Introducción a los Transformers

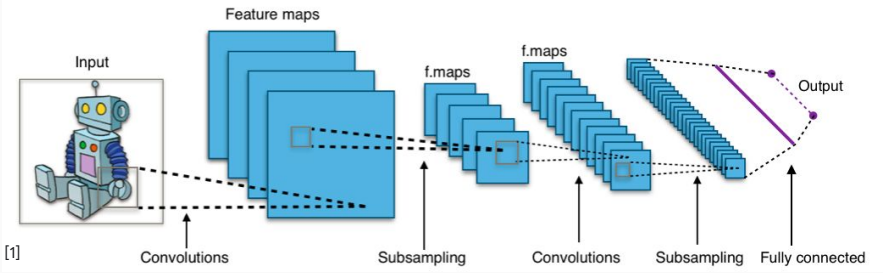
Basado en la conferencia de Lucas Beyer sobre Transformers

El paisaje clásico:

Una arquitectura por "especialidad"

Computer Vision

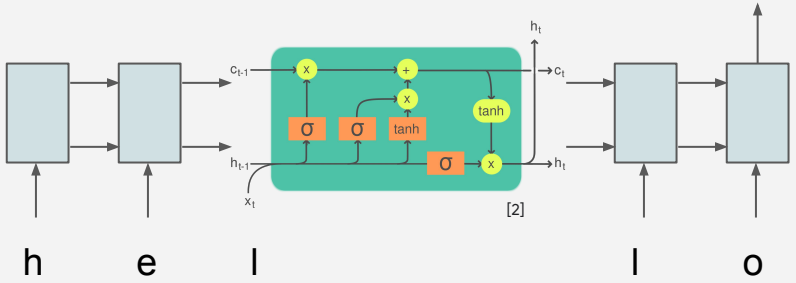
Convolutional NNs (+ResNets)



[1]

Natural Lang. Proc.

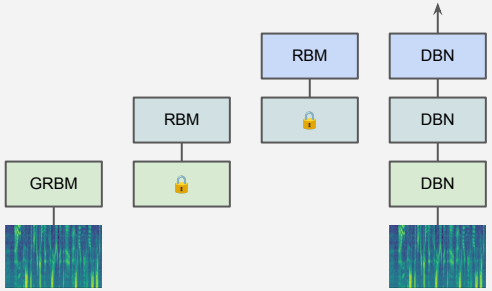
Recurrent NNs (+LSTMs)



h e l l o

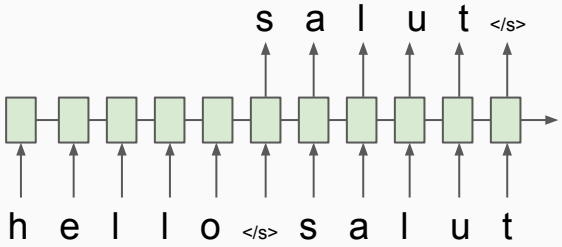
Speech

Deep Belief Nets (+non-DL)



Translation

Seq2Seq



RL

BC/GAIL

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \quad (17)$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta),$$
$$\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \quad (18)$$

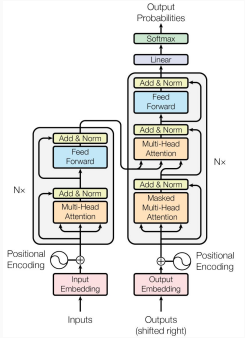
6: **end for**

[1] CNN image CC-BY-SA by Aphex34 for Wikipedia https://commons.wikimedia.org/wiki/File:Typical_cnn.png
[2] RNN image CC-BY-SA by GChe for Wikipedia https://commons.wikimedia.org/wiki/File:The_LSTM_Cell.svg

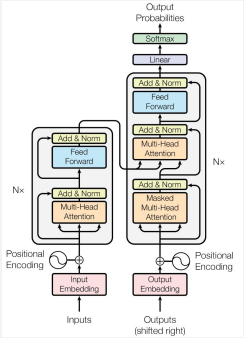
La adquisición del Transformer:

Una especialidad a la vez

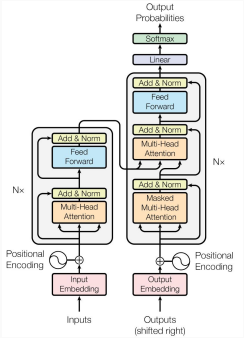
Computer Vision



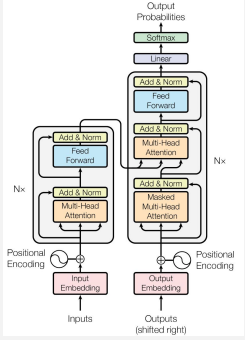
Natural Lang. Proc.



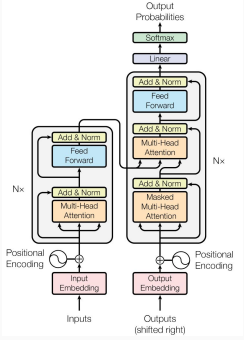
Reinf. Learning



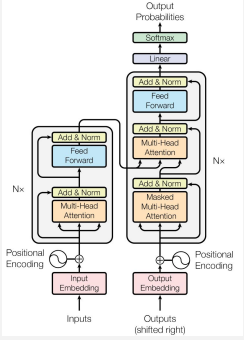
Speech



Translation



Graphs/Science



Transformer image source: "Attention Is All You Need" paper

Los origenes:

Traducción, alineación.

Neural Machine Translation by Jointly Learning to Align and Translate

2014, Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio

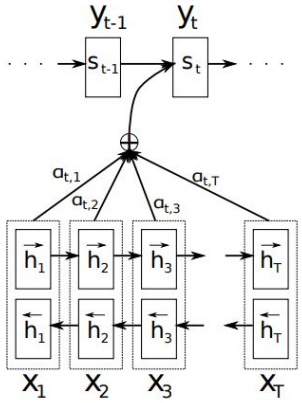
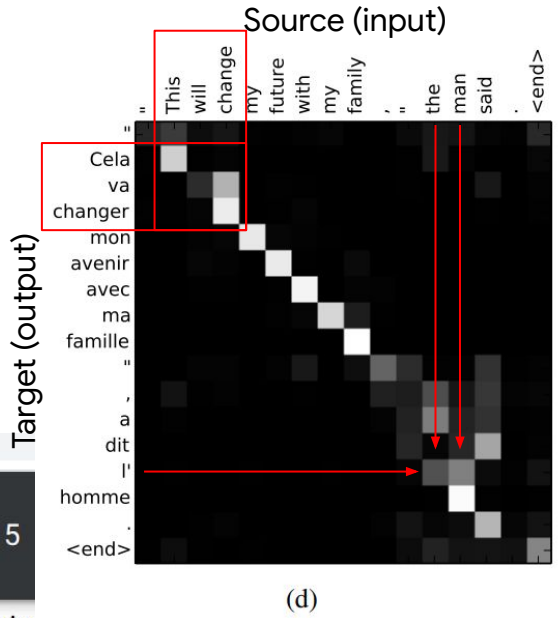


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.



4 / 15

attention 1/3

The probability α_{ij}, or its associated energy e_{ij}, reflects the importance of the annotation h_j with respect to the previous hidden state s_{i-1} in deciding the next state s_i and generating y_i. Intuitively, this implements a mechanism of attention in the decoder. The decoder decides parts of the source sentence to pay attention to. By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in the source sentence into a fixed-length vector. With this new approach the information can be spread throughout the sequence of annotations, which can be selectively retrieved by the decoder accordingly.

Attention Is All You Need

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

La atención es una función similar a una búsqueda "suave" en un diccionario de keys y query.

1. Los pesos de atención $a_{1:N}$ son similitudes query-key:

   $$\hat{a}_i = q \cdot k_i$$

   Normalizado vía softmax: $a_i = e^{\hat{a}i} / \sum_j e^{\hat{a}j}$

2. La salida z el promedio ponderado por atención de los valores $v_{1:N}$:

   $$z = \sum_i \hat{a}_i v_i = \hat{a} \cdot v$$

3. Usualmente, k y v se derivan de la misma entrada x:

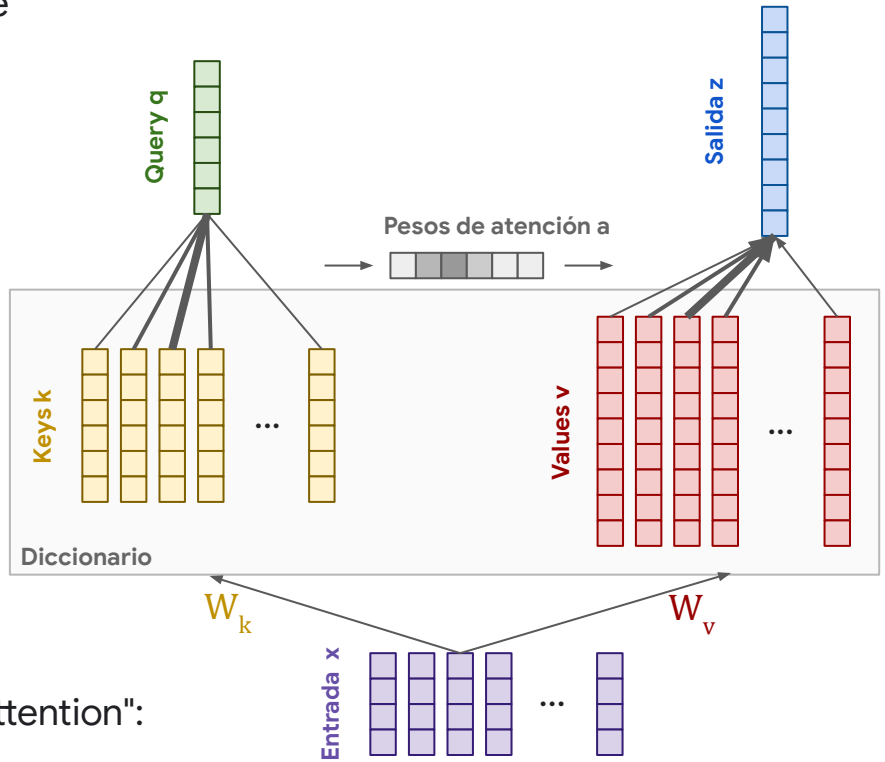
   $$k = W_k \cdot x \qquad v = W_v \cdot x$$

   El query q puede venir de una entrada separada y:

   $$q = W_q \cdot y$$

   O de la misma entrada x! Entonces lo llamamos "self attention":

   $$q = W_q \cdot x$$



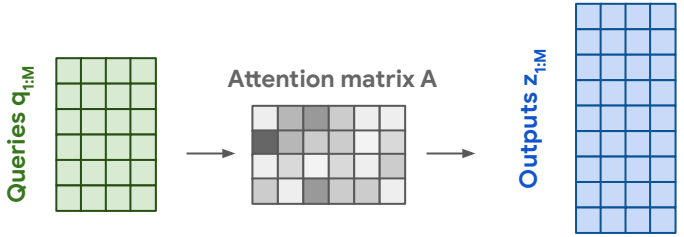
Attention Is All You Need

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

¡Pero eso no es todo! Hay algunos detalles más:

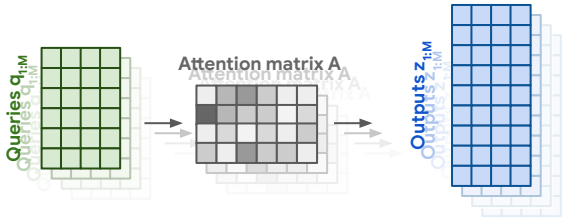
1. Usualmente usamos **muchas queries** $q_{1:M}$, no solo una.
   Apilarlas lleva a la matriz de Atención $\mathbf{A}_{1:N,1:M}$
   Y subsecuentemente a muchas salidas:

   $$z_{1:M} = \text{Attn}(q_{1:M}, x) = [\text{Attn}(q_1, x) \mid \text{Attn}(q_2, x) \mid \ldots \mid \text{Attn}(q_M, x)]$$



2. Usualmente usamos atención "**multi-head**". Esto significa que la operación se repite K veces y los resultados se concatenan a lo largo de la dimensión de características. Las **W** difieren.

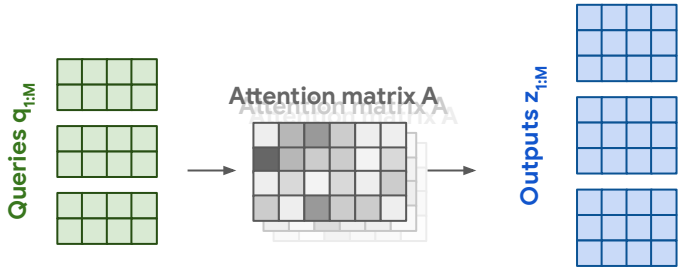
$$z_i = \begin{pmatrix} \text{Attn}_1(q_i, x) \\ \text{Attn}_2(q_i, x) \\ \ldots. \\ \text{Attn}_K(q_i, x) \end{pmatrix}$$



3. La formulación más comúnmente vista:

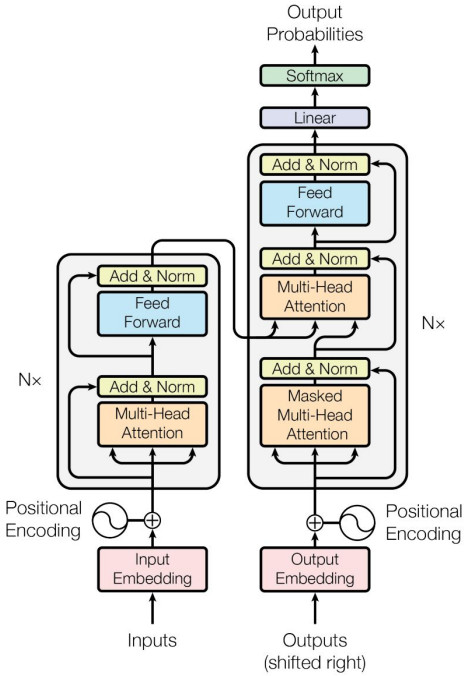
   $$z = \text{softmax}(QK'/\sqrt{d_{key}})V$$

   Observe que la complejidad es $O(N^2)$



Attention Is All You Need - La arquitectura Transformer

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

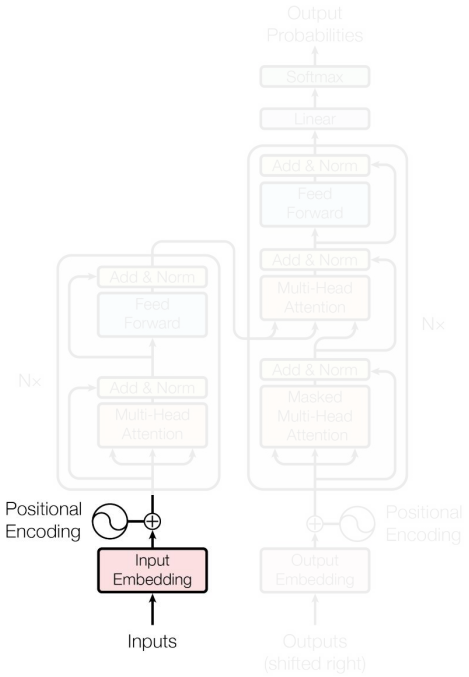


Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - La arquitectura Transformer

