Introducción al Deep Learning

Día 4: Arquitecturas Avanzadas. Transformers

Manuel Germán y David de la Rosa

Universidad de Jaén

(mgerman,drrosa)@ujaen.es

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

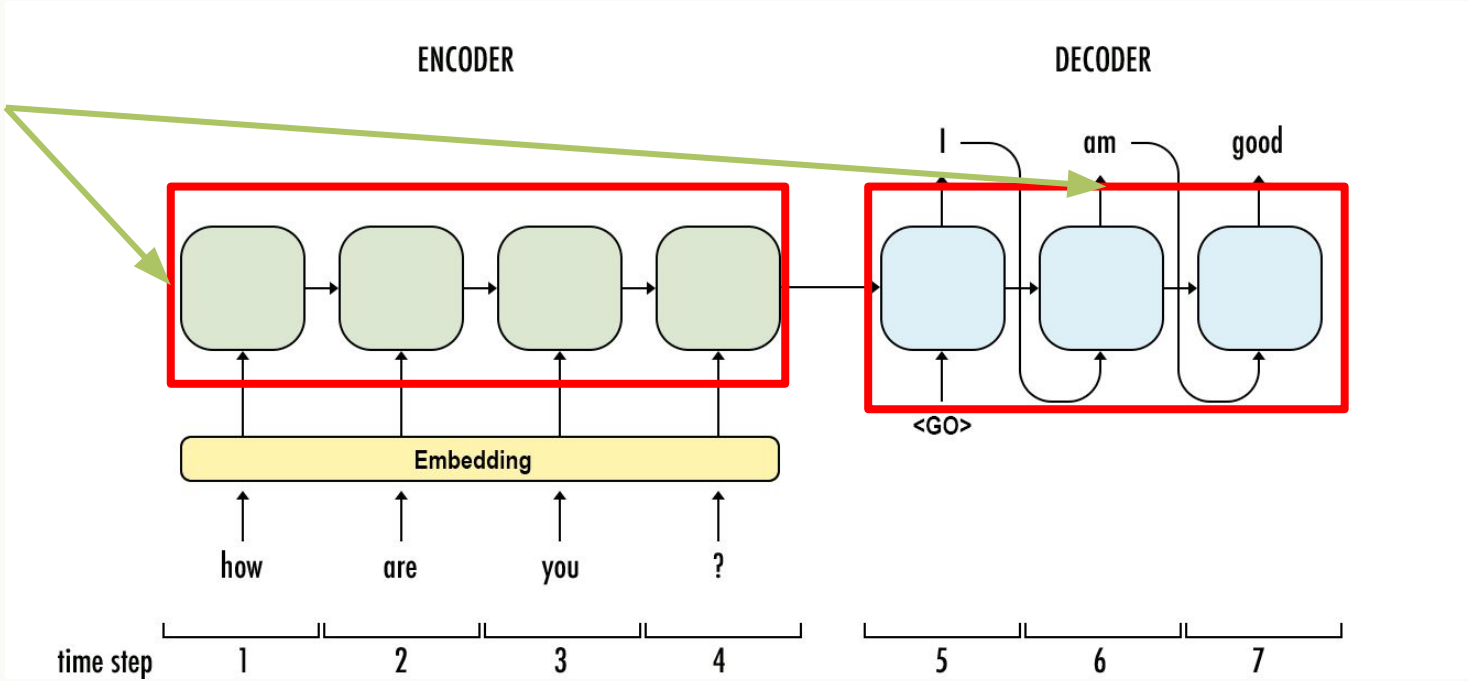
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

1

Motivación

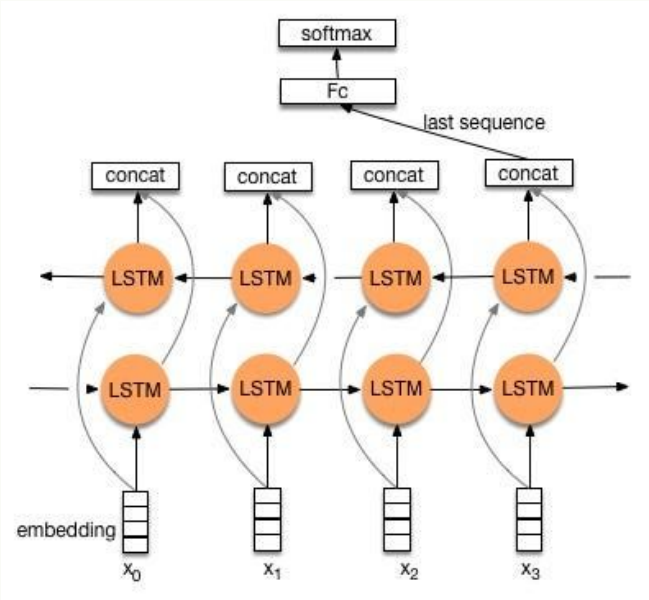
Modelos *sequence-to-sequence*



Modelos *sequence-to-sequence*

A lo largo del tiempo, aparecen mejoras que aumentan el rendimiento de las RNN

Bidireccionalidad

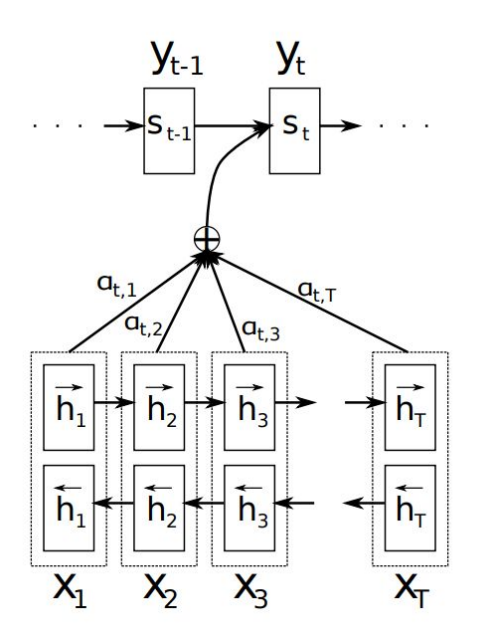


Modelos *sequence-to-sequence*

A lo largo del tiempo, aparecen mejoras que aumentan el rendimiento de las RNN

Bidireccionalidad

Atención



Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*. http://arxiv.org/abs/1409.0473

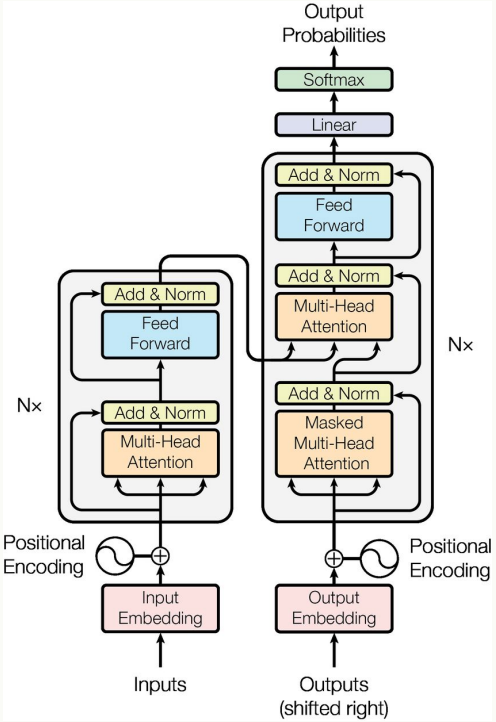
Sin embargo, la triada sigue presente…

Pérdida de dependencias temporales

Complejidad

Paralelización

2017: Nacimiento de los *Transformers*



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Procesan secuencias de manera más eficiente

