Transformers III

Clase 21

Dra. Wendy Aguilar

Modelos Generativos Profundos

UN ENFOQUE DESDE LA CREATIVIDAD COMPUTACIONAL

Entrenamiento del modelo GPT

gpt_listo_para_entrenar.ipynb

Google colab

```
gpt.fit(
    train_ds,
    epochs=EPOCHS,
    callbacks=[tensorboard_callback, text_generator],
)
```

```
Epoch 1/5
4060/4060 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - loss: 2.5973
generated text:
wine review : france : southwest france : bordeaux - style red blend : from one of 90 acres of two small vineyards in margaux , the garonne river . cabernet franc adds an older wine with

4060/4060 ━━━━━━━━━━━━━━━━━━━━ 212s 49ms/step - loss: 2.5972
Epoch 2/5
4059/4060 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - loss: 1.9768
generated text:
wine review : france : burgundy : merlot : rich and extracted [UNK] - textured . this wine needs a year going to have to be ready to drink .

4060/4060 ━━━━━━━━━━━━━━━━━━━━ 114s 28ms/step - loss: 1.9768
Epoch 3/5
4059/4060 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - loss: 1.8990
generated text:
wine review : us : california : cabernet sauvignon : spicy black fruit and cedar mark this wine known in malbec . integrated oak supports its red - cherry , oak and plum aromas on this f

4060/4060 ━━━━━━━━━━━━━━━━━━━━ 123s 30ms/step - loss: 1.8990
Epoch 4/5
4059/4060 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - loss: 1.8507
generated text:
wine review : brazil : protect : petite sirah : this wine dates - like [UNK] ' s standard [UNK] - smelling , it has the palate - based aromas of vanilla , cola , roasted meat and plum .

4060/4060 ━━━━━━━━━━━━━━━━━━━━ 122s 30ms/step - loss: 1.8507
Epoch 5/5
4060/4060 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - loss: 1.8236
generated text:
wine review : us : california : chardonnay : this has a beautiful chardonnay [UNK] notable for its vibrant , appley apple , banana and buttered toast flavors that are smooth and suave ,

4060/4060 ━━━━━━━━━━━━━━━━━━━━ 116s 28ms/step - loss: 1.8236
<keras.src.callbacks.history.History at 0x7e6c375ad490>
```

Generación de texto con nuestro modelo GPT

gpt.ipynb

Autoregresiva

Loop de generación:
1. Pasas el prompt → obtienes probabilidades del siguiente token.
2. **Generas una muestra** (un token) de esa distribución.
3. **Lo anexas** al prompt y repites (paso a paso).

3. Generate text using the Transformer

```
# Función auxiliar para visualizar qué palabras tienen más probabilidad
# de ser generadas y dónde está prestando atención el modelo.
def print_probs(info, vocab, top_k=5):
    for i in info:
        highlighted_text = []
        for word, att_score in zip(
            i["prompt"].split(), np.mean(i["atts"], axis=0)
        ):
            highlighted_text.append(
                '<span style="background-color:rgba(135,206,250,'
                + str(att_score / max(np.mean(i["atts"], axis=0)))
                + ');">'
                + word
                + "</span>"
            )
        highlighted_text = " ".join(highlighted_text)
        display(HTML(highlighted_text))

        word_probs = i["word_probs"]
        p_sorted = np.sort(word_probs)[::-1][:top_k]
        i_sorted = np.argsort(word_probs)[::-1][:top_k]
        for p, i in zip(p_sorted, i_sorted):
            print(f"{vocab[i]}:   \t{np.round(100*p,2)}%")
        print("--------\n")
```

`info`: una lista de diccionarios, cada uno con la información de una predicción (prompt, atenciones, probabilidades, etc.).
vocab: lista o diccionario que mapea índices → palabras. top_k: número de palabras más probables que se mostrarán (por defecto 5).

Itera sobre cada predicción

Crea una lista vacía donde se irá almacenando cada palabra con su color de fondo (según el nivel de atención).

Itera sobre cada palabra del *prompt* y su *score de atención promedio*.

Construye una cadena HTML para mostrar la palabra `word` con un **color azul (RGB 135,206,250)** cuya **transparencia (alfa)** depende de la magnitud de `att_score`.
Cuanto mayor el att_score, más oscuro el fondo → significa que el modelo prestó más atención a esa palabra.

Divide el *score* entre el máximo para normalizarlo entre 0 y 1.

Une todas las palabras coloreadas en una sola cadena HTML.

Muestra el texto coloreado directamente en la celda de Jupyter Notebook.

Extrae el vector de probabilidades (una distribución sobre el vocabulario).

Ordena las probabilidades de mayor a menor (`[::-1]`) y toma las `top_k`.

Imprime las `top_k` palabras más probables con sus porcentajes de probabilidad.

Generación de texto con nuestro modelo GPT

Autoregresiva

Loop de generación:

1. Pasas el prompt → obtienes probabilidades del siguiente token.
2. **Muestras** un token de esa distribución.
3. **Lo anexas** al prompt y repites (paso a paso).

Sampling vs. determinismo:

Al muestrear de la distribución hacemos el proceso **estocástico** (más variedad que tomar el **argmax**).

- Temperatura T:
  - Controla **qué tan "plana" o "afilada"** es esa distribución.
  - Matemáticamente se aplica:

$$p_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

z_i = logit (salida cruda del modelo para el token i)

T = parámetro de temperatura

```
class TextGenerator(callbacks.Callback):
```

Implementa **esa misma fórmula** de la temperatura aplicada al *softmax*, pero de forma computacional y simplificada. $\text{softmax}(z/T) = \frac{e^{z_i/T}}{\sum e^{z_j/T}} \approx \frac{(e^{z_i}/\sum e^{z_j})^{1/T}}{\sum (e^{z_j}/\sum e^{z_j})^{1/T}}$

```
def sample_from(self, probs, temperature):
    probs = probs ** (1 / temperature)
    probs = probs / np.sum(probs)
    return np.random.choice(len(probs), p=probs), probs
```

`probs` representa ya el resultado del **softmax clásico**, es decir $e^{z_i}/\sum e^{z_j}$

En lugar de volver a aplicar `exp()`, el código **reescala la distribución** elevando las probabilidades a $1/T$.

Normaliza los valores para que vuelvan a sumar 1. Esto equivale al denominador de la fórmula: $\sum_j e^{z_j/T}$

Muestrea aleatoriamente un índice según las probabilidades $probs$.

Generación de texto con nuestro modelo GPT

Autoregresiva

Loop de generación:

1. Pasas el prompt → obtienes probabilidades del siguiente token.
2. **Muestras** un token de esa distribución.
3. **Lo anexas** al prompt y repites (paso a paso).

Sampling vs. determinismo:

Al muestrear de la distribución hacemos el proceso **estocástico** (más variedad que tomar el **argmax**).

- Temperatura T:
  - Controla **qué tan "plana" o "afilada"** es esa distribución.
  - Matemáticamente se aplica:

$$p_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

z_i = logit (salida cruda del modelo para el token i)

T = parámetro de temperatura

Supongamos que el modelo predice la siguiente palabra tras *"wine review: germany:"*

Palabra	Logit original	p(T=1.0)	p(T=0.5)	p(T=1.5)
pfalz	4.1	**0.484**	**0.577**	**0.441**
mosel	3.9	0.397	0.388	0.386
rhein	2.7	0.119	0.035	0.173

- Al bajar la temperatura a 0.5 → la distribución se concentra más (pfalz domina).
- Al subirla a 1.5 → las probabilidades se reparten más equitativamente.

Sustituyendo $T = 1.0$

Primero calculamos los exponentes:

$$e^{z_i/T} = e^{z_i}$$

Este cociente controla **cuánto se exageran o suavizan las diferencias** entre logits.

$$e^{4.1} = 60.34$$
$$e^{3.9} = 49.40$$
$$e^{2.7} = 14.88$$

Calculamos el denominador (la suma total)

$$\sum_j e^{z_j/T} = 60.34 + 49.40 + 14.88 = 124.62$$

Normalizamos cada palabra

$$p_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

$$p_{\text{pfalz}} = \frac{60.34}{124.62} = 0.484$$
$$p_{\text{mosel}} = \frac{49.40}{124.62} = 0.396$$
$$p_{\text{rhein}} = \frac{14.88}{124.62} = 0.119$$

gpt.ipynb

Google colab

Temperatura	Efecto en el modelo	Características del texto generado
T < 0.5	Muy baja	El modelo se vuelve **determinista**. Tiende a repetir siempre las mismas palabras y frases. Genera texto **seguro pero aburrido**.
T ≈ 1.0	Equilibrada	Es el valor más **natural**: mantiene coherencia, pero permite algo de **diversidad**.
T > 1.2	Alta	Aumenta la aleatoriedad. El modelo puede generar **variaciones creativas**, pero también **errores gramaticales o semánticos**.
T > 2.0	Muy alta	La distribución se vuelve casi uniforme → el modelo **elige palabras al azar** (texto incoherente).

```
info = text_generator.generate(
    "wine review : us", max_tokens=80, temperature=1.0
)
```

```
generated text:
wine review : us : virginia : viognier : round and round in the mouth there ' s a floral
note that washes toward lemon - lime pith and melon aromas lead to a rounded texture with
a toasted backdrop of fatty peach , tropical notes and a tropical set up a low finish .
```

```
info = text_generator.generate(
    "wine review : italy", max_tokens=80, temperature=0.5
)
```

```
generated text:
wine review : italy : tuscany : red blend : this blend of 90 % sangiovese and 10 %
canaiolo opens with aromas of red berry , scorched earth and a whiff of leather . the
palate offers ripe plum , black cherry , clove and a hint of star anise alongside firm ,
fine - grained tannins .
```

- Mantiene estilo del corpus (formato “wine review”).
- Coherencia temática (uvas/regiones) mejora con T más baja.
- Más **variedad** y ocasionales imprecisiones con T alta.

¿Qué pasa si usas T=3?

gpt.ipynb

Google colab

El TransformerBlock devuelve **attention weights** por cabeza.

```
info = text_generator.generate(
    "wine review : germany", max_tokens=80, temperature=0.5
)
print_probs(info, vocab)
```

```
generated text:
wine review : germany : mosel : riesling : while demure in color , this off - dry riesling is intensely fruity and
refreshing , this off - dry riesling is surprisingly crisp and refreshing . it ' s a thirst - quenching sip , and
it ' s a perfect apéritif to pair with oysters or game .
```

Paso 1

```
wine review : germany
::          100.0%
zealand:    0.0%
-:          0.0%
grosso:     0.0%
blend:      0.0%
--------
```

Paso 2

```
wine review : germany :
mosel:      92.43000030517578%
rheinhessen: 6.03000020980835%
rheingau:   0.75%
franken:    0.44999998807907104%
baden:      0.20999999344348907%
```

Colores más intensos = **mayor atención** a esa palabra al decidir el siguiente token.

Región → País: para predecir una *región* (pfalz, mosel…), el modelo **atiende al país** (germany).

Paso 3

```
wine review : germany : mosel
::          99.77999877929688%
-:          0.2199999988079071%
blend:      0.0%
valley:     0.0%
hills:      0.0%
--------
```

Paso 4

```
wine review : germany : mosel :
riesling:   99.98999786376953%
grüner:     0.0%
[UNK]:      0.0%
pinot:      0.0%
sparkling:  0.0%
--------
```

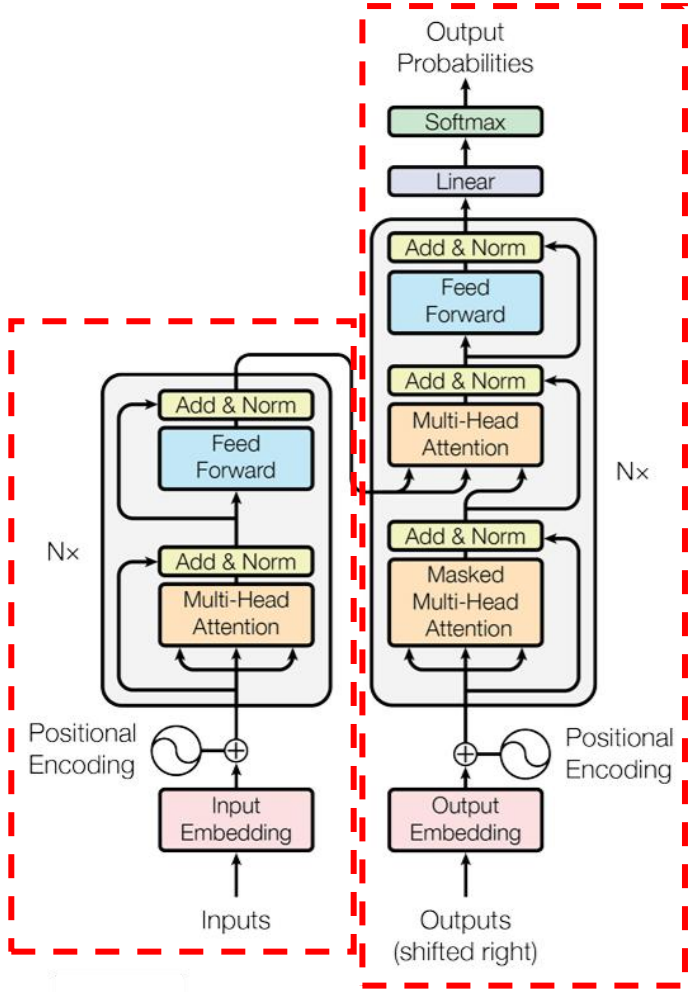
Arquitectura Transformer original

Tarea principal

Traducción

El modelo está compuesto por **dos bloques principales**:

Codificador



The Transformer - model architecture.

Decodificador

Arquitectura Transformer original

Tarea principal

Traducción

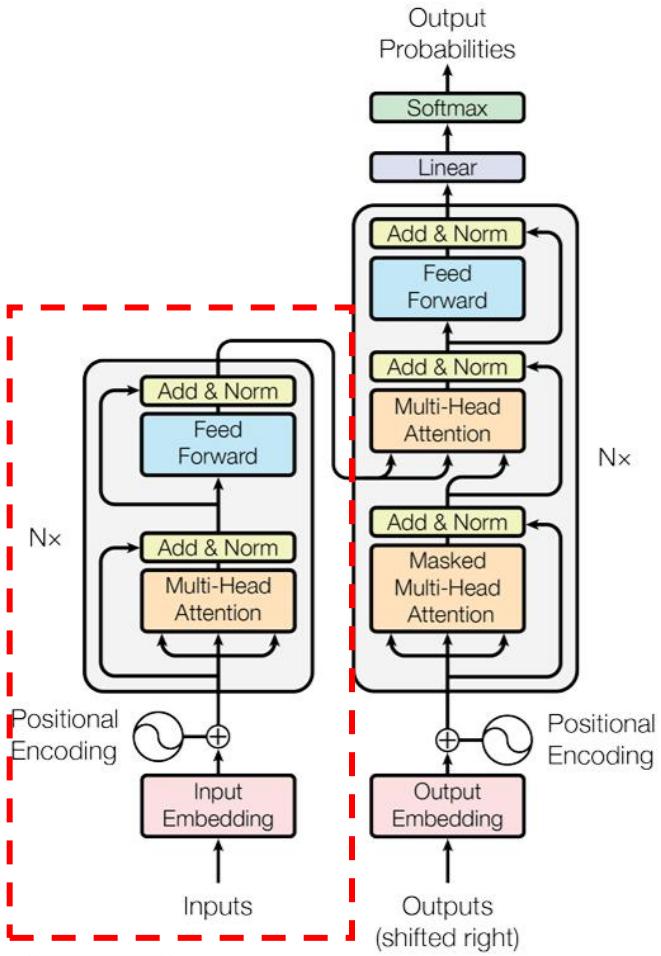
El **encoder** tiene como propósito **transformar:**

la secuencia de entrada

"The pink elephant tried to get into the car but it was too big"

en un conjunto de **representaciones contextuales** (vectores) que **capturan el significado global** del texto.