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Transformer image source: "Attention Is All You Need" paper

Entrada (Tokenización y) Embedding

El texto de entrada se divide primero en piezas. Pueden ser caracteres, palabras, "tokens":

```
"The detective investigated" -> [The_] [detective_] [invest] [igat] [ed_]
```

Los tokens son índices en el "vocabulario":

```
[The_] [detective_] [invest] [igat] [ed_] -> [3 721 68 1337 42]
```

Cada entrada del vocabulario corresponde a un vector de dimensión d_{model} aprendido.

```
[3 721 68 1337 42] -> [ [0.123, -5.234, ...], [...], [...], [...], [...] ]
```

Codificación posicional

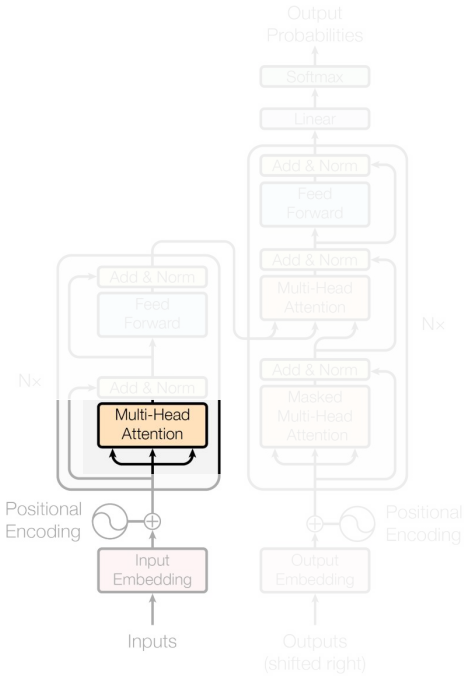
Recuerda que la atención es invariante a la permutación, ¡pero el lenguaje no lo es! ("The mouse ate the cat" vs "The cat ate the mouse")

Necesitamos codificar la posición de cada palabra; solo añade algo.

Algo cómo `[The_] + 10` `[detective_] + 20` `[invest] + 30` ... pero más inteligente.

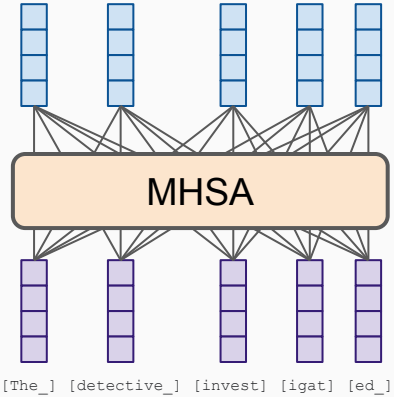
Attention Is All You Need - La arquitectura Transformer

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



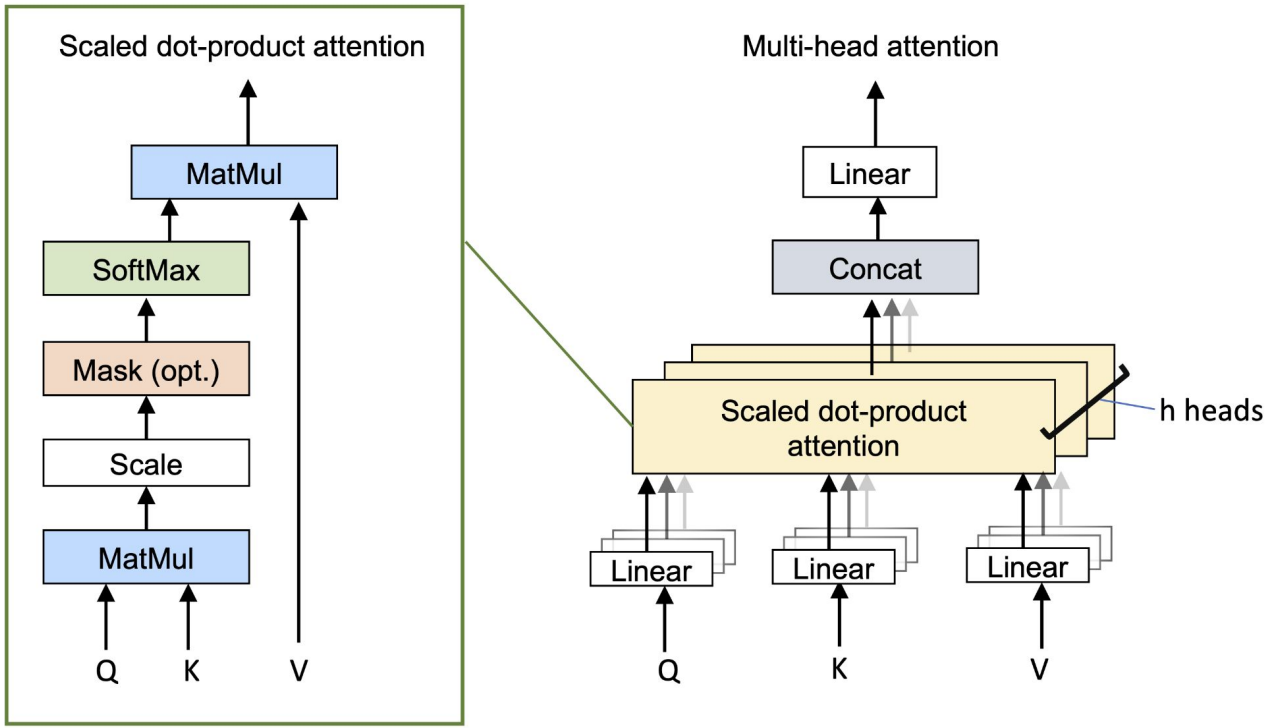
Multi-headed Self-Attention

Esto significa que la secuencia de entrada es para crear queries, keys, y values!
Cada token puede "mirar alrededor" de toda la entrada y decidir cómo actualizar su representación basándose en lo que ve.



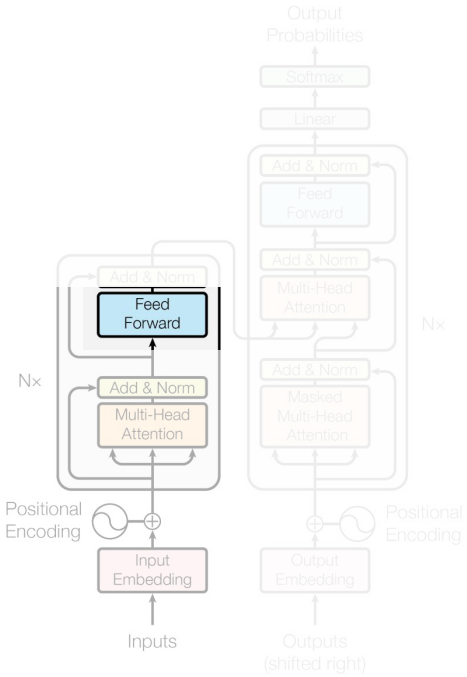
Transformer image source: "Attention Is All You Need" paper

Multi-head Attention (zoom)



Attention Is All You Need - La arquitectura Transformer

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

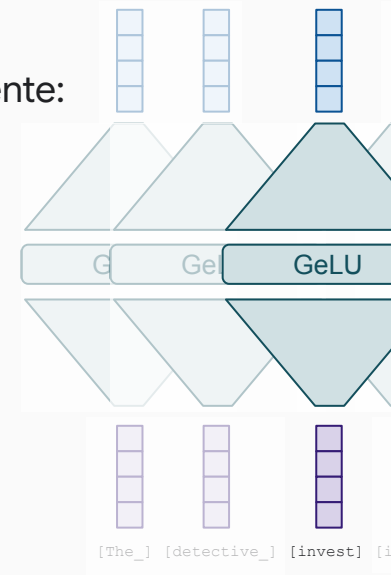


Transformer image source: "Attention Is All You Need" paper

MLP punto a punto

Una MLP simple aplicada a cada token individualmente:

$$z_i = W_2\,\mathrm{GeLU}(W_1 x + b_1) + b_2$$



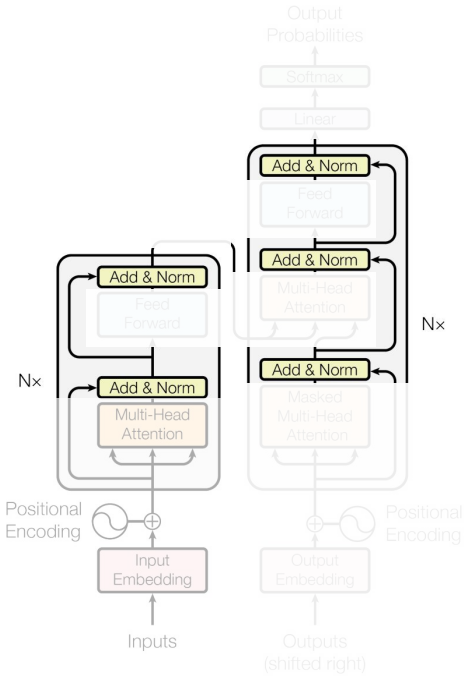
Piensa en ello como si cada token reflexionara por sí mismo sobre lo que ha observado previamente. Hay algunas evidencias débiles de que aquí es donde se almacena el "conocimiento del mundo" también.

Contiene la mayor parte de los parámetros. Cuando las personas crean modelos gigantes y sparse/moe, esto es lo que se vuelve gigante.

A algunas personas les gusta llamarlo convolución 1x1.

Attention Is All You Need - La arquitectura Transformer

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Conexiones residuales

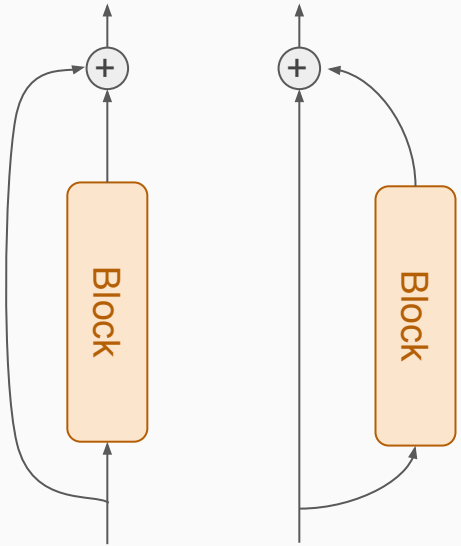
"Skip connection" == "Residual block"

La salida de cada módulo tiene exactamente la misma forma que su entrada.

Siguiendo los ResNets, el módulo calcula un "residuo" en lugar de un nuevo valor:

$$z_i = \text{Module}(x_i) + x_i$$

Esto mejora dramáticamente la capacidad de entrenamiento.



LayerNorm

La normalización también mejora dramáticamente el entrenamiento.

Hay **post-norm** (original)

$$z_i = \text{LN}(\text{Module}(x_i) + x_i)$$

y **pre-norm** (moderno)

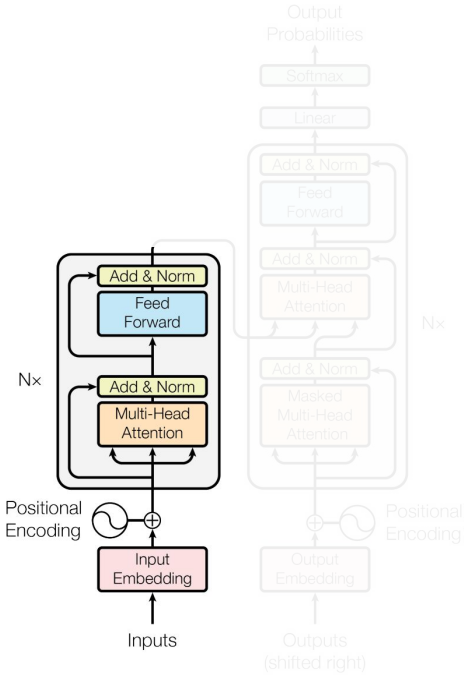
$$z_i = \text{Module}(\text{LN}(x_i)) + x_i$$

Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - La arquitectura Transformer

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Encoding / Encoder

Como las formas de entrada y salida son idénticas, podemos apilar N de estos bloques.

Típicamente, N=6 ("base"), N=12 ("large") o más.

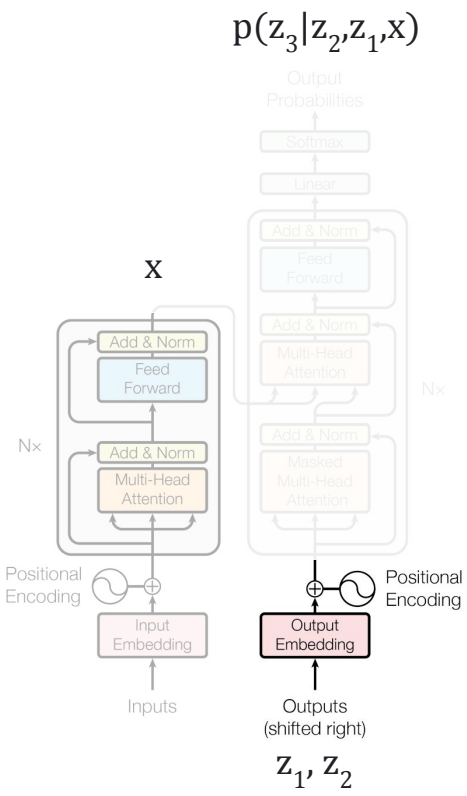
La salida del encoder es una versión "altamente procesada" (piensa: "de alto nivel, contextualizada") de los tokens de entrada, es decir, una secuencia.

Esto no tiene nada que ver con la salida solicitada todavía (piensa: traducción). Eso viene con el decoder..

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - La arquitectura Transformer

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Transformer image source: "Attention Is All You Need" paper

Decodificación / Decoder (alternativamente Generando / el Generador)

Lo que queremos modelas: $p(z|x)$

Por ejemplo, en traducción: $p(z \mid \text{"the detective investigated"})$ **∀z**

Parece imposible al principio, pero podemos descomponer exactamente en tokens:

$$p(z|x) = p(z_1|x)\ p(z_2|z_1,x)\ p(z_3|z_2,z_1,x)...$$

Es decir, podemos generar la respuesta un token a la vez.
Cada p es una pasada completa a través del modelo.