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

¡Y en paralelo! (cogedlo con pinzas, no es lo que parece)

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

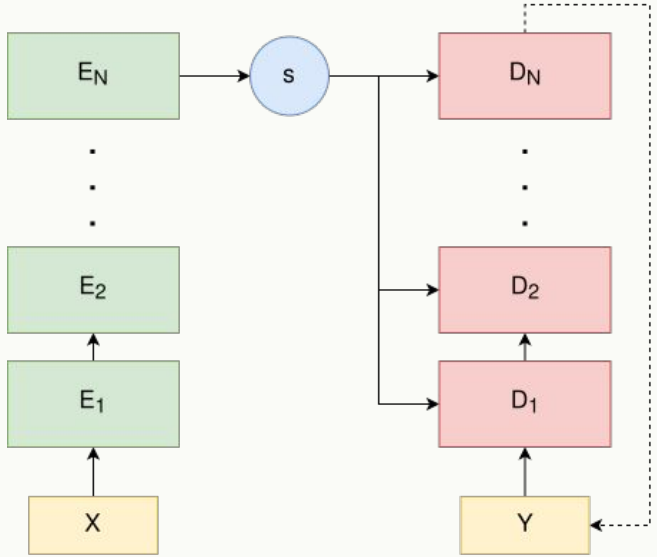
2

¿Qué es un transformer?

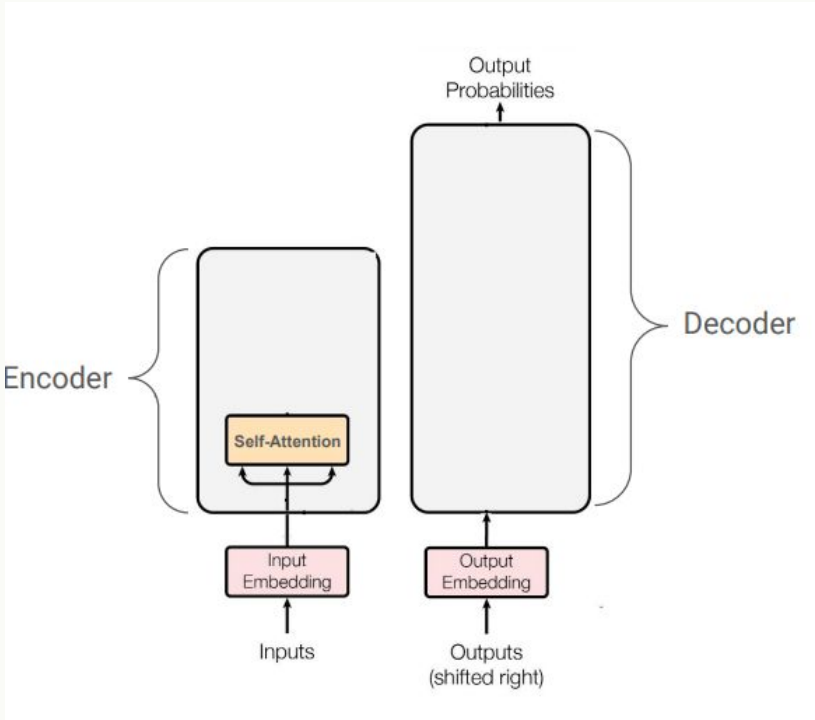
Idea general

Tipo de arquitectura compuesta de varios **codificadores** y **decodificadores**.

Los **codificadores** convierte las secuencias de entrada en **representaciones continuas** que luego los **decodificadores** usan para generar la salida paso a paso.



Codificador: *Self-Attention*



Entendiendo el mecanismo de atención

Es una especie de tabla hash “difusa”. Para encontrar un **valor**, comparamos unas **consultas** respecto a un conjunto de **claves**.

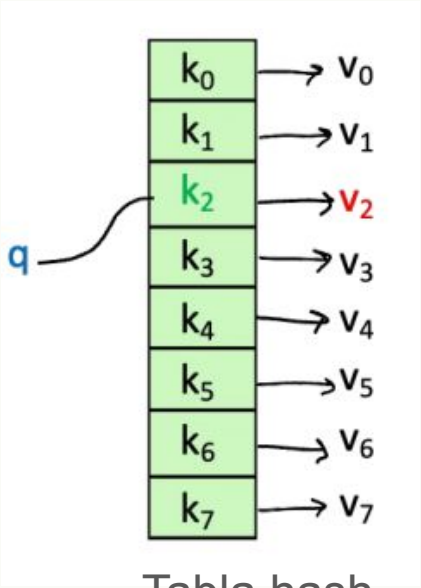
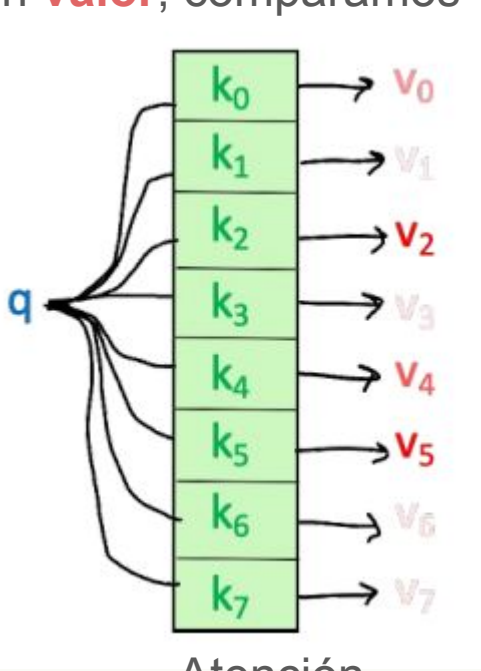