Token	Representación contextual (salida del encoder)	Interpretación semántica aproximada
The	[0.11, -0.09, 0.42, 0.03, -0.02, 0.20]	Determinante sin carga semántica fuerte.
pink	[0.60, 0.25, -0.05, 0.40, -0.22, -0.28]	Atributo cromático modificado por "elephant".
elephant	[0.78, 0.19, -0.25, 0.62, 0.15, -0.46]	Entidad principal (agente) central en el contexto.
tried	[0.40, 0.18, 0.15, 0.58, -0.20, 0.30]	Acción realizada por el sujeto "elephant".
car	[0.48, 0.22, -0.10, 0.34, 0.07, -0.29]	Objeto meta del verbo "get into".
it	[0.72, 0.11, 0.12, 0.44, 0.18, -0.35]	Referencia al sujeto "elephant".
big	[0.85, 0.21, -0.18, 0.50, 0.09, -0.39]	Atributo físico asociado a "it/elephant".



The Transformer - model architecture.

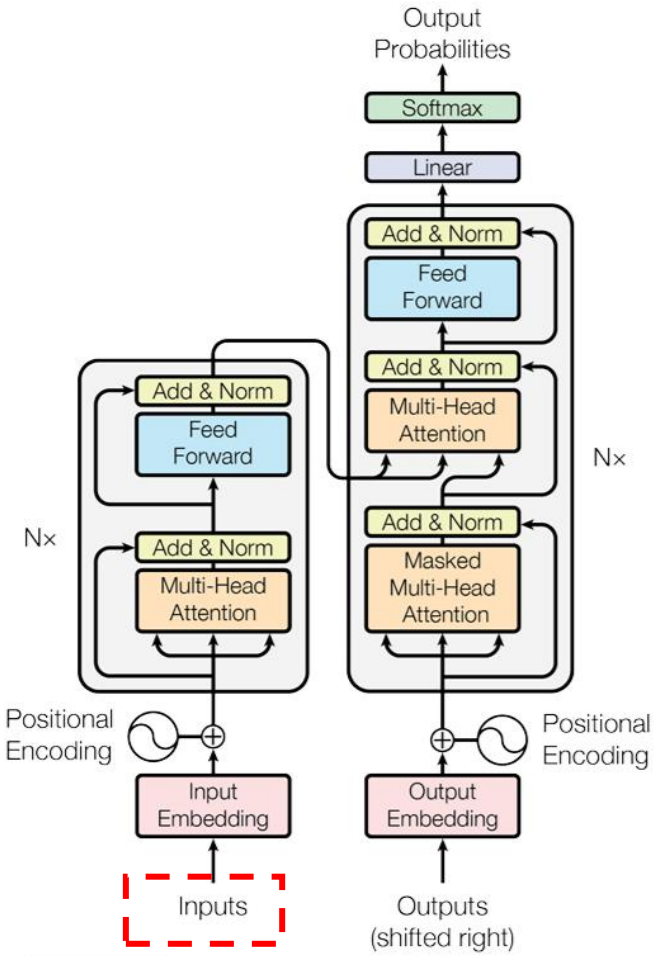
Arquitectura Transformer original

Tarea principal

Traducción

Entrada: la secuencia completa de tokens del texto fuente (por ejemplo, una oración en inglés).

"The pink elephant tried to get into the car but it was too big."



The Transformer - model architecture.

Arquitectura Transformer original

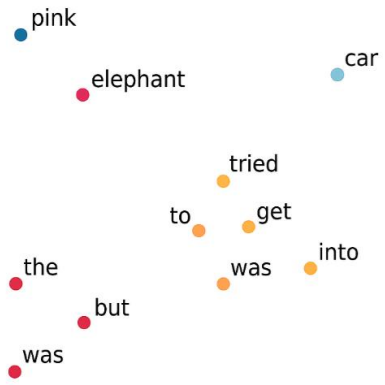
Tarea principal

Traducción

Capa de embedding + codificación posicional: convierte cada palabra y su posición en un vector (como vimos en las codificaciones sinusoidales).

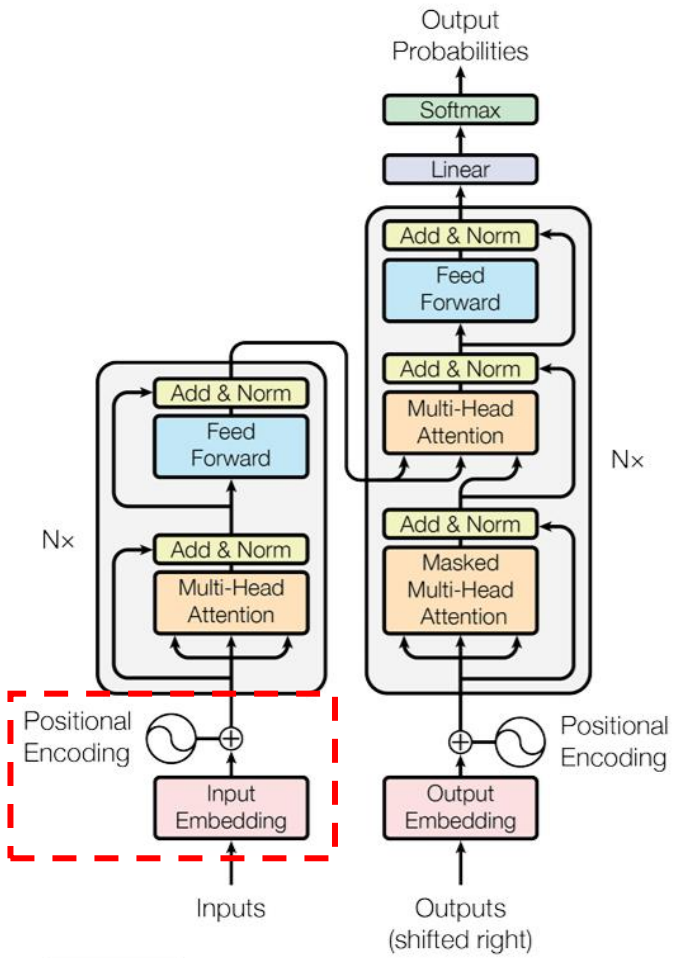
Resultado: una matriz de entrada $X \in \mathbb{R}^{n \times d}$, donde *n* es la longitud de la secuencia y *d* la dimensión del embedding.

Cada **columna** puede verse como una *dimensión semántica aprendida*.



Palabra (vocabulario)	dim_1: color / apariencia	dim_2: tamaño / forma	dim_3: movimiento / acción	dim_4: objetos / entorno	dim_5: emoción / contexto	... (dim_n)
the	0.02	0.01	0.00	0.03	0.00	...
pink	0.88	0.10	0.05	0.12	0.02	...
elephant	0.32	0.91	0.18	0.07	0.10	...
tried	0.04	0.08	0.92	0.06	0.12	...
to	0.01	0.00	0.05	0.02	0.00	...
get	0.03	0.09	0.88	0.11	0.15	...
into	0.02	0.03	0.61	0.75	0.10	...
the	0.02	0.01	0.00	0.03	0.00	...
car	0.10	0.12	0.22	0.94	0.05	...
but	0.01	0.02	0.03	0.01	0.84	...
it	0.00	0.05	0.10	0.03	0.67	...
was	0.01	0.02	0.07	0.02	0.80	...
too	0.02	0.08	0.09	0.04	0.93	...

big ...



The Transformer - model architecture.

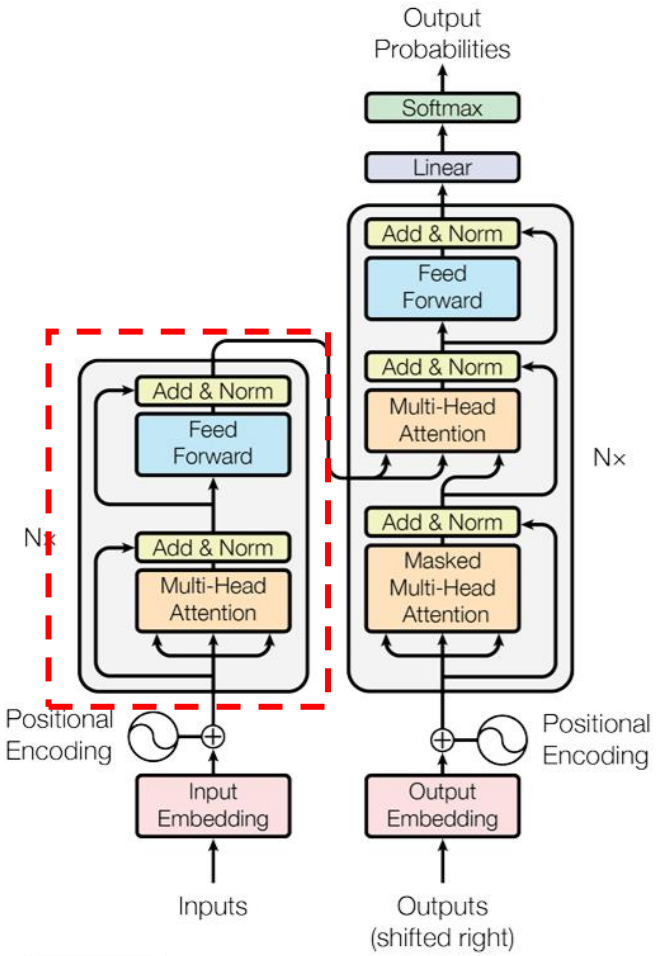
Arquitectura Transformer original

Tarea principal

Traducción

Mecanismo interno:

- Cada bloque (repetido *N* veces) tiene:
- Multi-Head Attention (auto-atención, sin máscara).
- Feed-Forward Network (red densa).
- Usa conexiones residuales y normalización (*Add & Norm*).

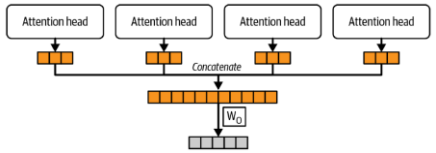


The Transformer - model architecture.

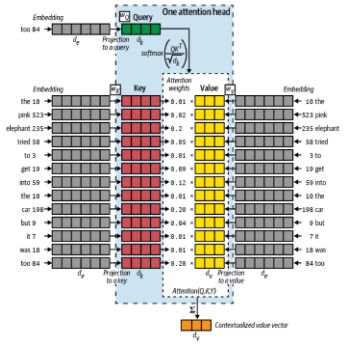
Arquitectura Transformer original

Multi-Head Self-Attention

- Cada token genera tres vectores: **Q (query)**, **K (key)** y **V (value)**.
- Se calcula el **producto punto entre todas las queries y keys**, midiendo cuánta atención debe prestar cada palabra a las demás.
- El resultado es una representación de cada palabra **en función de su relación con todas las demás**.

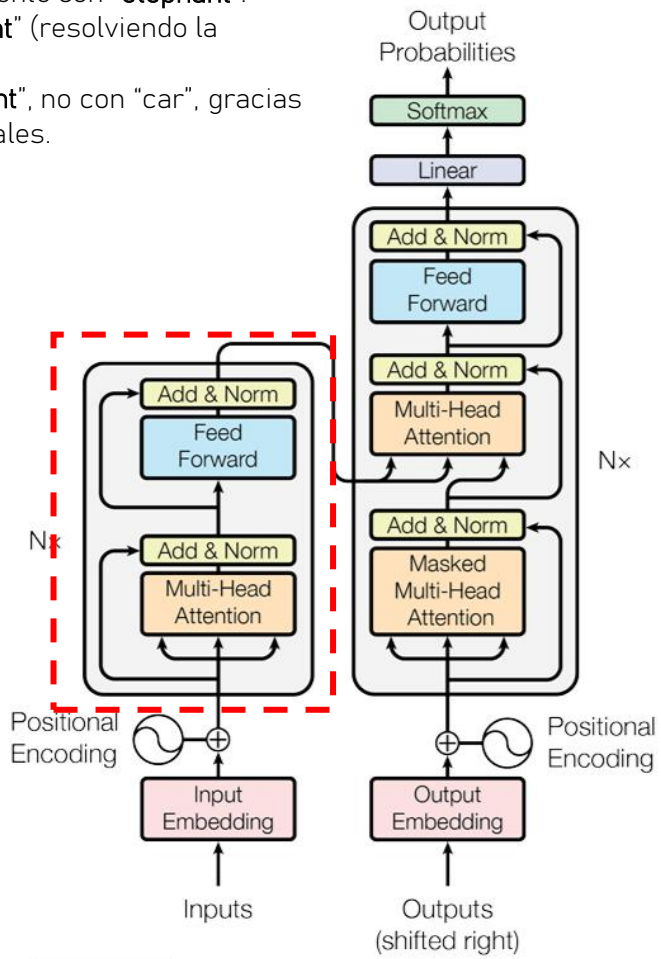


Capa de atención muticabeza



Funcionamiento de una cabeza de atención

- "**pink**" se relaciona fuertemente con "**elephant**".
- "**it**" se conecta con "**elephant**" (resolviendo la referencia).
- "**big**" se asocia con "**elephant**", no con "car", gracias a esas relaciones contextuales.



The Transformer - model architecture.

Feed-Forward Network (FFN)

- Una red densa aplicada a cada posición de la secuencia.
- Su función es **extraer características más abstractas y no lineales** de la atención anterior.

Residual Connections + Layer Normalization

- Se añade la entrada original a la salida de cada subcapa (*Add & Norm*).
- Esto estabiliza el entrenamiento y permite que la información fluya sin degradarse en redes profundas.

Arquitectura Transformer original

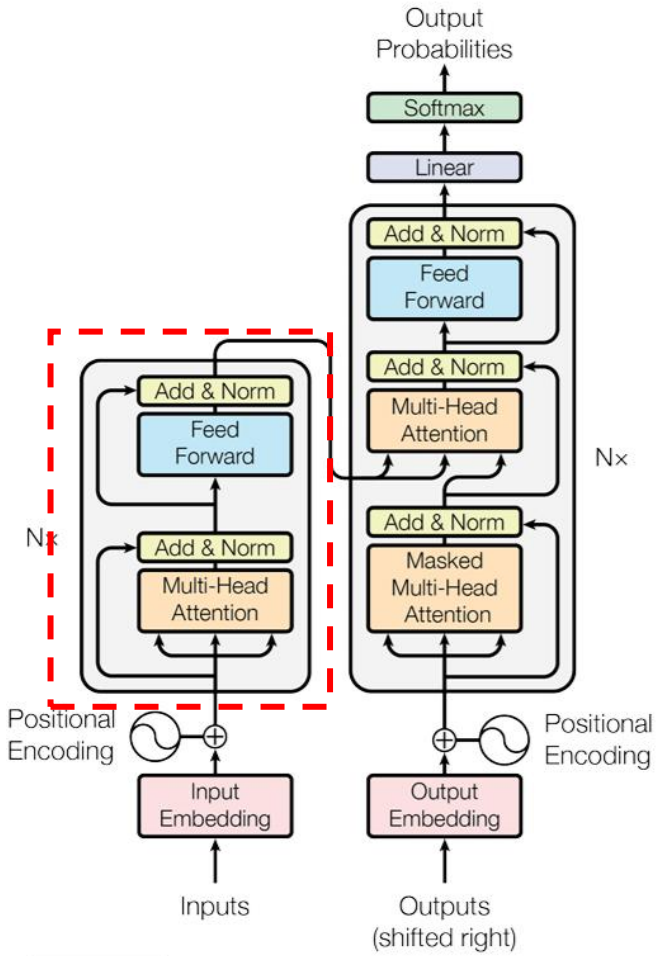
Resultado del Encoder

Al final de *N* capas (normalmente 6–12):

- Cada palabra está representada por un vector contextualizado h_i que captura su **significado dentro de toda la oración**.

