GPT

Generative Pre-trained Transformer

"Attention Is All You Need"

Universidad de Salamanca y AIR Institute

Pablo Chamoso
Raúl García

27/07/2023

https://bisite.usal.es - https://air-institute.com

Agenda

Parte I:

El futuro de la inteligencia artificial con ChatGPT

17:00 - 18:15

Parte II:

Nuevos modelos de negocio basados en IA generativa

18:15 - 18:45

Parte III:

Casos de éxito

18:45 - 19:30

Contenido

PARTE I

- Introducción
  - Bienvenida y presentación
  - Objetivos del taller
- Historia y bases de GPT
  - ¿Quién lo ha desarrollado y qué es?
  - Arquitectura
  - Evolución
- Funcionamiento y capacidades
  - Proceso de entrenamiento y *fine-tuning*
  - Características
  - Ventajas
  - Inconvenientes
  - Uso vía API
- Llama 2, de Meta (open source)

Contenido

PARTE II

- Estableciendo los mejores *prompts*
  - ¿Qué es un *prompt*?
  - Estrategias para formular preguntas
  - Aplicaciones prácticas
  - Ejemplos
- Problemas potenciales y seguridad
  - Confidencialidad
  - Límites del modelo
  - Respuestas erróneas o sesgadas.

Contenido

PARTE II

- Usos potenciales y modelos de negocio
  - Nuevas herramienta (chat GPT plus)
  - Modelos de negocio con IA generativa
  - Casos de éxito
- Preguntas y respuestas

Presentación

Presentación

- Pablo Chamoso
  chamoso@usal.es
- Raúl García
  rgarcia@air-institute.com

Pablo Chamoso

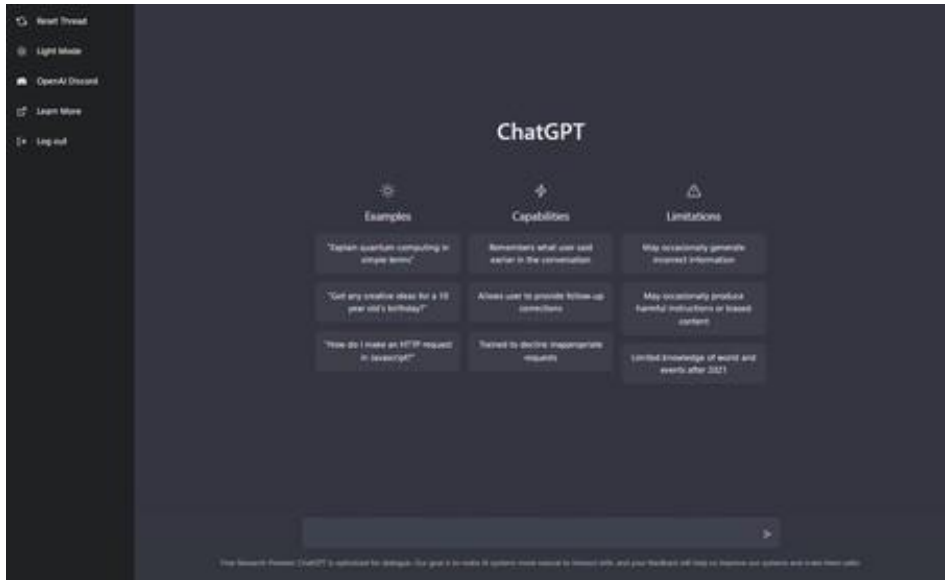
Raúl García Serrada

Objetivos

- Conocer el funcionamiento interno de ChatGPT
- Aprender a integrar ChatGPT en aplicaciones (vía API)
- Aprender a usar ChatGPT de forma eficiente
- Conocer nuevas formas de uso de ChatGPT en tareas diarias

ChatGPT

- ChatGPT es un chatbot basado en inteligencia artificial desarrollado en 2022 por OpenAI.
- El chatbot es un gran modelo de lenguaje, ajustado con técnicas de aprendizaje tanto supervisadas como de refuerzo. La versión básica se basa en el modelo GPT-3.5, mientras que la variante de pago emplea GPT-4.



https://chat.openai.com/

GPT: Generative Pre-trained Transformer

OpenAI

- GPT-4 (2023) es nombre del más reciente modelo de lenguaje preentrenado de OpenAI
- En esencia es una red neuronal profunda para generar textos coherentes y relevantes a partir de una entrada proporcionada por el usuario (prompt)
- Versiones anteriores del modelo:
  - GPT (2018)
  - GPT-2 (2019)
  - GPT-3 (2020)
  - GPT-3.5 (2022)
- Se popularizó por su herramienta generadora de imágenes DALL·E.

Bases

IA generativa

+

Transformer

Inteligencia artificial generativa

La inteligencia artificial generativa se refiere a sistemas de IA que pueden generar nuevos contenidos o datos que no estaban en su conjunto de entrenamiento original.

Estos sistemas no se limitan simplemente a clasificar o reconocer patrones en los datos existentes; en cambio, pueden crear algo nuevo basándose en lo que han aprendido.

Ejemplos y aplicaciones de IA generativa:

- **Modelos de lenguaje generativo (por ejemplo, GPT)**
- Redes Generativas Adversarias (GANs) (para imágenes principalmente)
- Música y sonido
- Diseño de fármacos
- Videojuegos

La ventaja de la inteligencia artificial generativa es que puede producir **resultados novedosos y valiosos** en una amplia variedad de campos, basándose en patrones y estructuras aprendidas de los datos con los que fue entrenada.

Sin embargo, también hay **desafíos éticos y técnicos** asociados con su uso, especialmente en términos de la originalidad del contenido generado y las posibles implicaciones de generar información falsa o engañosa.

Transformer: ¿Quién lo ha desarrollado?

Artículo original

- No solo GPT
  - También BERT o Llama 2
- Conferencia en 2017
- Investigadores de la **Universidad de Toronto** y de **Google**
- "*Attention is all you need*"
- Originalmente diseñado para traducción
- 3 innovaciones principales

- Pero antes…
  - Definición
  - Precedentes

Attention Is All You Need

Lo único que necesitas es atención

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

1 Introduction

Recurrent neural networks, long short-term memory [12] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [29, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [31, 21, 13].

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Transformer : ¿Qué es?

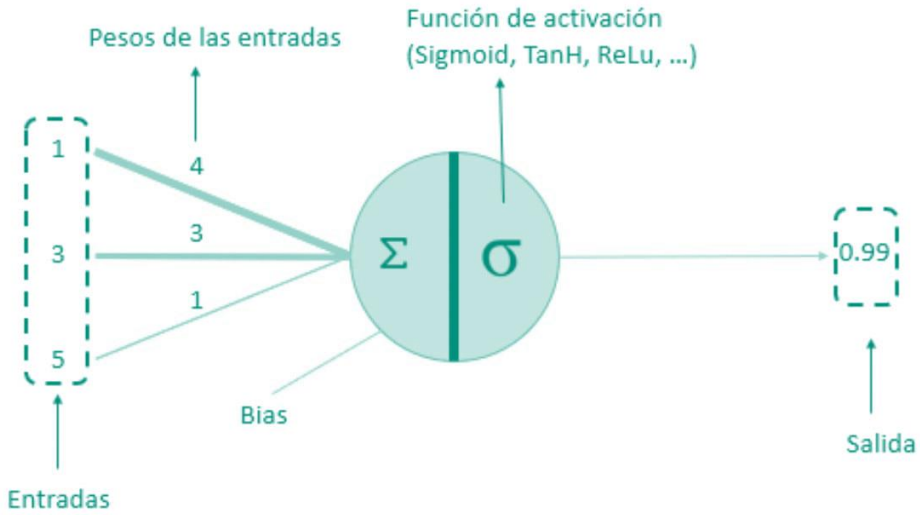
- "Una nu
  arquitec
  neurona
- Basada
  mecanis
  atenció
  prescinc
  recurrer
  convolu

Precedentes

- Neurona
- Redes neuronales (artificiales)
- Redes neuronales convolucionales
- Redes neuronales recurrentes
- Word embeddings

Neurona (I)