Tabla hash

Las representaciones q, k y v se **aprenden.**

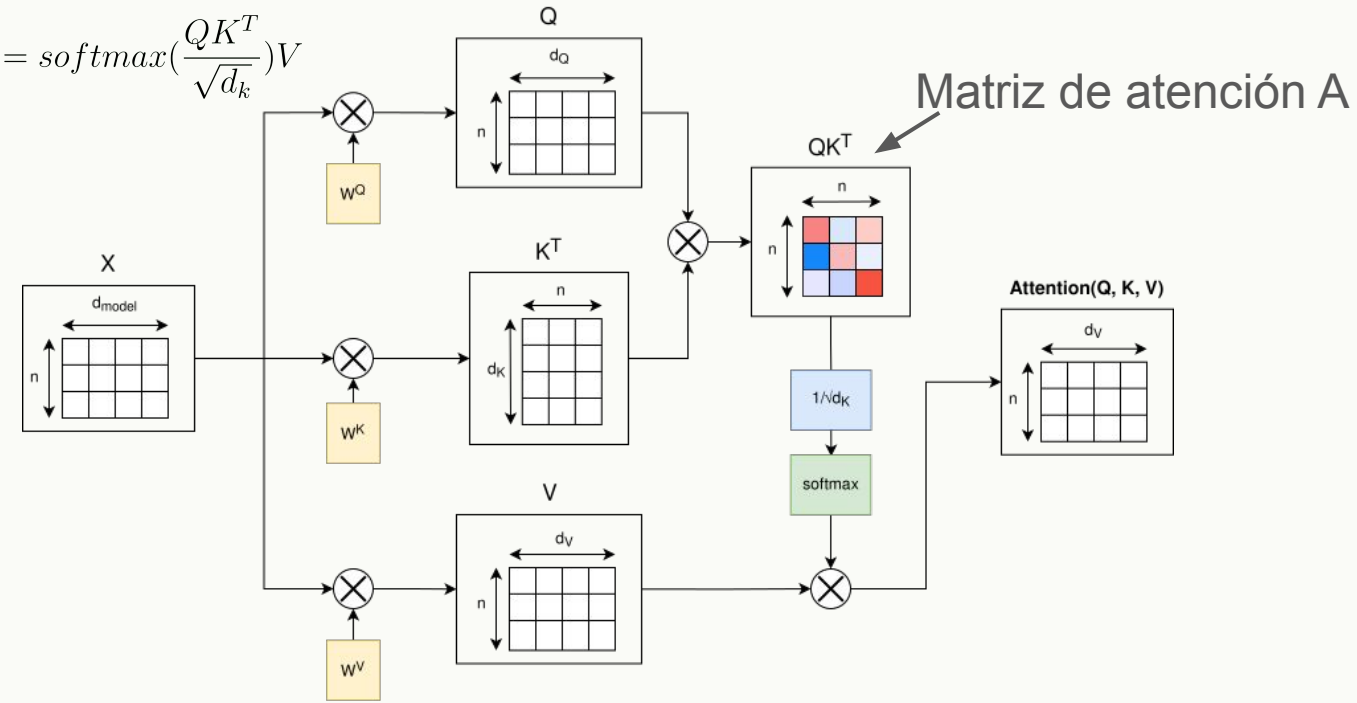


Atención

Entendiendo el mecanismo de atención

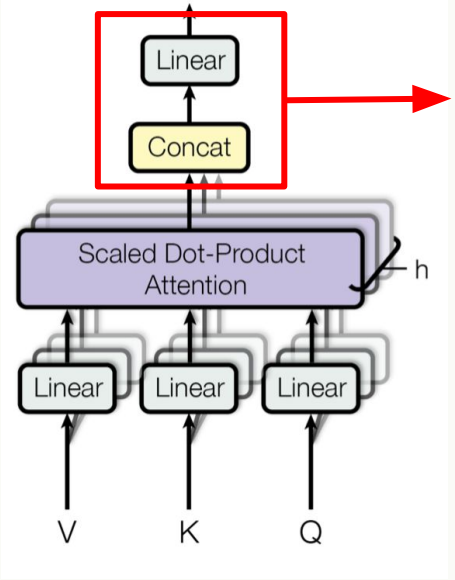
Pretende responder a la pregunta: ¿Cuán relevante es elemento *i* para el *j*?

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

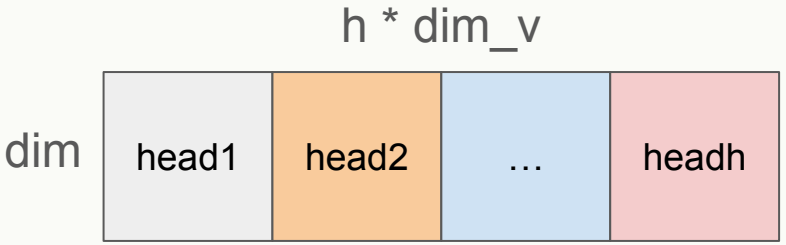


Atención multicabezal

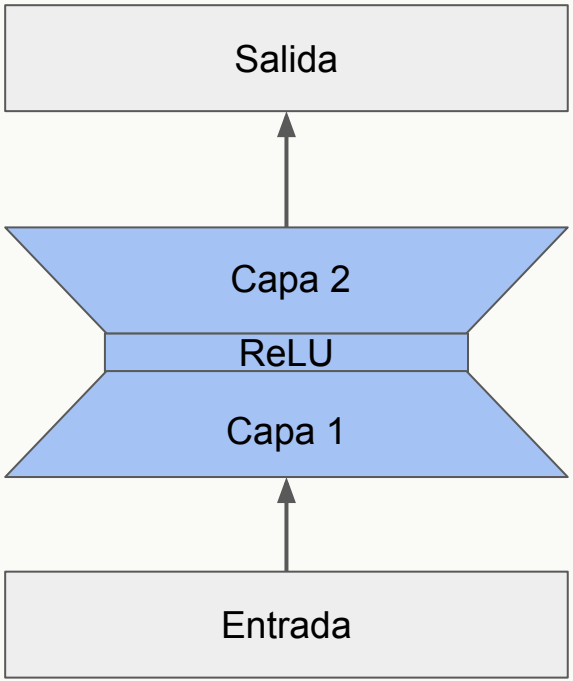
Podemos tener varias capas de atención en paralelo para luego combinarlas.
Cada “cabeza” prestará su atención a unas características específicas.

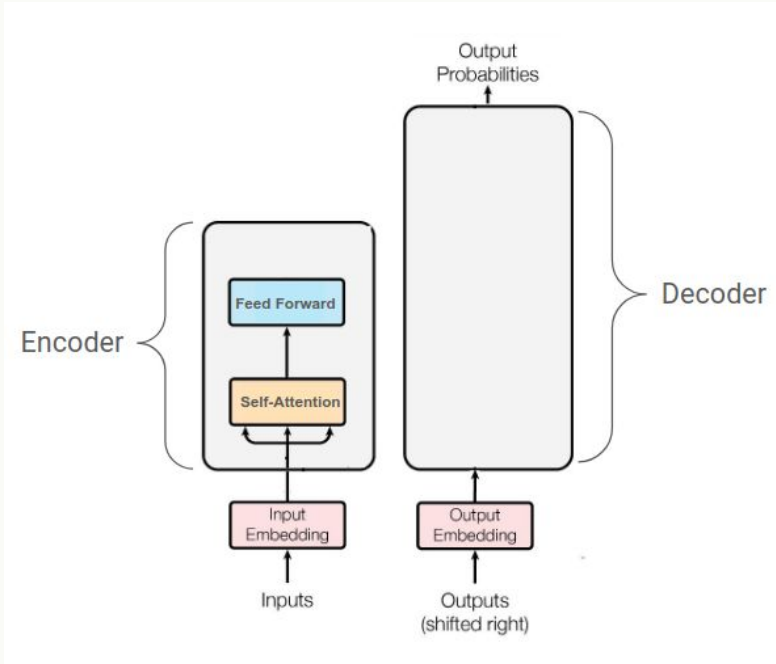


$$MultiHead(Q, K, V) = Concat(head_0, ..., head_h)W_O$$
$$head_i = Attention(QW_Q^i, KW_K^i, VW_V^i)$$



