

# Transformers: Razonamiento y Representación del Conocimiento

---

## Introducción

---

Las **redes neuronales** son modelos computacionales inspirados en el cerebro humano que procesan información mediante la combinación lineal del vector de entrada con los pesos de la red. Con el avance del **deep learning**, estas redes han evolucionado para incluir muchas capas, principalmente **convolucionales**, permitiendo aprender relaciones complejas en los datos.

### Limitaciones de las Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNNs) son especialmente efectivas en el procesamiento de datos donde las relaciones locales son importantes, como en imágenes y señales. Capturan relaciones entre datos que están localmente “cerca” entre sí. Sin embargo, presentan limitaciones al manejar **secuencias de datos largas**, donde las dependencias pueden abarcar largas distancias:

- **Lenguaje Natural:** Las relaciones entre palabras pueden ser a largo plazo, y palabras distantes en una oración pueden influir en el significado.
- **Secuencias de Video:** La comprensión de eventos puede requerir información de fotogramas distantes.

## Transformers

---

Los **Transformers** surgen como una solución para abordar las limitaciones de las CNNs y otros modelos al procesar secuencias largas de datos. Fueron introducidos por Vaswani et al. en 2017 en el artículo “Attention is All You Need”.

### Características Clave

- **Captura de Relaciones a Largo Plazo:** Pueden modelar dependencias entre elementos distantes en una secuencia.
- **Dependencia de la Entrada:** Las relaciones se aprenden de manera adaptativa en función de la entrada específica.

Por ejemplo, en las oraciones:

- “El gato está sobre la mesa.”
- “El gato, que es de color negro, está sobre la mesa.”

En ambas oraciones, la relación entre “gato” y “está” es importante, incluso si están separadas por varias palabras.

## Procesamiento de Secuencias en Transformers

---

El proceso general para manejar secuencias de datos en Transformers consta de tres pasos principales:

1. **Tokenización**
2. **Embeddings**
3. **Aprendizaje mediante Autoatención**

### 1. Tokenización

La **tokenización** consiste en separar la secuencia de datos de entrada en unidades discretas llamadas **tokens**.

#### Tipos de Tokenizadores

- **Character Tokenizer:** Divide la entrada en caracteres individuales.
- **Word Tokenizer:** Separa la entrada en palabras completas.
- **Sub-word Tokenizer:** Combina los anteriores, dividiendo en unidades menores que palabras pero mayores que caracteres.

## Byte-Pair Encoding (BPE)

El **GPT Tokenizer** utiliza el método de **Byte-Pair Encoding (BPE)**:

- **Inicio:** Comienza con un conjunto de n-gramas de 1 carácter.
- **Proceso Iterativo:**
  - Se identifican las parejas de caracteres adyacentes más frecuentes y se combinan en nuevos tokens.
  - Se repite hasta alcanzar el tamaño de vocabulario deseado.
- **Resultado:** Un vocabulario eficiente que equilibra entre tokens de un solo carácter y palabras completas.

**Nota:** GPT-4 utiliza un vocabulario de 100,256 n-gramas.

## 2. Embeddings

Los **embeddings** transforman los tokens en vectores numéricos que capturan características semánticas y sintácticas.

- **Proceso:**
  - A cada token se le asigna un vector en un espacio de alta dimensión.
  - Estos vectores se entrenan para que tokens con significados similares tengan representaciones cercanas.
- **Ajuste del Embedding:**
  - Utiliza una **red neuronal** (como una **CNN** o una capa densa) para aprender las representaciones vectoriales.

## 3. Mecanismo de Autoatención (Self-Attention)

El núcleo del Transformer es el **mecanismo de autoatención**, que permite al modelo enfocarse en diferentes partes de la secuencia al procesar cada elemento.

### Problemas con Convoluciones Largas

- **Limitaciones:**
  - Las convoluciones tradicionales tienen un alcance local limitado.
  - Para capturar dependencias a larga distancia, se necesitarían convoluciones con kernels muy grandes o muchas capas, lo cual es ineficiente.

### Introducción al Mecanismo de Autoatención

- **Idea Fundamental:** Permitir que cada elemento de la secuencia preste atención a todos los demás elementos, ponderando su influencia.
- **Función de Puntuación de Atención (Attention Scoring Function):**
  - Calcula una puntuación que indica la importancia de cada elemento en relación con los demás.

