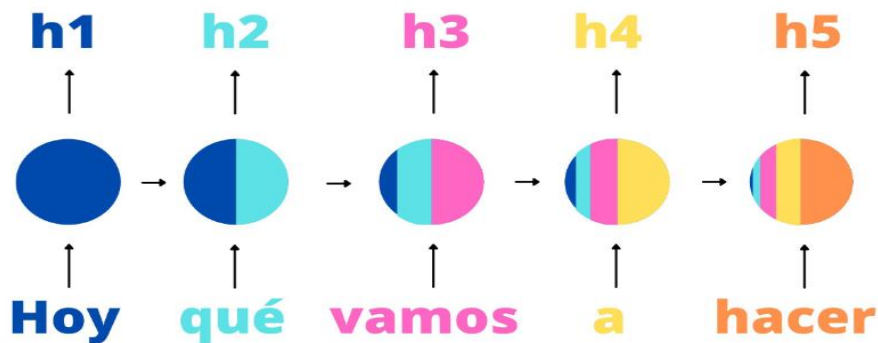
# Redes neuronales recurrentes



La RNN procesa secuencias mediante un bucle que itera a través de los elementos de la secuencia y mantiene un estado que contiene información relativa a lo procesado anteriormente.

# Redes neuronales recurrentes

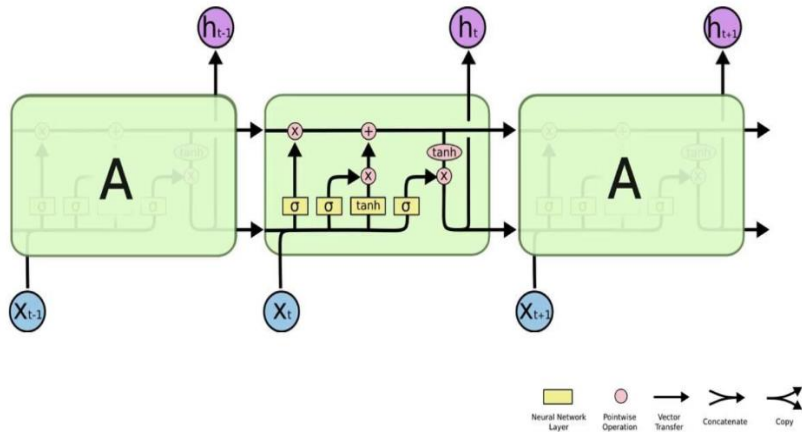
---



El procesamiento secuencial es una de las mayores limitaciones de las RNR ante largas secuencias

Cuanto más extensa es la secuencia, mayor es el desvanecimiento de los gradientes.

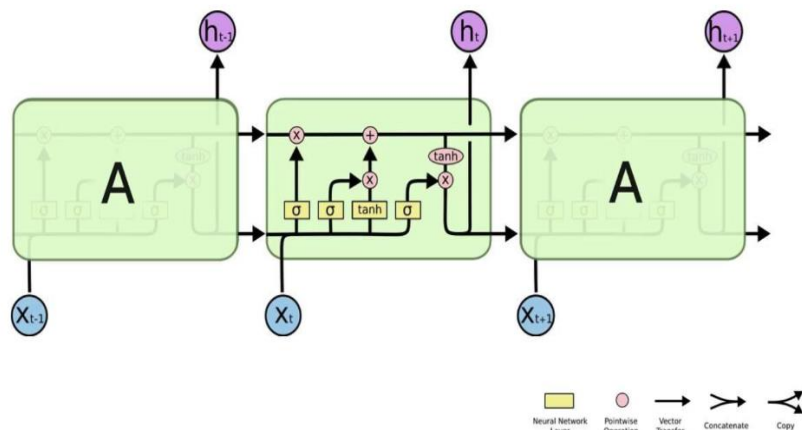
# LSTM



Variante de las RNN diseñadas para superar los problemas de desvanecimiento y explosión del gradiente.

Puerta de entrada  
Puerta de olvido  
Puerta de salida

# LSTM

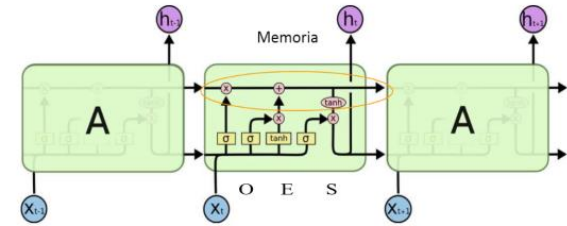


En las LSTM se añade la incorporación de las celdas de memoria.