Conocimientos básicos preliminares: el concepto de neurona



Pesos ajustables

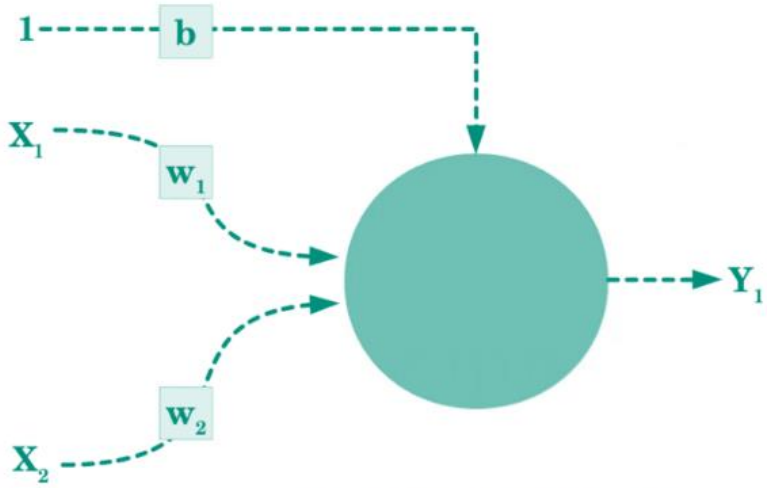
salida $\longrightarrow y = \theta\left(\sum_{i=1}^{d} w_i x_i\right)$

entradas

Modelo matemático de una neurona

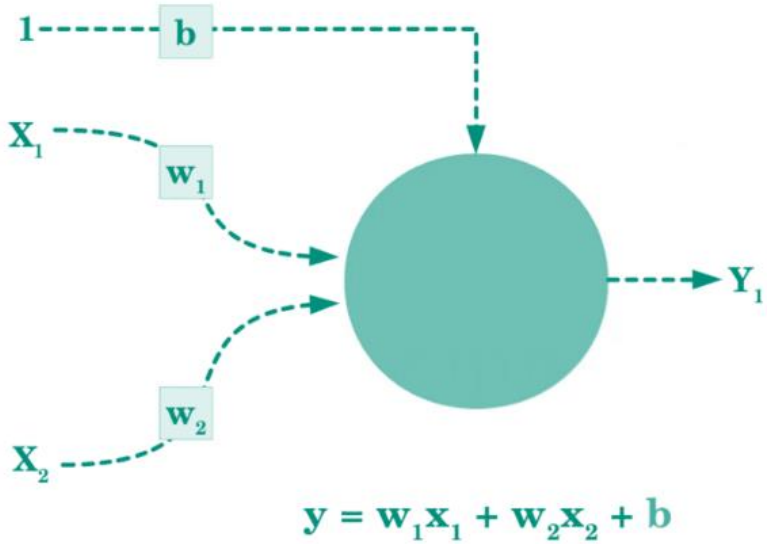
Neurona (II)

Asistir a las clases + hacer el trabajo = aprobar el curso



Neurona (III)

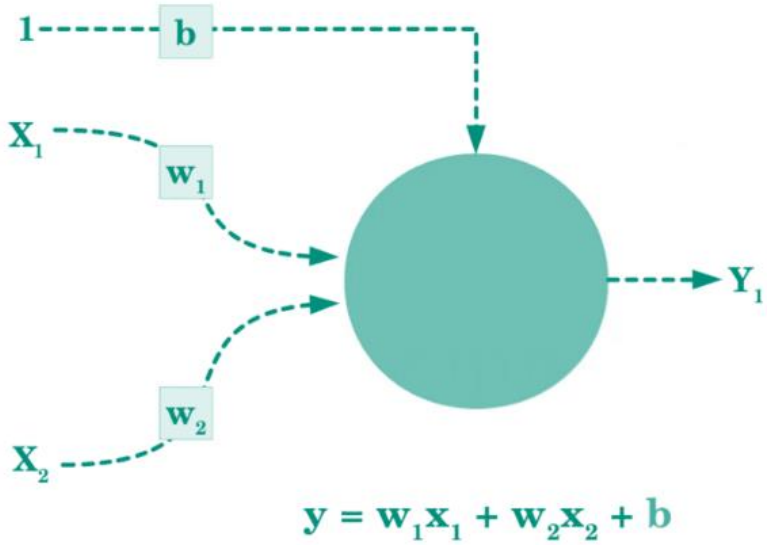
Asistir a las clases + hacer el trabajo = aprobar el curso



	1	0
X1		
X2		

Neurona (IV)

Asistir a las clases + hacer el trabajo = aprobar el curso



	1	0
X1		
X2		
y		

Neurona (V)

Asistir a las clases + hacer el trabajo = aprobar el curso

$WX + b <= 0 \rightarrow Y = 0$
$WX + b > 0 \rightarrow Y = 1$

Por simplificar, usamos una función de activación binaria

	1	0
X1		
X2		
y		

Neurona (VI)

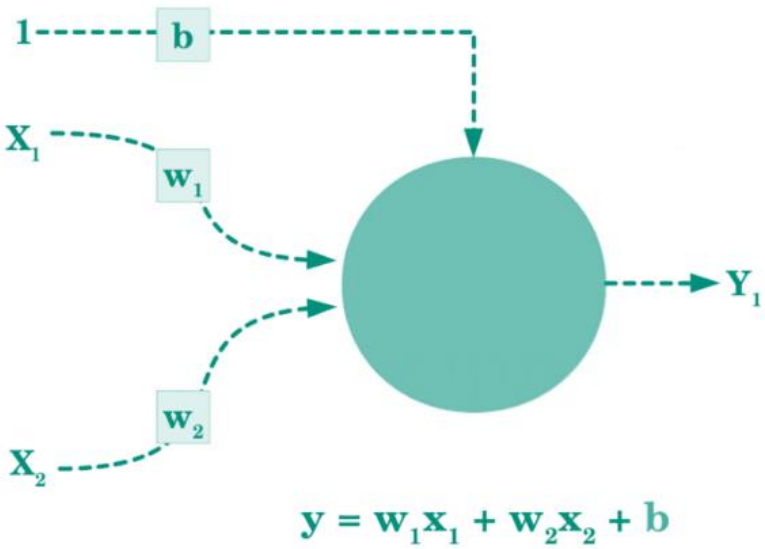
Asistir a las clases + hacer el trabajo = aprobar el curso

X1	X2	Target	Y
			?
			?
			?
			?

Neurona (VII)

Asistir a las clases + hacer el trabajo = aprobar el curso

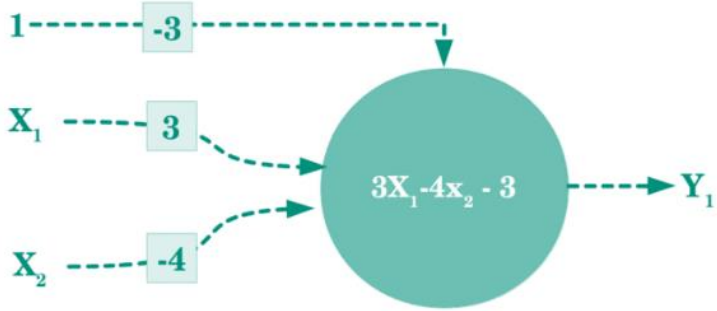
X1	X2	Target	Y
			?
			?
			?
			?



Neurona (VIII)

Asistir a las clases + hacer el trabajo = aprobar el curso

X1	X2	Target	Y
			?
			?
			?
			?

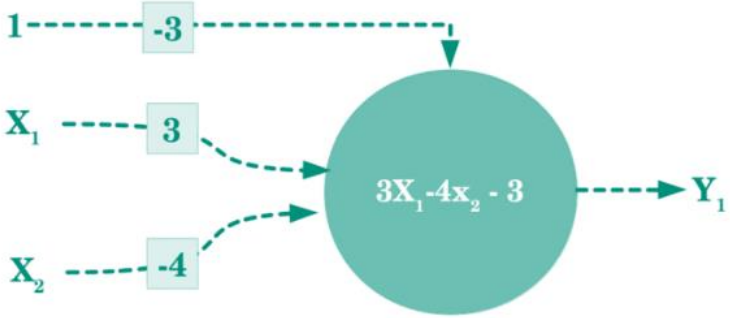


Neurona (IX)

Asistir a las clases + hacer el trabajo = aprobar el curso

X1	X2	Target	Y
			-3
			0
			-7
			-4

3*0-4*0-3=-3



← Esto debería ser > 0

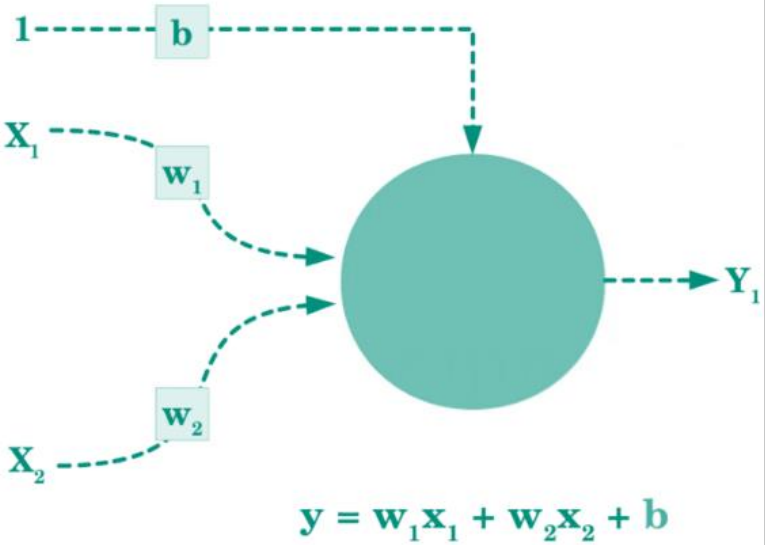
Neurona (X)

Asistir a las clases + hacer el trabajo = aprobar el curso

X1	X2	Target	Y
			?
			?
			?
			?