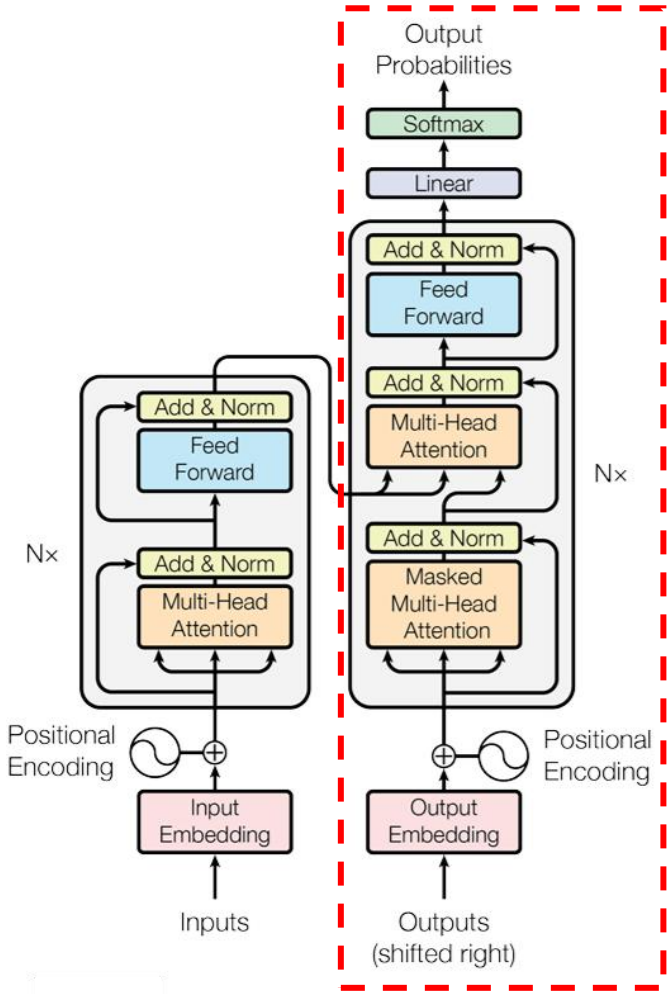
- El conjunto de todos esos vectores forma la **memoria interna del modelo**, que el **decoder** puede consultar durante la generación (a través de *cross-attention*).

Token	Representación contextual (salida del encoder)	Interpretación semántica aproximada
The	[0.11, -0.09, 0.42, 0.03, -0.02, 0.20]	Determinante sin carga semántica fuerte.
pink	[0.60, 0.25, -0.05, 0.40, -0.22, -0.28]	Atributo cromático modificado por "elephant".
elephant	[0.78, 0.19, -0.25, 0.62, 0.15, -0.46]	Entidad principal (agente) central en el contexto.
tried	[0.40, 0.18, 0.15, 0.58, -0.20, 0.30]	Acción realizada por el sujeto "elephant".
car	[0.48, 0.22, -0.10, 0.34, 0.07, -0.29]	Objeto meta del verbo "get into".
it	[0.72, 0.11, 0.12, 0.44, 0.18, -0.35]	Referencia al sujeto "elephant".
big	[0.85, 0.21, -0.18, 0.50, 0.09, -0.39]	Atributo físico asociado a "it/elephant".



The Transformer - model architecture.

Arquitectura Transformer original



The Transformer - model architecture.

El **decoder** tiene como propósito **generar una secuencia de salida**, paso a paso, **usando dos fuentes de información**:

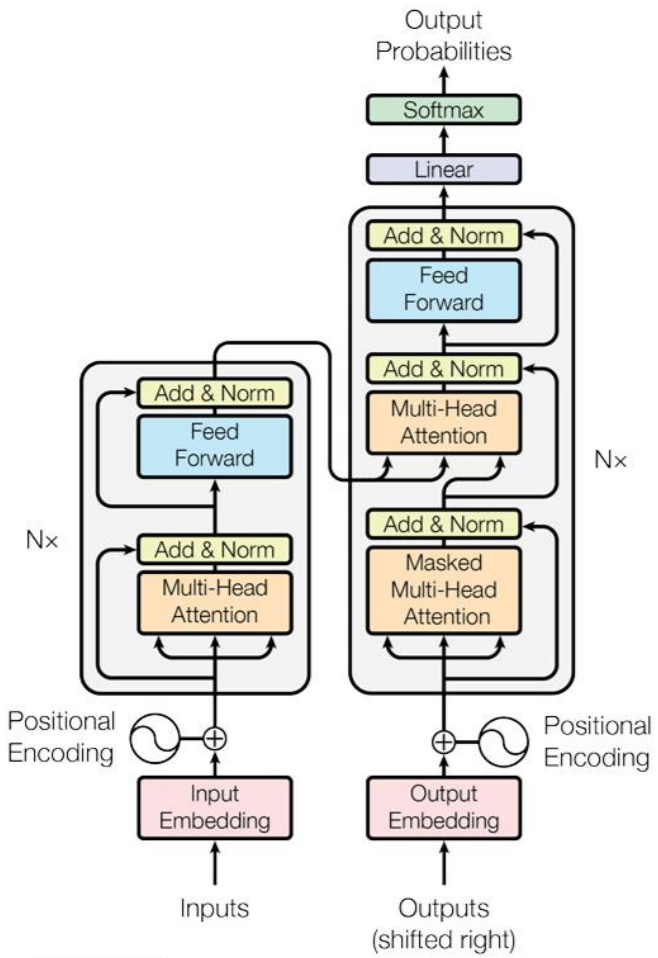
- **Lo que ya ha generado hasta el momento** (por ejemplo, en traducción: las palabras ya traducidas).
- **La representación codificada del texto de entrada**, proveniente del encoder (el significado global de la oración original).

Arquitectura Transformer original

Tarea principal

Traducción

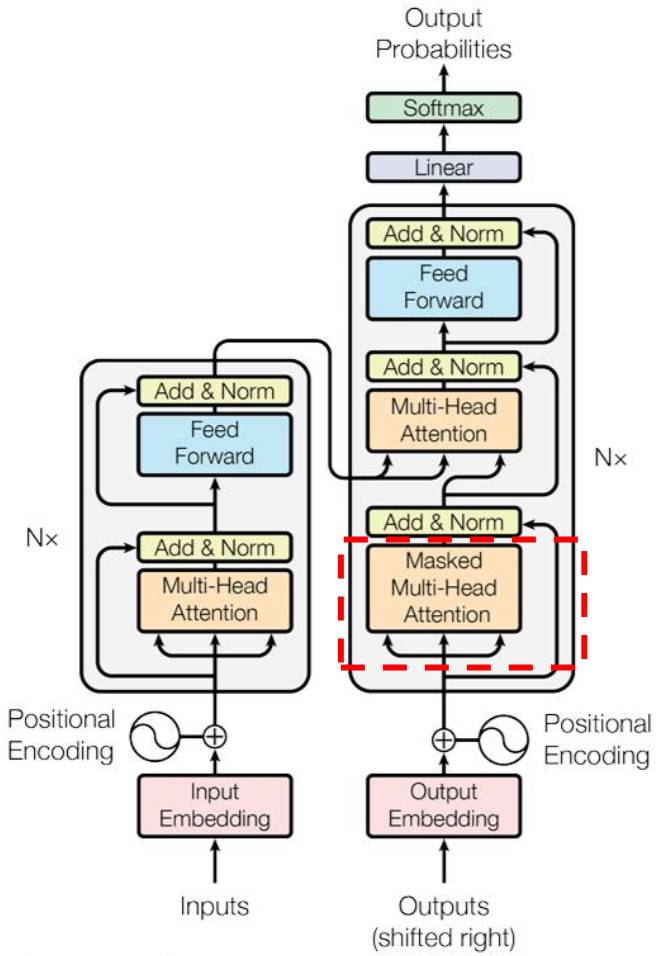
El *encoder* entiende el texto.



The Transformer - model architecture.

El *decoder* **lo usa para producir otro texto** (por ejemplo, una traducción, un resumen o la continuación de una frase).

Arquitectura Transformer original

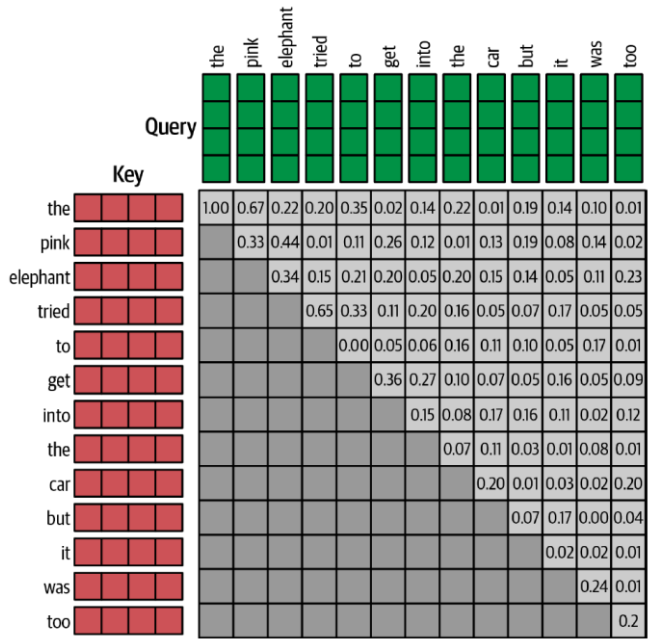


The Transformer - model architecture.

Capas internas:

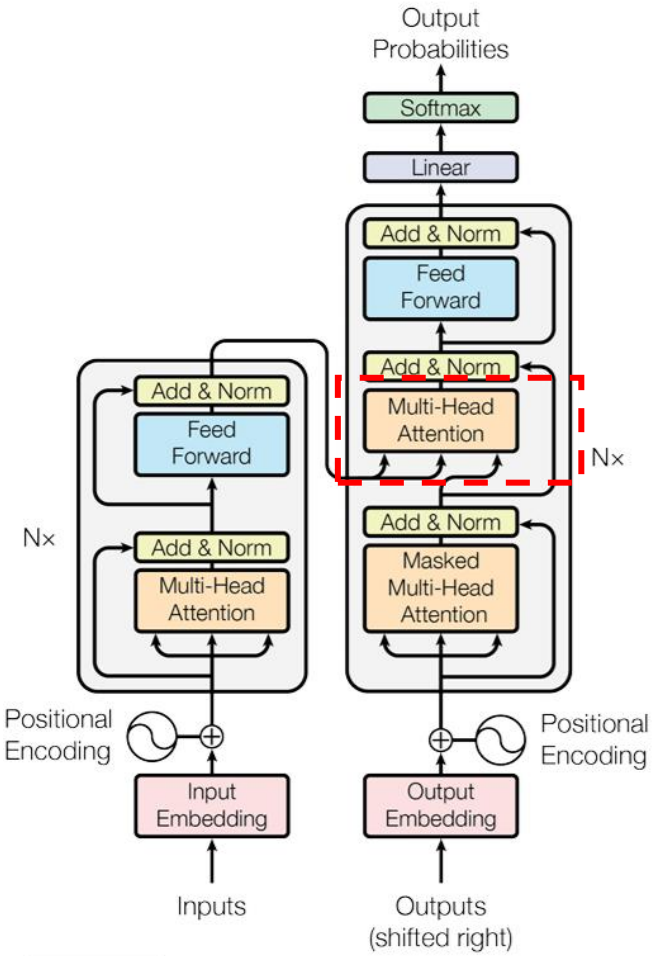
1. Masked Multi-Head Attention: auto-atención causal (solo mira tokens previos).

Si el modelo está generando *"The pink elephant"*, no puede mirar la palabra *"tried"* todavía.
Solo tiene acceso a "The" y "pink".



Arquitectura Transformer original

El **cross-attention** conecta las representaciones del texto fuente ("The pink elephant...") con las palabras que el decodificador está generando ("El elefante rosa..."), garantizando que las referencias y concordancias sean correctas.



The Transformer - model architecture.

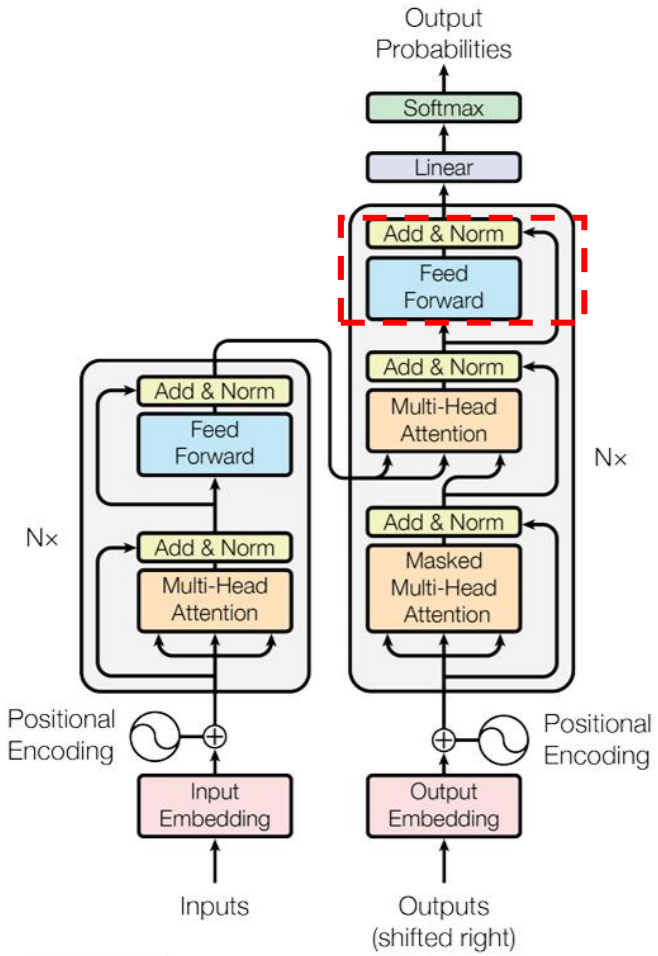
2. **Cross-Attention:** atiende a las salidas del encoder (permite "mirar" el texto fuente).

- Aquí es donde ocurre la "comunicación" entre el **encoder** y el **decoder**.
- Las queries provienen de la salida del bloque anterior del decoder, mientras que las keys y values provienen de la **salida del encoder**.

Así, el decoder puede **atender a las partes más relevantes de la entrada** para decidir qué generar.

En la traducción de
"The pink elephant tried to get into the car but it was too big"
al **español**,
cuando el **decodificador** genera la palabra **"era"**,
puede prestar atención al *embedding* contextual de **"elephant"** en el **codificador**,
para resolver correctamente la referencia (**"it" → "era [el elefante]"**).