Para generar $p(z_3|z_2,z_1,x)$:

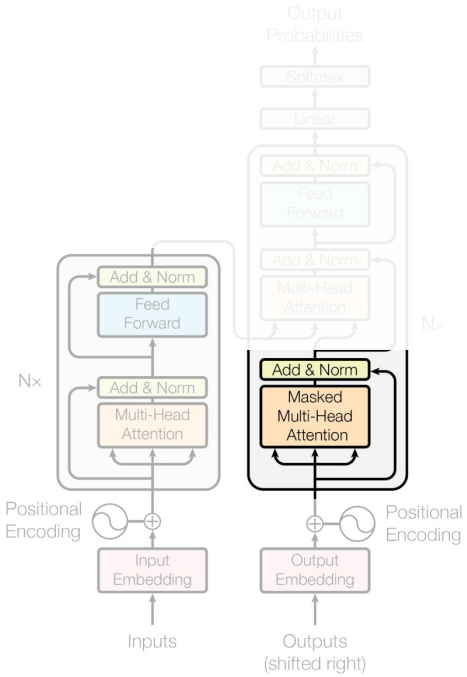
x viene del encoder,

z_1, z_2 es lo que hemos predicho hasta ahora, entra en el decoder.

Una vez que tenemos $p(z|x)$ aún necesitamos muestrear una oración como "le détective a enquêté". Muchas estrategias: *greedy*, *beam-search*, ...

Attention Is All You Need - La arquitectura Transformer

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Transformer image source: "Attention Is All You Need" paper

En tiempo de entrenamiento: Masked self-attention

Este es el mismo self-attention que en el encoder, para procesar lo que se ha decodificado hasta ahora, eg z_2,z_1 in $p(z_3|z_2,z_1,x)$, pero con un truco ...

Si tuviéramos que entrenar un sólo $p(z_3|z_2,z_1,x)$ a la vez: LENTO!

En su lugar, entrenar en todos los $p(z_i|z_{1:i},x)$ simultáneamente.

¿Cómo? En los pesos de atención para z_i, establecer todas las entradas $i{:}N$ to 0.

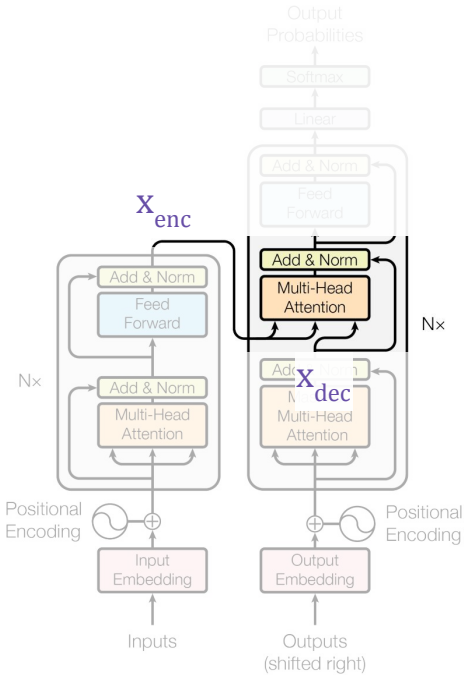
De esta manera, cada token solo ve los ya generados.

En tiempo de generación

No hay tal truco. Necesitamos generar un z_i a la vez. Esta es la razón por la que la decodificación autorregresiva es extremadamente lenta.

Attention Is All You Need - La arquitectura Transformer

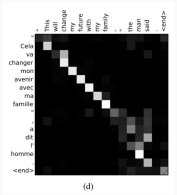
2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



"Cross" attention

Cada token decodificado puede "mirar" la salida del codificador:

$$\text{Attn}(q=W_q x_{dec}, k=W_k x_{enc}, v=W_v x_{enc})$$



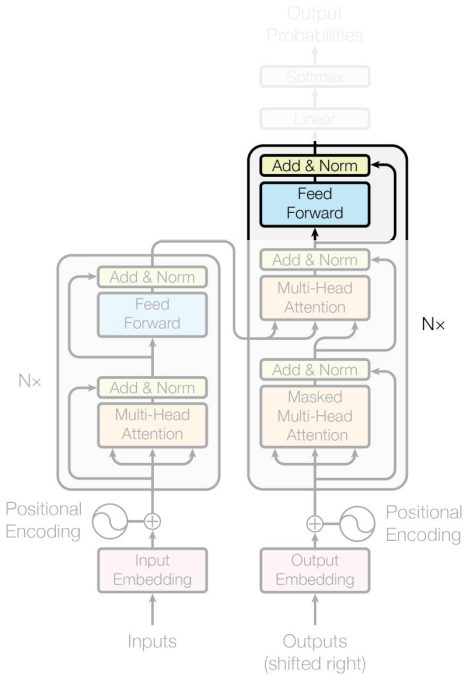
Esto es lo mismo que en el artículo de 2014.
Aquí es de donde viene el $|x$ en $p(z_3|z_2,z_1,x)$.

Debido a que el *self-attention* se usa tan ampliamente, la gente ha empezado a llamarla simplemente "*attention*".
Por lo tanto, ahora a menudo necesitamos llamar a esto explícitamente "*cross attention*".

Transformer image source: "Attention Is All You Need" paper ; heatmap image source: "Neural Machine Translation by Jointly Learning to Align and Translate" paper

Attention Is All You Need - La arquitectura Transformer

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

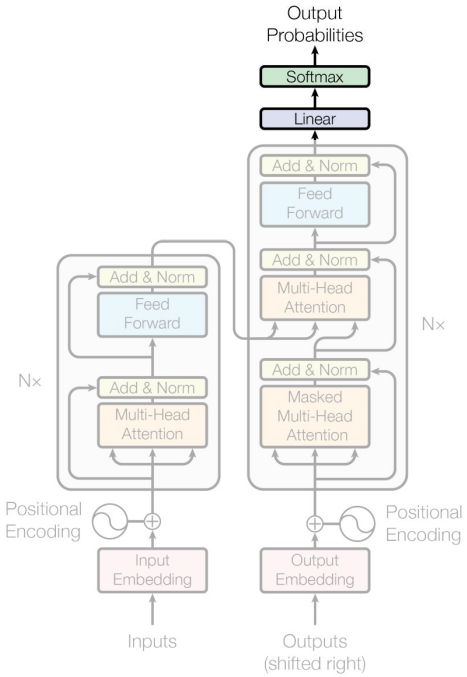


Feedforward and stack layers.

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - La arquitectura Transformer

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Transformer image source: "Attention Is All You Need" paper

Output layer

Asumamos que ya hemos generado K-1 tokens, generamos el siguiente.

El decoder se utilizó para recopilar toda la información necesaria para predecir una distribución de probabilidad para el siguiente token (K), sobre todo el vocabulario.

Simple:

proyección lineal del token K

normalización SoftMax

Attention Is All You Need - Summary and results

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

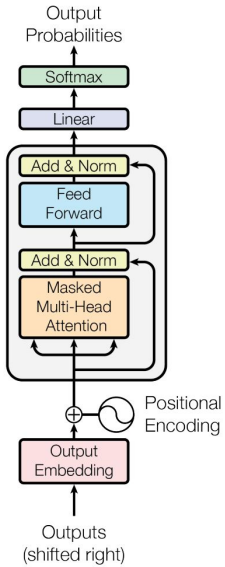
Animation source: https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html
Table source: "Attention Is All You Need" paper

La primera (1.5th) gran conquista:

Language Modeling / NLP

Decoder-only GPT

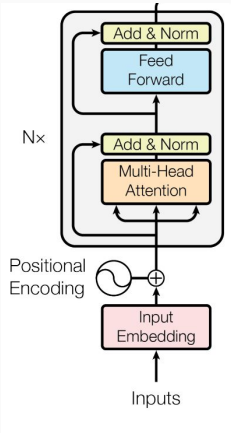
[sat_]



[START] [The_] [cat_]

Encoder-only BERT

[*] [*] **[sat_]** [*] **[the_]** [*]



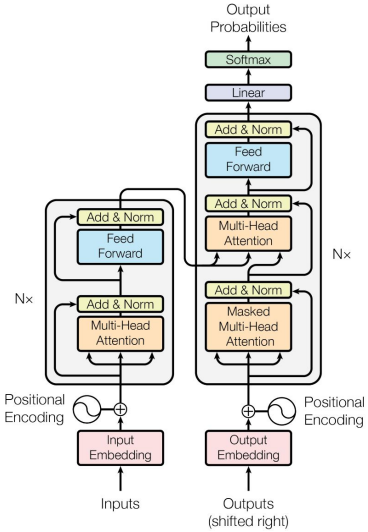
[The_] [cat_] **[MASK]** [on_] **[MASK]** [mat_]

Enc-Dec T5

Das ist gut.

A storm in Attala caused 6 victims.

This is not toxic.



Translate EN-DE: This is good.

Summarize: state authorities dispatched…

Is this toxic: You look beautiful today!

Transformer image source: "Attention Is All You Need" paper

GPT – Generative Pre-trained Transformer

- Fue desarrollado por OpenAI.
- Es unidireccional, ya que se entrena para predecir la siguiente palabra en una oración.

i. GPT (110 millones de parámetros) Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

ii. GPT-2 1(1500 millones de parámetros) Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners.

iii. GPT-3 (175 mil millones de parámetros) Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners.

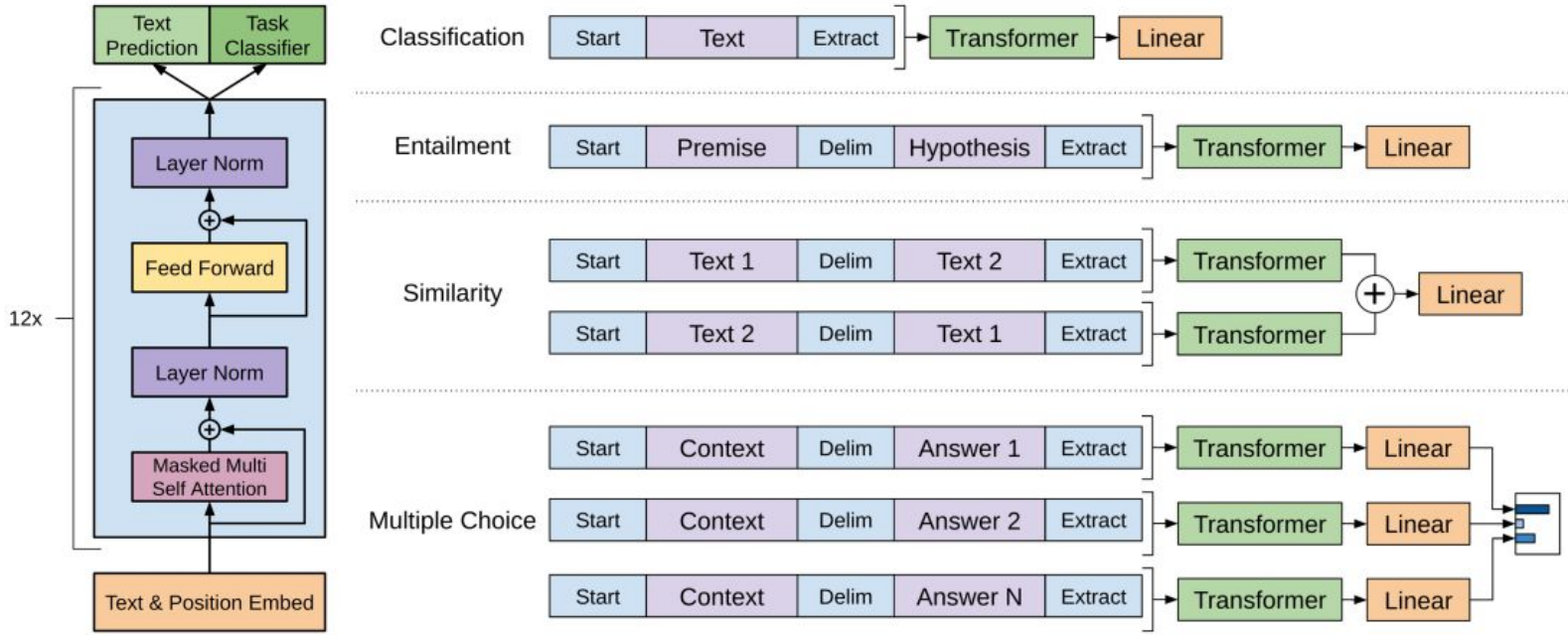
Conceptos importantes de GPT-1

- Cuello de botella: falta de datos etiquetados.
- El proceso de entrenamiento consiste de dos pasos (semi-supervisados):
  i. Pre-entrenamiento generativo (de datos sin etiquetar) -> aprendizaje auto-supervisado
  ii. *Fine-tuning* discriminativo (sobre datos etiquetados) -> aprendizaje supervisado
- Pre-entrenamiento sobre un conjunto de datos grande: BookCorpus (7000 libros).
- GPT está basado en la arquitectura del *decoder* del Transformer original.

Arquitectura de GPT-1 y Tareas Derivadas

Transformer architecture and training objectives

Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.



Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

GPT-1 Resultados

Method	Avg. Score	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	MNLI (acc)	QNLI (acc)	RTE (acc)
Transformer w/ aux LM (full)	74.7	45.4	91.3	82.3	82.0	**70.3**	**81.8**	**88.1**	**56.0**
Transformer w/o pre-training	59.9	18.9	84.0	79.4	30.9	65.5	75.7	71.2	53.8
Transformer w/o aux LM	**75.0**	**47.9**	**92.0**	**84.9**	**83.2**	69.8	81.1	86.9	54.4
LSTM w/ aux LM	69.1	30.3	90.5	83.2	71.8	68.1	73.7	81.1	54.6

BERT

Bidirectional Encoder Representations from Transformers

BERT

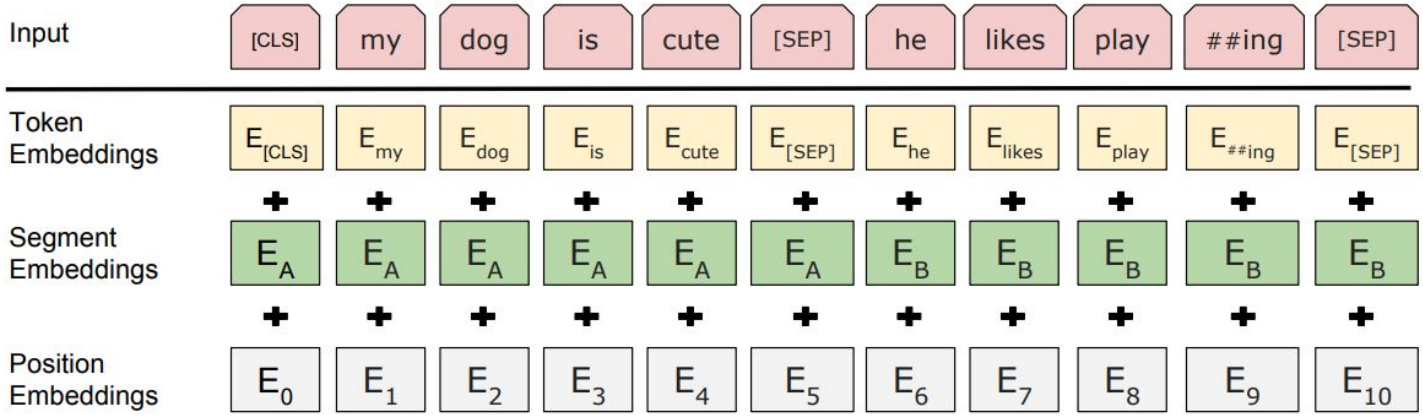
Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. (Google Research, 2018)

Arquitectura *encoder* de Transformer, multi-capa y bi-direccional

Es una arquitectura casi idéntica al transformer original, pero:

- Realiza enmascaramiento bidireccional
- Predice la oración siguiente como una tarea adicional de pre-entrenamiento

¿Cúales son las Entradas a BERT?