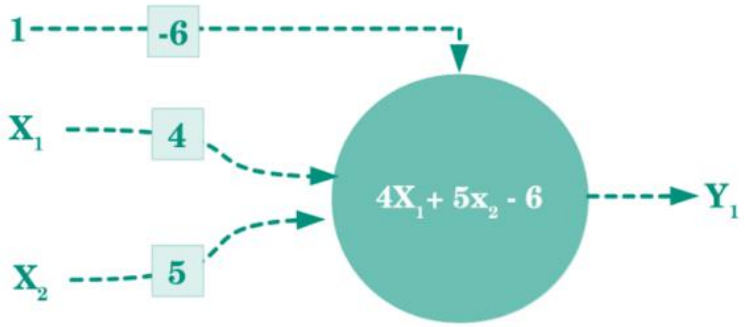
No se cumple, por tanto, otra época

¡¡Fuerza bruta hasta buscar unos pesos y un bias que nos dé lo que buscamos!!



Neurona (XI)

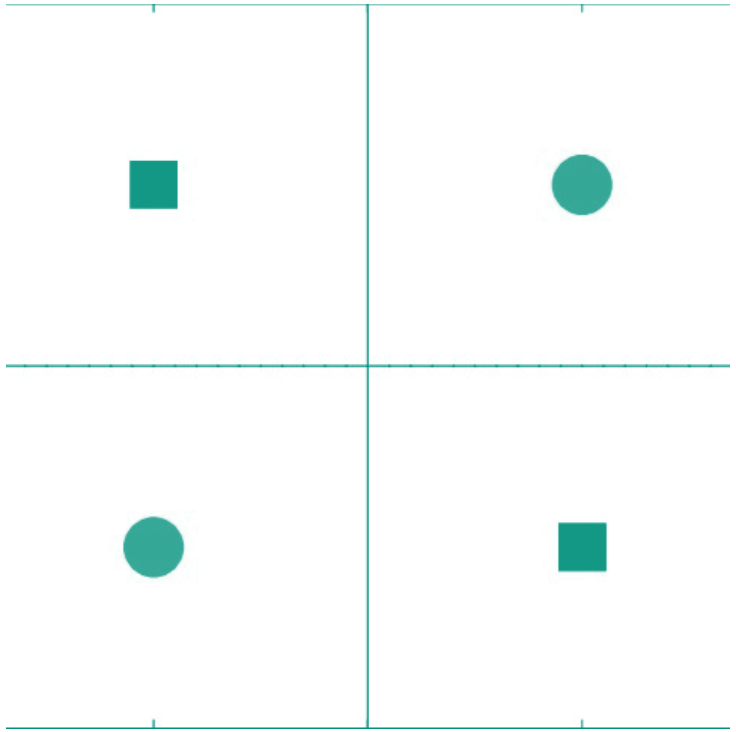
Asistir a las clases + hacer el trabajo = aprobar el curso

X1	X2	Target	Y	
			-6	4*0-5*0-6=-6
			-2	
			-1	
			3	¡Se cumple todo!

Neurona (XII)

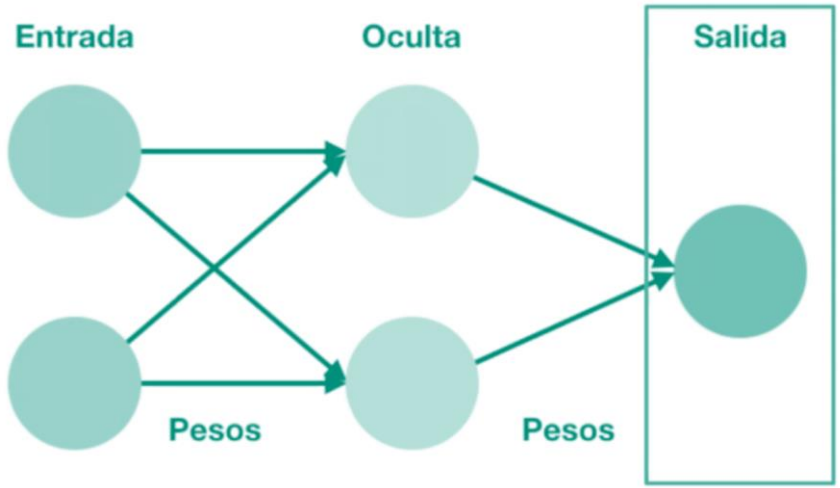
Problema: una sola neurona no puede representar toda la información

Ejemplo, problemas de linealidad como XOR:
¿Cómo agrupas esta imagen (círculos con círculos y cuadrados con cuadrados) con un solo corte lineal? Si es lineal, necesitas 2:



Red neuronal (I)

¡Necesitamos agrupaciones de neuronas!



Podemos incluso tener múltiples neuronas en la capa de salida:
multitarget output

Red neuronal (II)

Modelos útiles para analizar tipos de datos complicados como:

- Imágenes
- Vídeo
- Audio
- Texto

Hay múltiples tipos en función de los datos de entrada.

GPT-4 es un gran modelo multimodal que puede aceptar entradas de imagen y texto y emitir salidas de texto. Sin embargo, **GPT-4 no puede generar imágenes**. Es estrictamente un modelo lingüístico multimodal que puede recibir distintas formas de entrada de datos, pero que **solo puede devolver respuestas en lenguaje natural**.

CNN (I)

Red neuronal convolucional

- Adecuadas para imágenes
- Su diseño imita vagamente cómo el cerebro humano las procesa
- Desde 2012 muy utilizadas, por ejemplo, para etiquetar objetos en fotografías

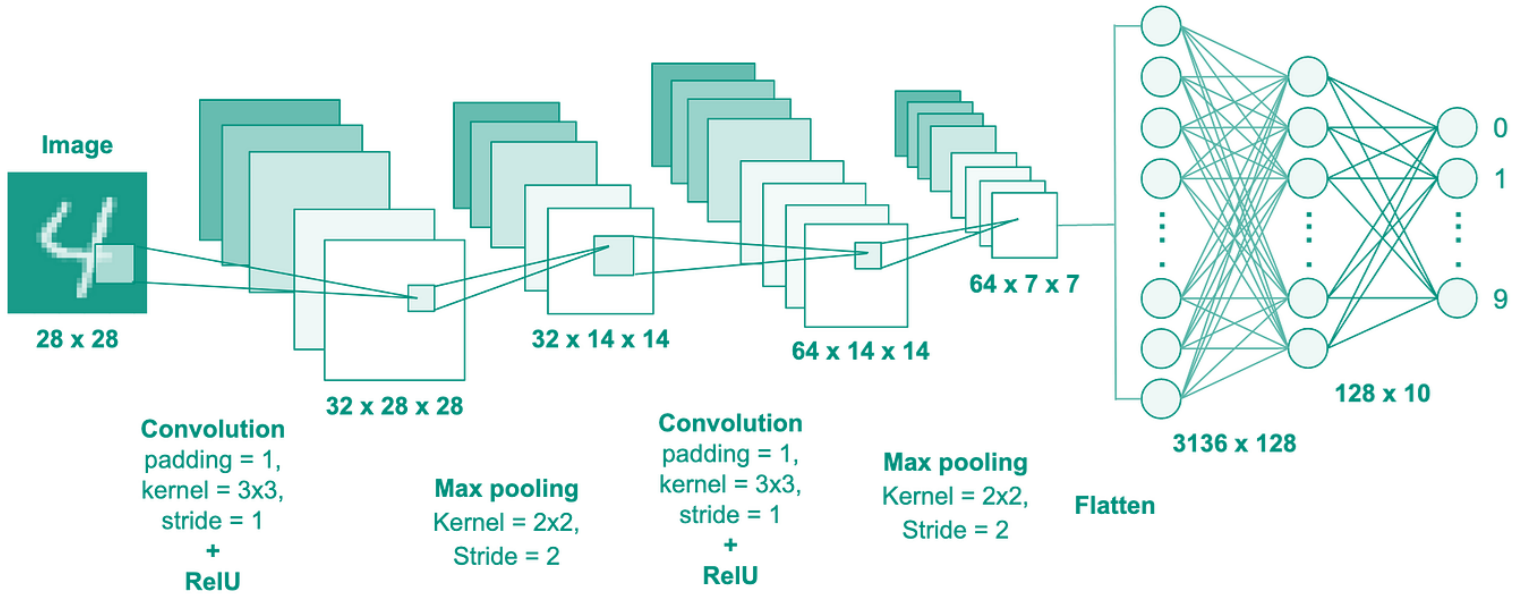
Estas capas se organizan en tres componentes principales: capas convolucionales, capas de activación y capas de agrupación.