Arquitectura Transformer original

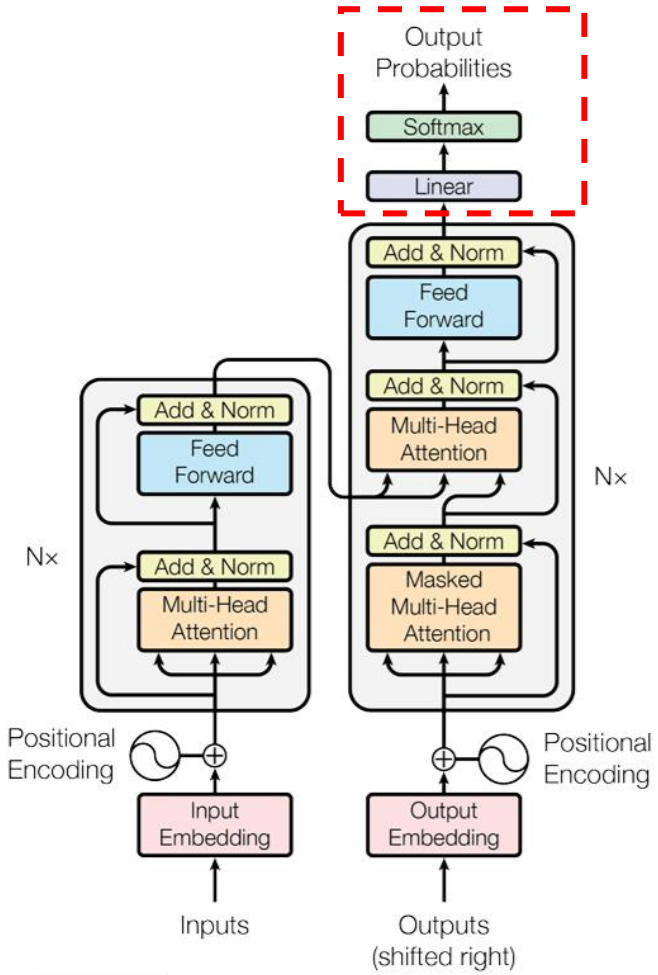


The Transformer - model architecture.

3. Feed-Forward + Add & Norm.

Igual que en el encoder: dos capas densas y normalización para refinar las representaciones intermedias.

Arquitectura Transformer original



The Transformer - model architecture.

- Pasa las representaciones contextuales por una **capa lineal** que proyecta cada vector del espacio semántico interno al espacio del vocabulario, asignando un *logit* a cada palabra posible.

Supongamos que estamos en el paso del decodificador donde el modelo ya ha generado:

"El elefante rosa intentó meterse al coche, pero"

Palabra del vocabulario	Logit (z_i)
el	1.2
elefante	–0.8
rosa	–0.3
intentó	0.5
meterse	1.1
al	0.2
coche	–0.7
era	**2.4**

Aplica *softmax* para transformar estos logits en **probabilidades**.

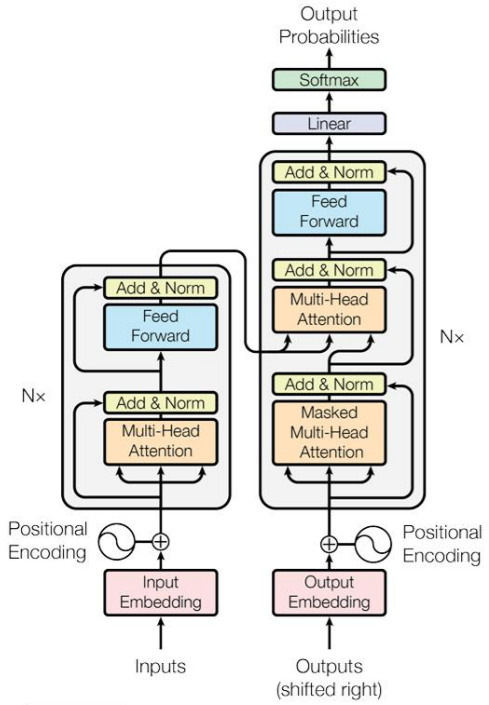
Tres grandes tipos de Transformers

Solo codificador

solo decodificador

codificador–decodificador

Cada uno se ajusta a diferentes tipos de tareas y se entrena de manera distinta.



The Transformer - model architecture.

Tres grandes tipos de Transformers

Solo codificador

Ejemplo:

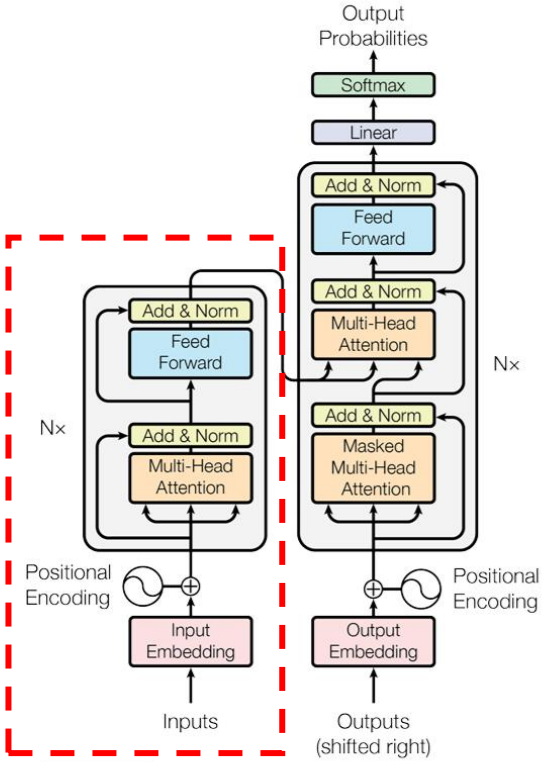
BERT (Bidirectional Encoder Representations from Transformers) – Google, 2018

Características:

- No usa enmascaramiento causal, porque no genera texto.
- Cada token puede atender a todos los demás tokens (contexto completo).
- Su objetivo es comprender el texto de entrada, no producirlo.
- Se entrena con tareas como *masked language modeling* (predecir palabras faltantes) y *next sentence prediction*.

Tareas típicas (fine-tuning):

- Clasificación de oraciones (opiniones positivas o negativas).
- Reconocimiento de entidades (nombres, lugares, organizaciones).
- Respuesta extractiva (encontrar fragmentos relevantes de un texto).



The Transformer - model architecture.

Tres grandes tipos de Transformers

Solo decodificador

Ejemplo:

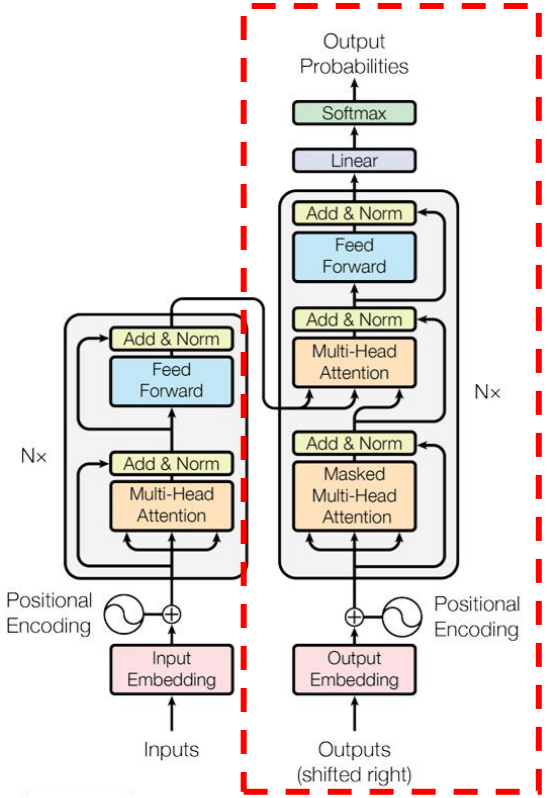
GPT (Generative Pre-trained Transformer) – OpenAI

Características:

- Usa **enmascaramiento causal (causal masking)** → cada palabra solo ve las anteriores.
- Genera texto **de manera autoregresiva**, palabra por palabra.
- No tiene bloque codificador; su entrada y salida son secuencias del mismo tipo.

Tareas típicas:

- **Generación de texto** (narrativas, poesía, código).
- **Completado de texto** ("The pink elephant tried to …").
- **Chatbots (ChatGPT).**



The Transformer - model architecture.

Tres grandes tipos de Transformers

Codificador- Decodificador

Ejemplos:

T5 (Text-to-Text Transfer Transformer) – Google
BART (Bidirectional and Auto-Regressive Transformers) – Meta
Transformer original (Vaswani et al., 2017)

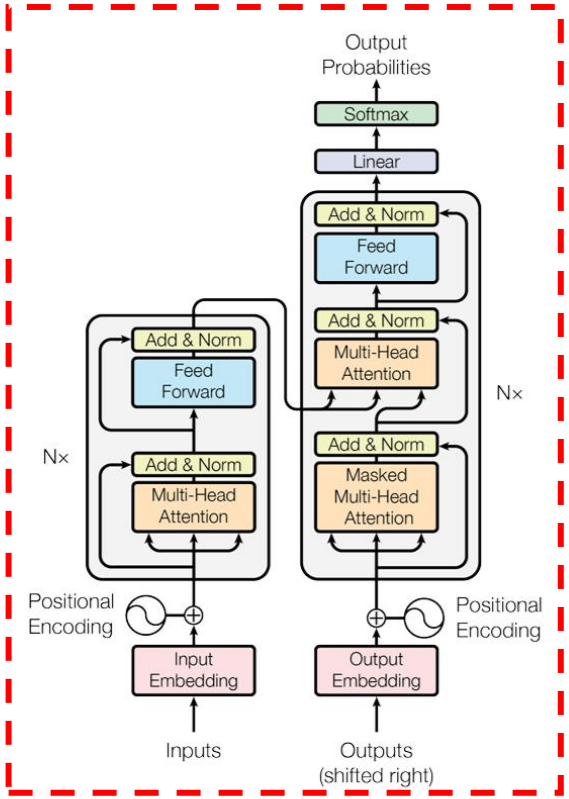
Características:

- Tiene dos componentes:
  - **Encoder** → procesa el texto fuente (por ejemplo, en inglés).
  - **Decoder** → genera la salida (por ejemplo, en español).

- El decoder **atiende tanto a las palabras anteriores** como a **las representaciones del encoder**.

Tareas típicas:

- **Traducción:** Inglés a Español
- **Resumen de texto:** Artículo a resumen.
- **Parafraseo:** Oración a reescritura equivalente.



The Transformer - model architecture.

¿El GPT que programamos es un Transformer de qué tipo?

¿Por qué?

GPT es un Transformer del tipo (solo) decodificador

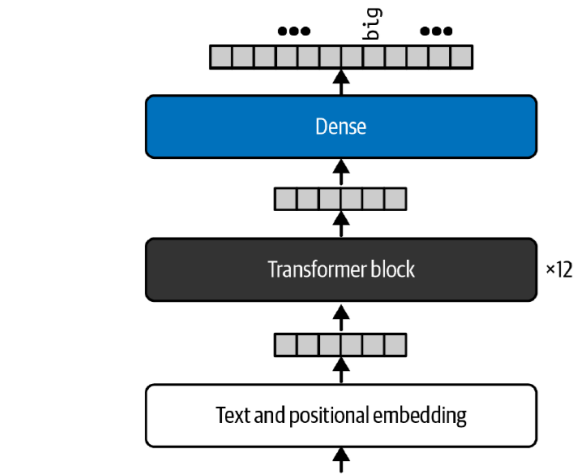
Tarea principal
Generación de texto

Se le llama así porque:

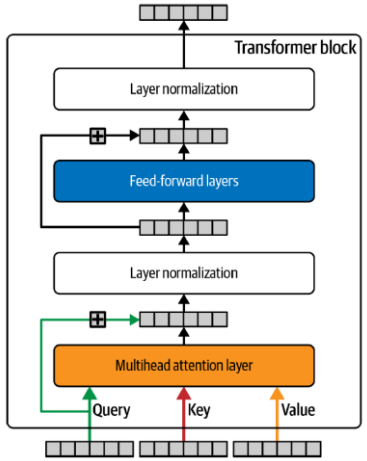
- Usa únicamente la mitad decodificadora de la arquitectura Transformer original presentada en *Attention Is All You Need* (Vaswani et al., 2017).
- y elimina la conexión con el encoder, porque su tarea no es traducir entre secuencias, sino predecir el siguiente token en una secuencia de texto.

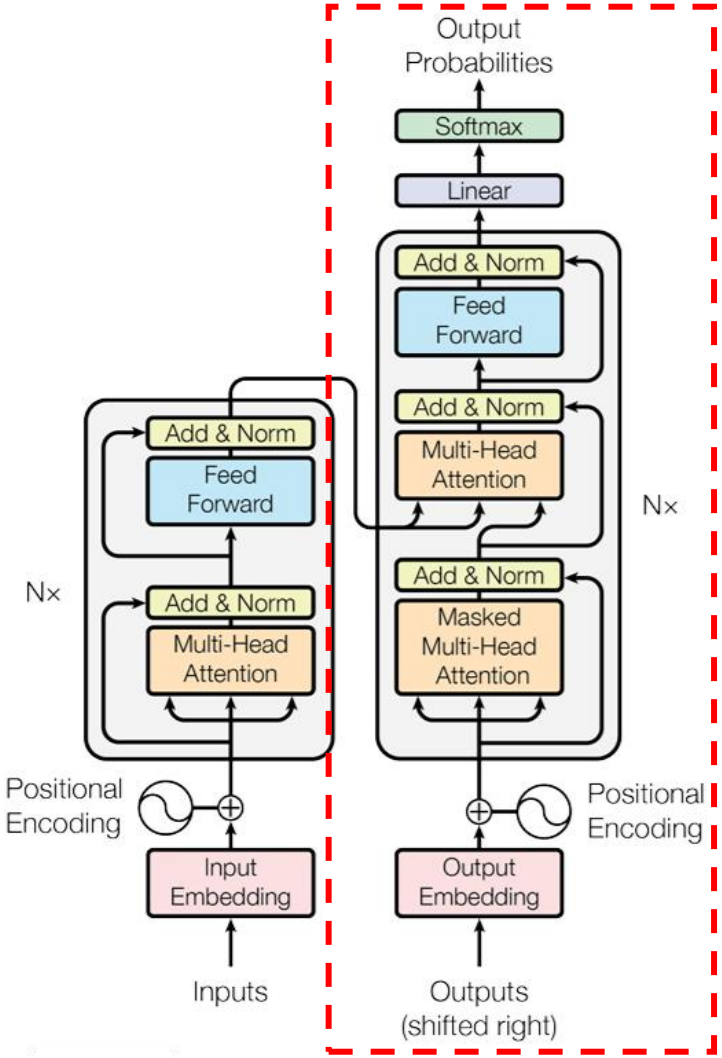
Por eso GPT:

- Usa auto-atención con máscara causal
- No usa cross-attention (no necesita un encoder)
- Funciona de forma autoregresiva, generando texto paso a paso



the pink elephant tried to get into the car but it was too





The Transformer - model architecture.

¿Por qué se le llama Transformer?

¿Qué transforma?

¿Por qué se le llama Transformer?

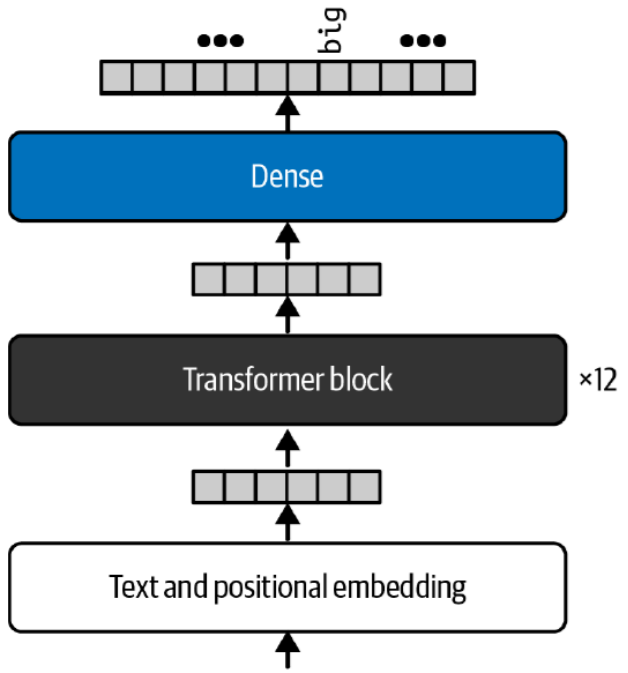
¿Qué transforma?

