

Transformadores

Procesamiento de lenguaje natural

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

5 de mayo de 2023



Antecedentes

- 1 Antecedentes
- 2 Arquitectura
- 3 Flujo residual
- 4 Cabezas de atención
- 5 Modelos de lenguaje de gran tamaño

Temas

1 Antecedentes

- Procesamiento de lenguaje natural (PLN)

Procesamiento de lenguaje natural (PLN)

Definición (PLN)

“El *Procesamiento del Lenguaje Natural (PLN)* es el campo de conocimiento de la Inteligencia Artificial que se ocupa de investigar la manera de **comunicar a las máquinas** con las personas mediante el uso de **lenguas naturales**, como el español, el inglés o el chino.”

Antonio Moreno^a

^a<https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/>

Componentes del PLN

Tradicionalmente al procesar lenguaje natural se pueden realizar los siguientes tipos de análisis, dependiendo de la aplicación a desarrollar:^[1]

Análisis morfológico o léxico Consiste en el análisis interno de las palabras que forman oraciones para extraer componentes léxicos, fragmentos de palabras que permiten identificar su categoría sintáctica y significado léxico.

Análisis sintáctico “Consiste en el análisis de la estructura de las oraciones de acuerdo con el modelo gramatical empleado (lógico o estadístico).”

Análisis semántico “Proporciona la interpretación de las oraciones, una vez eliminadas las ambigüedades morfosintácticas.”

Análisis pragmático “Incorpora el análisis del contexto de uso a la interpretación final. Aquí se incluye el tratamiento del lenguaje figurado (metáfora e ironía) como el conocimiento del mundo específico necesario para entender un texto especializado.”

^[1]Antonio Moreno Ibid.

Problemas de interés

Traducción automatizada Traducciones de textos de un idioma a otro, realizadas por la computadora.

Resúmenes de textos Extracción automática de resúmenes.

Pies de imágenes Dada una imagen, generar un texto que la describa. [2]

Clasificación de enunciados Dado un enunciado, indicar a qué clase pertenece. [3] Ej:

Basura Determinar si un correo es o no basura.

Análisis de sentimientos Dado un producto ¿los comentarios son positivos o negativos?

Verificación de hechos Revisa artículos y detecta *afirmaciones* vs *no afirmaciones* que puedan ser sujetas a verificación de veracidad.

[2] <https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

[3] Alammar s.f.

Arquitectura

- 1 Antecedentes
- 2 **Arquitectura**
- 3 Flujo residual
- 4 Cabezas de atención
- 5 Modelos de lenguaje de gran tamaño

Transformador (*Transformer*)

- Un *transformador* es una unidad de procesamiento diseñada para procesar lenguaje natural haciendo uso casi exclusivamente de **módulos de atención**.
- Originalmente se diseñó para realizar traducciones, pero hoy en día tiene muchas más aplicaciones.
- Observemos que el lenguaje natural es una secuencia temporal donde la función y significado de cada símbolo depende fuertemente de otros símbolos, probablemente a distancias considerables de él dentro de la frase. Ej:
*Usualmente el costo de la fruta no supera los \$30. Hoy encontré un mango que costaba \$60. ¡Qué **caro** estaba!*
¿A qué se refiere la palabra *caro*?
- Las redes recurrentes, incluso las LSTM no logran considerar correctamente relaciones entre datos tan separados.
- Algunos afirman que los transformadores son para el PLN lo que las capas convolucionales han sido para los problemas de visión.

Definición (Transformador)

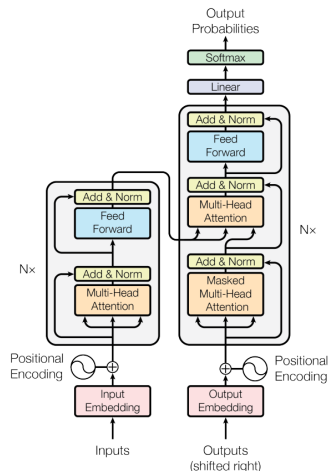
El *transformador* es un modelo de *transducción autorregresivo* que depende únicamente de **auto-atención** para calcular representaciones de sus entradas y salidas. Vaswani y col. 2017

- **Transducción:** f. Transformación de un tipo de señal en otro distinto. [Diccionario de la RAE].
- Que el modelo sea **autorregresivo** significa que, en cada paso, consume los símbolos generados previamente como entrada adicional a la hora de generar el siguiente.