La red neuronal es capaz de eliminar información previa y actualizar la memoria al recibir nuevo input.

# LSTM

---



Tres puertas : puerta de olvido (O), puerta de entrada (E) y puerta de salida (S).

- En la primera puerta o puerta de olvido se aplica una función sigmoideal para calcular cuánta información antigua se debe eliminar.
- En la puerta de entrada se determina cuánta nueva información se debe agregar al estado de memoria.
- En la puerta de salida se establece qué información del estado de memoria se usa en el cálculo de la salida de cada celda



---

**2017**

Vaswani et al.

A solid green horizontal bar at the bottom of the slide.

---

# 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

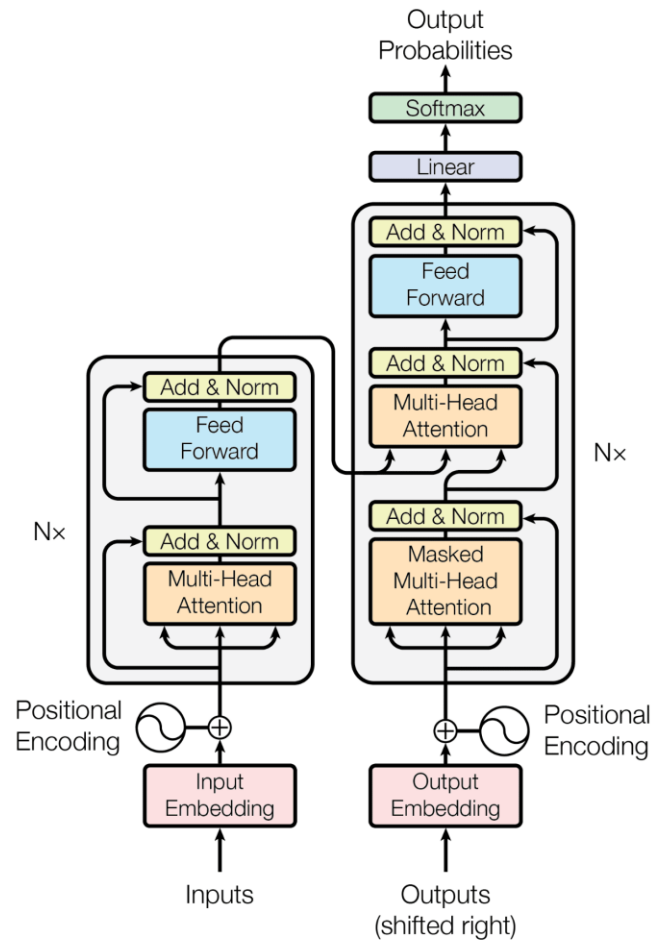
**Aidan N. Gomez\* †**  
University of Toronto  
aidan@cs.toronto.edu