Se llaman **Transformers** porque su arquitectura **transforma**
una secuencia de representaciones (embeddings) en representaciones contextuales,
paso a paso,
utilizando **mecanismos de atención** en lugar de recurrencia o convolución.

Tercera transformación: de significado a salida generada.

Segunda transformación: de números a significado.

Primera transformación: de texto a números.



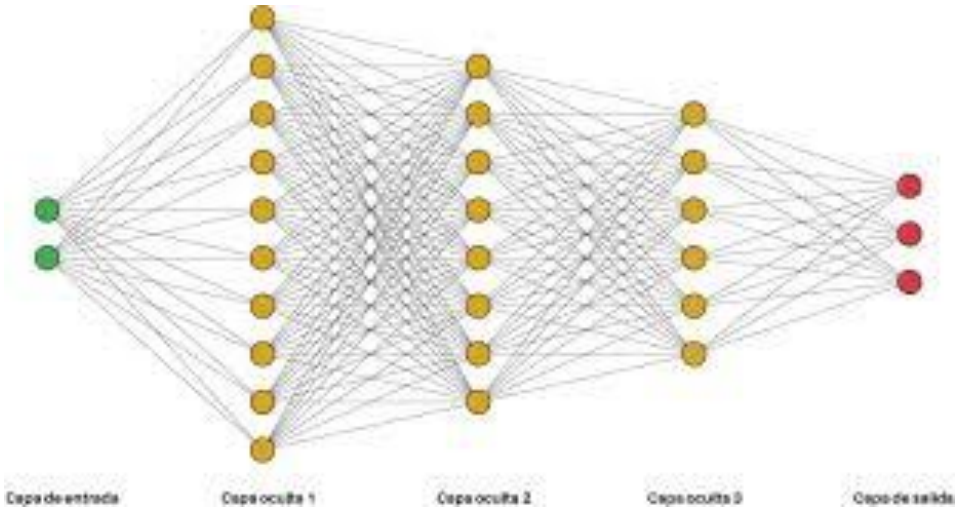
Finalmente, el bloque **Dense** toma esas representaciones y las **transforma en una predicción**, por ejemplo, la palabra siguiente ("big").

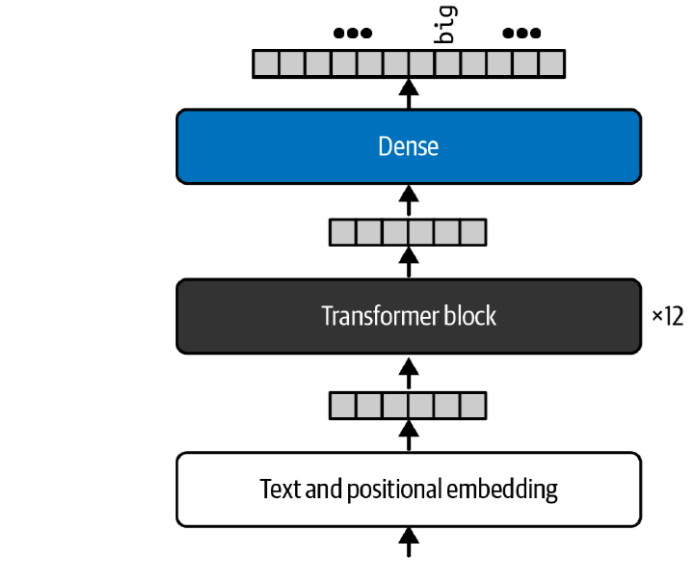
Esos vectores pasan por varias capas de **Transformer block**, donde cada una aplica **atención** y **normalización** para generar representaciones más profundas y contextuales.

Entrada textual ("The pink elephant...") se convierte en **vectores numéricos** mediante el bloque **Text and positional embedding**.

¿Por qué se le llama Transformer?

Todas las redes profundas transforman representaciones.
Pero los *Transformers* se llaman así porque su principio fundamental **es transformar directamente relaciones entre los elementos de la secuencia**, no solo aplicar filtros o recurrencias.





the pink elephant tried to get into the car but it was too

Evolución de GPT

Model	Date	Layers	Attention heads	Word embedding size	Context window	# parameters	Training data
GPT	Jun 2018	12	12	768	512	120,000,000	BookCorpus: 4.5 GB of text from unpublished books
GPT-2	Feb 2019	48	48	1,600	1,024	1,500,000,000	WebText: 40 GB of text from outbound Reddit links
GPT-3	May 2020	96	96	12,888	2,048	175,000,000,000	CommonCrawl, WebText, English Wikipedia, book corpora and others: 570 GB
GPT-4	Mar 2023	-	-	-	Variante hasta ~128 k tokens	No divulgado oficialmente	Multimodal, varios dominios de entrenamiento.
GPT-5	Ago 2025					No divulgado oficialmente	Sistema unificado multimodal (texto, imágenes, etc.)

Datos públicamente disponibles (como datos de Internet), datos licenciados de terceros, y otros datos obtenidos legalmente.

Evolución de GPT

Entre 2018 y 2021, escalar era la principal vía de mejora.

Durante la primera etapa del desarrollo de LLMs (GPT-1 ⟶ GPT-3), el patrón era claro:

Mayor tamaño del modelo y mas datos ⇒ mejor desempeño

- Los experimentos de *scaling laws* de OpenAI y DeepMind (Kaplan et al., 2020) mostraron que la **pérdida decrecía log-linealmente** con el aumento en cómputo, tamaño del dataset y número de parámetros.

- En esa época, el crecimiento en capacidades emergentes — razonamiento aritmético, coherencia contextual, generación de código— correlacionaba directamente con el escalamiento.

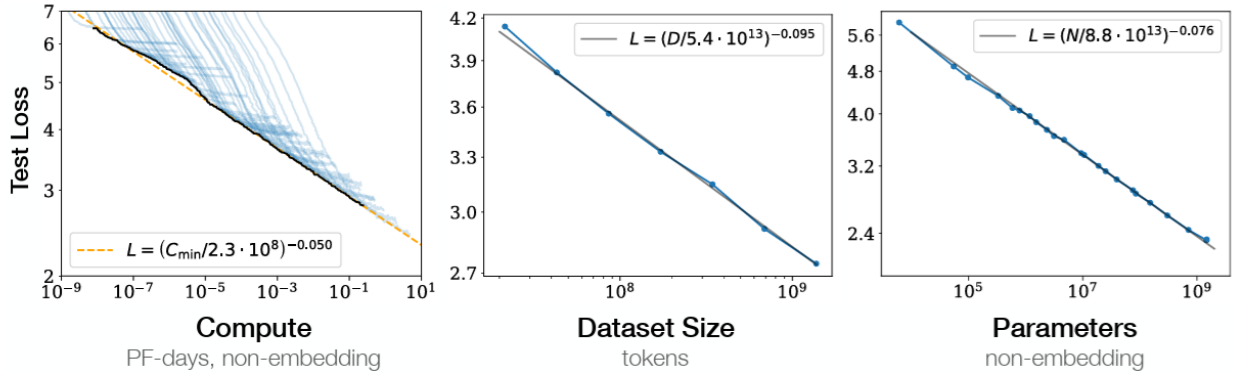


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). *Scaling laws for neural language models*. arXiv preprint arXiv:2001.08361.

Evolución de GPT

Actualmente: la afirmación ya no es completamente cierta

- A partir de GPT-4 y GPT-5, **el mero escalamiento deja de ser suficiente**.

- Los resultados de los modelos más recientes y de investigaciones paralelas (Google DeepMind 2024, OpenAI 2025) muestran que:

a) El *retorno marginal de escalar* está disminuyendo.
• Aumentar parámetros o datos ya no produce saltos drásticos de rendimiento.

• Los modelos comienzan a saturarse en tareas básicas (traducción, QA, inferencia común), donde el *benchmark* roza el 100 %.

• Ejemplo: pasar de GPT-4 a GPT-5 no ha producido un aumento espectacular en precisión, sino en **eficiencia, contexto y multimodalidad**.

b) El avance actual proviene de otras direcciones:

1. Aplicación de técnicas posteriores al **preentrenamiento** llamadas fine-tuning de alineación, como:

- **RLHF (Reinforcement Learning from Human Feedback)**

  Etapas:
  1. El modelo base genera varias respuestas posibles a una misma pregunta.
  2. Anotadores humanos las clasifican de mejor a peor según criterios de calidad (precisión, claridad, tono).
  3. Se entrena un **modelo de recompensa** (*reward model*) para predecir esas preferencias.
  4. El modelo principal se optimiza con **PPO** (Proximal Policy Optimization) para maximizar esa recompensa.

  Resultado:
  El modelo aprende no solo a ser correcto, sino también **a responder de manera preferida por humanos** (más útil, menos dañina, más educada).

  GPT-3.5 y ChatGPT 2022 fueron los primeros modelos públicos entrenados con RLHF.

Evolución de GPT

Actualmente: la afirmación ya no es completamente cierta

- A partir de GPT-4 y GPT-5, **el mero escalamiento deja de ser suficiente**.

- Los resultados de los modelos más recientes y de investigaciones paralelas (Google DeepMind 2024, OpenAI 2025) muestran que:

a) El *retorno marginal de escalar* está disminuyendo.
• Aumentar parámetros o datos ya no produce saltos drásticos de rendimiento.

• Los modelos comienzan a saturarse en tareas básicas (traducción, QA, inferencia común), donde el *benchmark* roza el 100 %.

• Ejemplo: pasar de GPT-4 a GPT-5 no ha producido un aumento espectacular en precisión, sino en **eficiencia, contexto y multimodalidad**.

b) El avance actual proviene de otras direcciones:

1. **Aplicación de técnicas posteriores al preentrenamiento** llamadas fine-tuning de alineación, como:

- Auto-crítica o *self-reflection*

  Idea:
  El propio modelo evalúa y corrige sus respuestas, sin intervención humana.
  Tras generar una respuesta, se le pide "reflexionar" sobre su exactitud o consistencia.
  Produce una nueva versión mejorada con base en esa evaluación.
  Este proceso puede iterarse (→ *self-consistency* o *tree of thoughts*).

  Resultado:
  Se entrena una forma de **razonamiento metacognitivo**, donde el modelo aprende a "pensar sobre sus propios errores".

Evolución de GPT

Actualmente: la afirmación ya no es completamente cierta

- A partir de GPT-4 y GPT-5, **el mero escalamiento deja de ser suficiente**.

- Los resultados de los modelos más recientes y de investigaciones paralelas (Google DeepMind 2024, OpenAI 2025) muestran que:

a) El *retorno marginal de escalar* está disminuyendo.

• Aumentar parámetros o datos ya no produce saltos drásticos de rendimiento.

• Los modelos comienzan a saturarse en tareas básicas (traducción, QA, inferencia común), donde el *benchmark* roza el 100 %.

• Ejemplo: pasar de GPT-4 a GPT-5 no ha producido un aumento espectacular en precisión, sino en **eficiencia, contexto y multimodalidad**.

b) El avance actual proviene de otras direcciones:

1. Aplicación de técnicas posteriores al **preentrenamiento** llamadas fine-tuning de alineación, como:

- "Reflection tuning" o *fine-tuning de reflexión*

  Un tipo de ajuste fino donde el modelo se entrena explícitamente en pares (*respuesta inicial*, *auto-revisión mejorada*) para reforzar comportamientos de **autocorrección**.
  Se genera un corpus de reflexiones (por el modelo o por humanos).
  Se entrena el modelo para imitar el patrón de "detectar un error y corregirlo razonadamente".

  Resultado:
  El modelo desarrolla un estilo de razonamiento más deliberativo, menos impulsivo y más preciso.

Evolución de GPT

Actualmente: la afirmación ya no es completamente cierta

- A partir de GPT-4 y GPT-5, **el mero escalamiento deja de ser suficiente**.

- Los resultados de los modelos más recientes y de investigaciones paralelas (Google DeepMind 2024, OpenAI 2025) muestran que:

a) El *retorno marginal de escalar* está disminuyendo.

• Aumentar parámetros o datos ya no produce saltos drásticos de rendimiento.

• Los modelos comienzan a saturarse en tareas básicas (traducción, QA, inferencia común), donde el *benchmark* roza el 100 %.

• Ejemplo: pasar de GPT-4 a GPT-5 no ha producido un aumento espectacular en precisión, sino en **eficiencia, contexto y multimodalidad**.

b) El avance actual proviene de otras direcciones:

1. **Aplicación de técnicas posteriores al preentrenamiento** llamadas fine-tuning de alineación.

2. **Arquitecturas híbridas y multimodales** (texto + imagen + voz + video).

3. **Inferencia más larga y razonamiento paso a paso (Tree of Thoughts, self-consistency)**.

4.**Entrenamiento con datos de mayor calidad, no solo más volumen**.

Evolución de GPT

Actualmente: la afirmación ya no es completamente cierta

- A partir de GPT-4 y GPT-5, **el mero escalamiento deja de ser suficiente**.

- Los resultados de los modelos más recientes y de investigaciones paralelas (Google DeepMind 2024, OpenAI 2025) muestran que:

a) El *retorno marginal de escalar* está disminuyendo.

• Aumentar parámetros o datos ya no produce saltos drásticos de rendimiento.

• Los modelos comienzan a saturarse en tareas básicas (traducción, QA, inferencia común), donde el *benchmark* roza el 100 %.

• Ejemplo: pasar de GPT-4 a GPT-5 no ha producido un aumento espectacular en precisión, sino en **eficiencia, contexto y multimodalidad**.

b) El avance actual proviene de otras direcciones:

1. **Aplicación de técnicas posteriores al preentrenamiento** llamadas fine-tuning de alineación.

2. **Arquitecturas híbridas y multimodales** (texto + imagen + voz + video).

3. **Inferencia más larga y razonamiento paso a paso (Tree of Thoughts, self-consistency)**.

4.**Entrenamiento con datos de mayor calidad, no solo más volumen**.

c) El límite físico-económico se aproxima.

• Escalar 10× los parámetros exige **energía y cómputo** prohibitivos.

• Los *gains* de rendimiento siguen una curva logarítmica: enormes costos, mejoras pequeñas.

• De ahí el cambio de enfoque hacia modelos más "eficientes y reflexivos".

OpenAI pide al Gobierno de EE.UU. que genere más energía eléctrica para no perder contra China

Sam Altman acaba de enviar una carta al Gobierno estadounidense con un mensaje claro: **necesitamos 100 gigavatios de nueva capacidad energética cada año** si queremos mantener el liderazgo en IA frente a China. Para dimensionar la magnitud, China añadió 429 gigavatios el año pasado mientras Estados Unidos apenas llegó a 51. La brecha es brutal y OpenAI está tocando todas las alarmas mientras construye Stargate, su red de centros de datos valorada en $500.000 millones.

El argumento de Altman convierte la energía en un tema de seguridad nacional, no solo un servicio público.

La IA está redefiniendo qué significa ser una potencia tecnológica, y resulta que todo empieza con electricidad, mucha electricidad.

c) El límite físico-económico se aproxima.