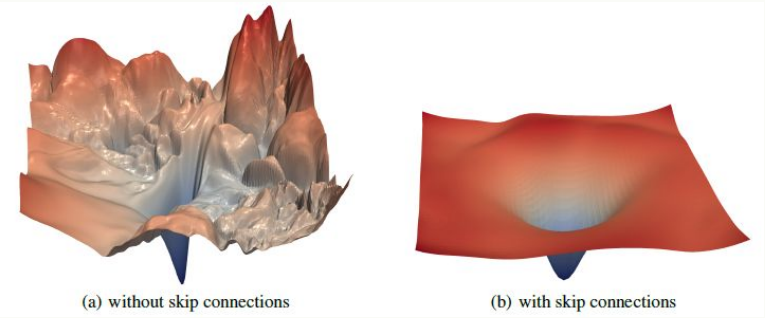
Codificador: *Feed-Forward Layer*

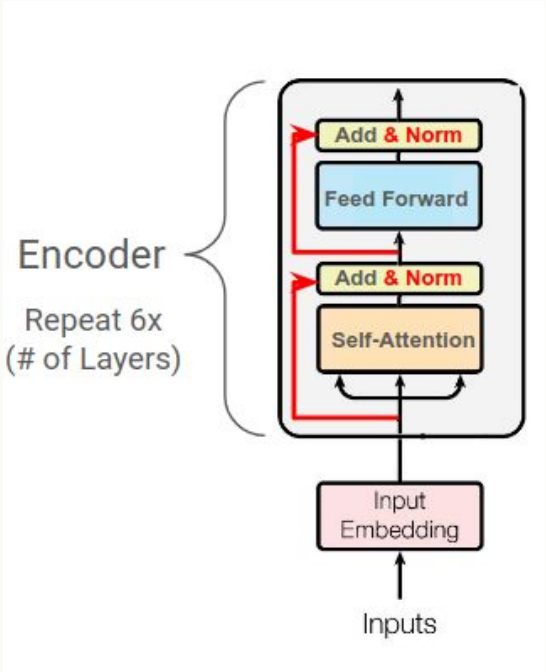




Mejorando el entrenamiento: Conexiones residuales

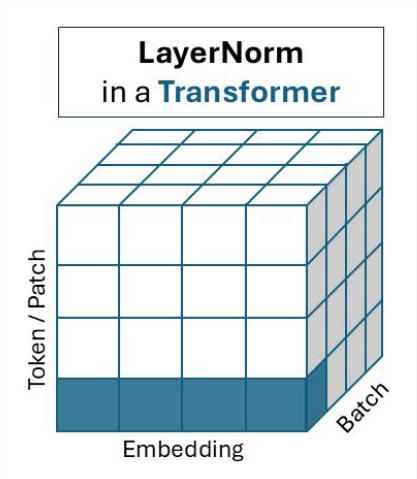
Las **conexiones residuales** permiten mitigar el olvido o la distorsión de información importante. A efectos prácticos:





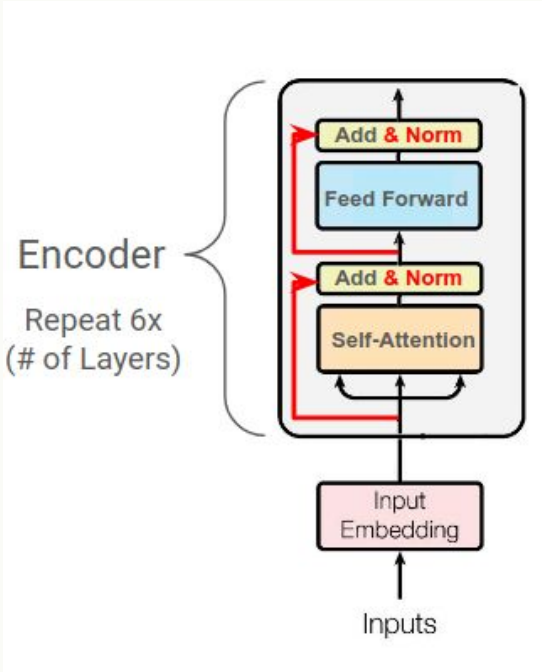
Mejorando el entrenamiento: Normalización por capas

Mitigar explosión/desvanecimiento de gradiente estandarizando cada característica de los datos de entradas.



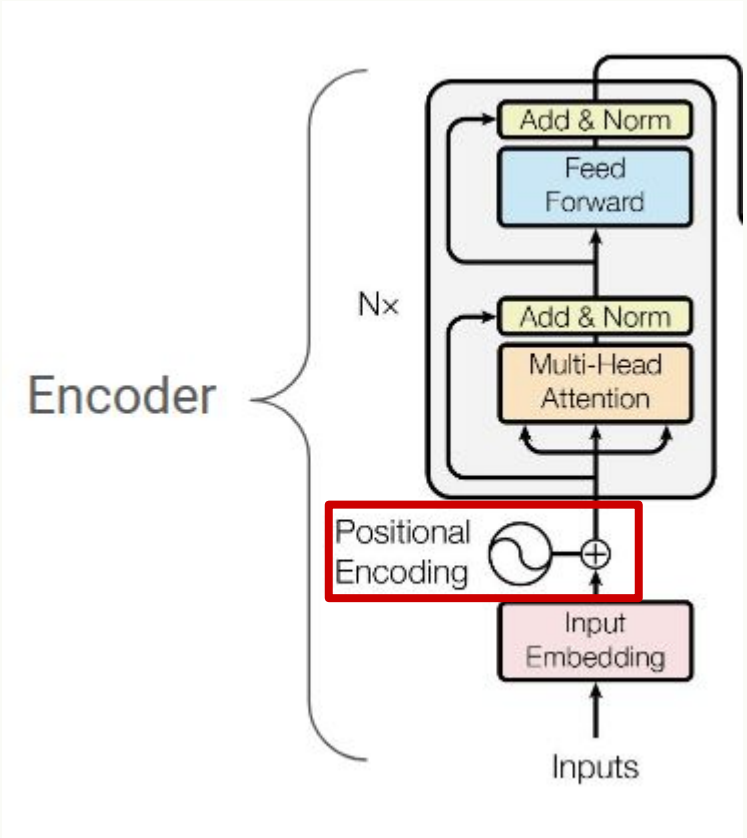
https://docs.pytorch.org/docs/stable/generated/torch.nn.LayerNorm.html

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$



Codificación posicional

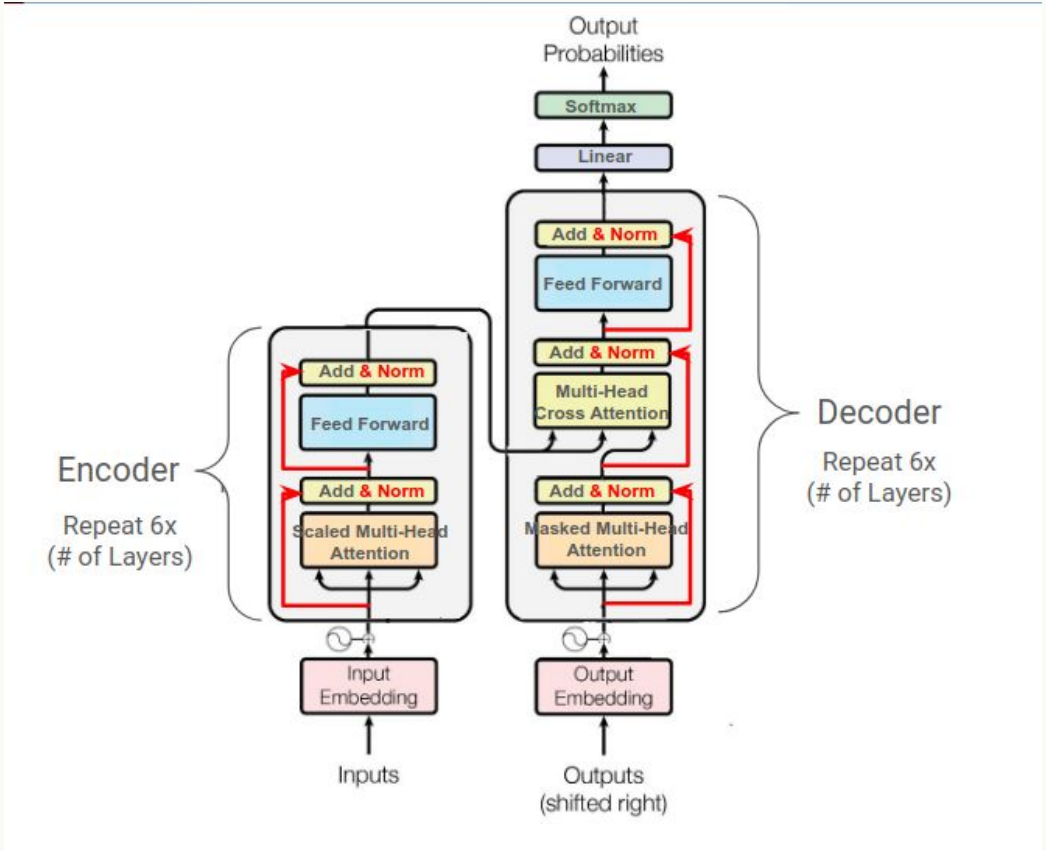
https://erdem.pl/2021/05/understanding-positional-encoding-in-transformers



Decodificador

Es **muy similar** al codificador.