Atención es todo lo que necesitas

Arquitectura original propuesta por Vaswani y col. 2017.

- Tiene una estructura codificador-decodificador.
- Se propuso para realizar traducciones entre idiomas.
- No contiene unidades recurrentes.



Codificador-decodificador

Codificador Recibe una secuencia de representaciones de símbolos (*tokens*):

$$(x_1, \dots, x_n)$$

y la convierte en una secuencia de representaciones continuas:

$$z = (z_1, \dots, z_n)$$

Decodificador Dada z genera la secuencia de salida

$$(y_1, \dots, y_m)$$

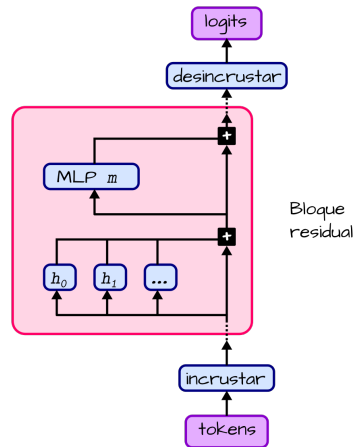
un símbolo a la vez. (Vaswani y col. 2017)

Transformadores decodificadores

Estructura genérica de un transformador decodificador autoregresivo.

- Un *bloque residual*^a tiene dos elementos:
 - 1 Una capa de atención con múltiples *cabezas*, que operan en paralelo.
 - 2 Un perceptrón multicapa.

^aOriginalmente llamado *incrustación* (*embedding*), pero que según Elhage y col. 2021 presenta propiedades diferentes a otras incrustaciones.



Resumen en: <https://www.lesswrong.com/posts/rEPnce975Fid9v5qv/brief-notes-on-transformers>

Temas

- 2 Arquitectura
 - Codificador
 - Decodificador
 - Entrenamiento

Preparación I

Para procesar los datos, estos pasan por dos etapas:

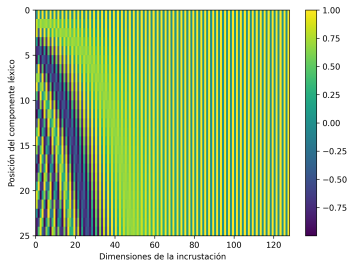
Incrustación Los componentes léxicos (*tokens*) que forman un enunciado entran como vectores *one hot* y son multiplicados por la matriz de incrustación (*embedding*) generando vectores de dimensión d_m :

$$X_E = W^E \text{tokens}$$

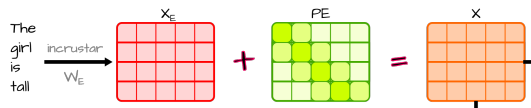
Codificación posicional Se genera un vector también de tamaño d_m que caracteriza la posición de la palabra en el enunciado:

$$\begin{aligned} PE_{(\text{pos}, 2i)} &= \sin(\text{pos}/10000^{2i/d_m}) \\ PE_{(\text{pos}, 2i+1)} &= \cos(\text{pos}/10000^{2i/d_m}) \end{aligned}$$

Preparación II



(a) Codificación posicional



(b) Proceso de incrustación

Finalmente ambos vectores se suman:

$$X = X_E + PE$$

Atención I

Este módulo se compone de varias cabezas H_i y un flujo residual que conserva el valor de X para la capa siguiente.

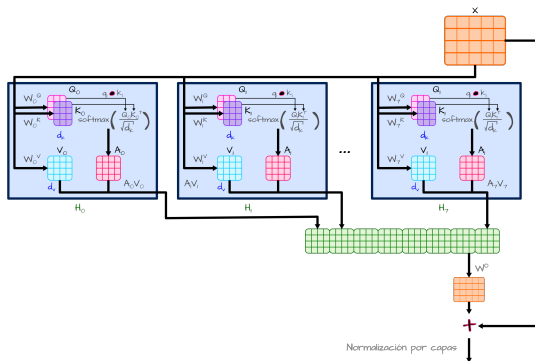


Figura: Módulo de atención con varias cabezas.