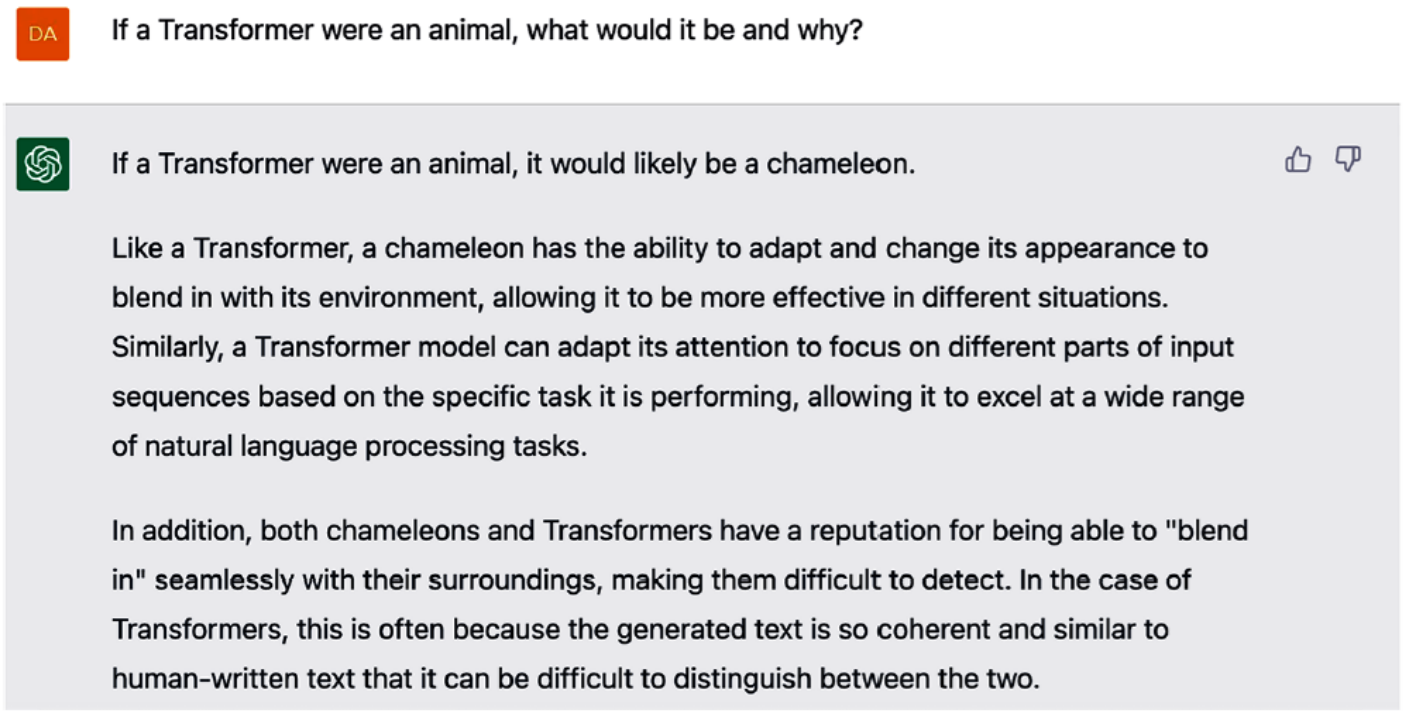
• Escalar 10× los parámetros exige **energía y cómputo** prohibitivos.

• Los *gains* de rendimiento siguen una curva logarítmica: enormes costos, mejoras pequeñas.

• De ahí el cambio de enfoque hacia modelos más "eficientes y reflexivos".

ChatGPT

- ChatGPT fue lanzado públicamente en **noviembre de 2022** como una interfaz conversacional sobre los modelos de lenguaje de OpenAI.
- Su lanzamiento representó la **primera versión ampliamente accesible** de un modelo generativo conversacional.
- La versión inicial estaba basada en **GPT-3.5**, una evolución de GPT-3 **ajustada específicamente para mantener conversaciones coherentes y naturales**.
- A diferencia de los modelos previos, ChatGPT **mantiene el contexto entre mensajes**: comprende referencias, reformulaciones y continuidad temática.



ChatGPT

¿Cómo se entrenó?

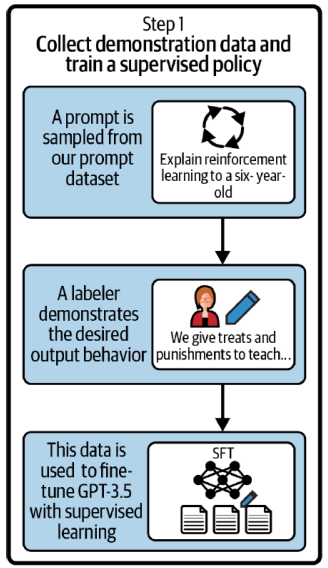
ChatGPT

¿Cómo se entrenó?

1. Supervised Fine-Tuning (SFT)

- Se recopilan ejemplos de conversaciones humanas (*prompts + respuestas ideales*).

- GPT-3.5 se ajusta mediante **aprendizaje supervisado** para imitar ese estilo conversacional humano.

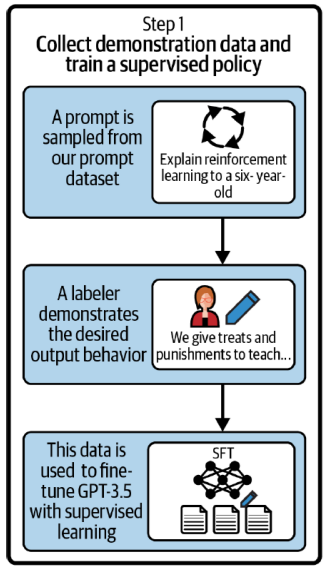


ChatGPT

¿Cómo se entrenó?

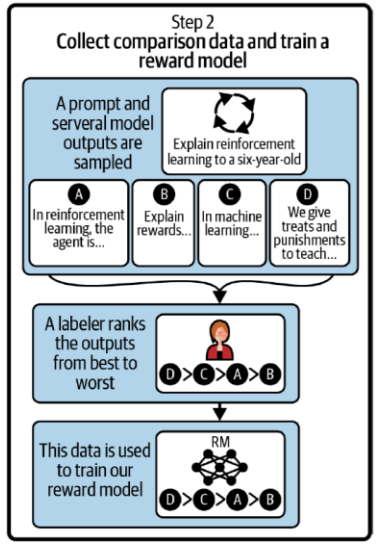
1. Supervised Fine-Tuning (SFT)

- Se recopilan ejemplos de conversaciones humanas (*prompts + respuestas ideales*).
- GPT-3.5 se ajusta mediante **aprendizaje supervisado** para imitar ese estilo conversacional humano.



2. Reward Modeling (RM)

- Evaluadores humanos califican múltiples respuestas generadas por el modelo (de mejor a peor).
- Se entrena un **modelo de recompensa** que aprende a predecir cuál respuesta sería preferida por un humano.

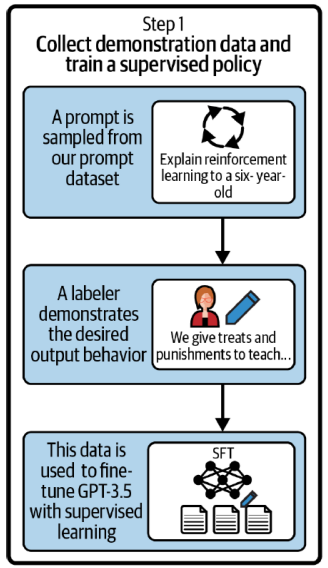


ChatGPT

¿Cómo se entrenó?

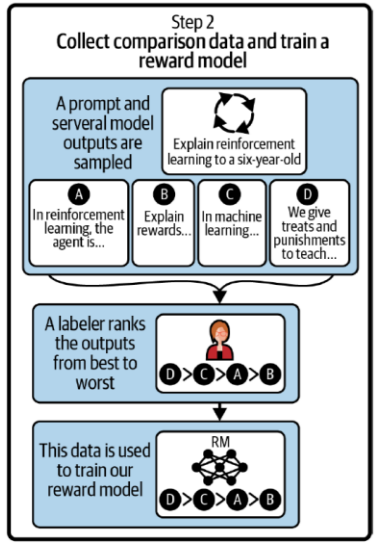
1. Supervised Fine-Tuning (SFT)

- Se recopilan ejemplos de conversaciones humanas (*prompts + respuestas ideales*).
- GPT-3.5 se ajusta mediante **aprendizaje supervisado** para imitar ese estilo conversacional humano.



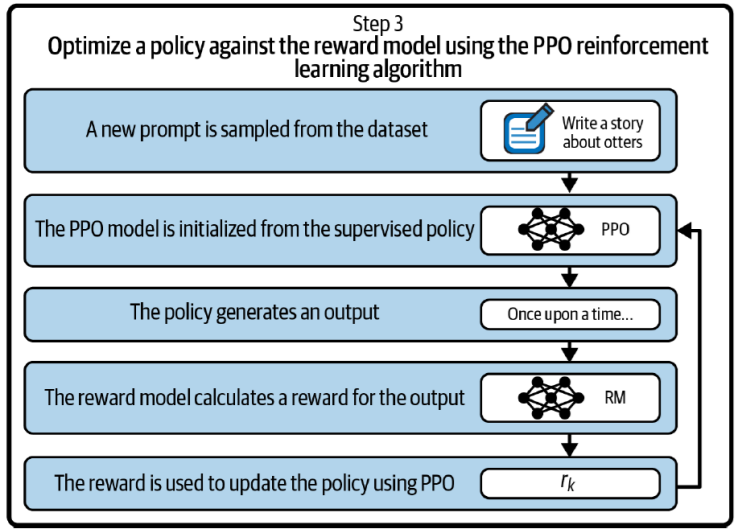
2. Reward Modeling (RM)

- Evaluadores humanos califican múltiples respuestas generadas por el modelo (de mejor a peor).
- Se entrena un **modelo de recompensa** que aprende a predecir cuál respuesta sería preferida por un humano.



3. Reinforcement Learning (PPO)

- El modelo ajustado se convierte en una política dentro de un entorno de conversación.
- Se usa **Proximal Policy Optimization (PPO)** para ajustar los pesos del modelo y **maximizar la "recompensa"** otorgada por el modelo de preferencia.



ChatGPT

¿Cómo se entrenó?

1. Supervised Fine-Tuning (SFT)

- Se recopilan ejemplos de conversaciones humanas (*prompts + respuestas ideales*).
- GPT-3.5 se ajusta mediante **aprendizaje supervisado** para imitar ese estilo conversacional humano.

2. Reward Modeling (RM)

- Evaluadores humanos califican múltiples respuestas generadas por el modelo (de mejor a peor).
- Se entrena un **modelo de recompensa** que aprende a predecir cuál respuesta sería preferida por un humano.

3. Reinforcement Learning (PPO)

- El modelo ajustado se convierte en una política dentro de un entorno de conversación.
- Se usa **Proximal Policy Optimization (PPO)** para ajustar los pesos del modelo y **maximizar la "recompensa"** otorgada por el modelo de preferencia.

ChatGPT aprende no solo a completar texto, sino a **responder como preferiría un humano**.

ChatGPT

Limitaciones

- A pesar de su capacidad, ChatGPT **puede "alucinar" información incorrecta o inventada**.

- OpenAI afirma que GPT-5 "**is significantly less likely to hallucinate than our previous models**".
- En su artículo "Why language models hallucinate", OpenAI aclara que aunque se han reducido, las alucinaciones **"still occur"** incluso con GPT-5.
- Según análisis externos, GPT-5 mostró una tasa de error factual más baja (~1.4%) frente a versiones anteriores (~1.8%) en ciertos tests de "grounded hallucination".
- El modelo aún depende de lo que "ha visto" en su entrenamiento: su conocimiento tiene un **corte temporal** y no siempre integra información posterior al corte. (Lo mismo aplica a versiones previas).

ChatGPT

Limitaciones

- A pesar de su capacidad, ChatGPT **puede "alucinar" información incorrecta o inventada**.

- Tiende a responder **con exceso de confianza**, incluso cuando no dispone de información verificable.

- Depende totalmente del texto visto en su entrenamiento y de la formulación del *prompt*.

- OpenAI afirma que GPT-5 "**is significantly less likely to hallucinate than our previous models**".

- En su artículo "Why language models hallucinate", OpenAI aclara que aunque se han reducido, las alucinaciones **"still occur"** incluso con GPT-5.

- Según análisis externos, GPT-5 mostró una tasa de error factual más baja (~1.4%) frente a versiones anteriores (~1.8%) en ciertos tests de "grounded hallucination".

- El modelo aún depende de lo que "ha visto" en su entrenamiento: su conocimiento tiene un **corte temporal** y no siempre integra información posterior al corte. (Lo mismo aplica a versiones previas).

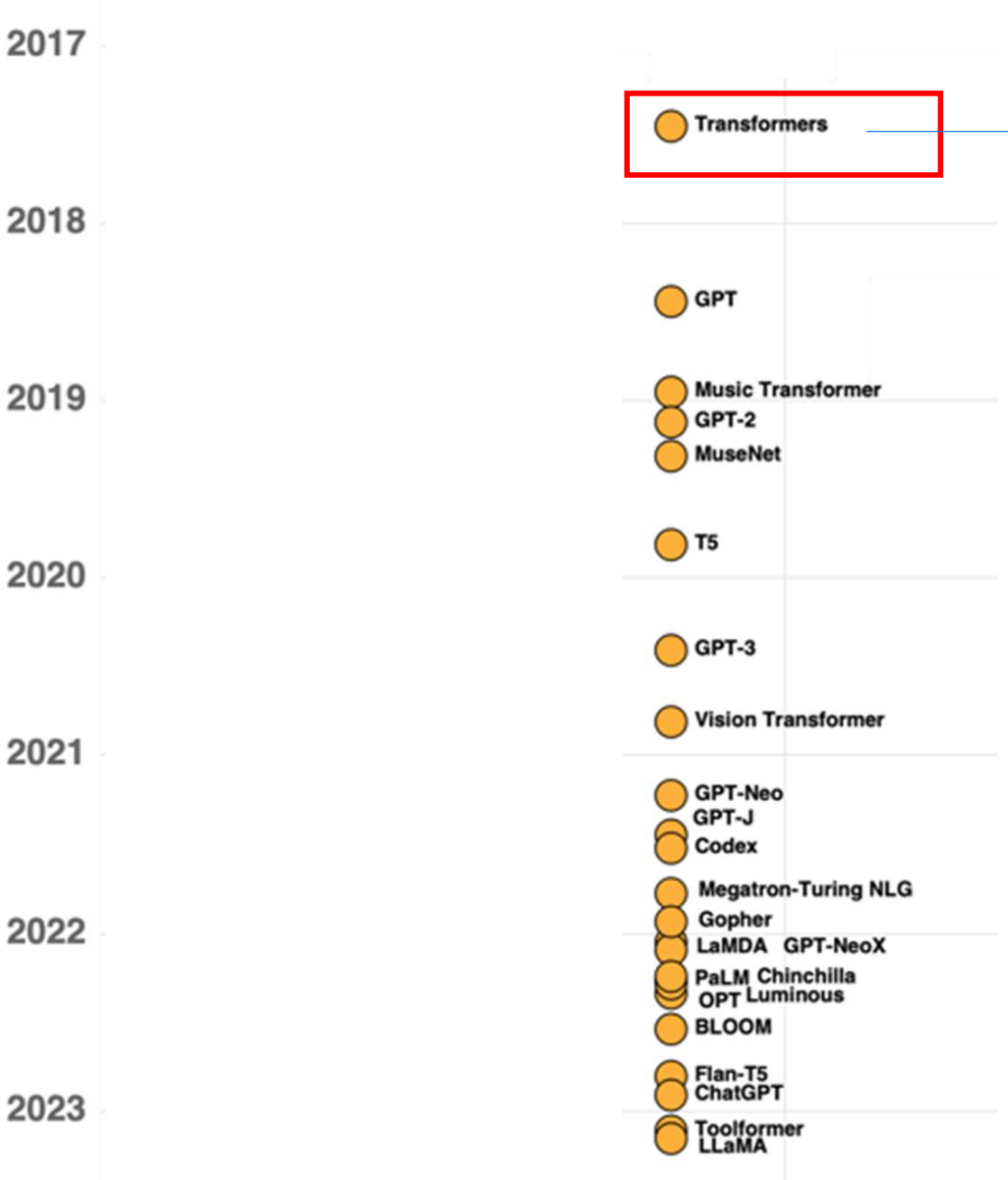
Impacto de ChatGPT

- ChatGPT marcó un **punto de inflexión** en la percepción pública de la IA generativa.
- Su éxito mostró cómo los **Transformers** pueden producir texto extenso, coherente y creativo.
- Inspiró una nueva generación de **interfaces conversacionales y co-creativas**.