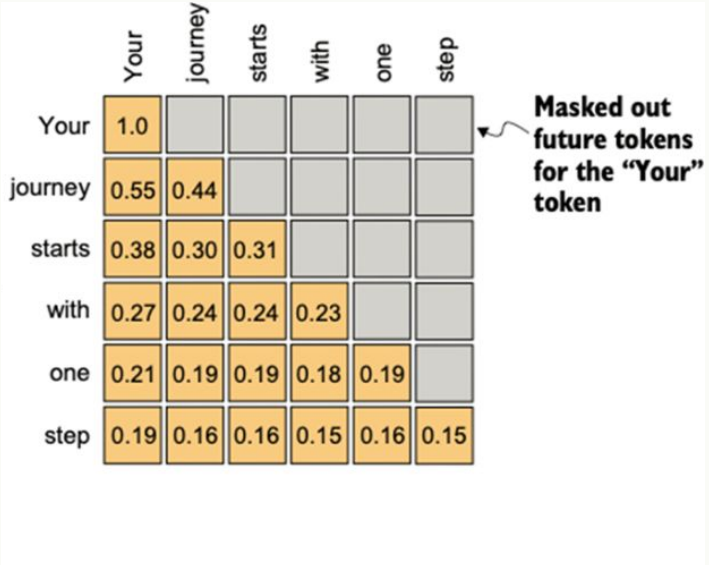
- Posee **dos capas de atención**. La primera se denomina *enmascarada* y la segunda *cruzada*.
- Su salida se usa para realizar la **predicción**.



Atención enmascarada

Un elemento **solo** puede atender a él mismo o a elementos que están **antes** que él.

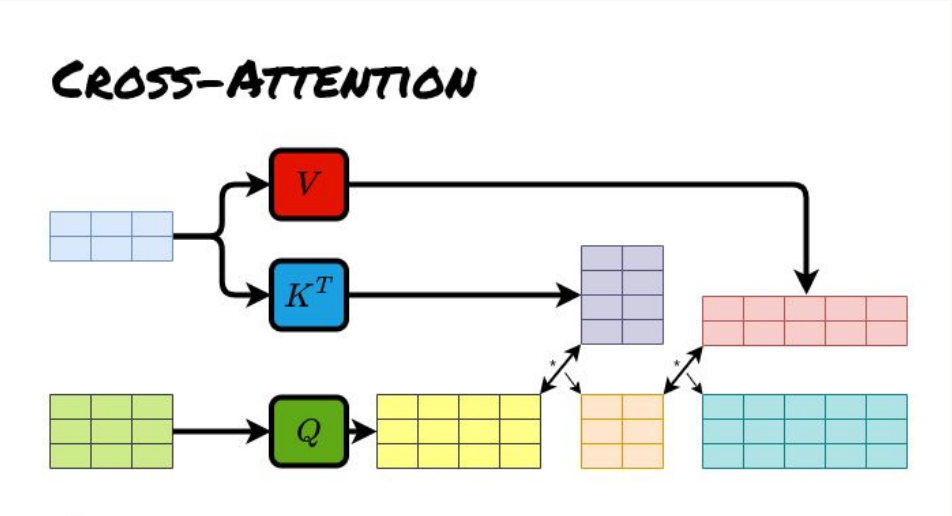
Se emplea una **máscara** para anular las posiciones de la matriz que se correspondan con elementos posteriores.



Atención cruzada

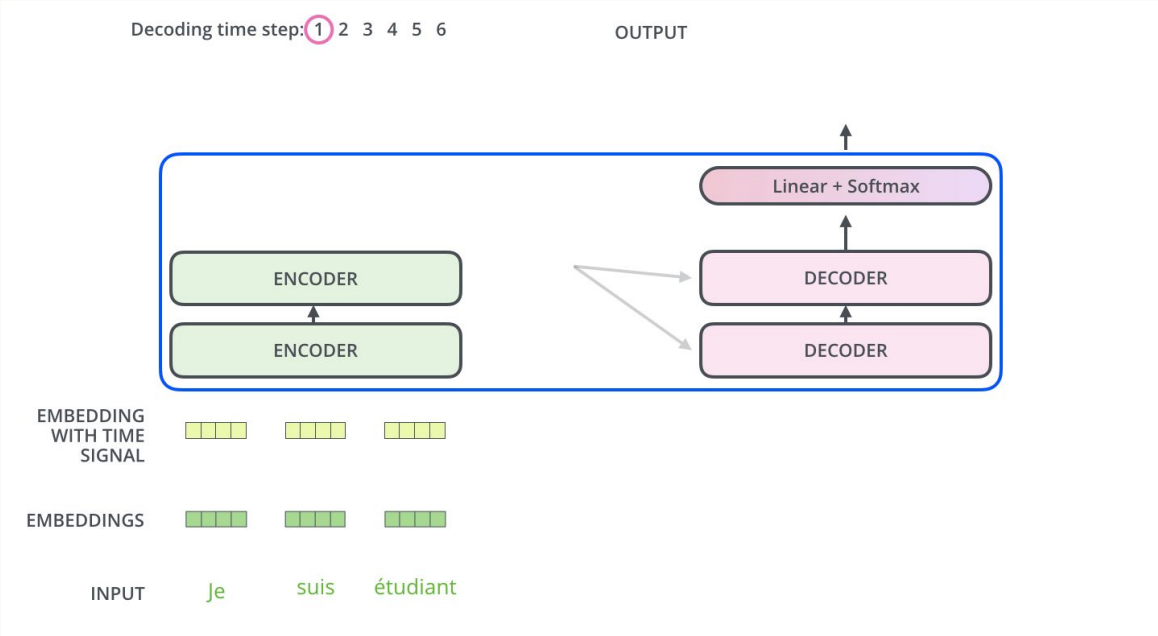
Logra que el decodificador se centre en los elementos de la entrada adecuados.

Las **claves** y los **valores** se obtienen de la salida de la pila de **codificadores**, mientras que las **consultas** se obtienen a partir de la entrada del **decodificador**.