- **Capas convolucionales**: Estas capas son responsables de extraer características importantes de las imágenes mediante filtros o "*kernels*". Los filtros se deslizan sobre la imagen, realizando operaciones matemáticas para detectar bordes, formas o patrones específicos.
- **Capas de activación**: Después de la convolución, se aplican funciones de activación (como ReLU) para agregar no linealidad y aumentar la capacidad de la red para aprender relaciones complejas.
- **Capas de agrupación**: Estas capas reducen el tamaño de la representación de la imagen, disminuyendo la cantidad de parámetros y haciendo que la red sea más eficiente en el procesamiento.

A medida que la información pasa por estas capas, la CNN va aprendiendo a reconocer características más abstractas y complejas, permitiendo la identificación de objetos, personas o cualquier cosa que se le haya entrenado para reconocer.

CNN (II)

Red neuronal convolucional



CNN (III)

Ejemplo en código[1,2]

```
# Importamos las librerías necesarias
import tensorflow as tf
from tensorflow.keras import layers, models, datasets

# Cargamos el conjunto de datos MNIST
(train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data()

# Normalizamos los datos para que los valores de los píxeles estén entre 0 y 1
train_images = train_images.reshape((60000, 28, 28, 1)).astype("float32") / 255
test_images = test_images.reshape((10000, 28, 28, 1)).astype("float32") / 255

# Creamos el modelo de la CNN
model = models.Sequential([
  layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
  layers.MaxPooling2D((2, 2)),
  layers.Conv2D(64, (3, 3), activation='relu'),
  layers.MaxPooling2D((2, 2)),
  layers.Conv2D(64, (3, 3), activation='relu'),
  layers.Flatten(),
  layers.Dense(64, activation='relu'),
  layers.Dense(10, activation='softmax')
])

# Compilamos el modelo
model.compile(optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])

# Entrenamos el modelo con los datos de entrenamiento
model.fit(train_images, train_labels, epochs=5, batch_size=64)

# Evaluamos el rendimiento del modelo en el conjunto de datos de prueba
test_loss, test_acc = model.evaluate(test_images, test_labels)

print(f'\nAccuracy on test data: {test_acc:.2f}')
```

TensorFlow

conv2d_input	input:	[(None, 28, 28, 1)]
InputLayer	output:	[(None, 28, 28, 1)]

conv2d	input:	(None, 28, 28, 1)
Conv2D	output:	(None, 26, 26, 32)

max_pooling2d	input:	(None, 26, 26, 32)
MaxPooling2D	output:	(None, 13, 13, 32)

conv2d_1	input:	(None, 13, 13, 32)
Conv2D	output:	(None, 11, 11, 64)

max_pooling2d_1	input:	(None, 11, 11, 64)
MaxPooling2D	output:	(None, 5, 5, 64)

conv2d_2	input:	(None, 5, 5, 64)
Conv2D	output:	(None, 3, 3, 64)

flatten	input:	(None, 3, 3, 64)
Flatten	output:	(None, 576)

dense	input:	(None, 576)
Dense	output:	(None, 64)

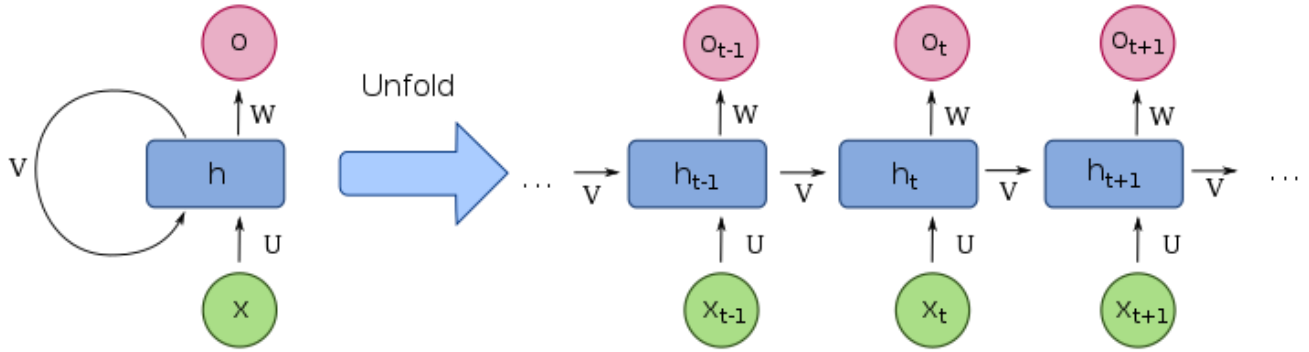
dense_1	input:	(None, 64)
Dense	output:	(None, 10)

1) este código no se corresponde con la ilustración de la diapositiva anterior
2) este código lo ha generado ChatGPT en pocos segundos

RNN (I)

Red neuronal recurrente

- Adecuadas para texto.
- Funcionan de forma **secuencial**. Motivo: el orden en el que aparece cada palabra, es algo muy a tener en cuenta en los diferentes idiomas:
  - Me baño en el río ; Me río en el baño.
- Además, es la forma en la que interpreta el texto la mente humana, de palabra en palabra secuencialmente.
- Las RNN prestan atención a cada palabra de forma individual y secuencialmente.

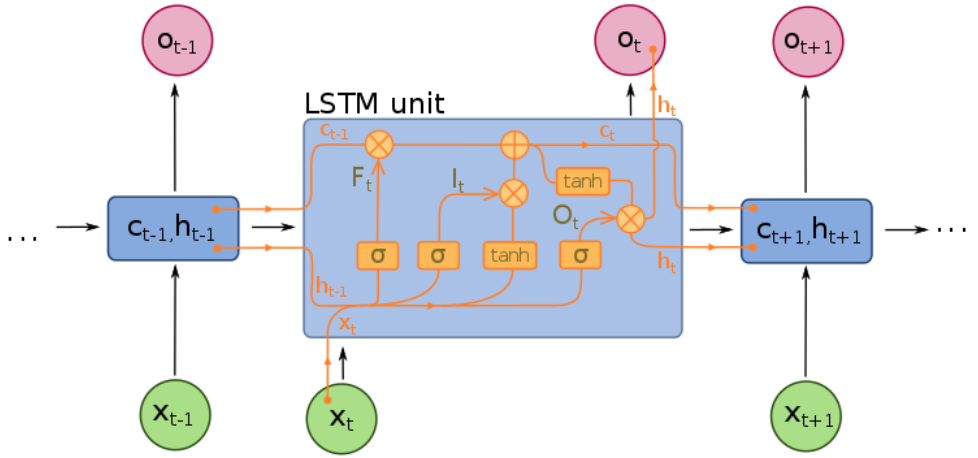


RNN (II)

PROBLEMA 1:

- En **párrafos muy largos**, cuando se analizaba el final del párrafo ya se había perdido el contexto de lo que se decía al principio.

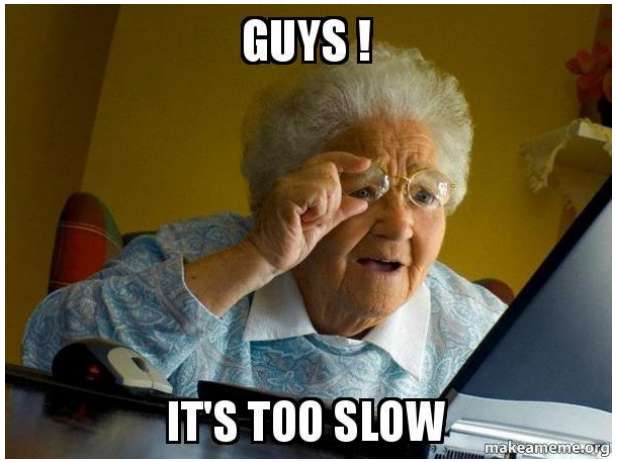
Solucionado por las redes LSTM (Long Short-Term Memory):



RNN (III)

PROBLEMA (aún peor) 2:

- Por el hecho de procesar las palabras **de forma secuencial** y por tanto no haber paralelización, el entrenamiento es lento y no se puede acelerar por muchas GPUs que se utilicen.
- **Evolucionarlo no es sencillo** porque no se puede entrenar con muchísimos datos al ser lento.