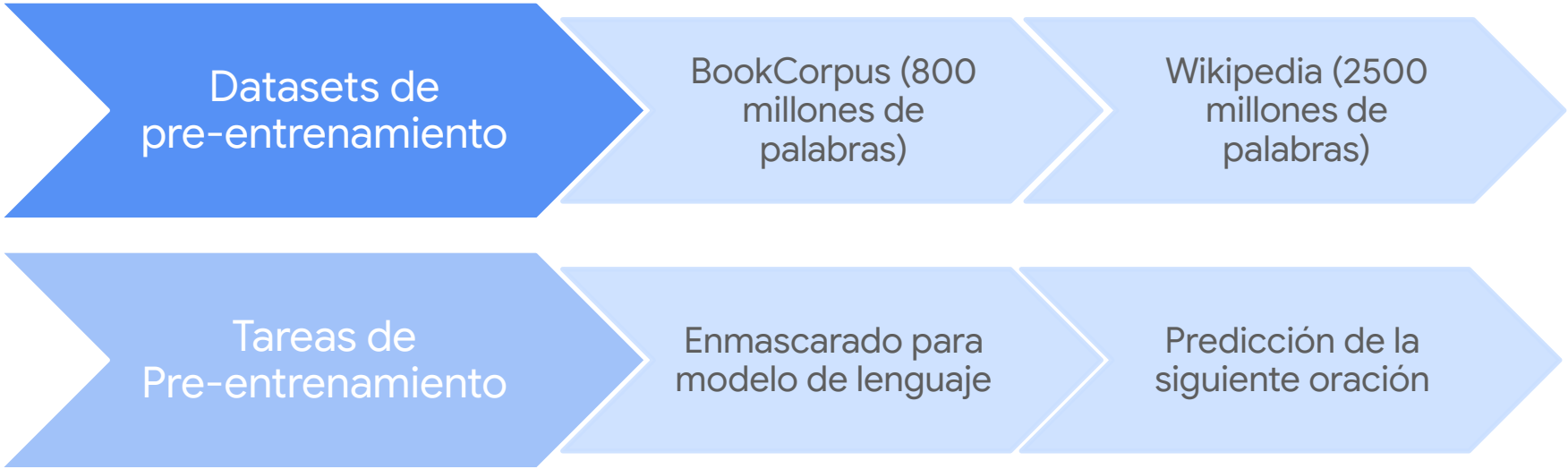


BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Token embeddings y WordPiece embeddings con un tamaño de vocabulario de 30,000

Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. (Google Research, 2018)

BERT: Tareas de Pre-entrenamiento



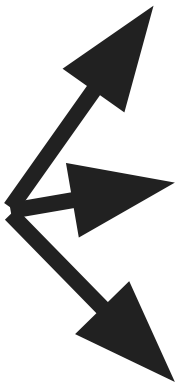
BERT: Tarea de pre-entrenamiento 1

Masked Language Model

Oración de entrada

Mi transformer favorito es Bumblebee

Marca el 15% de las palabras



80%: lo reemplaza con [MASK]

10%: lo reemplaza con una palabra aleatoria (prime)

10%: lo deja como está (favorito) para simular un escenario de fine-tuning

BERT: Tarea de pre-entrenamiento 1

Masked Language Model

Oración de entrada

Mi transformer favorito es Bumblebee

Oración aleatoriamente enmascarada

Mi transformer [MASK] es Bumblebee

0.1%	juguete
...	....
12 %	favorito
...	...
2%	bonito

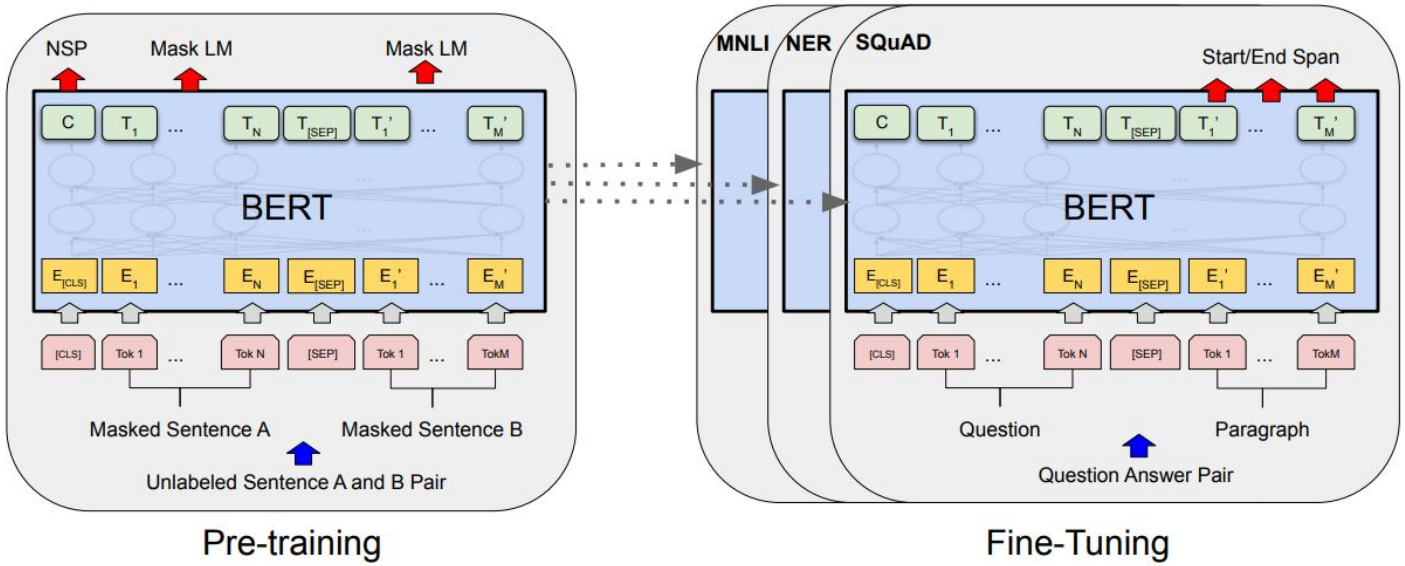
BERT: Tarea de Pre-entrenamiento 2

Predicción de la siguiente oración

Tarea de clasificación binaria balanceada (50% es “sigue”, 50% “no sigue”)

Entrada	[CLS] Me esta gustando [MASK] charla sobre [SEP] Transformers [MASK] aprendo bastante [SEP]
Etiqueta	Sigue
Entrada	[CLS] Me esta gustando [MASK] charla sobre [SEP] perritos [MASK] son lindos y bonitos [SEP]
Etiqueta	No sigue

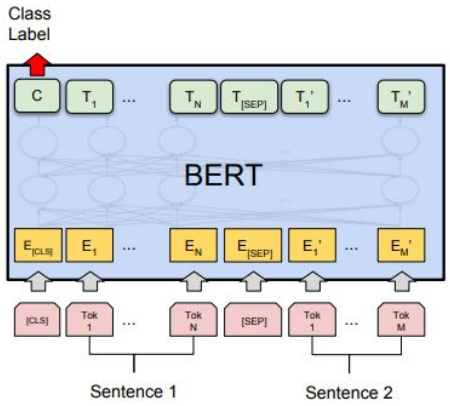
BERT – Pre-training y Tareas Posteriores



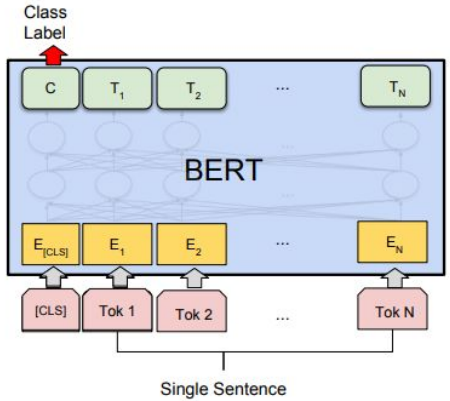
Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. (Google Research, 2018)

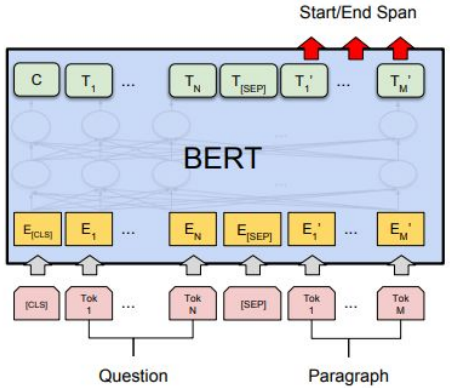
BERT – Pre-training y Tareas Posteriores



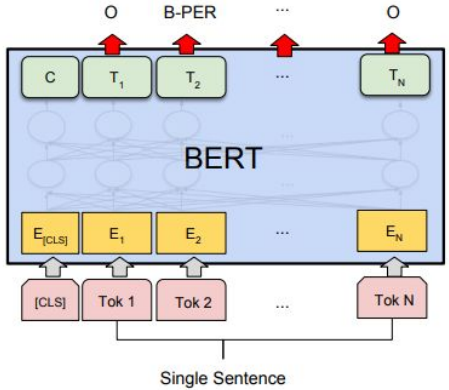
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. (Google Research, 2018)

Enfoques de Entrenamiento de un Transformer

1. **Pre-entrenamiento**: sobre conjunto de datos grandes (aprendizaje auto-supervisado).

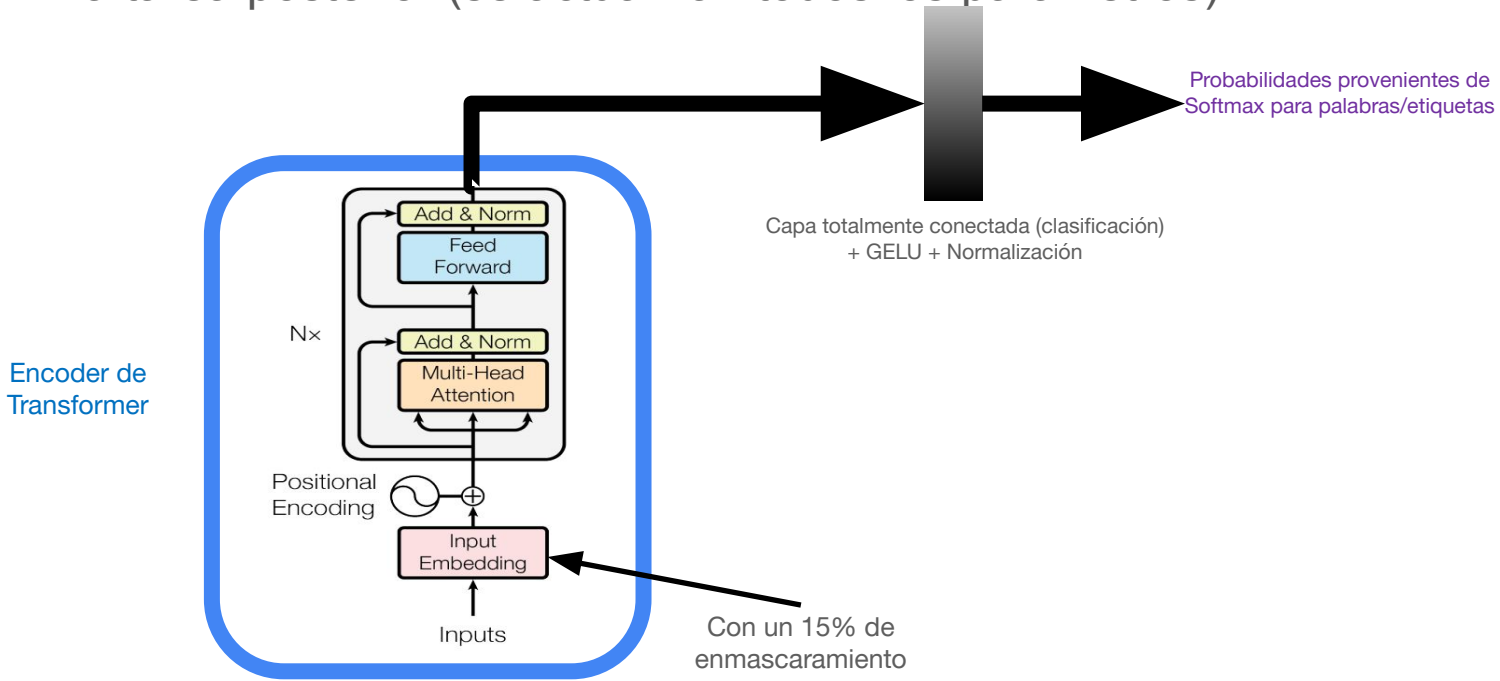
2. **Entrenamiento para tareas posteriores**: sobre conjuntos de datos etiquetados (aprendizaje supervisado)

   i. Enfoque de fine-tuning

   i. Enfoque con base en extracción de características

BERT Pre-entrenamiento y Enfoque de Fine-Tuning

- Se añade una capa de clasificación
- Se entrena todo el modelo sobre el conjunto de datos etiquetado para realizar la tarea posterior (se actualizan todos los parámetros)



Resultados BERT vs GPT-1

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The “Average” column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.8 BERT and OpenAI GPT are single model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

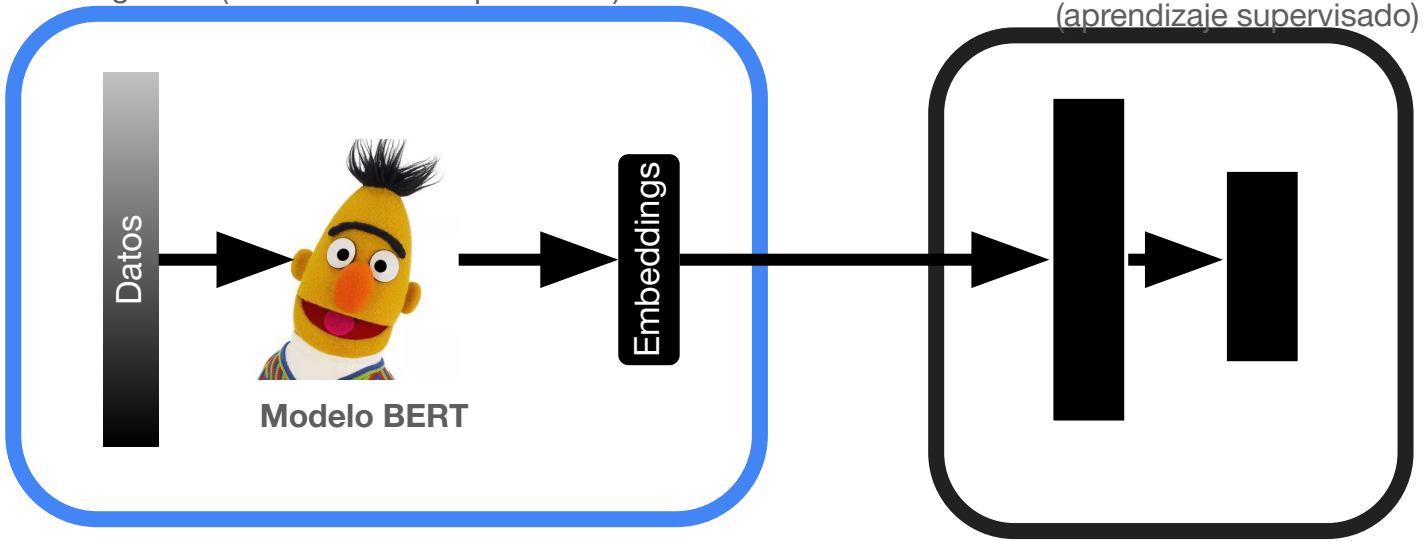
Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. (Google Research, 2018)

BERT Pre-entrenamiento y Entrenamiento con base en características

- Se congela a BERT después del pre-entrenamiento
- Se crean los embeddings a partir de BERT para el conjunto de datos etiquetado para la tarea (posterior) y se entrena un nuevo modelo con base en esos embeddings.