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

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

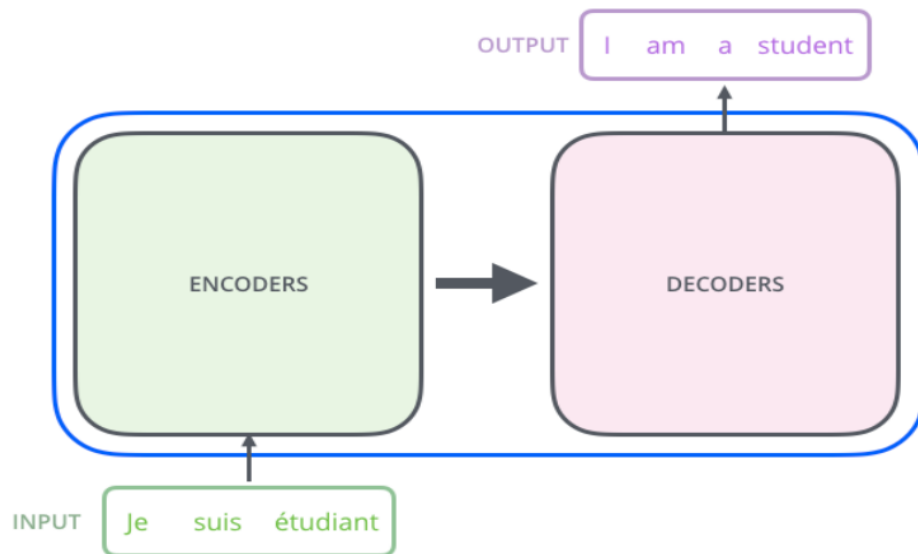


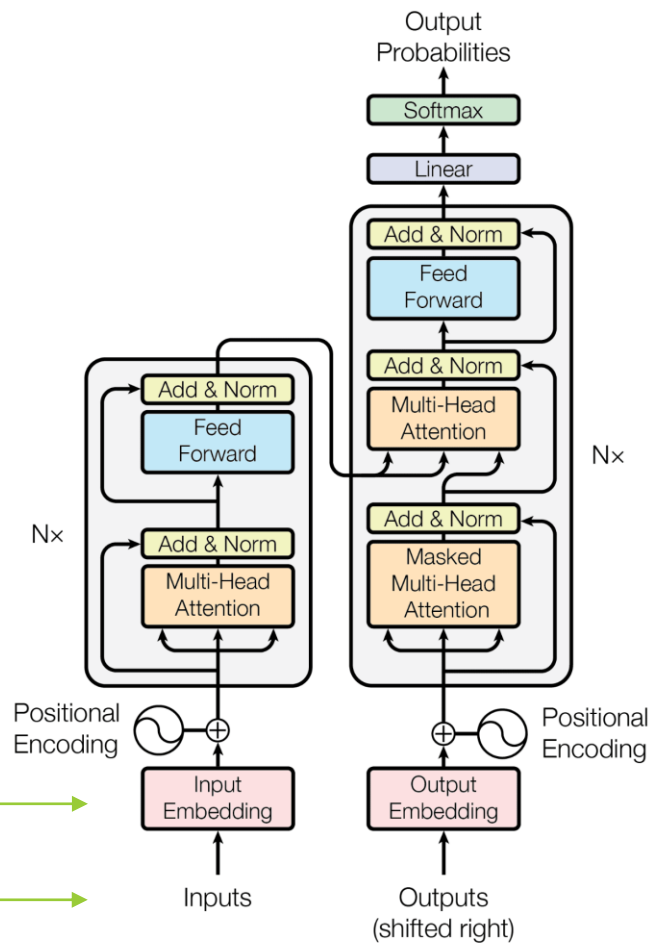
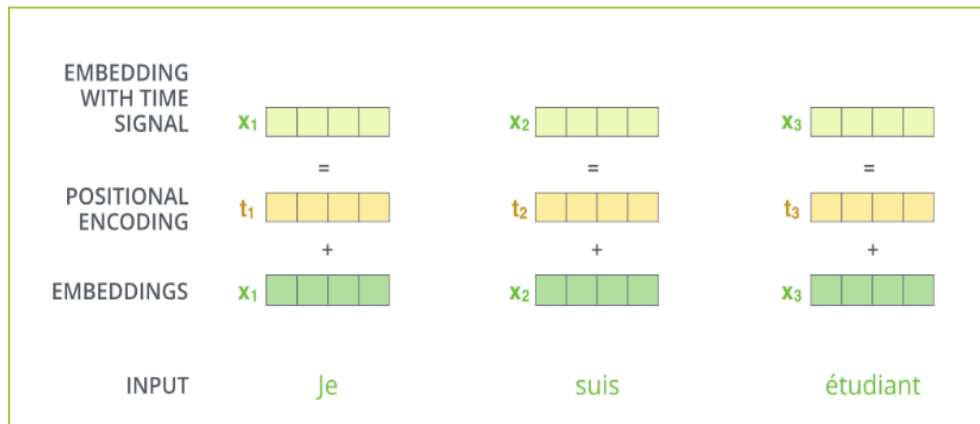
# Arquitectura Transformer