Word embeddings

Incrustaciones de palabras (técnica de NLP)

- Permite pasar de texto a matrices de números como entrada
- "Vectorizar el texto"
- Muchas estrategias
  La más sencilla de entender: codificaciones one hot

	cat	mat	on	sat	the
the =>	0	0	0	0	1
cat =>	1	0	0	0	0
sat =>	0	0	0	1	0
...	...				

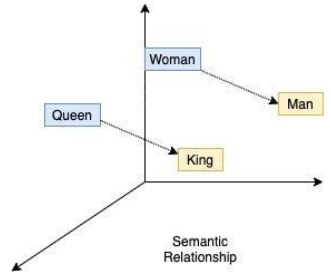
- Artículo más relevante (2013):

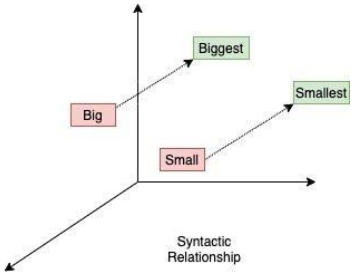
Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Word2vec es una técnica para el procesamiento de lenguaje natural publicada en 2013. El algoritmo Word2vec utiliza un modelo de red neuronal para aprender asociaciones de palabras a partir de un gran corpus de texto.

Más sobre word2vec:

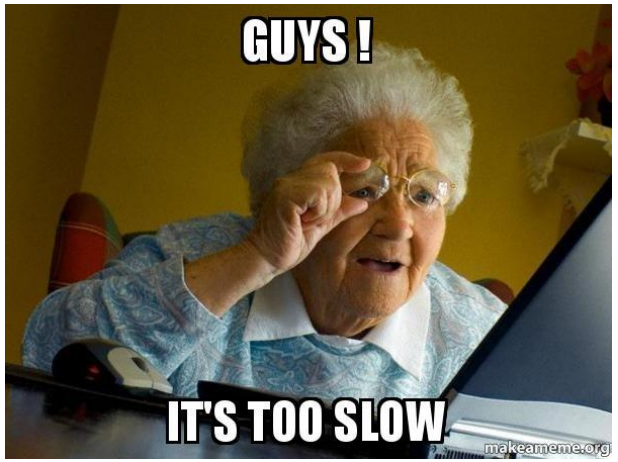
- https://www.tensorflow.org/tutorials/text/word2vec?hl=es-419
- https://towardsdatascience.com/word2vec-research-paper-explained-205cb7eecc30





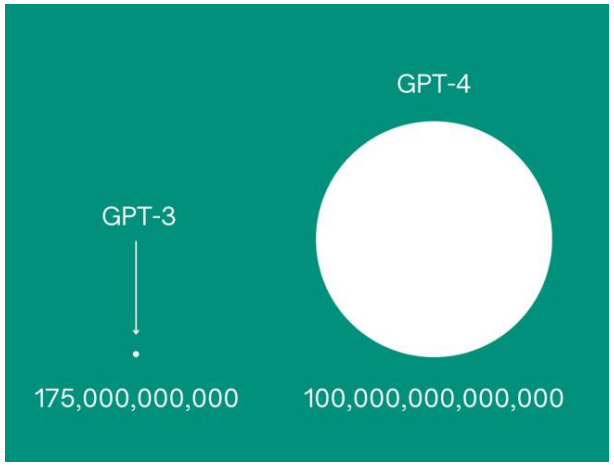
RNN vs Transformer

RNN: cada palabra se procesa de una en una
Transformer: podemos analizar todas las palabras de la frase al mismo tiempo



Transformer (I)

- La principal diferencia con las RNN es que se puede paralelizar
- Es decir, con el hardware adecuado, se pueden entrenar modelos mucho más grandes, que supone que los resultados pueden llegar a ser mejores.
- ¿Cómo de grandes? GPT-3 se entrenó con 175.000 millones de parámetros y 45TB de datos (de texto), incluyendo casi toda la información que estaba pública en Internet (Surface web)



- Es decir, los modelos transformer pueden ser escalables gracias a la paralelización, permitiendo trabajar con enormes cantidades de datos.

Transformer (II)

¿Cómo funcionan?

- Hemos hablado de 3 innovaciones principales que se presentan en el artículo original y que hacen que los resultados sean tan buenos:

1. Positional encoding (codificación posicional)
2. Attention (atención)
3. Self-attention (autoatención)

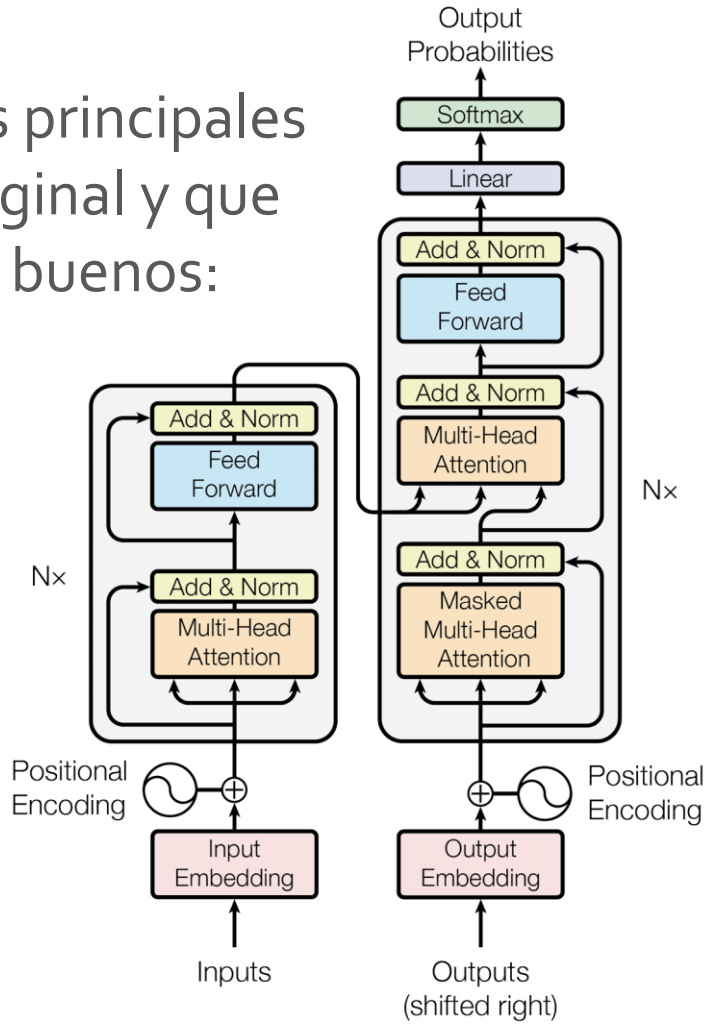
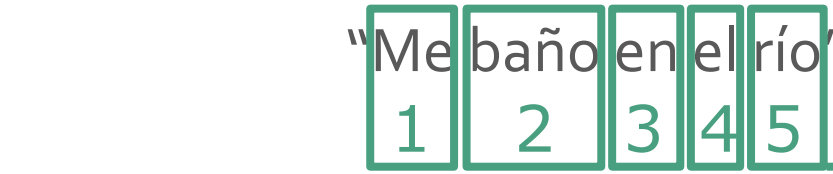


Figure 1: The Transformer - model architecture.

Transformer (III)

1. Positional encoding (codificación po

A diferencia de las RNN, en lugar de analizar pa
dentro de una frase, se basa en asignar una posi
cada una de las palabras y esa posición va a forr
entrada de la red neuronal.



Se binarizan para no tener números con mucho peso (solo 0 ó 1 y no el 345) y evita problemas de repeticiones si se normalizara a valores entre 0 y 1.
… y permite usar funciones sinusoidales

Con esto se consigue que la información sobre el orden de la palabra pase a estar en el propio dato y no en la estructura de la red, por lo que según se va entrenando la red con muchísimos datos textuales, la red aprende cómo interpretar esos positional encodings, aprendiendo la importancia del orden de las palabras a partir de los propios datos.