Atención II

Cabeza El procesamiento dentro de cada cabeza se aplica en varias etapas:

- 1 Por cada componente léxico se calculan tres vectores: consulta Q y llave K de tamaño d_k y valor V de tamaño d_v :

$$Q = W_Q X$$

$$K = W_K X$$

$$V = W_V X$$

- 2 Se calcula la matriz de atención con el producto punto de la consulta y cada llave correspondiente a los demás componentes:

$$A = \text{softmax} \left(\frac{1}{\sqrt{d_k}} [q_i \cdot k_j]_{i,j} \right) \quad \text{también escrito:}$$

Atención III

también escrito:

$$A = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right)$$

- 3 *Auto-atención* Multiplicar los valores por los vectores de atención, como efecto se mantendrán los valores de la palabra a la que más atención se debe poner, para traducir la palabra en cada posición, y se realiza una suma pesada:

$$H = AV$$

Atención IV

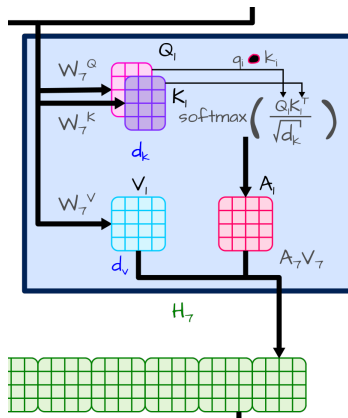


Figura: Cabeza

Atención V

Multicabeza Se concatenan las salidas de todas las cabezas y se combinan mediante la matriz W^O

- Se ejecutan las cabezas de atención y sus resultados se suman al flujo residual:

$$Z_h = H_h W_h^O$$
$$X = X + \sum_{h \in H} Z_h$$

- Originalmente se ve a W^O como una gran matriz de modo que:

$$Z = \text{Concatena}(H_0, H_1, \dots)$$
$$X = X + ZW^O$$

Atención VI

Normalización por capas Se normalizan los valores para todas las características para cada entrada, en lugar de normalizar característica por característica para todo el lote. De este modo se elimina la dependencia del lote. [4]

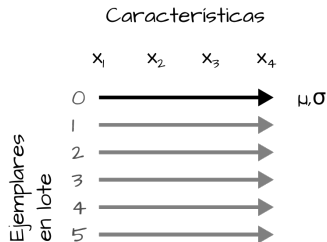


Figura: Normalización por capas.

Atención VII

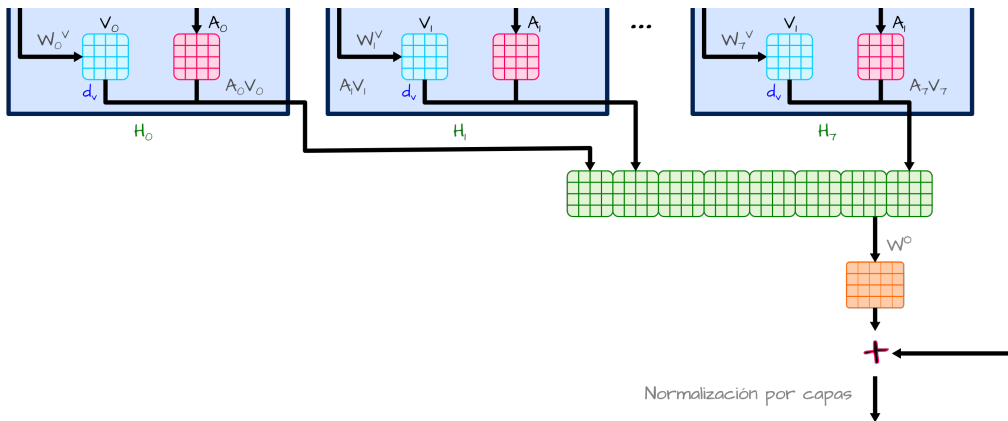


Figura: Concatenación y normalización por capas

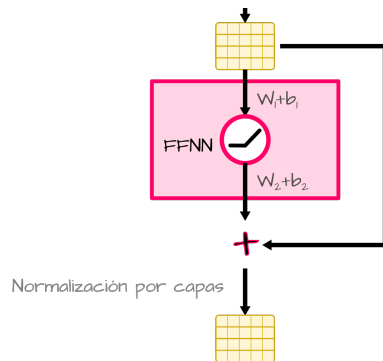
[4] Fuente: [What is Layer Normalization?](#) en "Build Better Deep Learning Models with Batch and Layer Normalization" por Bala Priya C.