# Transformers

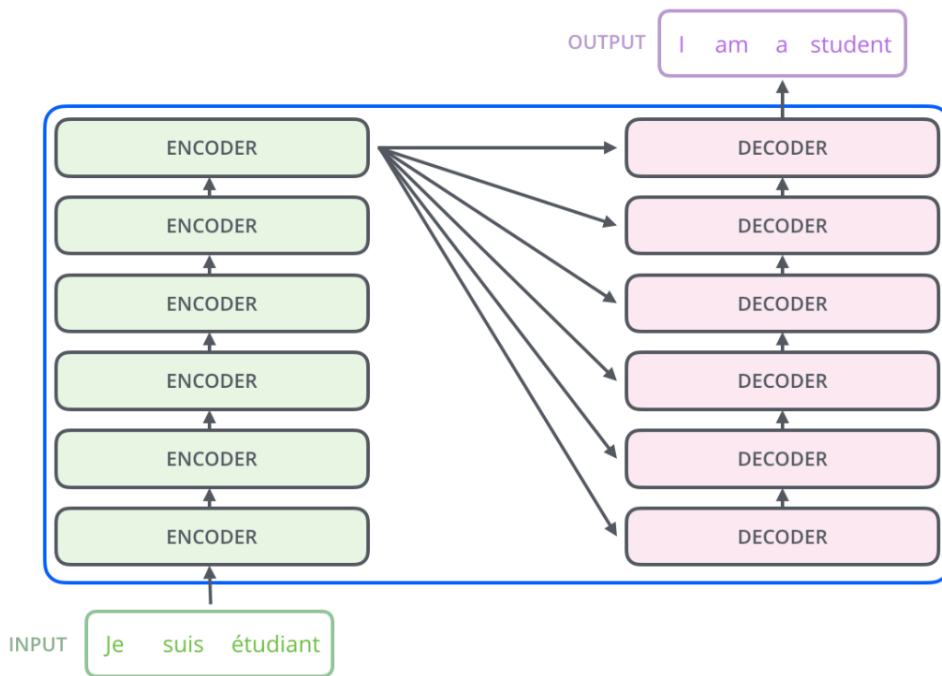
---





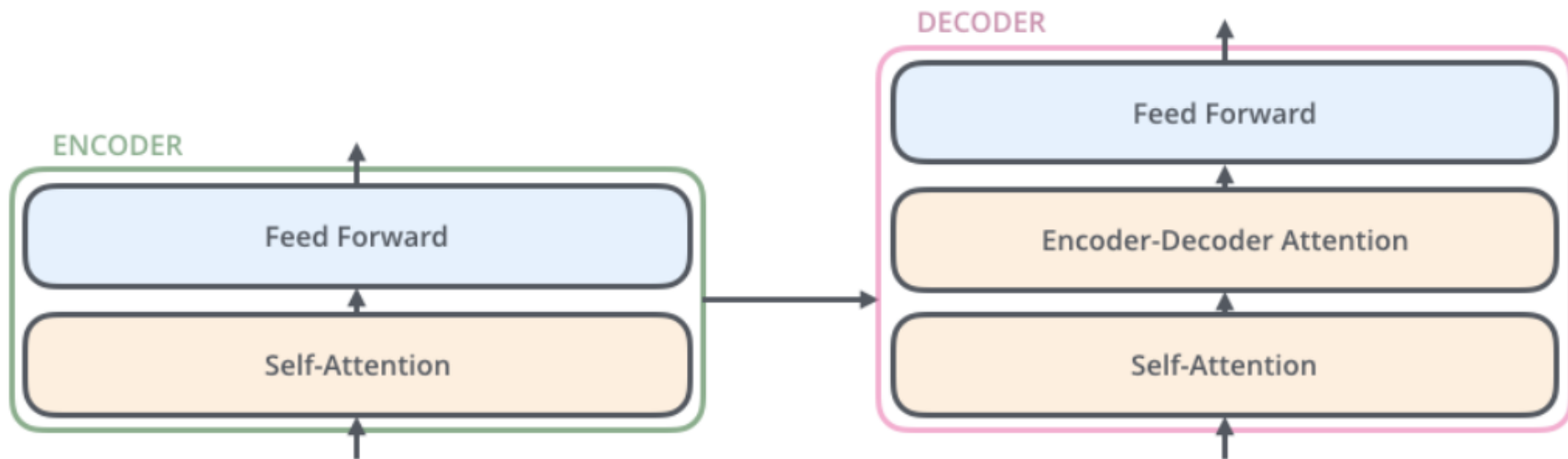
# Transformers

---

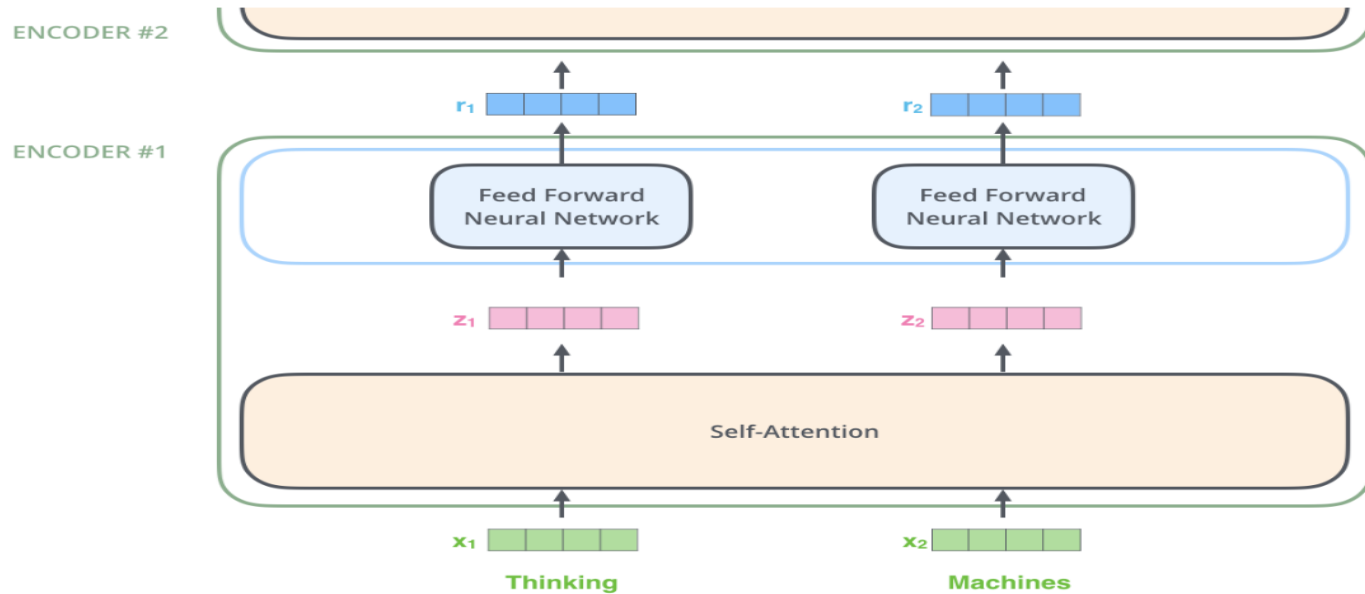


# Transformers

---

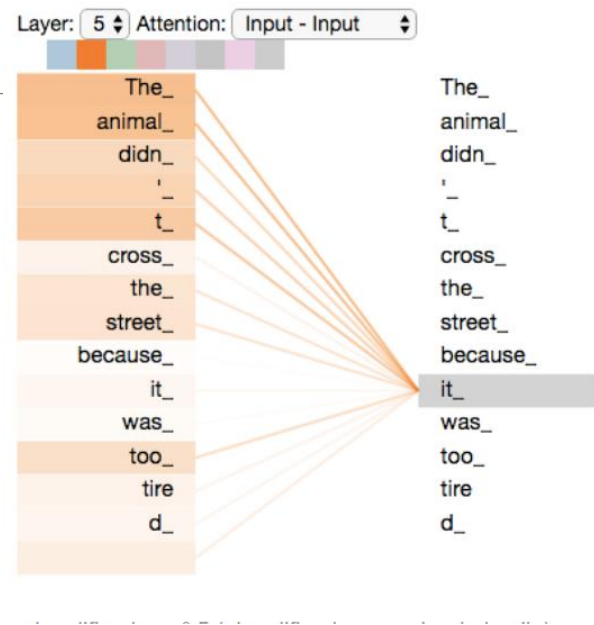


# Encoder

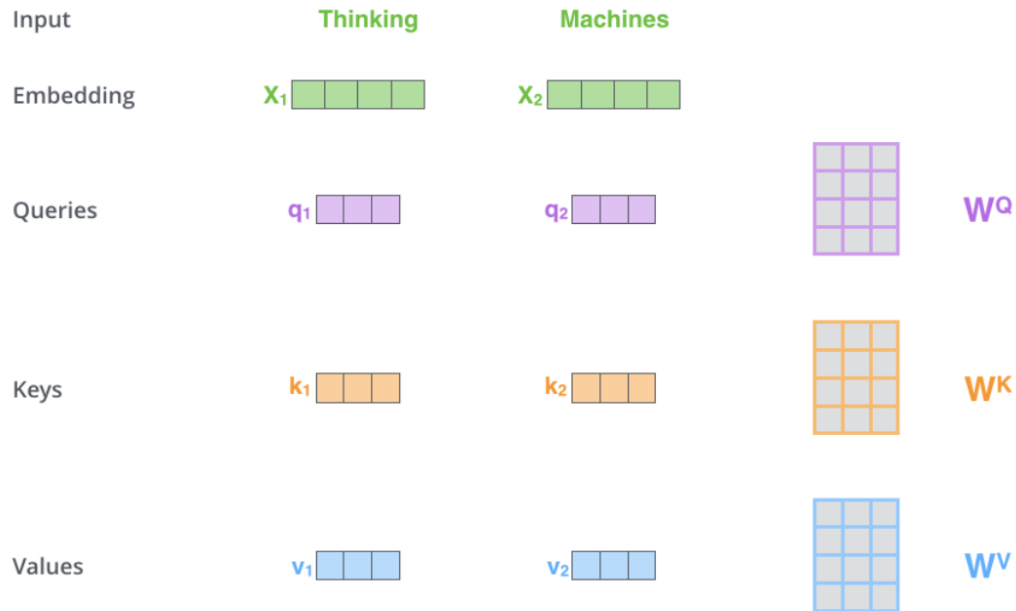


# Auto atención

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



# Auto atención





Input

Embedding

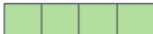
Queries

Keys

Values

Score

Thinking

$x_1$  

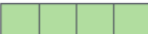
$q_1$  

$k_1$  

$v_1$  

$q_1 \cdot k_1 = 112$

Machines

$x_2$  

$q_2$  

$k_2$  

$v_2$  

$q_1 \cdot k_2 = 96$

Input

Embedding

Queries

Keys

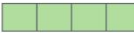
Values

Score

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

Softmax

Thinking

$x_1$  

$q_1$  

$k_1$  

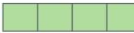