Esto contribuye a que los modelos transformer sean mucho más fáciles de entrenar que las RNN.

Transformer (IV)

2. Attention (atención)

Transformer original: orientado a traducción
El orden de las palabras importa.

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Pero la "atención" ya existía, ejemplo del paper
Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473

"The agreement on **the European Economic Area** was signed in August 1992"

Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau Jacobs University Bremen, Germany
KyungHyun Cho **Yoshua Bengio*** Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1 INTRODUCTION

Neural machine translation is a newly emerging approach to machine translation, recently proposed by Kalchbrenner and Blunsom (2013), Sutskever *et al.* (2014) and Cho *et al.* (2014b). Unlike the traditional phrase-based translation system (see, e.g., Koehn *et al.*, 2003) which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

Most of the proposed neural machine translation models belong to a family of *encoder–decoders* (Sutskever *et al.*, 2014; Cho *et al.*, 2014a), with an encoder and a decoder for each language, or involve a language-specific encoder applied to each sentence whose outputs are then compared (Hermann and Blunsom, 2014). An encoder neural network reads and encodes a source sentence into a fixed-length vector. A decoder then outputs a translation from the encoded vector. The whole encoder–decoder system, which consists of the encoder and the decoder for a language pair, is jointly trained to maximize the probability of a correct translation given a source sentence.

A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. Cho *et al.* (2014b) showed that indeed the performance of a basic encoder–decoder deteriorates rapidly as the length of an input sentence increases.

In order to address this issue, we introduce an extension to the encoder–decoder model which learns to align and translate jointly. Each time the proposed model generates a word in a translation, it (soft-)searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

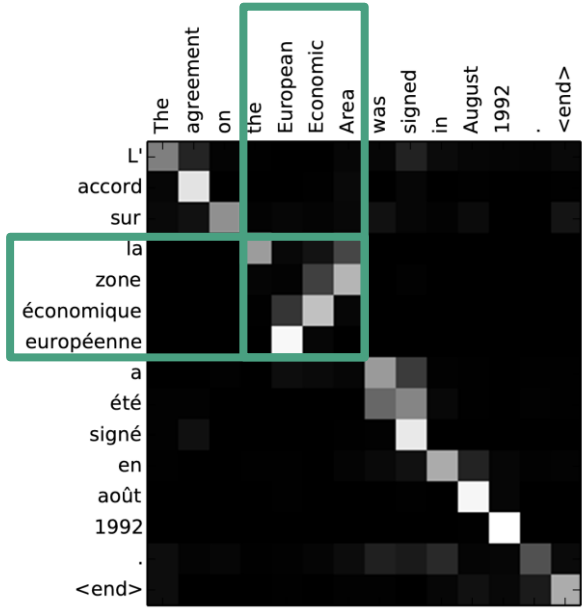
*CIFAR Senior Fellow

Si traducimos palabra por palabra:
"the European Economic Area"
tendremos: "el europeo económico espacio"
en lugar de: "el espacio económico europeo"
Para el problema del orden, ya teníamos una solución
Pero en francés, además del orden, cambia el género:
"la zone économique européene"

Transformer (V)

2. Attention (atención)

¿A qué palabras del texto de entrada pone **atención** para predecir cada palabra de la salida?



Matriz de atención

A partir de millones de ejemplos, el modelo es capaz de aprender todos estos aspectos sobre el género, plurales, etc.

Transformer (VI)

3. Self-attention (autoatención o atención auto-regresiva)

Principal novedad, no vista antes, **una vuelta de tuerca a la atención**.

La atención tradicional permite encontrar la relación entre palabras, muy útil en traducción, pero lo que **queremos es conocer todos las las características de un lenguaje** para poder realizar cualquier tarea relacionada con ese lenguaje.

Las redes neuronales, a medida que analizan cantidades inmensas de datos, en este caso relacionados con texto, comienzan a construir una representación interna o comprensión del lenguaje de forma automática.

Por ejemplo, aprenden que

Desarrollador de software = programador = informático

¡son sinónimos!

Y así con todas las reglas de la gramática, género y tiempos verbales.

Transformer (VII)

3. Self-attention (autoatención)

Cuanto mejor aprende a base de ver ejemplos, más tareas relacionadas con el lenguaje es capaz de realizar.

"La barca está en el río"
"Cuando juega el Barça me río"

Se puede desambiguar gracias a que conoce el contexto por las palabras que rodean la palabra.
barca -> río (agua) ; juega -> río (diversión)

La autoatención ayuda a entender una palabra en el contexto de las palabras que la rodean. Cuando procesan río en la primera frase, se atiende a la palabra barca. En la segunda se atiende a la palabra juega.

Entrenamiento (I)

- Preentrenamiento

Los modelos Transformer, como BERT o GPT, comienzan su entrenamiento con un proceso de preentrenamiento en grandes conjuntos de datos, como todo el contenido de Wikipedia o texto extraído de la web.

Durante este preentrenamiento, el modelo aprende a predecir palabras palabras siguientes en secuencias (en el caso de GPT), permitiéndole adquirir una comprensión general del lenguaje y capturar patrones, relaciones semánticas y sintácticas entre palabras.

Entrenamiento (II)

- Transferencia de conocimiento (transfer learning)

Una vez preentrenado, el modelo Transformer ha adquirido una vasta cantidad de conocimiento del lenguaje. Esta base de conocimiento se puede transferir a tareas más específicas mediante el proceso de "*fine tuning*" (ajuste fino).

En otras palabras, en lugar de entrenar un modelo desde cero para cada nueva tarea, podemos aprovechar la información ya aprendida durante el preentrenamiento.

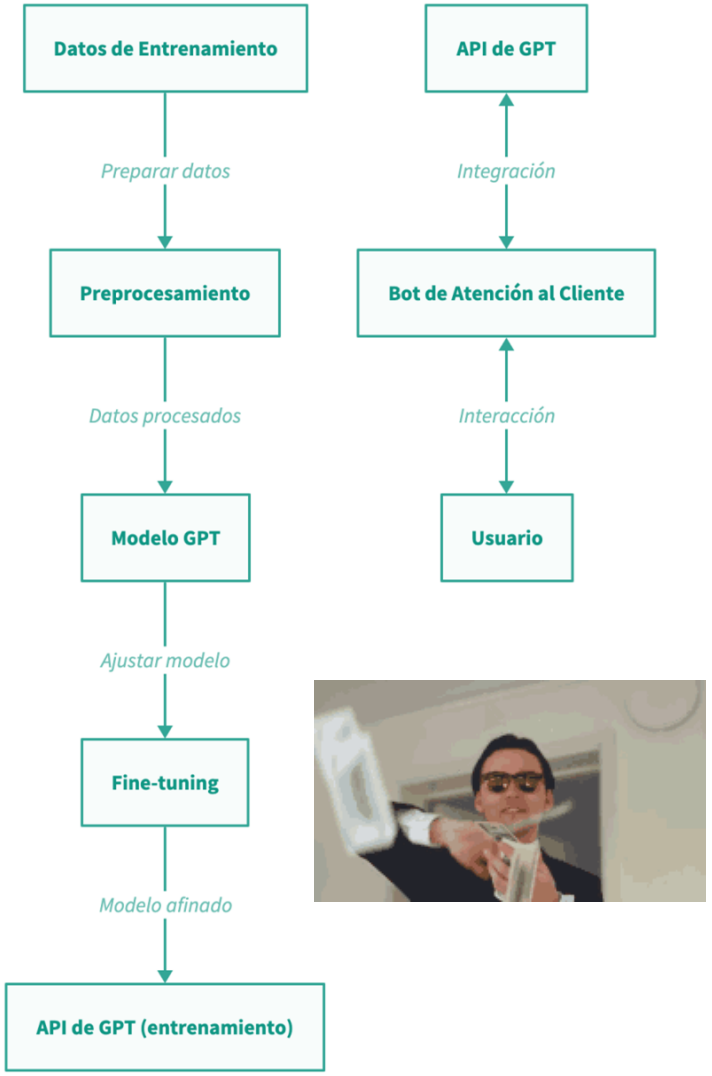
Entrenamiento (III)

- *Fine tuning* (ajuste Fino) en datos específicos de la tarea

Después del preentrenamiento, el modelo se ajusta en un conjunto de datos más pequeño y específico para una tarea particular (por ejemplo, clasificación de sentimientos, reconocimiento de entidades nombradas, etc.).