ChatGPT

No es solo un modelo que genera texto:
es el resultado de combinar **Transformers, aprendizaje por refuerzo y retroalimentación humana**
para crear un agente capaz de comunicarse, razonar y adaptarse.

La era Transformer

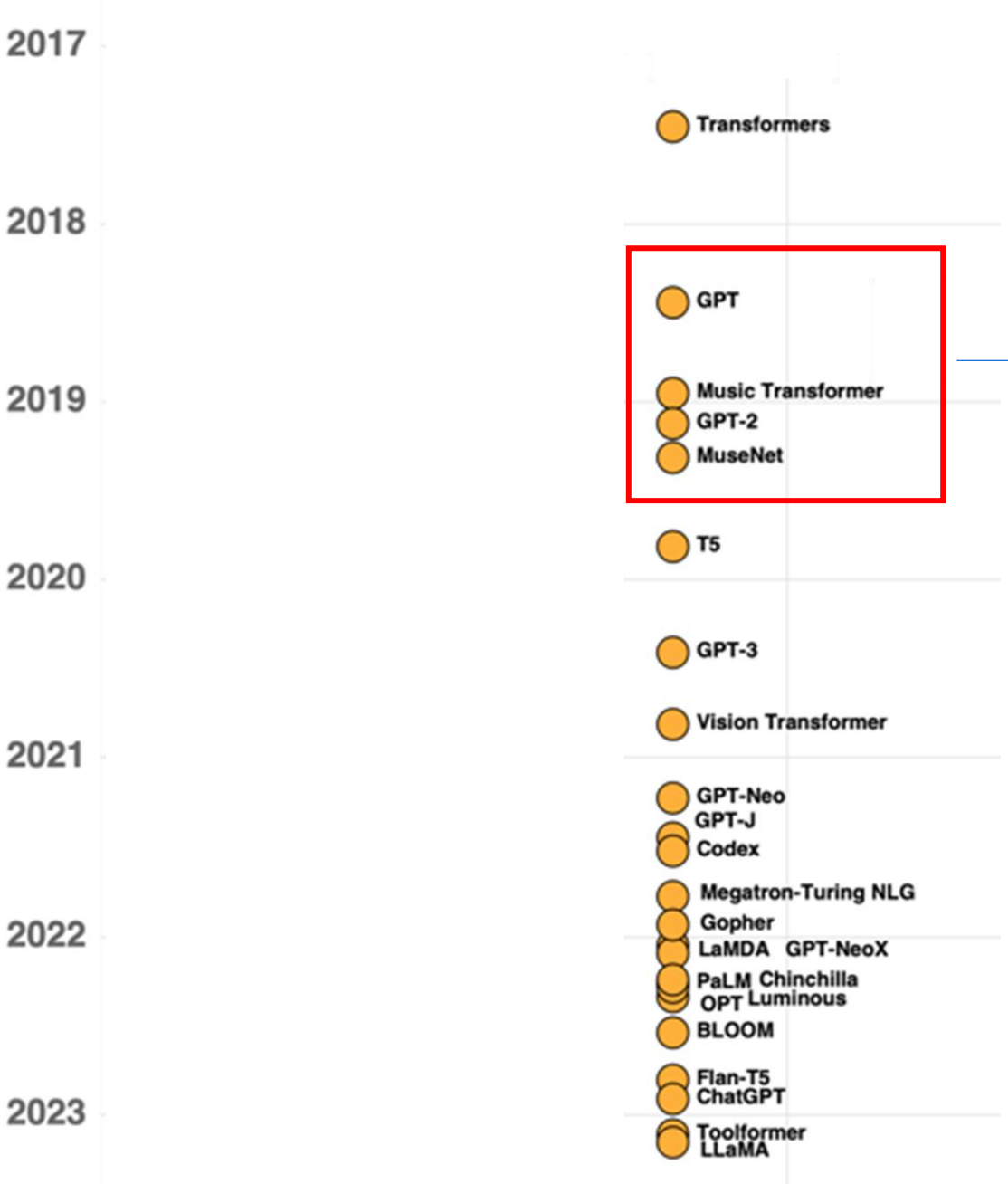


2017 – El inicio

- **Transformers** (Vaswani et al., 2017) cambiaron completamente la forma de procesar secuencias: → sustituyeron la recurrencia por **auto-atención** paralelizable.

- Su diseño sirvió como **fundamento universal** para modelos de texto, imagen, música y multimodales.

La era Transformer



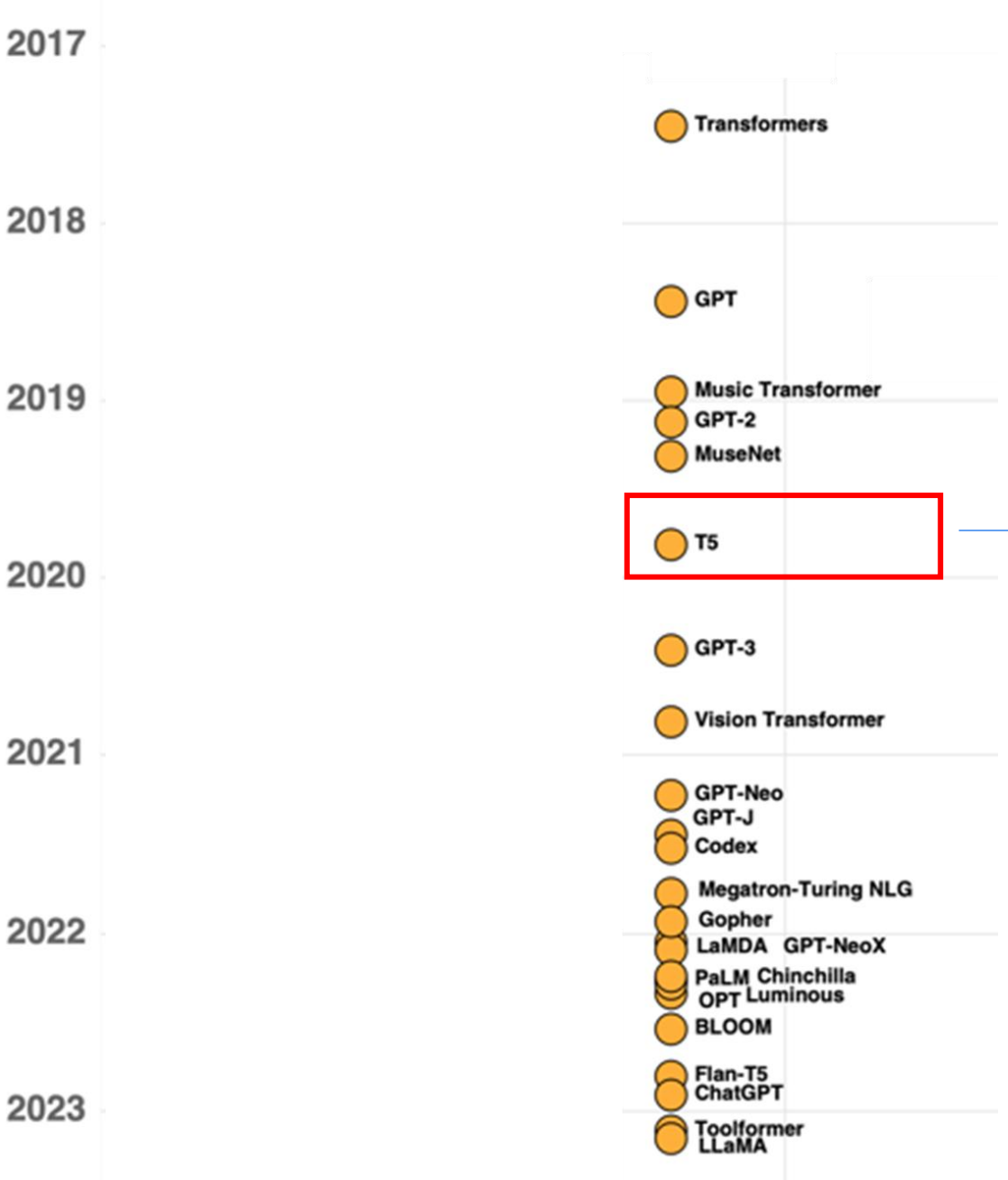
2018–2019 – Los primeros modelos de lenguaje

GPT (Generative Pre-trained Transformer, 2018):
→ primer *Transformer decoder-only*, entrenado para predecir el siguiente token.

GPT-2 (2019):
→ demostró la capacidad de los Transformers para generar texto extenso, coherente y creativo.
→ marcó el inicio de los **modelos de lenguaje autoregresivos a gran escala**.

Music Transformer (2019):
→ aplicó la atención a secuencias musicales, extendiendo el concepto más allá del texto.

La era Transformer



2020 – Modelos híbridos y multitarea

T5 (Text-To-Text Transfer Transformer):

→ propuso unificar todas las tareas de NLP como problemas de *traducción de texto a texto*.
→ enfoque *encoder-decoder*, predecesor de los modelos multitarea modernos.

Los Transformers empezaron a **superar a las RNNs y CNNs** en casi todas las tareas de procesamiento secuencial.

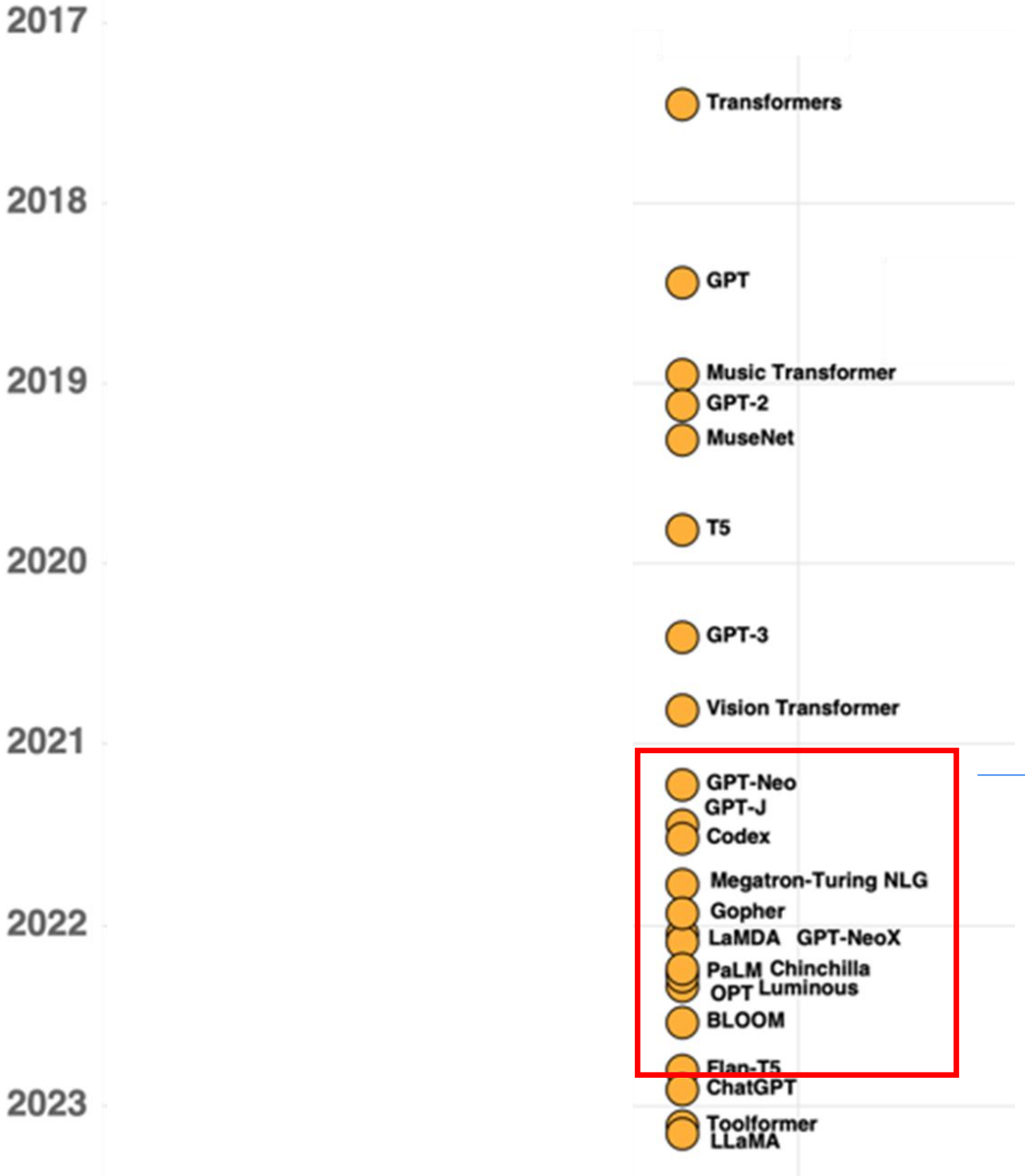
La era Transformer

2021 – Escalamiento masivo

GPT-3
→ 175 mil millones de parámetros, aprendizaje *in-context* sin reentrenamiento.
→ mostró que "**más tamaño = más capacidad de generalización**".

Vision Transformer (ViT):
→ llevó la arquitectura Transformer al dominio visual.
→ demostró que la atención puede reemplazar convoluciones.