$v_1$  

$q_1 \cdot k_1 = 112$

14

0.88

Machines

$x_2$  

$q_2$  

$k_2$  

$v_2$  

$q_1 \cdot k_2 = 96$

12

0.12

Input

Embedding

Queries

Keys

Values

Score

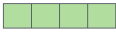
Divide by  $8 (\sqrt{d_k})$

Softmax

Softmax  
X  
Value

Sum

Thinking

$x_1$  

$q_1$  

$k_1$  

$v_1$  

$q_1 \cdot k_1 = 112$

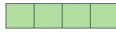
14

0.88

$v_1$  

$z_1$  

Machines

$x_2$  

$q_2$  

$k_2$  

$v_2$  

$q_2 \cdot k_2 = 96$

12

0.12

$v_2$  

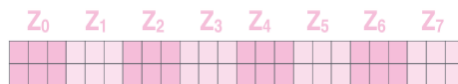
$z_2$  

$$\text{softmax} \left( \frac{\begin{matrix} \text{Q} \\ \begin{matrix} \text{1x2 purple} \\ \text{1x2 purple} \end{matrix} \end{matrix} \times \begin{matrix} \text{K}^T \\ \begin{matrix} \text{2x1 orange} \\ \text{2x1 orange} \end{matrix} \end{matrix}}{\sqrt{d_k}} \right) \begin{matrix} \text{V} \\ \begin{matrix} \text{1x2 blue} \\ \text{1x2 blue} \end{matrix} \end{matrix}$$
$$= \begin{matrix} \text{Z} \\ \begin{matrix} \text{1x2 pink} \\ \text{1x2 pink} \end{matrix} \end{matrix}$$

# Encoder

## Atención multicabezal

1) Concatenate all the attention heads



2) Multiply with a weight matrix  $W^O$  that was trained jointly with the model

$\times$



3) The result would be the  $Z$  matrix that captures information from all the attention heads. We can send this forward to the FFNN

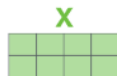


# Encoder

1) This is our input sentence\*

Thinking  
Machines

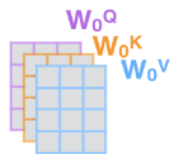
2) We embed each word\*



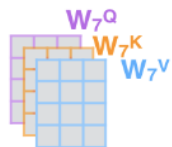
\* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



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



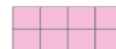
...