### Definición de Matrices Entrenables

Se definen tres matrices entrenables que transforman los embeddings de entrada:

- **Query (Q):** Representa la consulta que hacemos sobre otros elementos.
- **Key (K):** Representa la clave que otros elementos usan para comparar con la consulta.
- **Value (V):** Contiene la información que se agregará a la salida.

### Cálculo en la Capa de Autoatención

### 1. Cálculo de Q, K y V:

$$\begin{aligned} \mathbf{Q} &= \mathbf{XW}^Q \\ \mathbf{K} &= \mathbf{XW}^K \\ \mathbf{V} &= \mathbf{XW}^V \end{aligned}$$

Donde:

- $\mathbf{X}$  es la matriz de embeddings de entrada.
- $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$  son matrices de pesos entrenables.

### 2. Cálculo de Puntuaciones de Atención:

$$\text{Atención}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right) \mathbf{V}$$

Donde  $d_k$  es la dimensión de los vectores de clave.

### 3. Salida:

- Se obtiene un nuevo conjunto de representaciones que incorporan información global de la secuencia.

## Multi-Head Attention (Atención Multi-Cabeza)

En lugar de calcular una única función de atención, se utilizan múltiples cabezas de atención para capturar diferentes tipos de relaciones.

#### • Proceso:

- Se realizan varias operaciones de autoatención en paralelo, cada una con sus propias matrices  $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$ .
- Las salidas de cada cabeza se concatenan y pasan a través de una capa lineal final.

#### • Ventajas:

- Permite que el modelo preste atención a diferentes posiciones y relaciones en simultáneo.
- Mejora la capacidad de captura de patrones complejos.

## Arquitectura del Transformer

---

El Transformer está compuesto por la repetición de bloques que combinan mecanismos de autoatención y redes neuronales feed-forward.

### Bloque Básico del Transformer

1. **Capa de Multi-Head Attention**
2. **Adición y Normalización:** Residual connection seguida de Layer Normalization.
3. **Capa Feed-Forward:** Red neuronal con una o más capas ocultas.
4. **Adición y Normalización:** Otra residual connection y layer normalization.

#### Pre-normalización vs. Post-normalización

- **Pre-normalización:** La normalización se aplica antes del bloque.
- **Post-normalización:** La normalización se aplica después del bloque.

### Modelo Básico del Transformer

#### 1. Tokenización y Embeddings:

- Se tokeniza y se embeben los datos de entrada.
- Se añaden **positional embeddings** para incorporar información sobre el orden de los tokens.

## 2. Aplicación de Bloques del Transformer:

- Se aplican uno o más bloques básicos descritos anteriormente.

## 3. Capa de Salida:


- Dependiendo de la tarea (e.g., clasificación, traducción), se aplica una capa de salida apropiada.

# Ejemplo de Aplicación: Traducción Automática

---

En tareas de traducción:

- **Entrada:** Secuencia de tokens en el idioma de origen.
- **Proceso:**
  - El encoder procesa la secuencia de entrada y genera representaciones internas.
  - El decoder utiliza estas representaciones junto con mecanismos de atención para generar la secuencia traducida.
- **Salida:** Secuencia de tokens en el idioma de destino.

 Transformer en Traducción

# Bibliografía Recomendada

---

- "Alice's Adventures in a Differentiable Wonderland" de Simone Scardapane.

Este recurso proporciona una introducción accesible y amigable a los conceptos detrás de los Transformers y el deep learning.

# Conclusiones

---

Los **Transformers** han revolucionado el campo del procesamiento de lenguaje natural y han sido adaptados a otras áreas como visión por computador y generación de secuencias.

- **Ventajas:**
  - Capturan dependencias a largo plazo de manera efectiva.
  - Paralelización eficiente durante el entrenamiento.
  - Flexibilidad para diversas tareas de secuencias.
- **Avances Recientes:**
  - Modelos como **GPT-3** y **GPT-4** demuestran el poder de los Transformers en generación de texto coherente y tareas complejas.
- **Desafíos:**
  - Alto costo computacional y requerimientos de datos masivos para entrenar grandes modelos.

---

Este resumen ha explorado los fundamentos de los Transformers, su arquitectura y su importancia en la representación del conocimiento y el razonamiento en secuencias de datos. Comprender estos conceptos es esencial para avanzar en el desarrollo de sistemas de inteligencia artificial capaces de manejar información compleja y secuencial como lenguaje natural y video.