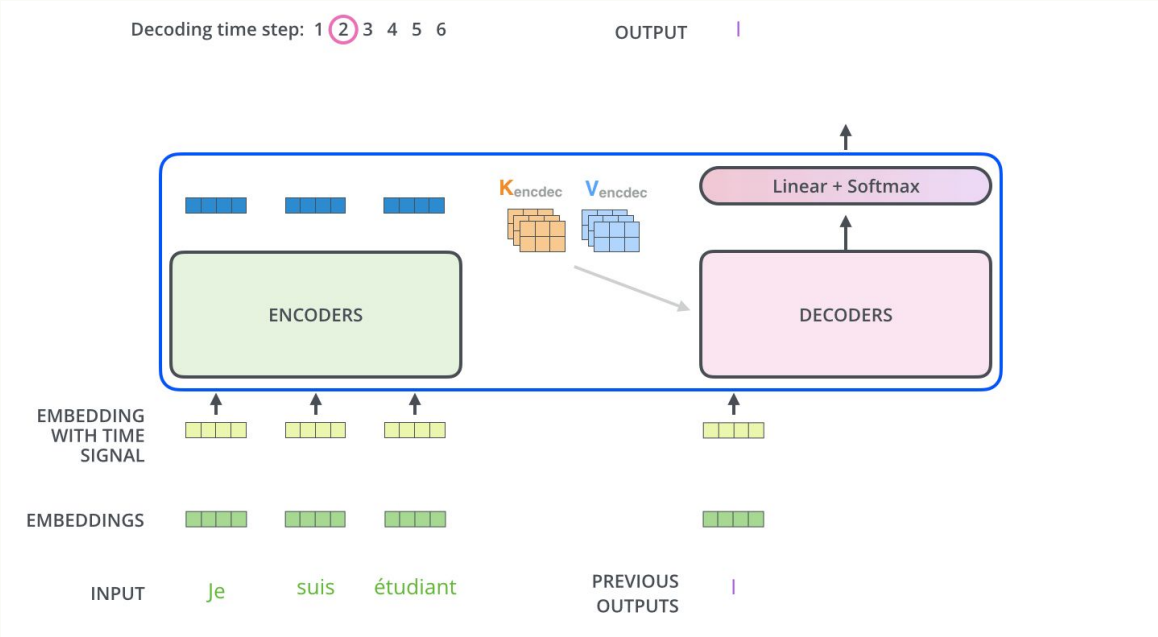
Inferencia

La entrada se procesa en la pila de decodificadores. Su salida es usada por la pila de decodificadores para realizar la tarea



Inferencia

La inferencia finaliza tras emitir una cantidad de elementos predeterminada o cuando el modelo lo considera conveniente.



3

Aplicaciones

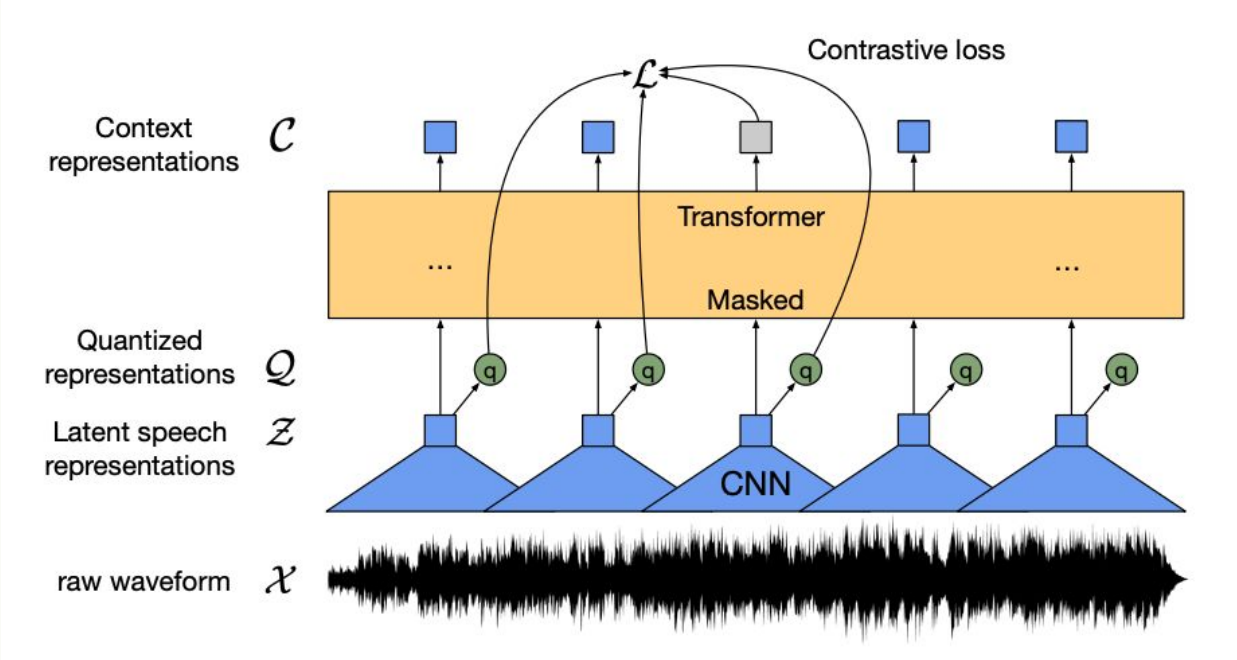
Modularidad

En función de la tarea usaremos los *encoders*, los *decoders* o ambos.