Durante este proceso, los pesos del modelo se ajustan ligeramente para adaptarse mejor a la tarea específica, mientras que aún se benefician del conocimiento general del lenguaje adquirido durante el preentrenamiento.

Entrenamiento (IV)



Características (I)

El uso de modelos preentrenados y luego afinados permite alcanzar el estado del arte en muchas tareas de NLP con menos datos de entrenamiento específicos de la tarea.

El ajuste fino es generalmente más rápido que el entrenamiento desde cero, ya que el modelo ya ha aprendido una gran cantidad de estructuras lingüísticas útiles durante el preentrenamiento.

Aunque el *fine tuning* es eficaz, es posible que un modelo ajustado finamente sea sensible a cambios en la distribución de los datos o que sobreajuste en conjuntos de datos pequeños.

Es esencial tener cuidado y evaluar el rendimiento del modelo en datos no vistos y considerar técnicas como la regularización para evitar el sobreajuste.

Características (II)

- Generación de Texto
  - Una vez entrenado, GPT puede generar texto de manera autónoma. Al proporcionarle una entrada o "prompt", GPT generará una continuación basada en lo que ha aprendido.
  - La longitud y el estilo de la respuesta pueden variar según cómo se configure el modelo y los hiperparámetros de generación.
- Consideraciones Prácticas
  - Aunque GPT es impresionante en su capacidad de generar texto, no garantiza la precisión factual. Puede producir información incorrecta o sesgada basándose en los datos con los que fue entrenado.
  - GPT, especialmente en sus versiones más grandes como GPT-3, requiere una cantidad significativa de recursos computacionales, lo que puede ser una limitación para algunos usuarios o aplicaciones.

Ventajas (I)

- **Mecanismo de atención**: La arquitectura Transformer utiliza la atención auto-regresiva para ponderar diferentes partes de la entrada de manera diferente, lo que permite al modelo centrarse en las partes relevantes de la entrada para una tarea específica.
- **Memoria a largo plazo**: Gracias al mecanismo de atención, los Transformers pueden capturar relaciones a largo plazo en los datos, lo que es crucial para muchas tareas de NLP.
- **Paralelización**: A diferencia de las arquitecturas recurrentes, los Transformers permiten un procesamiento paralelo de los datos, lo que facilita entrenamientos más rápidos en hardware moderno, especialmente en GPUs.
- **Adaptabilidad**: Los Transformers han demostrado ser efectivos en una amplia variedad de tareas de NLP, desde traducción automática hasta generación de texto y comprensión del lenguaje natural. Además, también se han aplicado con éxito en otras áreas, como visión por computadora.

Ventajas (II)

- **Modelos preentrenados**: Una ventaja significativa que ha emergido del enfoque de los Transformers es la idea del preentrenamiento y ajuste fino. Modelos como BERT, GPT o Llama 2 se entrenan primero en grandes cantidades de texto (preentrenamiento) y luego se afinan en tareas específicas. Esto ha llevado a un rendimiento del estado del arte en una amplia gama de tareas con menos datos de entrenamiento específicos de la tarea.
- **Capacidad**: Los modelos Transformer, especialmente las versiones más recientes, tienen una gran cantidad de parámetros, lo que les permite aprender y generalizar patrones complejos en los datos.
- **Flexibilidad**: Los Transformers pueden ser utilizados tanto para tareas de clasificación como generativas. Por ejemplo, BERT es excelente para tareas de clasificación, mientras que GPT es conocido por su capacidad generativa.
- **Interpretabilidad mejorada**: A través de los mapas de atención, los modelos Transformer pueden proporcionar intuiciones sobre qué partes del texto se consideraron más importantes para decisiones específicas, lo que puede ser útil para el análisis y la interpretación.

Inconvenientes (I)

- **Requisitos de computación**: Los modelos Transformer, en particular las versiones más grandes, requieren una gran cantidad de recursos computacionales tanto para el entrenamiento como para la inferencia. Esto puede hacer que no sean adecuados para aplicaciones en tiempo real o para dispositivos con limitaciones de recursos.
  - El modelo más pequeño, GPT-3 "pequeño", requiere un mínimo de 16 GB de RAM y una GPU con al menos 8 GB de VRAM para funcionar de manera efectiva.
  - El modelo más grande, GPT-3 "175B", requiere 600 GB de RAM y una GPU especializada con 250 GB de VRAM.

Inconvenientes (II)

- **Tamaño del modelo**: Algunas variantes de los modelos Transformer, como GPT-3 o BERT-large, tienen miles de millones de parámetros, lo que puede complicar su despliegue en entornos con restricciones de memoria.
- **Consumo energético**: El entrenamiento de modelos Transformer de gran escala consume una cantidad significativa de energía, lo que plantea preocupaciones tanto en términos de costos como de impacto medioambiental.
- **Sobreajuste**: En conjuntos de datos más pequeños, los modelos Transformer pueden ser propensos al sobreajuste debido a su alta capacidad.
- **Sensibilidad a la entrada**: Aunque los Transformers son robustos en general, pueden ser sensibles a pequeñas alteraciones en la entrada (por ejemplo, a ataques adversarios) y generar respuestas completamente diferentes.

Inconvenientes (III)

- **Interpretabilidad**: A pesar de que los mapas de atención pueden proporcionar alguna intuición sobre cómo funcionan los modelos, los Transformers siguen siendo, en gran medida, "cajas negras". Determinar exactamente cómo llegan a decisiones específicas puede ser complicado.
- **Generalización limitada**: A pesar de su capacidad, los Transformers no siempre generalizan bien fuera de los contextos en los que fueron entrenados. Por ejemplo, pueden generar texto que suena plausible pero que es factualmente incorrecto o incoherente.
- **Biases y prejuicios**: Los modelos Transformer aprenden a partir de los datos con los que son entrenados. Si esos datos contienen biases o prejuicios, el modelo también los adquirirá. Esto ha llevado a preocupaciones sobre la propagación y amplificación de estereotipos y prejuicios.
- **Dependencia del preentrenamiento**: Muchos de los beneficios de los modelos Transformer provienen del preentrenamiento en grandes conjuntos de datos, lo que puede no ser factible para todos los usuarios o aplicaciones.

Uso vía API

https://platform.openai.com/apps

ChatGPT →

Interact with our flagship language models in a conversational interface

DALL·E →

Create realistic images and art from a description in natural language

API →

Integrate OpenAI models into your application or business

Uso vía API: **curl**

https://platform.openai.com/docs/api-reference/introduction

```
curl https://api.openai.com/v1/models \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -H "OpenAI-Organization: org-BbVyalB1LIIiIGWWnxtDm6si”

curl https://api.openai.com/v1/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
     "model": "gpt-3.5-turbo",
     "messages": [{"role": "user", "content": ”AIR"}],
     "temperature": 0.7
   }'
```

Uso vía API: **Node.js library**

https://platform.openai.com/docs/api-reference/introduction

npm install openai

https://platform.openai.com/account/org-settings

```
import { Configuration, OpenAIApi } from "openai";
const configuration = new Configuration({
    organization: "org-BbVyalB1LIIiIGWWnxtDm6si",
    apiKey: process.env.OPENAI_API_KEY,
});
const openai = new OpenAIApi(configuration);
const response = await openai.listEngines();
```

https://platform.openai.com/account/api-keys

Uso vía API: **Python bindings**

https://platform.openai.com/docs/api-reference/introduction

python™

```
pip install openai
```

```
import os
import openai
openai.organization = "org-BbVyalB1LIIiIGWWnxtDm6si"
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Model.list()
```

NAME	KEY	CREATED	LAST USED
K	sk-...jqYN	30 may 2023	Never

+ Create new secret key

Uso vía API: **¡No solo Chat!**

https://platform.openai.com/docs/api-reference/introduction

También con imágenes

```
curl https://api.openai.com/v1/images/generations \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
    "prompt": ”Genera la foto de 3 científicos",
    "n": 3,
    "size": ”1080x1350"
  }'
```

Uso vía API: **Aplicaciones base (quickstart) (I)**

Versión para node y para python (Flask)