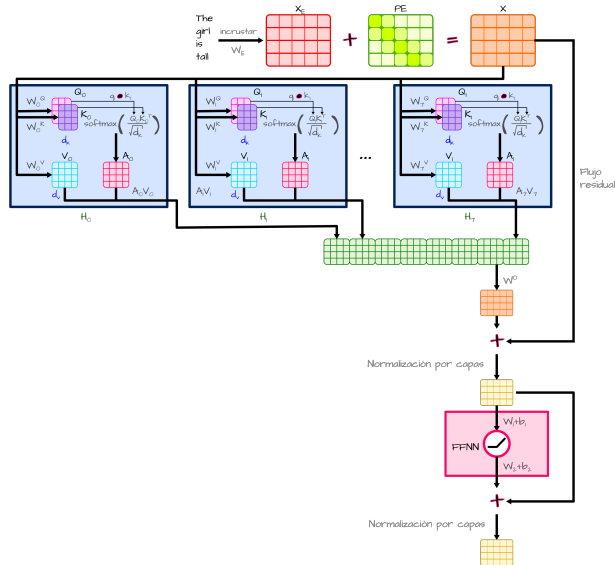
Perceptrón multicapa

- Una vez que se han procesado las entradas para considerar los efectos que deben tener unos componentes léxicos sobre otros, se procede a realizar transformaciones en el espacio vectorial mediante un **perceptrón multicapa**.

$$\text{FFN} = \max(0, XW_1 + B_1)W_2 + B_2$$

- Se termina con otra normalización por capas.
- La matriz a la salida tiene las mismas dimensiones que la matriz de entrada X .





Temas

- 2 Arquitectura
 - Codificador
 - Decodificador
 - Entrenamiento

Decodificador I

Codificador Se generan matrices K y V a partir de la salida del último bloque residual del codificador. Serán usadas tal cual por todos los bloques residuales del decodificador.

Preparación Autorregresión.

- En cada paso, el decodificador recibe como entrada a los componentes léxicos generados hasta ahora por él mismo.
- Se utilizan los mismos parámetros para incrustar y la codificación posicional que usa el codificador.

Bloque residual decodificador Tiene tres subcapas:

Auto-atención Recibe la versión codificada de los componentes léxicos generados hasta ahora.

Decodificador II

Atención codificador-decodificador Genera la matriz Q a partir la subcapa de auto-atención (palabras generadas hasta ahora) y utiliza las matrices K y V obtenidas de codificador.

Perceptrón multicapa FFN de dos capas y termina con normalización por capas.

Desincrustación Se multiplica por la matriz de desincrustación (*unembedding*) y se aplica softmax para determinar la probabilidad que tiene cada palabra de ser la siguiente:

$$\text{logits} = XW^u$$

$$Y = \text{softmax}(\text{logits})$$

se elige la palabra con la mayor probabilidad como salida para este paso.

Decodificador III

Existe también la posibilidad de mantener las n mejores transducciones en cada paso.

Final El proceso de transducción termina cuando el decodificador produce el símbolo especial `<eof>` (*end of sentence*), que también se debe encontrar en el vocabulario.

Temas

- 2 Arquitectura
 - Codificador
 - Decodificador
 - Entrenamiento

Parámetros

- Tanto las subcapas como las incrustaciones producen salidas de tamaño $d_m = 512$.
- Los pesos para las capas de incrustación se multiplican por $\sqrt{d_m}$.
- Número de cabezas de atención $h = 8$.
- Tamaño de perceptrones multicapa internos $d_{ff} = 2048$
 - Codificador
 - 6 bloques residuales para codificación.
 - Decodificador
 - 6 bloques residuales para decodificación.