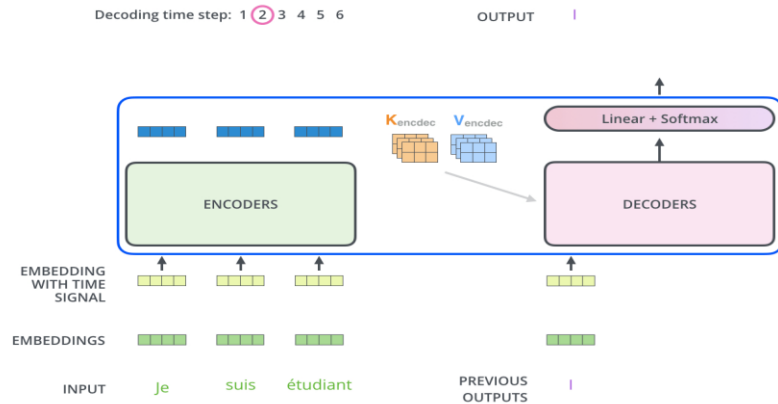
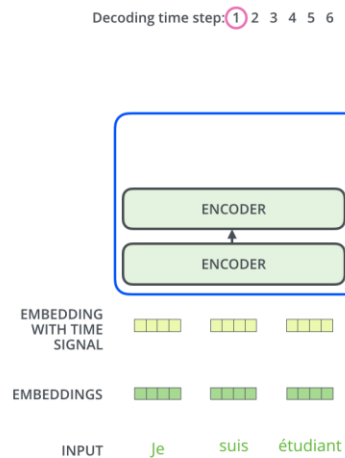
$W^O$