1. **Se baja a BERT**, pre-entrenado en un corpus grande (de forma auto-supervisada)

2. **Se entrena con base en las características** extraídas sobre una tarea específica (aprendizaje supervisado)



Resultados: fine-tuning y características

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. (Google Research, 2018)

GPT-2

Language Models are Unsupervised Multitask Learners

GPT – Generative Pre-trained Transformer

- Fue desarrollado por OpenAI.
- Es unidireccional, ya que se entrena para predecir la siguiente palabra en una oración.

i. GPT (110 millones de parámetros) Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

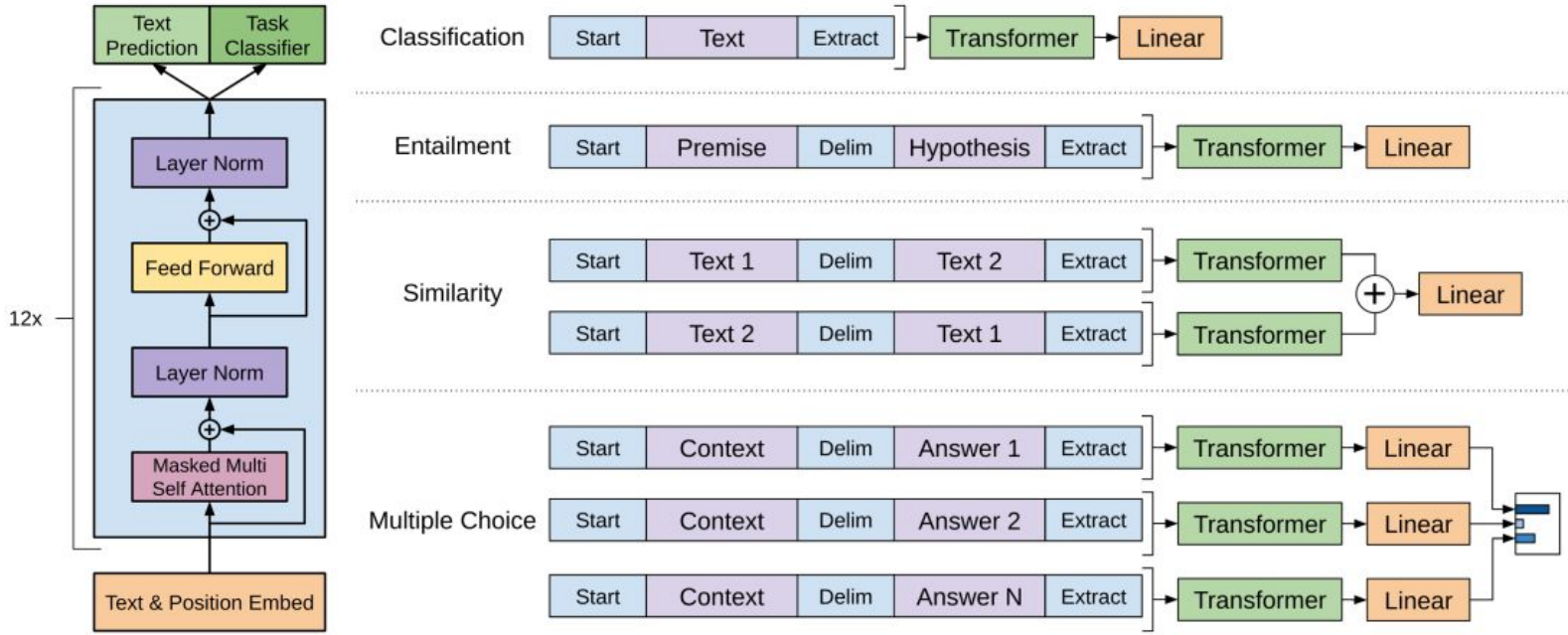
ii. GPT-2 1(1500 millones de parámetros) Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners.

iii. GPT-3 (175 mil millones de parámetros) Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners.

Arquitectura de GPT-1 y Tareas Derivadas

Transformer architecture and training objectives

Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.



Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

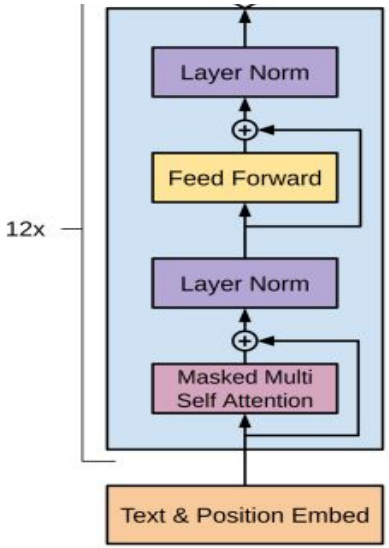
Conceptos importantes en GPT-2

Es unidireccional como GPT-1

Comparada con GPT-1:

- Tiene un modelo más grande
- Conjunto de datos sin etiquetar más grande
- No usa fine-tuning (usa zero-shot transfer)

Arquitectura de GPT-2



- En general, es similar a GPT-1 (un decoder de Transformer)
- Hay pequeños cambios cómo reacomodación de las capas de normalización y las capas residuales.
- Se incrementa el tamaño del vocabulario de 30,000 a 50,257
- Se incrementa el tamaño del contexto, pasando de 512 a 1024 tokens
- Pasa de ser un modelo de 110 millones de parámetros a 1500 millones.

Datasets de Entrenamiento para GPT-2

WebText (millones de páginas web)

Conjunto de datos enfatizados en alta calidad

Con base en posts de Reddit con más de 3 de karma:

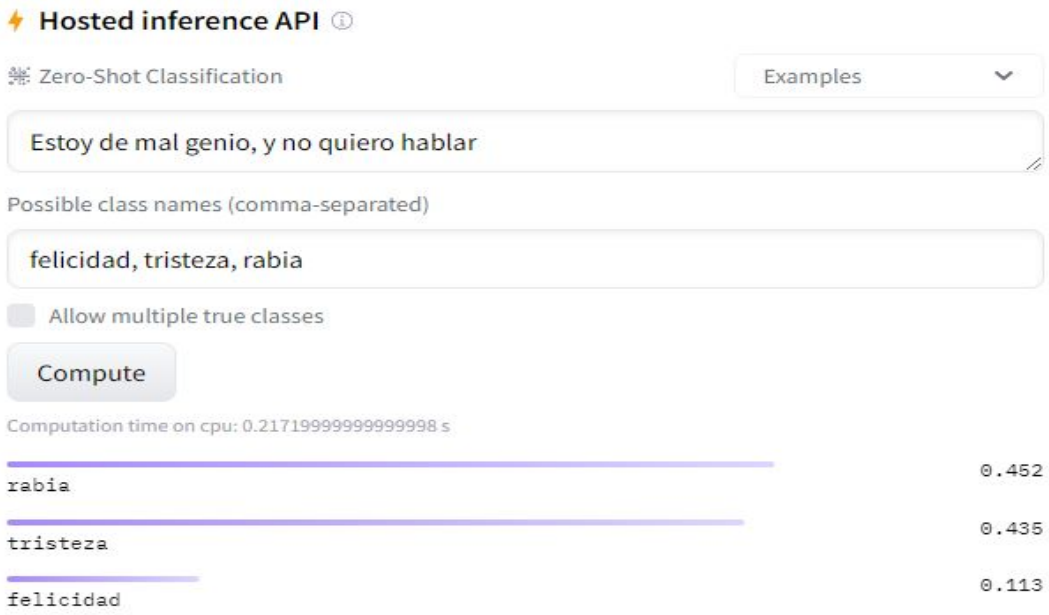
Se encuentran 45 millones de links a páginas webs

Después de preprocesar y limpiar se obtienen 8 millones de documentos

40 GB de texto

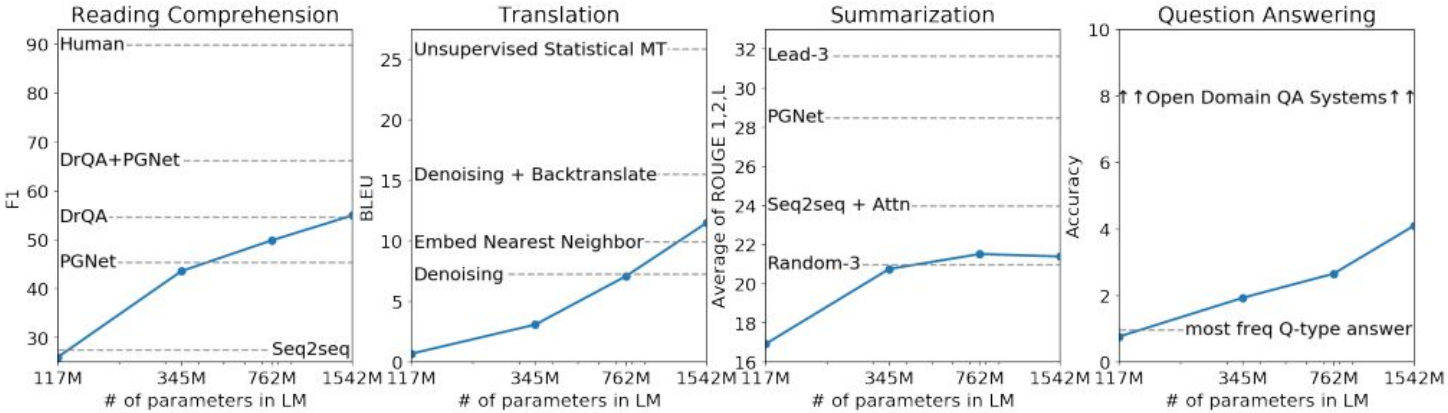
¿Qué es Zero-Shot Transfer (learning)?

A diferencia de GPT-1, no se necesita ninguna instrucción específica, ni ningún reacomodamiento para realizar tareas específicas



https://huggingface.co/Recognai/zeroshot_selectra_small

Resultados GPT



Zero-shot task performance of WebText LMs as a function of model size on many NLP tasks. Reading Comprehension results are on CoQA (Reddy et al., 2018), translation on WMT-14 Fr-En (Artetxe et al., 2017), summarization on CNN and Daily Mail (See et al., 2017), and Question Answering on Natural Questions (Kwiatkowski et al., 2019). Section 3 contains detailed descriptions of each result.

Radford, A . et al. (2019). Language models are unsupervised multitask learners.

Resultados GPT

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

Radford, A . et al. (2019). Language models are unsupervised multitask learners.

GPT-3

Language Models are Few-Shot Learners

GPT – Generative Pre-trained Transformer

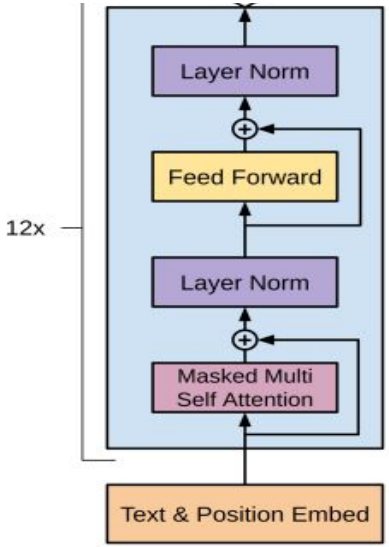
- Fue desarrollado por OpenAI.
- Es unidireccional, ya que se entrena para predecir la siguiente palabra en una oración.

i. GPT (110 millones de parámetros) Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

ii. GPT-2 1(1500 millones de parámetros) Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners.

iii. GPT-3 (175 mil millones de parámetros) Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners.

Arquitectura de GPT-3



- En general, es similar a GPT-1 (un decoder de Transformer)
- 175 mil millones de parámetros en vez de 1500 millones (tiene más capas, etc.)
- Se dobla el tamaño de contexto (2048 tokens en vez de 1024)
- Más grandes embeddings de palabras (12,8k en vez de 1,6k)
- Usa mecanismo de atención de un Sparse Transformer

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences With Sparse Transformers, 2019.

Datasets de Entrenamiento para GTP-3

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Datasets used to train GPT-3. “Weight in training mix” refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

Brown, T. B., et al. (2020). Language models are few-shot learners.