Entender | Generar | Ambos

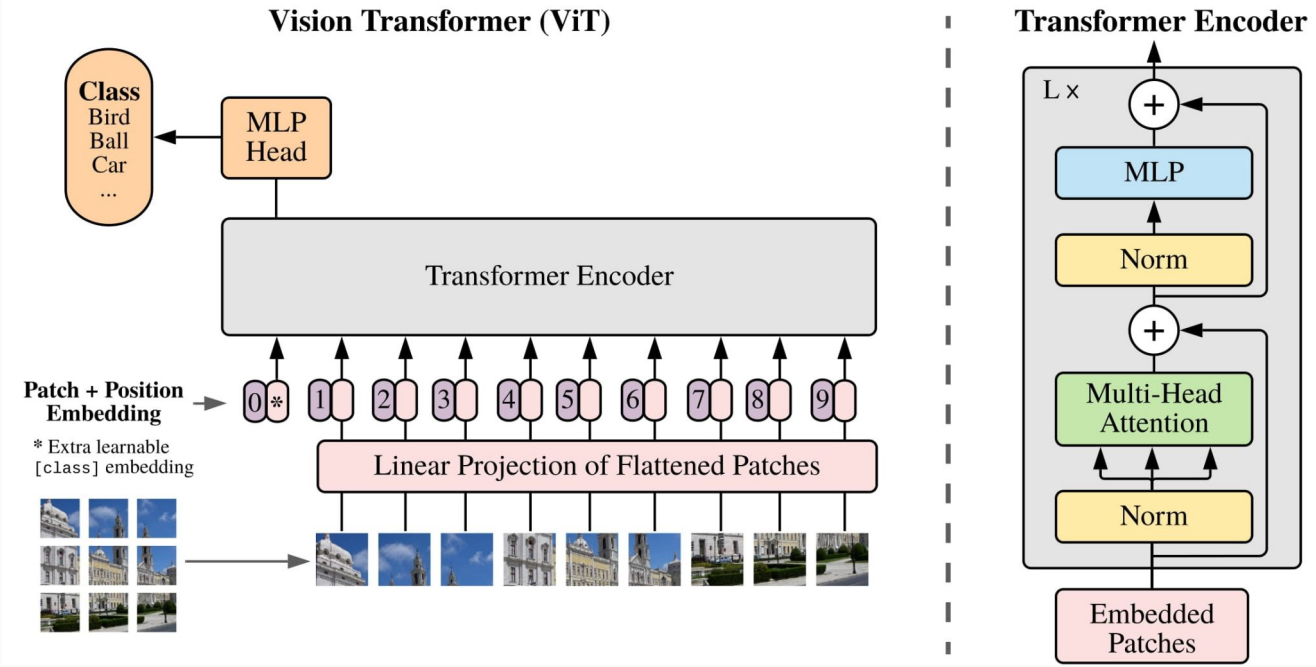
Tareas

Clasificación de audio y reconocimiento automático del habla: Wav2Vec2



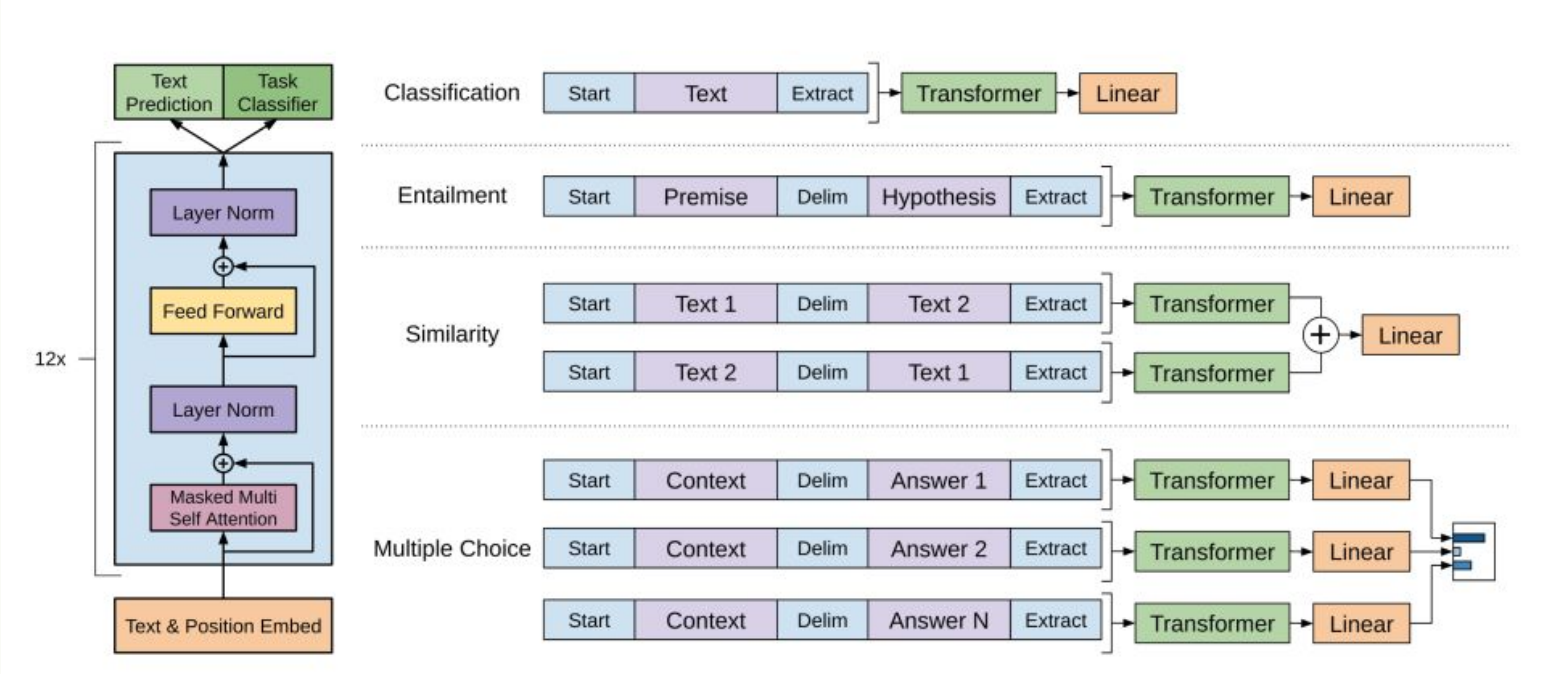
Tareas

Clasificación de imágenes: ViT



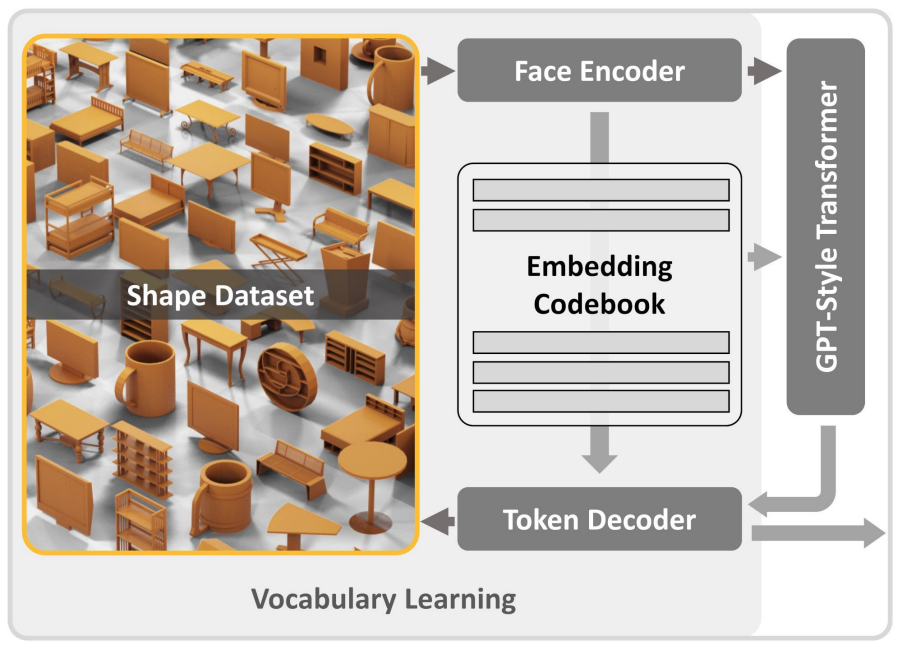
Tareas

Generación y clasificación de texto: GPT-2



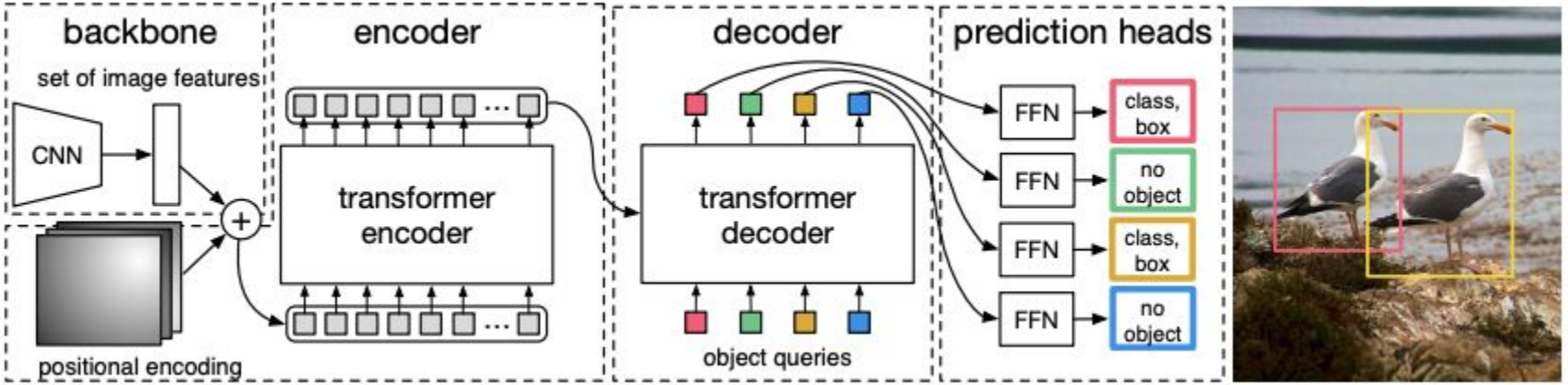
Tareas

Generación de mallas de triángulos: MeshGPT



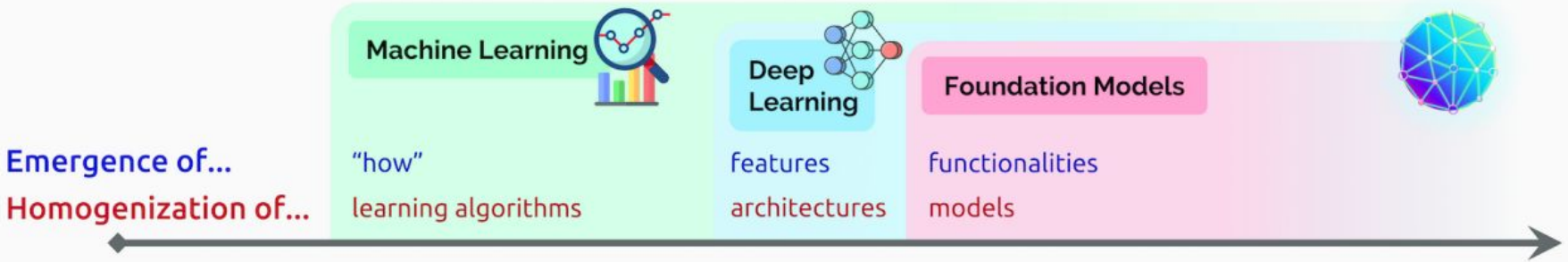
Tareas

Detección de objetos: DETR

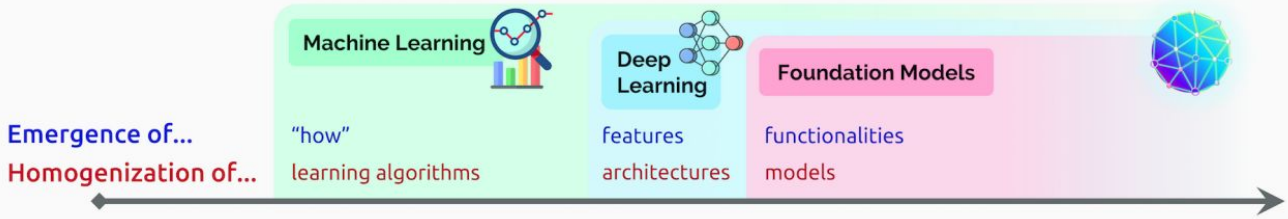


Modelos pre-entrenados y Modelos fundacionales

La evolución de la IA muestra un **aumento** en la **emergencia** de capacidades y la **homogeneización** de algoritmos, arquitecturas y modelos.

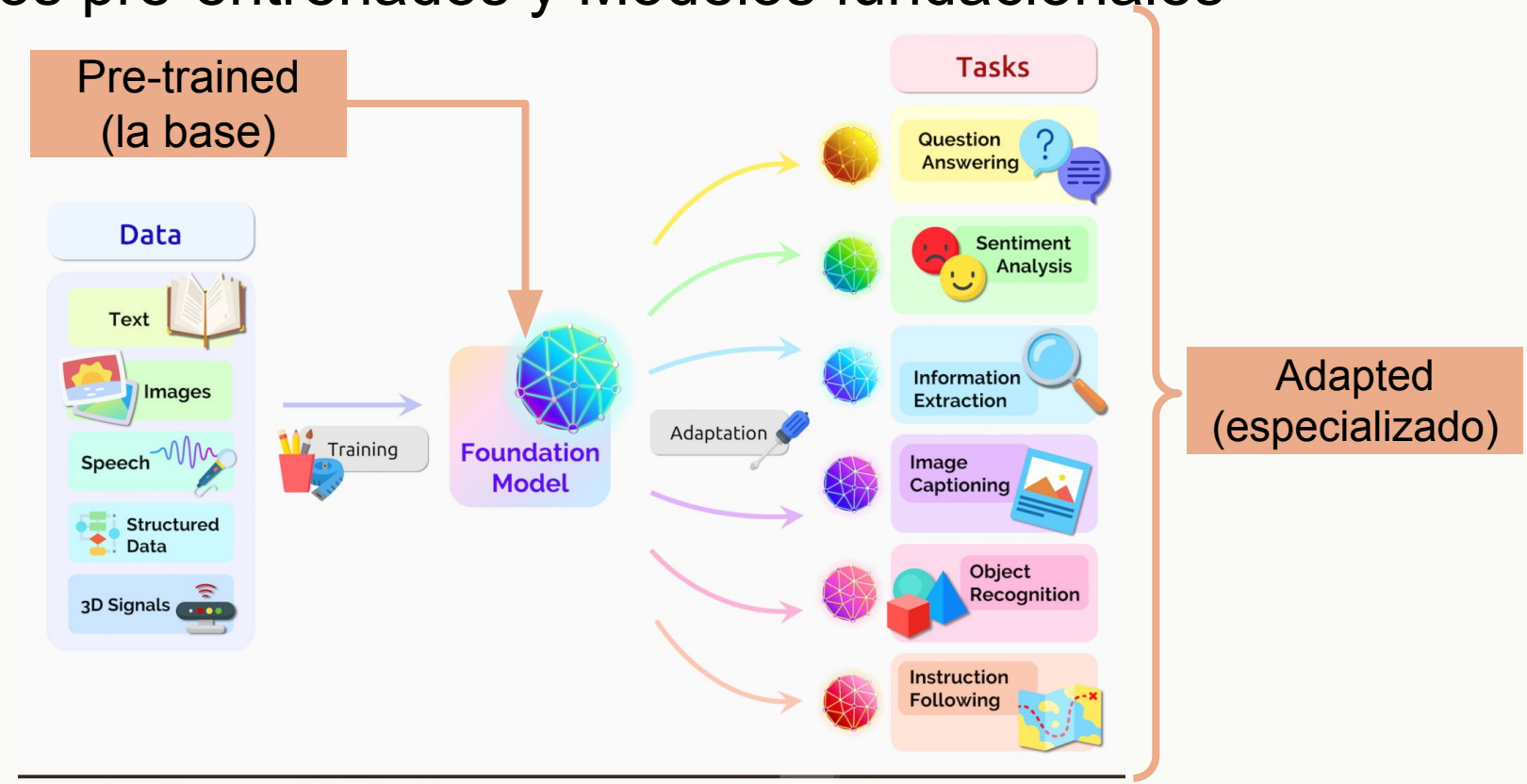


Modelos pre-entrenados y Modelos fundacionales



	Emerge	Homogeneización de
Aprendizaje Automático	**Aprender** a partir de la experiencia	Un **algoritmo** puede servir para varias tareas
Aprendizaje Profundo	Integración del aprendizaje de la **representación**	La **neurona artificial** como su base
Modelos Fundacionales	**Nuevas** funcionalidades debido a su tamaño	Un **modelo** puede servir para varias tareas

Modelos pre-entrenados y Modelos fundacionales



Introducción al Deep Learning

Día 4: Arquitecturas Avanzadas. Transformers

Manuel Germán y David de la Rosa

Universidad de Jaén

(mgerman,drrosa)@ujaen.es

Material Complementario

- https://jalammar.github.io/illustrated-transformer/
- https://llm-class.github.io/schedule.html (Week 4 lectures)
- Implementación (no optimizada) de un transformer desde cero https://github.com/mgermanm0/transformer-pytorch
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). Dive into deep learning. Cambridge University Press. - https://d2l.ai/ https://arxiv.org/abs/2106.11342 Capítulo 11
- Ejemplos HuggingFace https://huggingface.co/docs/transformers/notebooks
- Ejemplos detallados de las tareas https://huggingface.co/docs/transformers/tasks_explained