La era Transformer



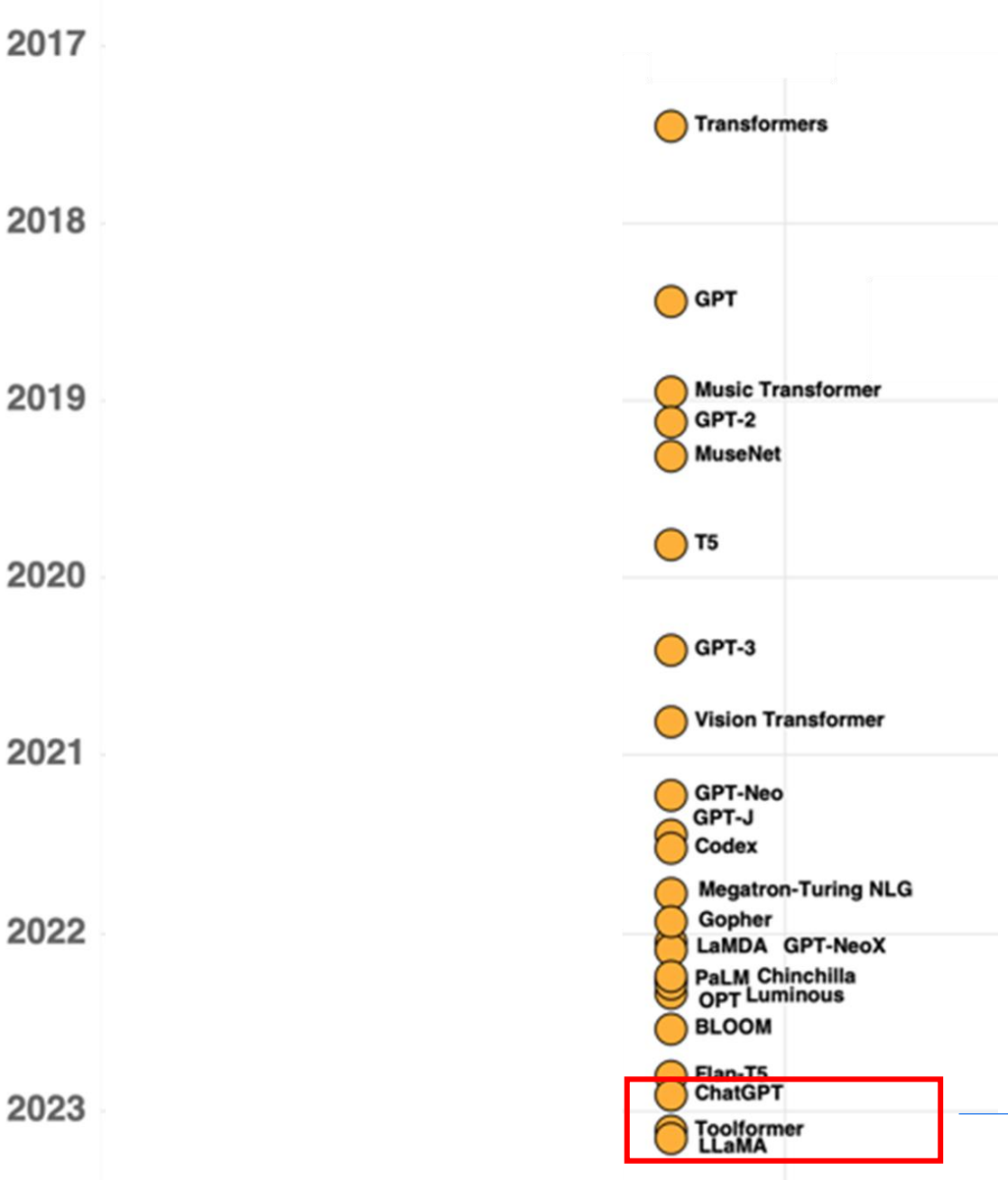
2022 – Democratización y variantes abiertas

Surgen versiones **abiertas y alternativas**:
→ **GPT-Neo, GPT-J, GPT-NeoX, BLOOM**, entrenados por la comunidad.

Los grandes laboratorios (Google, Meta, DeepMind) crean modelos como:
→ **PaLM, LaMDA, Gopher, Chinchilla**, cada uno optimizando tamaño, eficiencia y datos.

Se empieza a hablar del fenómeno de **alineación** (InstructGPT, RLHF).

La era Transformer



2023 – Conversación y multimodalidad

ChatGPT (2022–2023):
→ primera interfaz masiva sobre un modelo de lenguaje.
→ ajustado con RLHF para mantener diálogo natural y seguro.

Toolformer, LLaMA:
→ amplían el alcance del razonamiento y la interacción con herramientas externas.

GPT-4:
→ transición hacia **modelos multimodales**, capaces de procesar texto e imágenes.

Hasta GPT-4 y GPT-4o, los modelos eran principalmente **reactivos**:
respondían a un *prompt* → generaban una salida → se detenían.

… 2024-2025

Surge el concepto de **Agentic AI** (IA agentiva):
modelos que **razonan, planifican, actúan y se retroalimentan** en entornos complejos, combinando varios componentes:

- LLMs como núcleo cognitivo (razonamiento y lenguaje)
- **Memoria** para mantener contexto a largo plazo
- **Herramientas externas** (búsqueda, ejecución de código, navegación web, etc.)
- **Acción autónoma** (planificación y toma de decisiones iterativa)

Pasamos de “asistentes conversacionales” a “agentes autónomos de propósito general

- El corazón de estos agentes **sigue siendo un Transformer** (GPT, Claude, Gemini, etc.),
  pero ahora se le rodea de módulos que **extienden su agencia**:

Este cambio marca el paso de **IA generativa → IA agentiva**, donde los modelos pueden interactuar proactivamente con el mundo digital.

Modelos Multimodales I

Clase 21

Dra. Wendy Aguilar

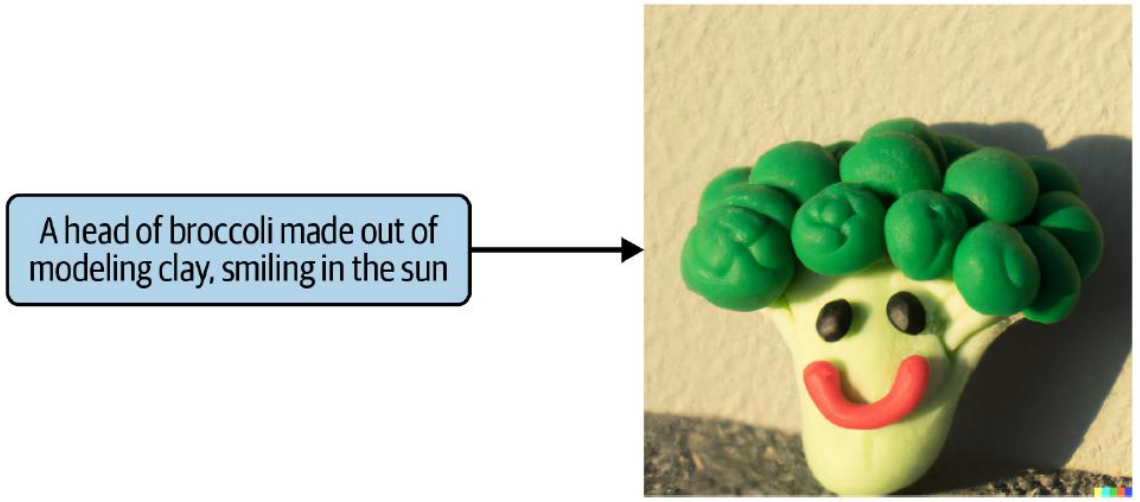
Modelos Generativos Profundos

UN ENFOQUE DESDE LA CREATIVIDAD COMPUTACIONAL

Modelos multimodales

- Los modelos multimodales aprenden a **relacionar y convertir información entre distintos tipos de datos**, como texto, imágenes, audio o video.
- Representan un avance hacia una **inteligencia artificial más general**, capaz de comprender y generar contenido combinando distintas modalidades sensoriales.
- En los últimos años, los modelos más impresionantes en generación creativa han sido **multimodales**, especialmente los de **visión y lenguaje**.

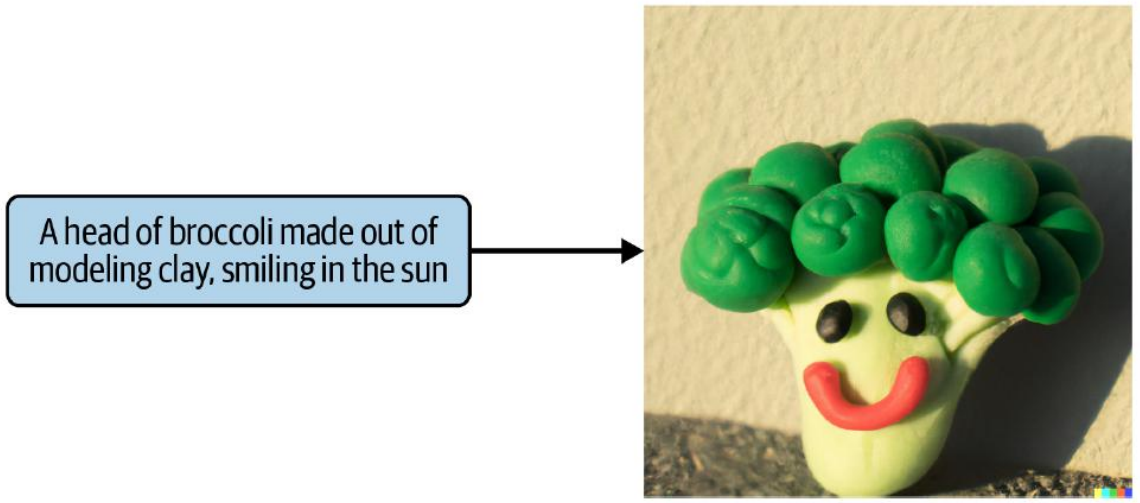
Generar imágenes a partir de texto (text-to-image).



El desafío del aprendizaje multimodal

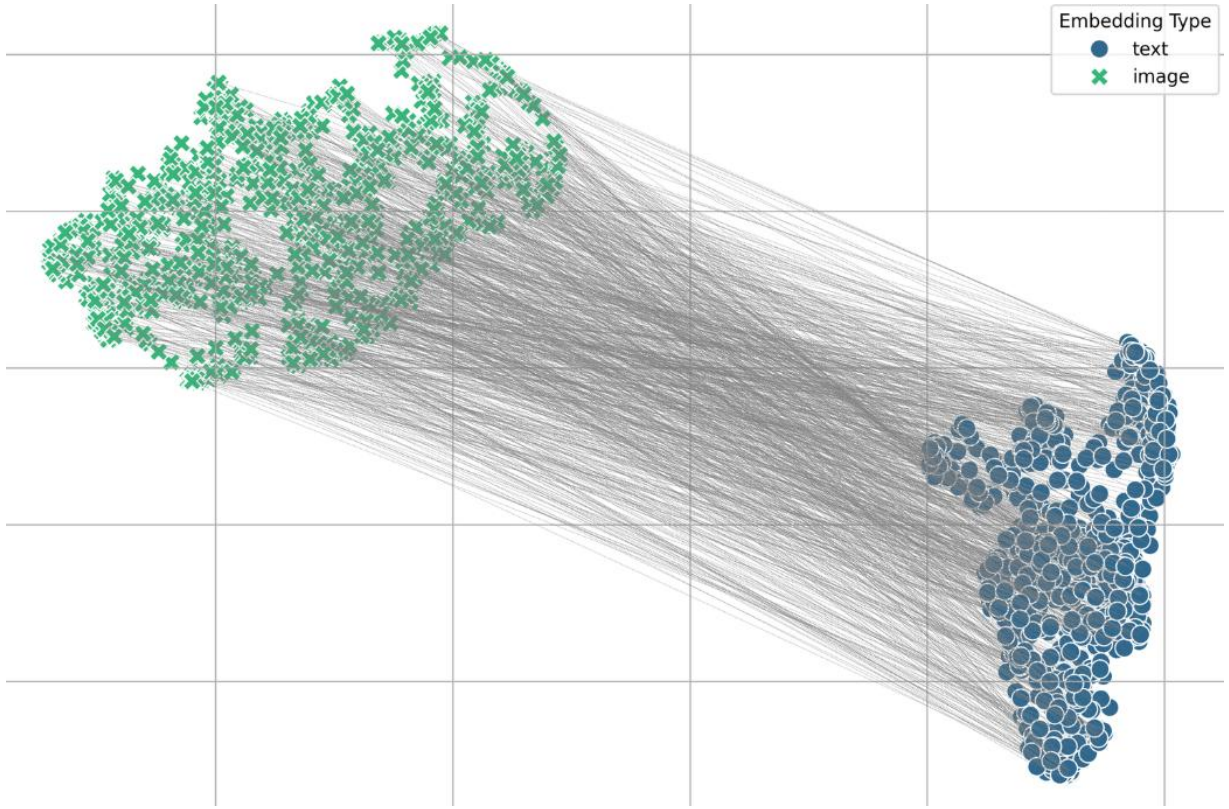
- Cada dominio (texto e imagen) presenta ya una gran complejidad por sí mismo.
- El reto añadido es **aprender una representación compartida** que permita traducir fielmente conceptos entre dos modalidades (texto e imagen).
- El modelo debe aprender a **"cruzar el puente" entre los dominios**, preservando el significado y los matices contextuales.
- Implica tanto **comprensión semántica** del lenguaje como **síntesis visual** coherente y realista.

Generar imágenes a partir de texto (text-to-image).



Traducción entre dominios

- El texto se codifica en un **espacio semántico latente**.
- Las imágenes se codifican en un **espacio visual latente**.
- El modelo multimodal aprende una **intersección o alineación** entre ambos espacios.
- Esto le permite generar imágenes que **corresponden al significado del texto**.

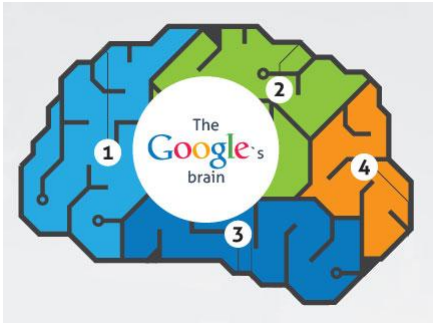


Implicaciones

- Los modelos multimodales abren paso a **sistemas creativos co-generativos**: texto → imagen → texto → video.
- Son la base de **aplicaciones de IA generativa** en diseño, arte, educación y comunicación.
- Se desea crear modelos que integren **visión, lenguaje, audio y acción**, capaces de razonar sobre el mundo de forma integrada.

Modelos representativos (2022-2023)

- DALL·E 2 (OpenAI) → Codifica texto e imagen en el espacio CLIP y usa *diffusion* para la generación.
- Imagen (Google Brain) → Usa un modelo de lenguaje muy potente y un decodificador de difusión para lograr fotorealismo extremo.
- Stable Diffusion (Stability AI, CompVis, Runway) → Modelo de difusión latente, eficiente y abierto, que popularizó la generación de imágenes creativas.
- Flamingo (DeepMind) → Modelo multimodal *few-shot* capaz de razonar y generar lenguaje condicionado por imágenes o video.



CompVis

runway

Google DeepMind

Modelos representativos (2022-2023)

- DALL·E 2 (OpenAI) → Codifica texto e imagen en el espacio CLIP y usa *diffusion* para la generación.
- Imagen (Google Brain) → Usa un modelo de lenguaje muy potente y un decodificador de difusión para lograr fotorealismo extremo.
- Stable Diffusion (Stability AI, CompVis, Runway) → Modelo de difusión latente, eficiente y abierto, que popularizó la generación de imágenes creativas.
- Flamingo (DeepMind) → Modelo multimodal *few-shot* capaz de razonar y generar lenguaje condicionado por imágenes o video.

Alexa, Angelo, Carlos, Rodrigo y Roy.
Miércoles 12 nov

Stable Diffusion

1. Contexto y motivación introductoria

Por qué surge Stable Diffusion, situándolo dentro del desarrollo histórico y técnico de los modelos generativos. La idea es que el público entienda **qué problema resolvió** y **por qué fue un punto de quiebre** en la IA generativa.

2. Arquitectura General

Explicar:

- La estructura interna del modelo.
- Cómo interactúan estos módulos para generar una imagen desde texto.

3. Proceso de Entrenamiento y Generación

- Teoría
- No intenten entrenar un modelo desde cero (innecesario y costoso).

4. Experimentación y Control Creativo

- Mostrar aplicaciones y control de resultados.
- Ejemplos reproducibles en **Hugging Face Spaces o Colab**.