Requisito: tener Node o Python instalado y configurado

Paso 1: descargar de GitHub

node https://github.com/openai/openai-quickstart-node

git clone https://github.com/openai/openai-quickstart-node.git

python https://github.com/openai/openai-quickstart-python

git clone https://github.com/openai/openai-quickstart-python.git

Uso vía API: **Aplicaciones base (quickstart) (II)**

Paso 2: crear API key y configurarla

cd openai-quickstart-node

cp .env.example .env

cd openai-quickstart-python

cp .env.example .env

Uso vía API: **Aplicaciones base (quickstart) (III)**

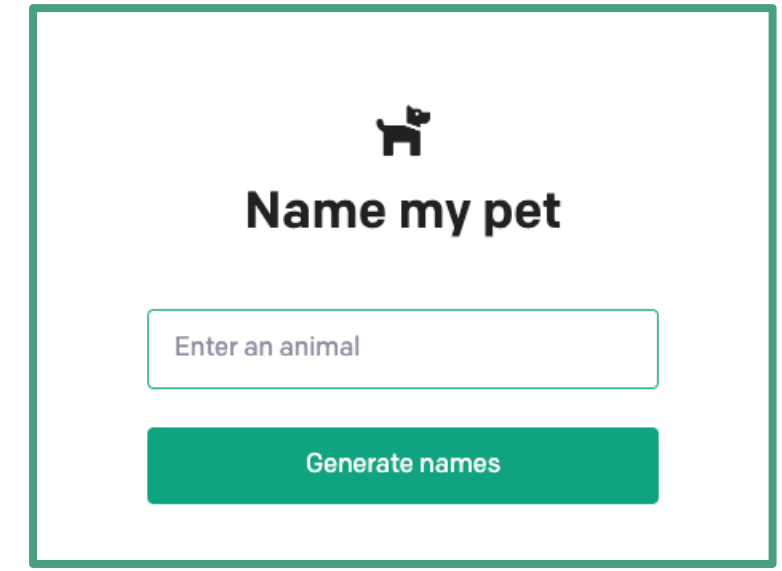
Paso 3: ejecutar la aplicación

node

npm install

npm run dev

Entrar en http://localhost:3000



python™

python -m venv venv

. venv/bin/activate

pip install -r requirements.txt

flask run

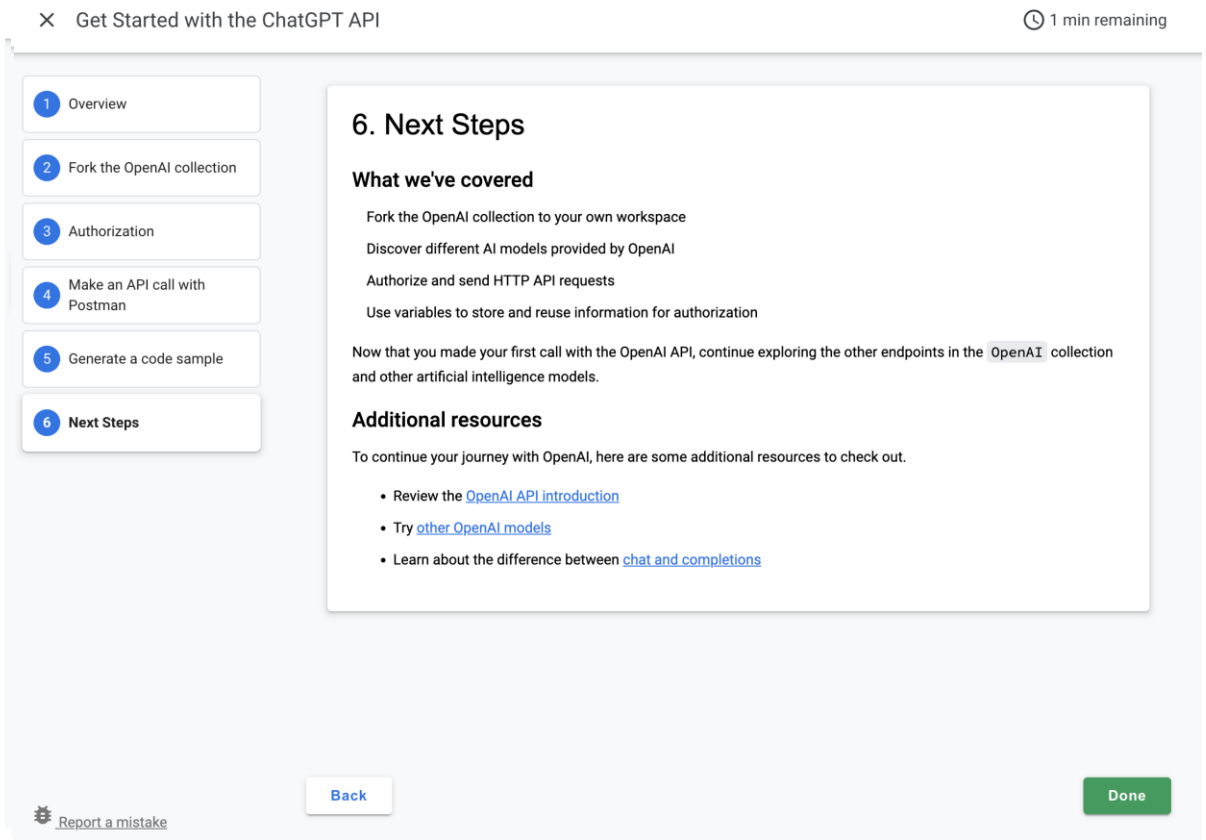
Entrar en http://localhost:5000

Uso vía API: **Con Postman**

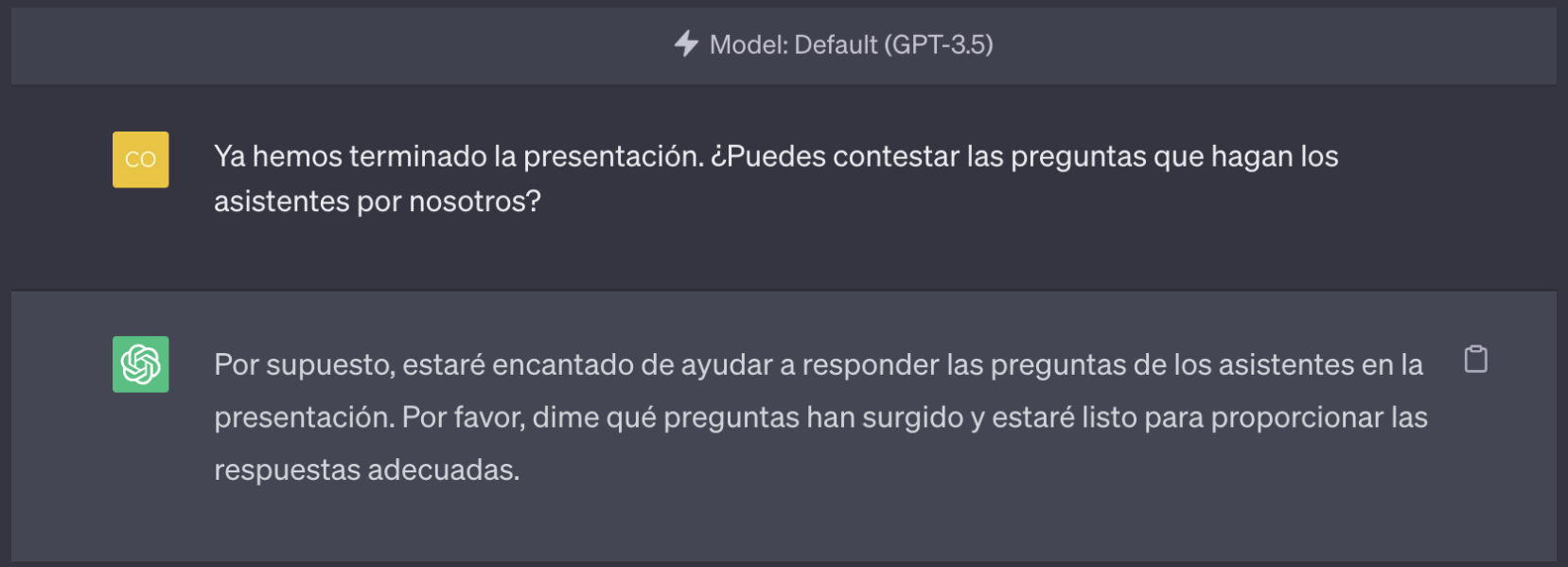
En menos de 10 minutos

POSTMAN

https://quickstarts.postman.com/guide/chatgpt/index.html



Preguntas y respuestas



VNiVERSiDAD D SALAMANCA

AIR INSTITUTE

https://bisite.usal.es – https://air-institute.com

Gracias

CO Escríbenos una frase corta de agradecimiento por su atención para que lo podamos poner en la última diapositiva de la presentación. Que sea ingeniosa.

"Gracias por alimentar nuestras mentes con su atención voraz."

Contacto

chamoso@usal.es

rgarcia@air-institute.com

VNiVERSiDAD D SALAMANCA

AIR INSTITUTE

https://bisite.usal.es – https://air-institute.com