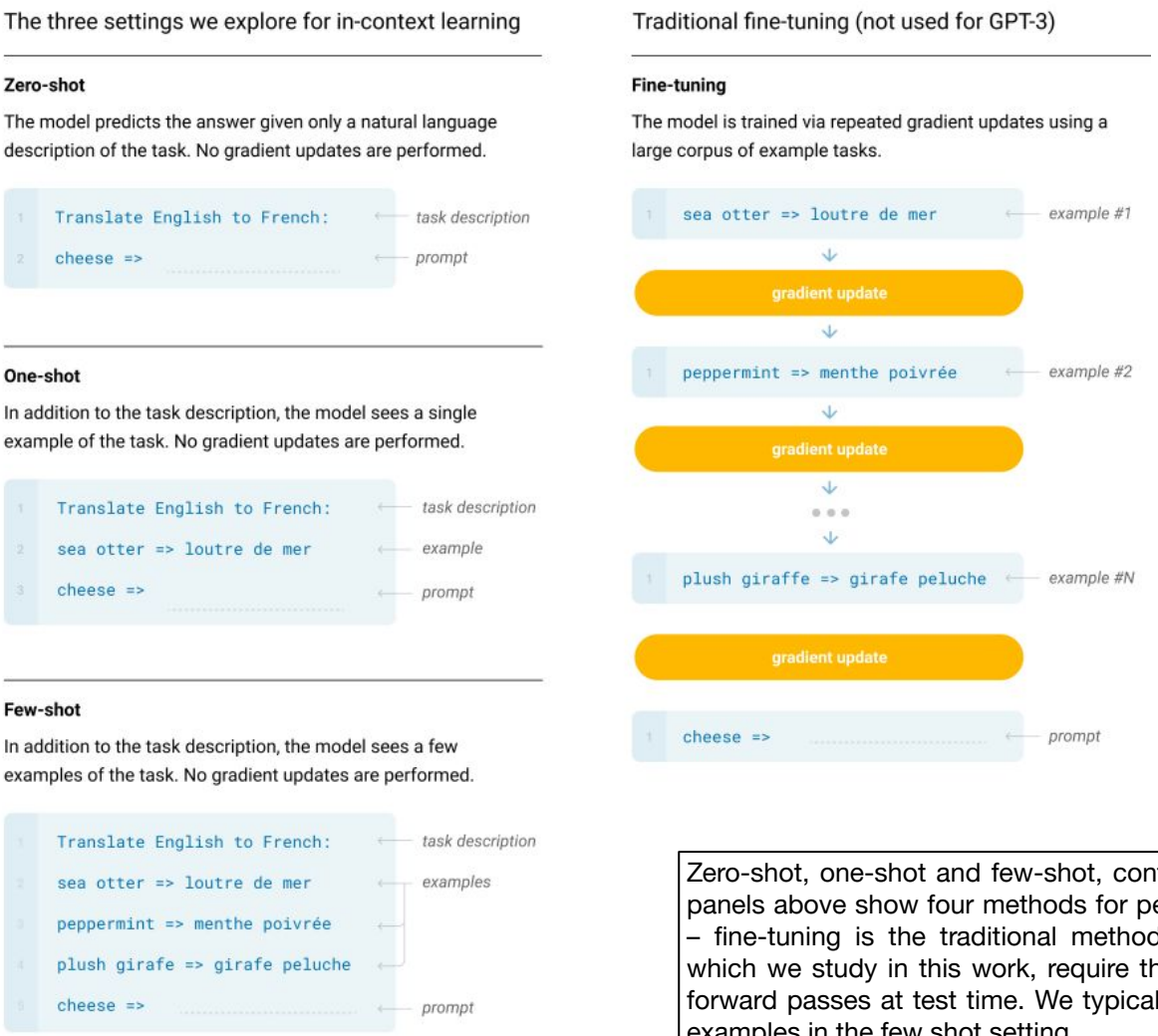
GPT-3 Aprende Tareas de Forma Implícita

Mientras aprende a predecir la palabra siguiente

Brown, T. B., et al. (2020). Language models are few-shot learners.

Language model meta-learning. During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term “in-context learning” to describe the inner loop of this process, which occurs within the forward-pass upon each sequence. The sequences in this diagram are not intended to be representative of the data a model would see during pre-training, but are intended to show that there are sometimes repeated sub-tasks embedded within a single sequence.

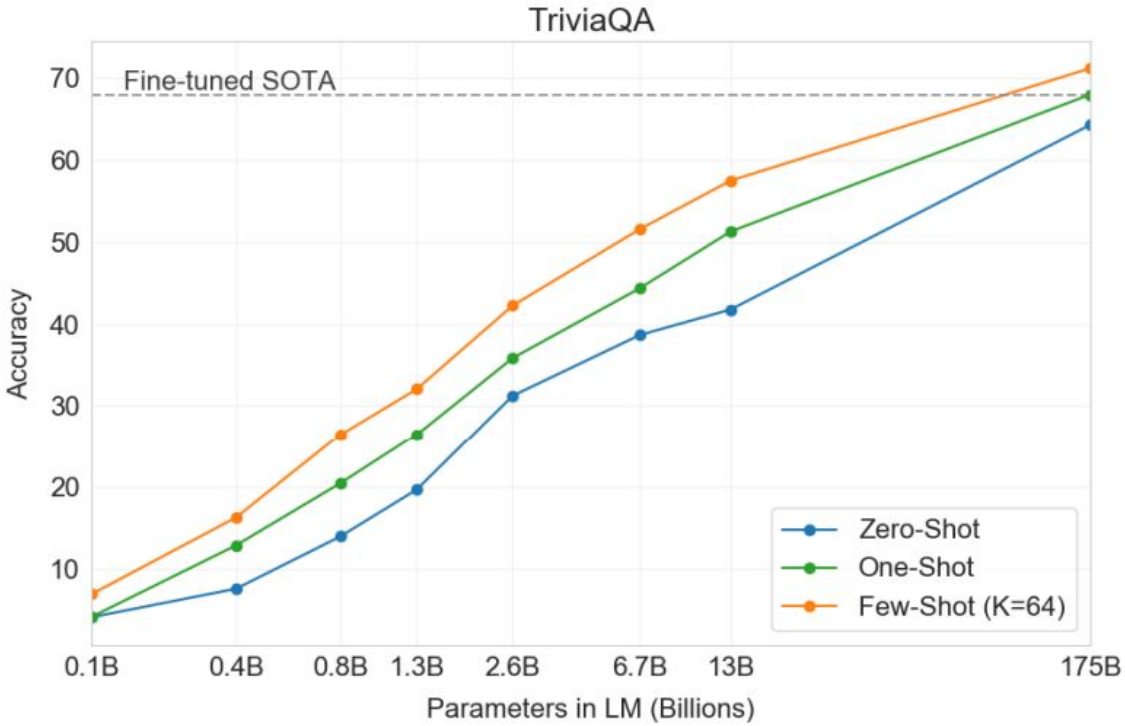
Mostrando Ejemplos Vs Fine-Tuning



Zero-shot, one-shot and few-shot, contrasted with traditional fine-tuning. The panels above show four methods for performing a task with a language model – fine-tuning is the traditional method, whereas zero-, one-, and few-shot, which we study in this work, require the model to perform the task with only forward passes at test time. We typically present the model with a few dozen examples in the few shot setting.

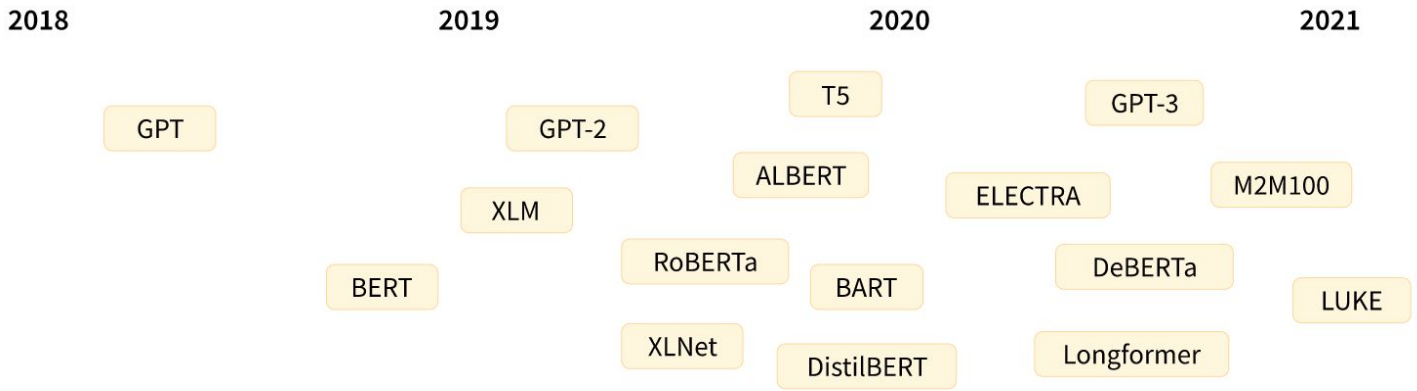
Brown, T. B., et al. (2020). Language models are few-shot learners.

Algunos de los Muchos Resultados



On TriviaQA GPT3's performance grows smoothly with model size, suggesting that language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG

Arquitecturas Populares de Transformers



https://huggingface.co/

Costo de Transformers

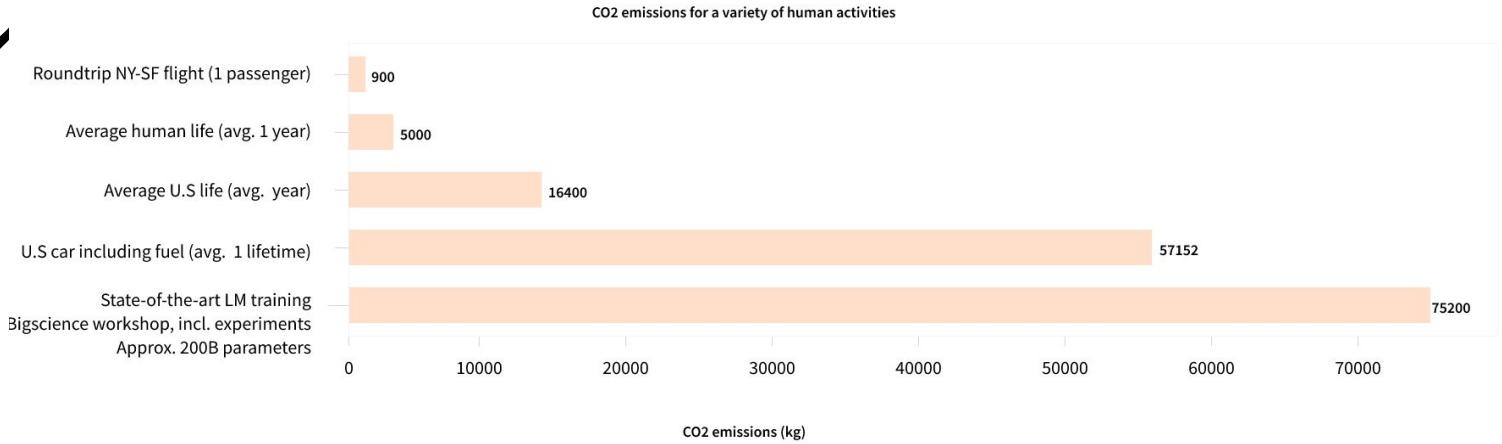
Los Transformers no son Baratos

✔ Entrenar modelos de Transformers es muy costoso:

- $2.5k - $50k (110 million parameter model)
- $10k - $200k (340 million parameter model)
- $80k - $1.6m (1.5 billion parameter model)

Sharir, O., Peleg, B., & Shoham, Y. (2020). The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*.

✔



La segunda gran conquista:

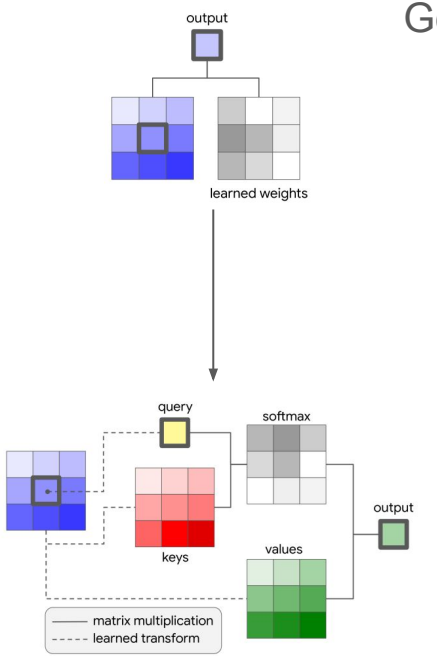
Computer Vision

Enfoques previos

1. Sobre pixeles pero localizados o factorizados

Muchos trabajos anteriores intentaron introducir la self-attention a nivel de píxeles.

Para 224px^2, eso es una secuencia de longitud 50k, lo cual es demasiado!



Generalmente reemplaza conv 3x3 en ResNet:

stage	output	ResNet-50	LR-Net-50 (7×7, m=8)
res1	112×112	7×7 conv, 64, stride 2	**1×1, 64** **7×7 LR, 64, stride 2**
		3×3 max pool, stride 2	3×3 max pool, stride 2
res2	56×56	[1×1, 64 3×3 conv, 64 1×1, 256] ×3	[1×1, 100 **7×7 LR, 100** 1×1, 256] ×3
res3	28×28	[1×1, 128 3×3 conv, 128 1×1, 512] ×4	[1×1, 200 **7×7 LR, 200** 1×1, 512] ×4
res4	14×14	[1×1, 256 3×3 conv, 256 1×1, 1024] ×6	[1×1, 400 **7×7 LR, 400** 1×1, 1024] ×6
res5	7×7	[1×1, 512 3×3 conv, 512 1×1, 2048] ×3	[1×1, 800 **7×7 LR, 800** 1×1, 2048] ×3
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params		**25.5**×10^6	**23.3**×10^6
FLOPs		**4.3**×10^9	**4.3**×10^9

Resultados:

Suelen ser "normales", nada del otro mundo

No justifica una mayor complejidad

No justifica la desaceleración por las convoluciones

Ejemplos:

Non-local NN (Wang et.al. 2017)

SASANet (Stand-Alone Self-Attention in Vision Models)

HaloNet (Scaling Local Self-Attn for Parameter Efficient...)

LR-Net (Local Relation Networks for Image Recognition)

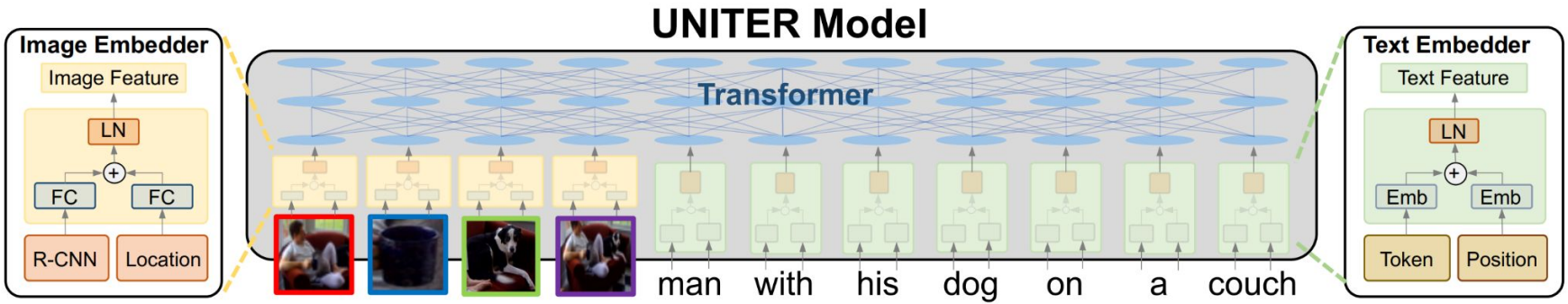
SANet (Exploring Self-attention for Image Recognition)

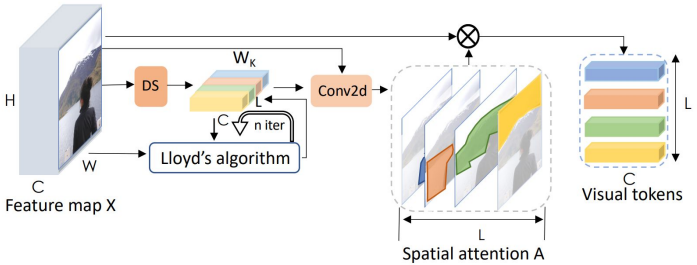
...