$Z$



# Decoder



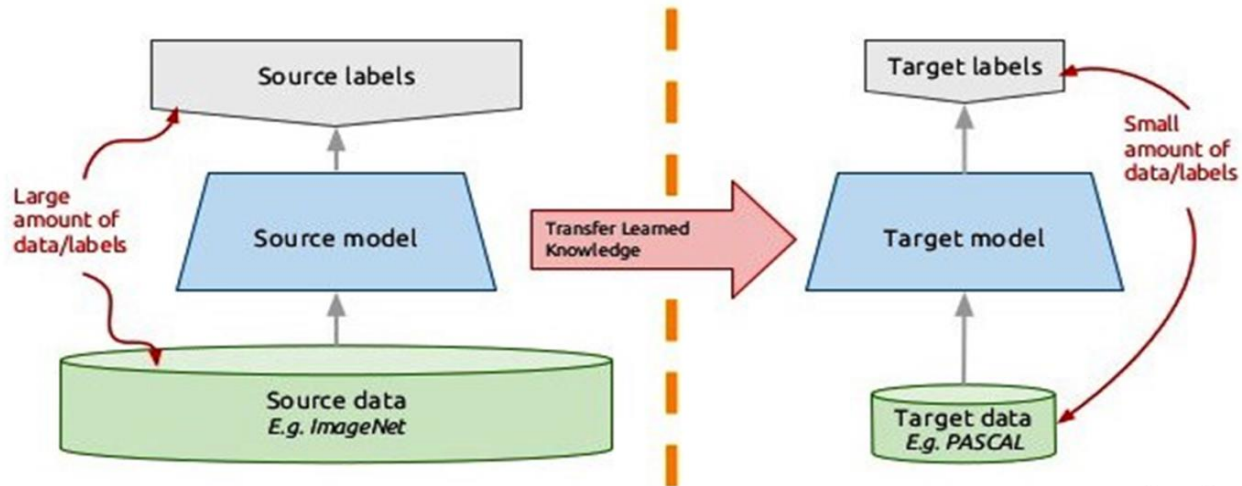
# Índice

---

1. Aprendizaje profundo
2. Modelos de lenguaje
- 3. Transfer learning**

# Transfer learning

## Transfer learning: idea



# Problemas Deep Learning

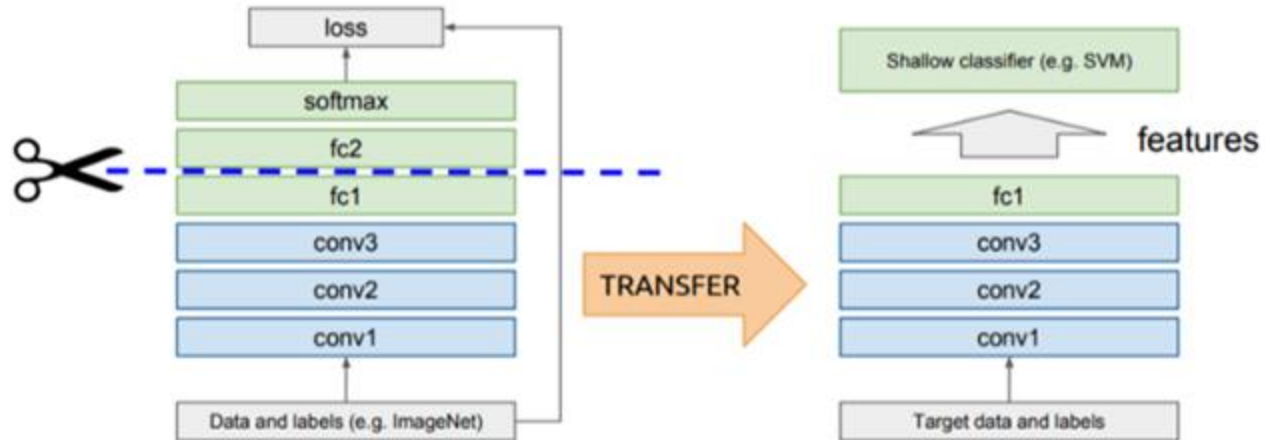
---

- Requieren grandes cantidades de datos
- Requieren uso de múltiples GPUs

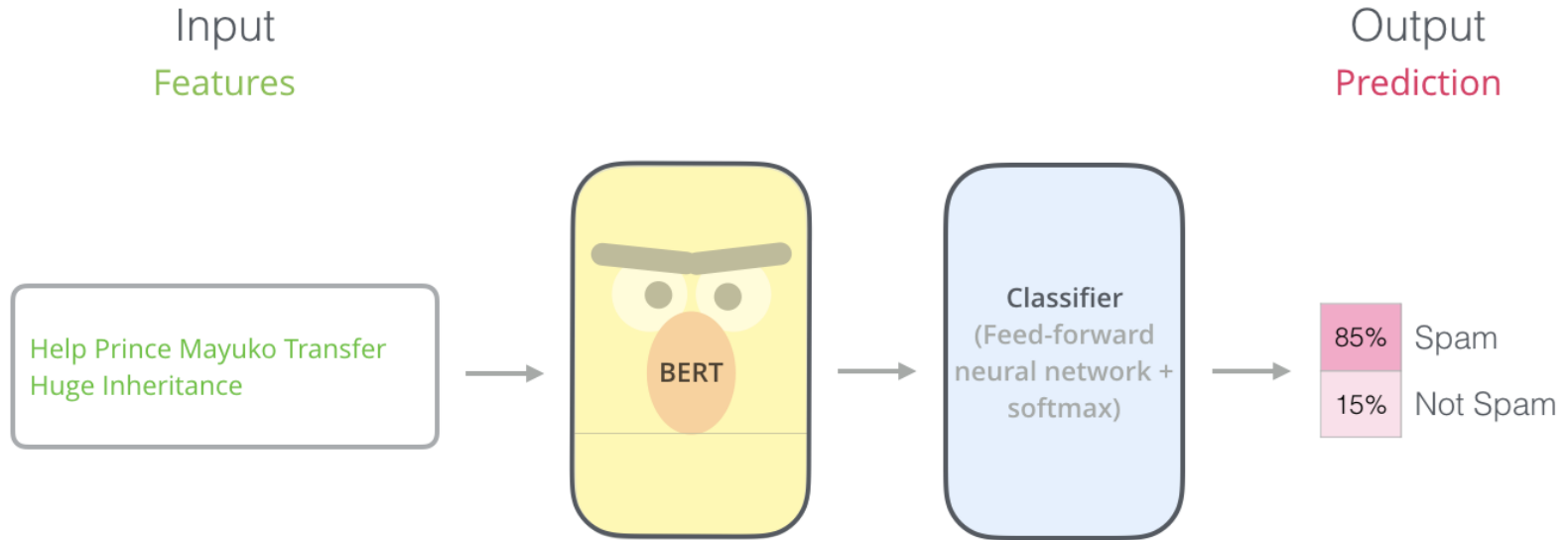


# Transfer learning

## Feature extractor



# Aplicaciones



---

LAB