Aprendizajes sobre el entrenamiento I

- Se usaron dos conjuntos de datos:
 - Inglés-Alemán. 4.5 millones de enunciados con 37,000 componentes léxicos.
 - Inglés-Francés. 36 millones de enunciados con 32,000 componentes léxicos.
- Se organizaron los lotes con enunciados de longitudes semejantes.
- Cada enunciado contenía alrededor de 25,000 componentes léxicos tanto en la entrada como en la salida.
- Se utiliza la **entropía cruzada** o la **divergencia de Kullback–Leibler** como función de error, para medir la distancia entre la distribución de probabilidad de las palabras probables y el vector *one hot* esperado.
- Se entrenó con Adam, variando la tasa de aprendizaje conforme progresaba el entrenamiento y Dropout.
- El proceso en detalle se puede consultar en Vaswani y col. 2017. Se tardaron 3 días y medio entrenando.

Flujo residual

- 1 Antecedentes
- 2 Arquitectura
- 3 Flujo residual
- 4 Cabezas de atención
- 5 Modelos de lenguaje de gran tamaño

Flujo residual

- Todos los componentes de un transformador:

- Incrustamiento de símbolos (*token embedding*).
- Cabezas de atención.
- Capas MLP.
- Desincrustamiento.

se comunican entre sí leyendo y escribiendo a subespacios diferentes del flujo residual.

- Por ello es útil descomponer el flujo residual en estos canales de comunicación, que corresponden a los caminos a través del modelo. Elhage y col. 2021

Cabezas de atención

Atención múltiple I

- El punto crucial de los transformadores es su capacidad para poner atención a subespacios diferentes de la representación codificada en el flujo residual en paralelo.

Reescribiendo las operaciones de las cabezas en notación tensorial se obtiene:

$$\begin{aligned}
 h(x)_i &= W_O \left(\sum_j A_{i,j} W_V x_j \right) \\
 h(x) &= (\text{Id} \otimes W_O) \cdot (A \otimes \text{Id}) \cdot (\text{Id} \otimes W_V) \cdot x \\
 &= \left(\underbrace{A}_{\substack{\text{Multiplicación} \\ \text{a través de} \\ \text{las posiciones}}} \otimes \underbrace{W_O W_V}_{\substack{\text{Multiplicación} \\ \text{de vectores} \\ \text{en cada posición}}} \right) \cdot x
 \end{aligned}$$

Atención múltiple II

Interpretación: La cabeza de atención mueve información de un subespacio del flujo residual de un componente léxico (*token*) a otro subespacio del flujo residual de otro componente léxico (*token*).

- $W_O W_V$ describe de qué subespacio se lee y a qué subespacio se escribe.
- A indica de qué *token* a qué *token* se mueve la información.

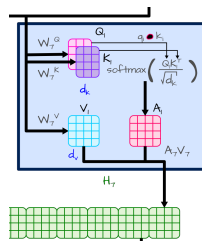


Figura: Cabeza

Independencia de las cabezas

- **[Eficiencia]** Los vectores resultantes de aplicar cada una de las cabezas z^{h_1}, z^{h_2}, \dots son concatenados para obtener la salida de la capa de atención al multiplicarla por la matriz W_O^H :

$$H = W_O^H \begin{bmatrix} z^{h_1} \\ z^{h_2} \\ \dots \end{bmatrix}$$

- **[Interpretación]** Dado que es una operación lineal, es posible reescribir como:

$$H = W_O^H \begin{bmatrix} z^{h_1} \\ z^{h_2} \\ \dots \end{bmatrix} = [W_O^{h_1}, W_O^{h_2}, \dots] \cdot \begin{bmatrix} z^{h_1} \\ z^{h_2} \\ \dots \end{bmatrix} = \sum_i W_O^{h_i} z^{h_i}$$

revelando que la operación original se puede interpretar como la acción independiente de cada cabeza, siendo multiplicada por su propia matriz, para luego sumar las contribuciones de todas al flujo residual. Elhage y col. 2021

Características de las cabezas

- Cada cabeza se puede entender como una operación independiente, el resultado de cada una contribuye a la suma que se agrega al flujo residual.
- Cada cabeza realiza dos operaciones:
 - Circuito consulta-llave QK (*Query-Key*) que calcula el patrón de atención.
 - Circuito salida-valor OV (*Output-Value*) que calcula cómo afecta cada símbolo (*token*) a la salida cuando se le pone atención.
- La composición de las cabezas de atención incrementan considerablemente la expresividad de los transformadores.
- Hay 3 formas en las que las cabezas se pueden componer y corresponden a las llaves, consultas y valores.
- Las composiciones **llave-consulta** son muy distintas a las composiciones de **valores**.