Image credit: Stand-Alone Self-Attention in Vision Models by Ramachandran et.al.
Image credit: Local Relation Networks for Image Recognition by Hu et.al.

Enfoques previos

2. Globalmente, después/dentro de una CNN completa, ¡o incluso un detector/segmentador!





Contras:

El resultado es muy complejo, a menudo una arquitectura de entrenamiento de múltiples etapas.

No es a partir de píxeles, es decir, el transformer no puede "aprender a corregir" los errores de la CNN (a menudo congelada).

Ejemplos:

DETR (Carion, Massa et.al. 2020)
UNITER (Chen, Li, Yu et.al. 2019)
VisualBERT (Li et.al. 20190)
Visual Transformers (Wu et.al. 2020)
ViLBERT (Lu et.al. 2019)
etc...

Image credit: UNITER: UNiversal Image-TExt Representation Learning by Chen et.al.
Image credit: Visual Transformers: Token-based Image Representation and Processing for Computer Vision by Wu et.al.

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

2020, A Dosovitskiy, L Beyer, A Kolesnikov, D Weissenborn, X Zhai, T Unterthiner, M Dehghani, M Minderer, G Heigold, S Gelly, J Uszkoreit, N Houlsby

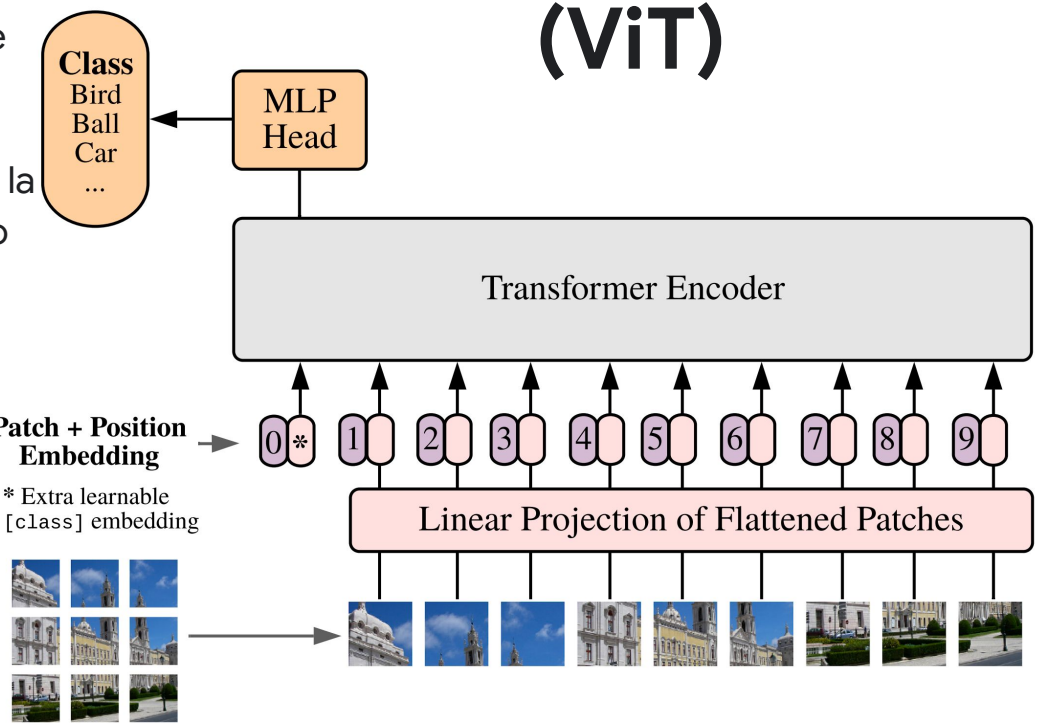
Muchos trabajos anteriores intentaron introducir la self-attention a nivel de píxel.

Para 224px², eso es una longitud de secuencia de 50k, ¡demasiado!

Por lo tanto, la mayoría de los trabajos restringen la atención a vecindarios locales de píxeles, o como mecanismo de alto nivel sobre las detecciones.

El **avance clave** en el uso de la arquitectura completa del Transformer, de forma independiente, fue **"tokenizar" la imagen cortandola en parches** de 16px², y tratar cada parche como un token, por ejemplo, incrustandolo (embedding) en el espacio de entrada.

Vision Transformer (ViT)

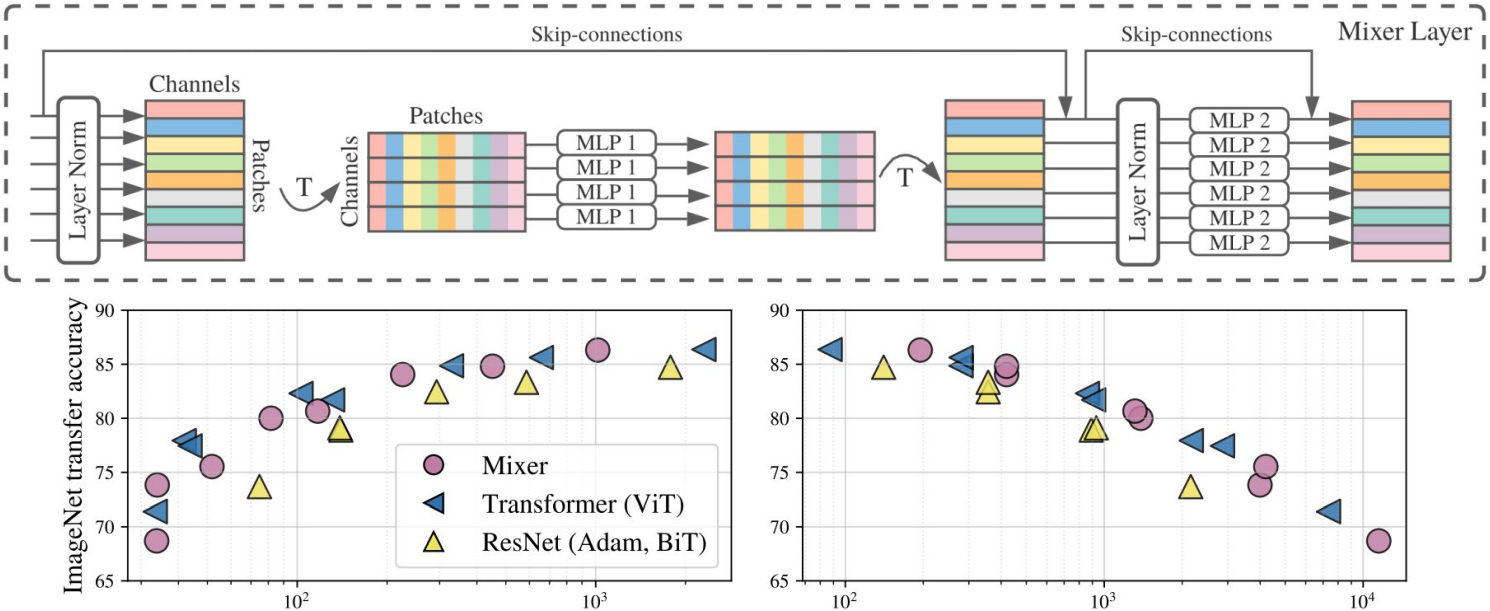


Side-note: MLP-Mixer

2020, I Tolstikhin, N Houlsby, A Kolesnikov, L Beyer, X Zhai, T Unterthiner, J Yung, A Steiner, D Keysers, J Uszkoreit, M Lucic, A Dosovitskiy

Después de que ViT respondiera a la pregunta "¿Son realmente necesarias las convoluciones para procesar imágenes?" con un NO...
Nos preguntamos si la auto-atención es realmente necesaria

El papel de la self-attention es "mezclar" información entre tokens.
Otra forma simple de lograr esto es "transponer" los tokens y pasarlos por un MLP



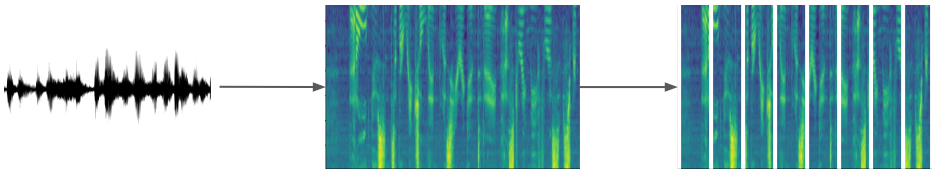
La tercera gran conquista:

Speech

Conformer: Convolution-augmented Transformer for Speech Recognition

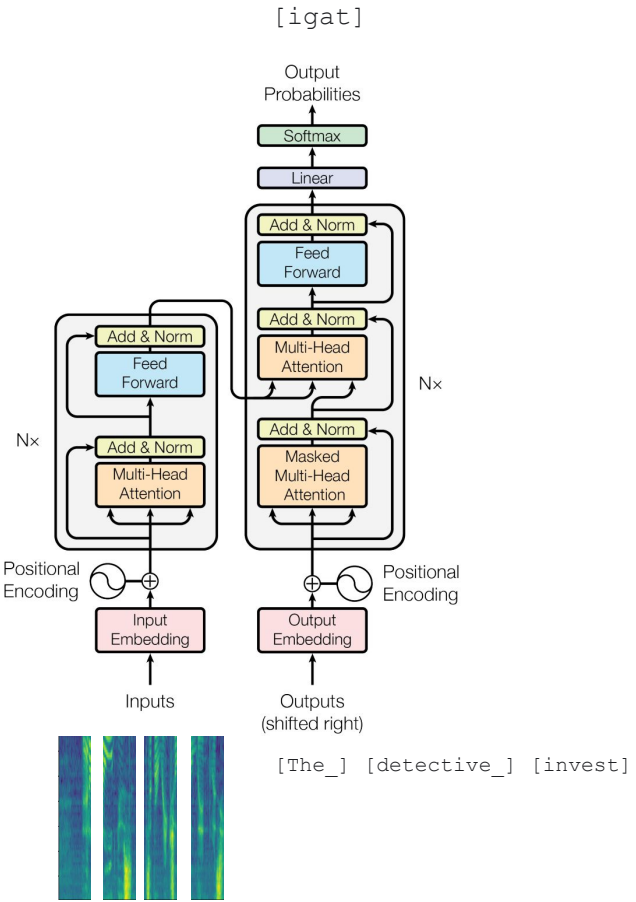
2020, A Gulati, J Qin, C-C Chiu, N Parmar, Y Zhang, J Yu, W Han, S Wang, Z Zhang, Y Wu, R Pang

Mayormente la misma historia que en visión por computadora.
Pero con espectrogramas en lugar de imágenes.



Conformer añade un tercer tipo de bloque usando convoluciones, y reordena ligeramente los bloques, pero en general es muy parecido al transformer.

Existe como variante encoder-decoder, o como variante solo-encoder con pérdida CTC.

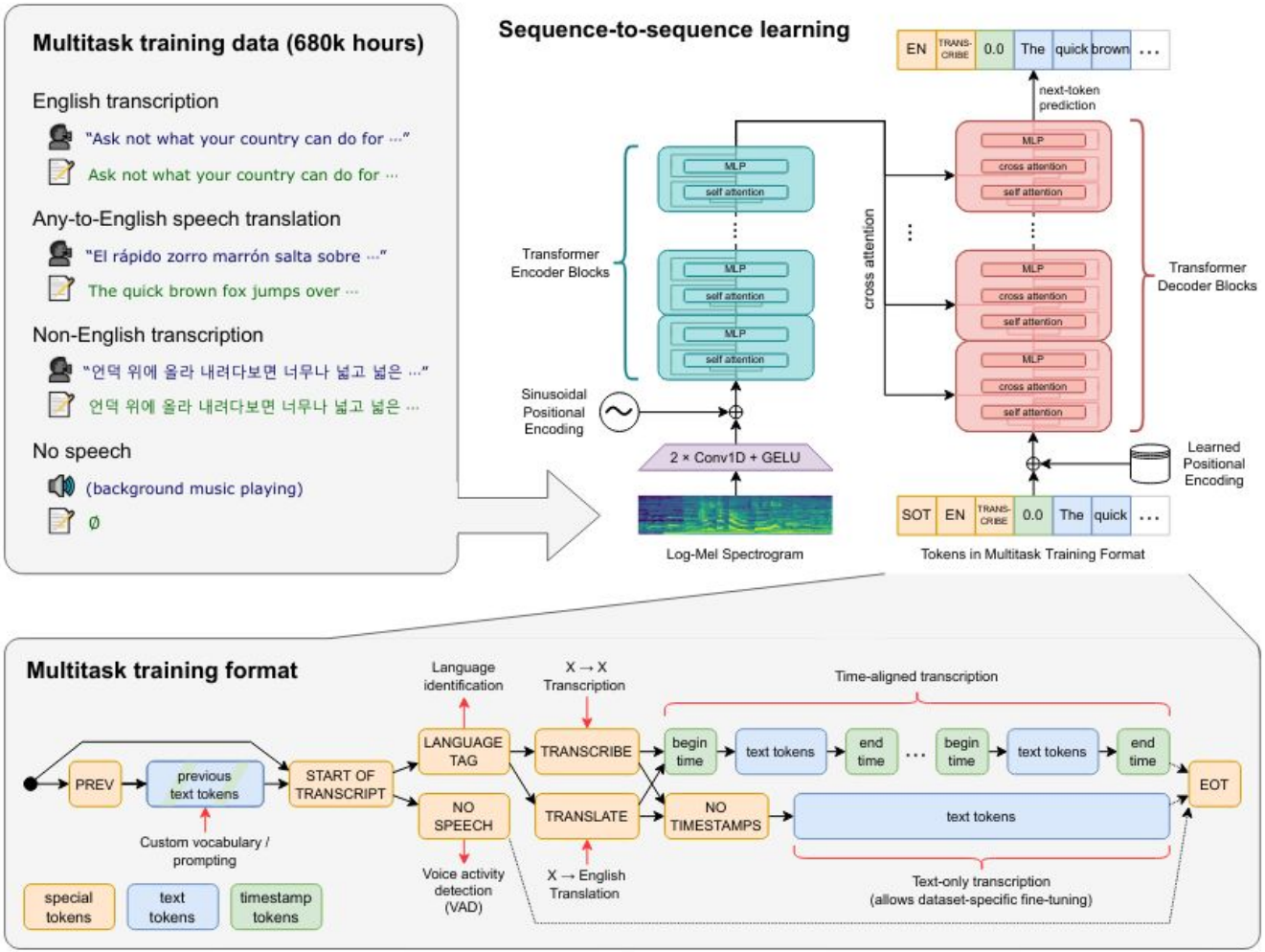


Transformer image source: "Attention Is All You Need" paper

Whisper

Robust Speech Recognition via Large-Scale Weak Supervision

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, Ilya Sutskever



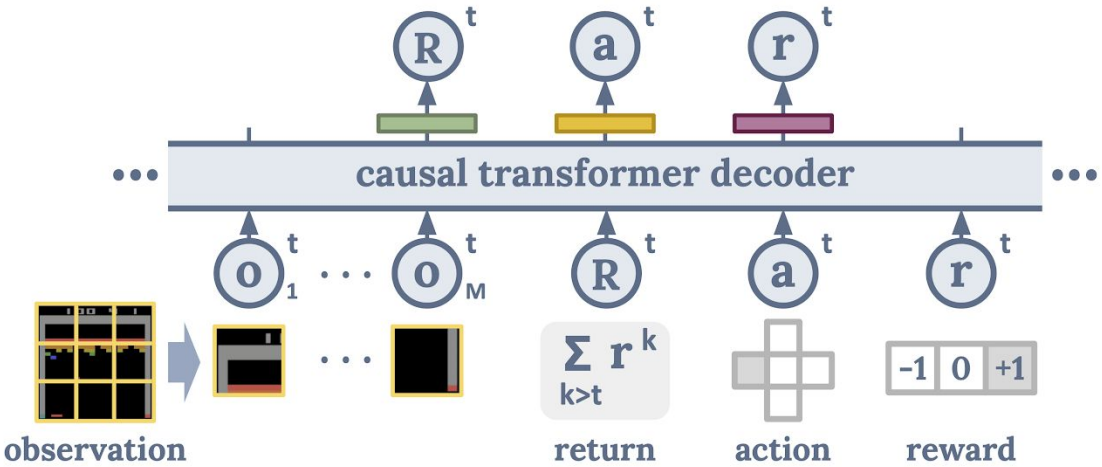
The fourth big takeover:

Reinforcement Learning

Decision Transformer: Reinforcement Learning via Sequence Modeling

2021, L Chen, K Lu, A Rajeswaran, K Lee, A Grover, M Laskin, P Abbeel, A Srinivas, I Mordatch

Cast the (supervised/offline) RL problem into a sequence ("language") modeling task:



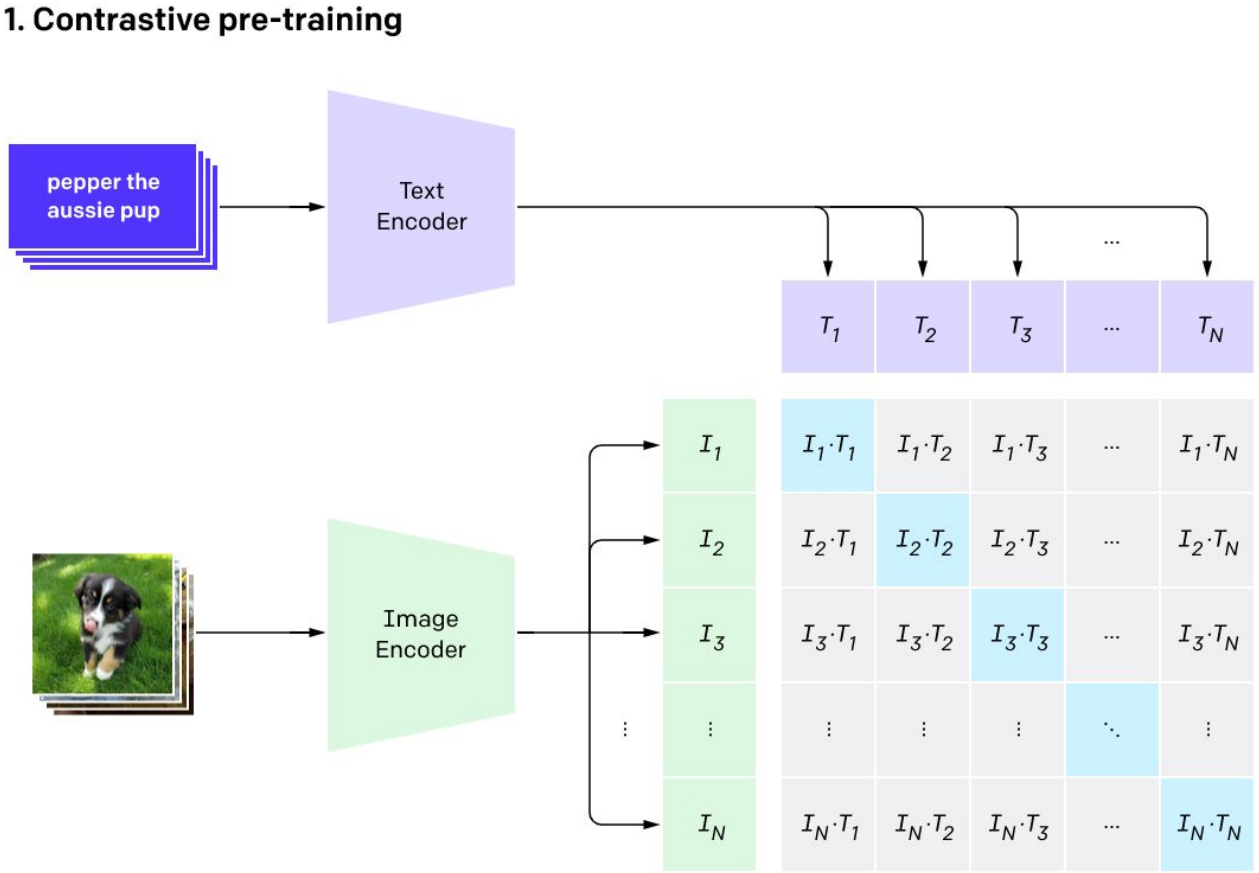
Can generate/decode sequences of actions with desired return (eg skill)

The trick is prompting: `"The following is a trajectory of an expert player: [obs] ..."`

Slide credit: Igor Mordatch

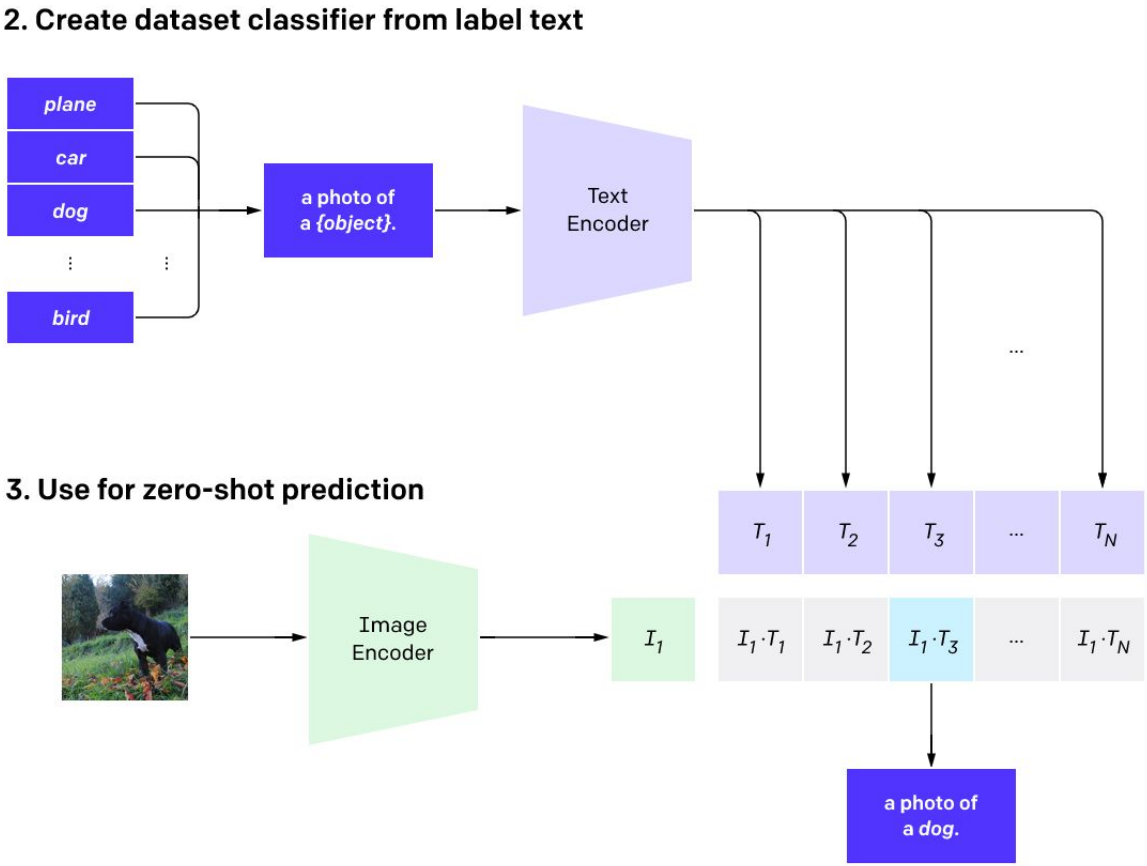
La unificación de las especialides con el Transformer

Multimodal - OpenCLIP



https://wandb.ai/johnowhitaker/openclip-benchmarking/reports/A-Deep-Dive-Into-OpenCLIP-from-OpenAI--VmlldzoyOTIzNzIz

Multimodal - OpenCLIP



Cualquier cosa que puedas tokenizar puedes entrar al Transformer

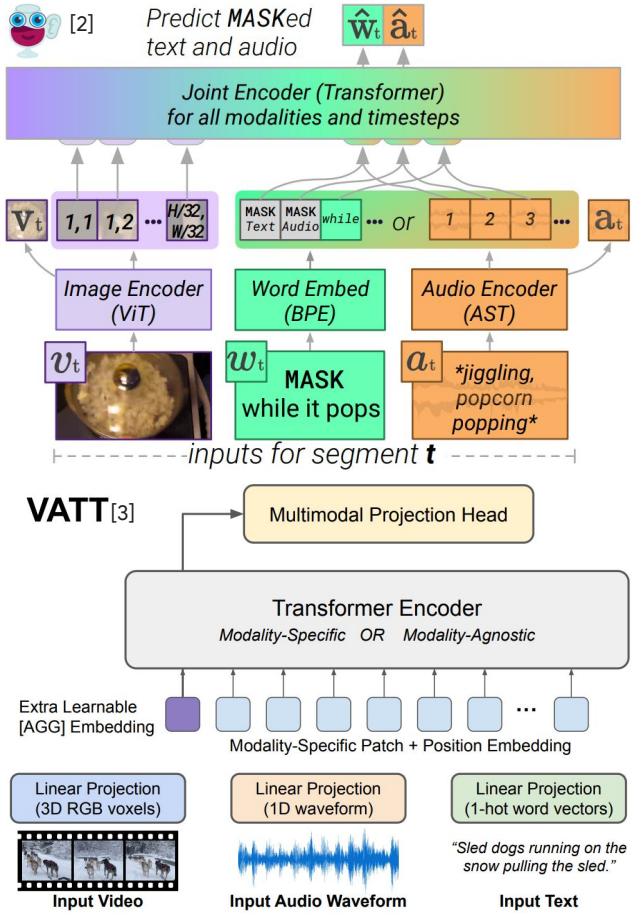
ca 2021 and onwards

Tokenice diferentes modalidades, cada una a su manera (una especie de "parcheo"), y envíelas todas juntas a un Transformer...

Parece que simplemente funciona...

¡Actualmente hay una explosión de trabajos haciendo esto!

[1]



Images from:

[1] LIMoE by B Mustafa, C Riquelme, J Puigcerver, R Jenatton, N Houlsby

[2] MERLOT Reserve by R Zellers, J Lu, X Lu, Y Yu, Y Zhao, M Salehi, A Kusupati, J Hessel, A Farhadi, Y Choi

[3] VATT by H Akbari, L Yuan, R Qian, W-H Chuang, S-F Chang, Y Cui, B Gong

A note on
Efficient Transformers

A note on Efficient Transformers

The self-attention operation complexity is $O(N^2)$ for sequence length N.

We'd like to use large N:

- Whole articles or books
- Full video movies
- High resolution images

Many O(N) approximations to the full self-attention have been proposed in the past two years.

Unfortunately, none provides a clear improvement. They always trade-off between speed and quality.

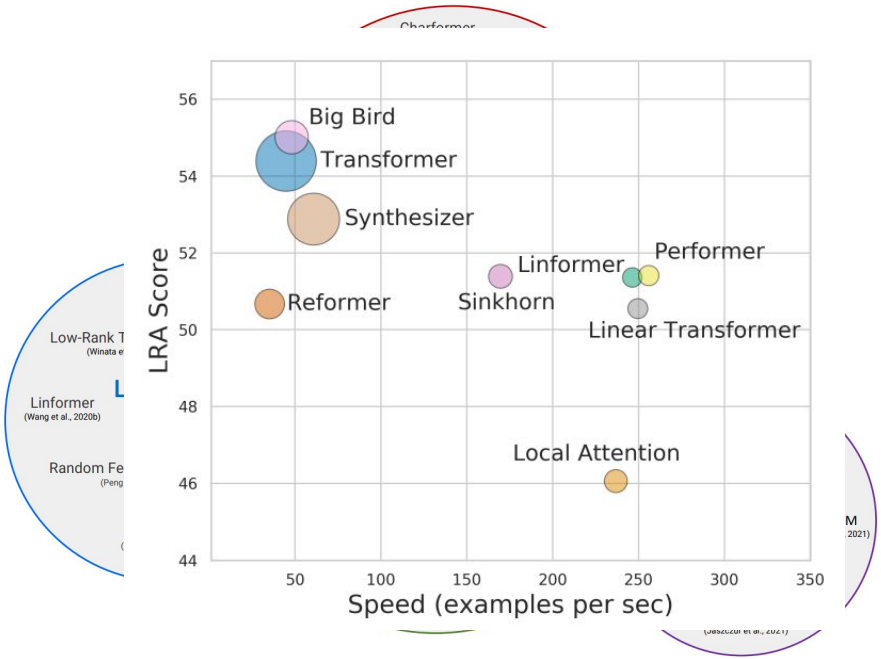


Figure 2: Taxonomy of Efficient Transformer Architectures.

Based on "Efficient Transformers: A Survey" by Yi Tay, Mostafa Dehghani, Dara Bahri, Donald Metzler
and "Long Range Arena: A Benchmark for Efficient Transformers" by Y Tay, M Dehghani, S Abnar, Y Shen, D Bahri, P Pham, J Rao, L Yang, S Ruder, D Metzler

Gracias por su Atención

Transformer image source: "Attention Is All You Need" paper