Modelos de lenguaje de gran tamaño

- 1 Antecedentes
- 2 Arquitectura
- 3 Flujo residual
- 4 Cabezas de atención
- 5 Modelos de lenguaje de gran tamaño

GPT I

- GPT viene del inglés *Generative Pre-trained Transformer*.

GPT-2^[5]

“Cuando un modelo de lenguaje grande es entrenado sobre un conjunto de datos grande y diverso puede tener un buen desempeño para varios dominios y otros conjuntos de datos”.
(Radford, Wu y col. 2018)

- Su arquitectura y filosofía de entrenamiento se expone en los artículos (Radford, Wu y col. 2018; Radford, Narasimhan y col. 2018).
- Usa como base el modelo de Vaswani, moviendo y agregando algunas capas de normalización por capas e incrementa el número de parámetros.

GPT II

- Para entrenarla se utilizó aprendizaje no supervisado para extraer un modelo del lenguaje inglés. Por *modelo del lenguaje* entenderemos estimaciones de probabilidades del estilo:

$$p(s_{n-k}, \dots, s_n | s_1, \dots, s_{n-k-1})$$

donde (s_1, s_2, \dots, s_n) es una secuencia de símbolos.

- Incidentalmente incluyó un poco de francés incrustado en los conjuntos de entrenamiento en frases como:
"I'm not the cleverest man in the world, but like they say in French: Je ne suis pas un imbecile [I'm not a fool]."

GPT III

- El objetivo de los autores fue utilizar la red sin más entrenamiento para resolver tareas más generales de la forma:

$$p(\text{salida}|\text{entrada}, \text{tarea})$$

considerando que la especificación de la tarea misma esté contenida en el texto de entrada con frases como “*answer the question*”.

[5] <https://towardsdatascience.com/examining-the-transformer-architecture-part-1-the-openai-gpt-2-controversy-feced54363bb5>

Un modelo de lenguaje multitarea I

Con adaptaciones muy básicas, usaron el modelo para resolver las tareas siguientes:

Modelado del lenguaje Tarea inicial con la que se entrenó el modelo.

Libro para niños Es una prueba que consiste en predecir cuál de entre 10 opciones es la palabra correcta que se omitió en un texto.

LAMBADA Predecir la última palabra en enunciados donde un ser humano requeriría al menos 50 componentes léxicos de contexto para realizar una predicción correcta.

Reto del esquema de Winograd Prueba la capacidad de un sistema para razonar por sentido común al medir su habilidad para resolver ambigüedades en un texto.

Un modelo de lenguaje multitarea II

- Comprensión de lectura** Debe responder preguntas estilo “¿por qué?” a partir de la historia leída.
- Traducción** Usaron el poco francés en el texto para detectar si podía traducir algo.
- Responder preguntas** “Who played john connor in the original terminator? → Arnold Schwarzenegger” Ok, fue Edward Furlong, esta la tuvo mal, pero respondió muchas otras bien.
- Resúmenes** Dado un texto extraer un número limitado de frases que transmitan la mayor cantidad de información al respecto.

Aprendizajes

- En 7 de 8 pruebas que se reportan en el primer artículo, Chat-GPT ya competía con el estado del arte en su tiempo; fallaba al tratar de extraer resúmenes.
- En experimentos posteriores se procedería a realizar un postentrenamiento para realizar ajustes finos según la tarea, lo cual incrementaría el desempeño del sistema.
- Los experimentos de Radford, Wu y col. 2018 les llevaron a concluir que el pre-entrenamiento de grandes modelos de lenguaje mediante aprendizaje no supervisado ha sido altamente exitoso porque, al aprender a resolver una tarea, el sistema ya está aprendiendo implícitamente a resolver otras también sin necesidad de supervisión o adaptación.

Parámetros para GPT-2

- Tiene alrededor de 1.5 mil millones de parámetros (10X más que el modelo original).
- Fue entrenado a partir del texto de 8 millones de sitios de internet (~40GB de texto).
- Tiene 48 capas.
- Lo pueden descargar de Github: <https://github.com/openai/gpt-2>

GPT-3

GPT-3^[6]

- 175 mil millones de parámetros.
Hey, hay unos 100.000 millones de neuronas en un cerebro... todavía le falta.
- Tiene 96 capas que alternan bloques de atención con perceptrones multicapa (MLP).
- Pueden descargar la capa de incrustación <https://github.com/openai/GPT-3-Encoder>.

[6] <https://transformer-circuits.pub/2021/framework/index.html>, <https://arxiv.org/pdf/2005.14165.pdf> 

GPT-4

- Plataforma multimodal, recibe como entradas texto e imágenes, responde con texto.
- 6 meses de entrenamiento incluyendo pruebas adversariales.
- Ajuste fino utilizando aprendizaje por refuerzo con retroalimentación humana.
- 82 % menos probable que responda a solicitudes de contenido no permitido
- 40 % más probable que responda con hechos que GPT-3.5 en las evaluaciones internas de OpenAI.
- Entrenado en las supercomputadoras para IA Azure de Microsoft.
- **Limitaciones:**
 - Sesgos sociales
 - Alucinaciones
 - Sugerencias adversariales (*adversarial prompt*)
- Se encuentra disponible en ChatGPT Plus y como API para que los desarrolladores programen aplicaciones y servicios.

<https://openai.com/product/gpt-4>

Cerrado por ahora

“Given both the competitive landscape and the safety implications of large-scale models like GPT-4, this report contains no further details about the architecture (including model size), hardware, training compute, dataset construction, training method, or similar.”

OpenAI (2023)

<https://arxiv.org/pdf/2303.08774.pdf>

La función apenas comienza I

- A partir de la llegada de los transformadores están apareciendo nuevas variantes que buscan mejorar aún más el rendimiento de este componente:

Difusores Diffuser: Efficient Transformers with Multi-hop Attention Diffusion for Long Sequences por Aosong Feng, Irene Li, Yuang Jiang, Rex Ying, <https://arxiv.org/abs/2210.11794>.

Además de PLN se aplican en DALL-E 2, Google's Imagen y Midjourney.

Ya hay librerías listas para que realicen sus propios experimentos:

<https://towardsdatascience.com/hugging-face-just-released-the-diffusers-library-846f32845e65>.







La función apenas comienza II





Figura: “Estudiantes de Ciencias de la Computación peleando con sus computadoras”.

Transformadores dispersos Introducen factorizaciones a las matrices de atención y otras técnicas para reducir el uso de memoria y permitir operaciones sobre miles de pasos en las secuencias. Utilizan la misma arquitectura para imágenes, audio y texto a partir de bytes. (Child y col. 2019)

Referencias I

-  Alammar, Jay (2020). *The Illustrated Transformer*. URL: <https://jalammar.github.io/illustrated-transformer/>.
-  — (s.f.). *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*. URL: <http://jalammar.github.io/illustrated-bert/>.
-  Child, Rewon y col. (2019). «Generating Long Sequences with Sparse Transformers». En: URL: <https://arxiv.org/pdf/1904.10509.pdf>.
-  Elhage, Nelson y col. (2021). «A Mathematical Framework for Transformer Circuits». En: *Transformer Circuits Thread*. <https://transformer-circuits.pub/2021/framework/index.html>.
-  Huang, Austin y col. (2022). *The Annotated Transformer*. URL: <http://nlp.seas.harvard.edu/annotated-transformer/>.
-  Radford, Alec, Karthik Narasimhan y col. (2018). «Improving Language Understanding by Generative Pre-Training». En.

Referencias II

-  Radford, Alec, Jeffrey Wu y col. (2018). «Language Models are Unsupervised Multitask Learners». En: URL: https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.
-  Vaswani, Ashish y col. (2017). «Attention is All you Need». En: *Advances in Neural Information Processing Systems*. Ed. por I. Guyon y col. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Creative Commons
Atribución-No Comercial-Compartir Igual

