**Resumen de “Attention Is All You Need”**

**📌 Objetivo del artículo**

Proponer una nueva arquitectura llamada **Transformer**, que elimina por completo el uso de redes recurrentes (RNNs) y convolucionales (CNNs), utilizando únicamente mecanismos de **atención** para procesar secuencias de datos, como el lenguaje natural.

**🏗️ Arquitectura del Transformer**

**1. Codificador-Decodificador (Encoder-Decoder)**

* El modelo se divide en dos partes:
  + **Codificador**: procesa la secuencia de entrada.
  + **Decodificador**: genera la secuencia de salida.
* Ambos están compuestos por bloques repetidos que contienen mecanismos de atención y capas feed-forward.

**2. Mecanismo de Atención**

* Introduce el concepto de **“Self-Attention”** (atención a uno mismo), que permite al modelo ponderar la importancia de cada palabra en una oración respecto a las demás.
* Fórmula clave:  
  Attention(Q,K,V)=softmax(QKTdk)VAttention(*Q*,*K*,*V*)=softmax(*dk*​​*QKT*​)*V*  
  donde:
  + Q*Q*: consultas (queries)
  + K*K*: claves (keys)
  + V*V*: valores (values)
  + dk*dk*​: dimensión de las claves

**3. Multi-Head Attention**

* En lugar de una sola atención, se usan múltiples “cabezas” de atención en paralelo para capturar diferentes relaciones semánticas.

**4. Positional Encoding**

* Como no hay recurrencia, se añade información de posición a los embeddings para que el modelo entienda el orden de las palabras.

**⚙️ Ventajas del Transformer**

* **Paralelización**: A diferencia de las RNNs, permite procesar todas las palabras de una secuencia al mismo tiempo.
* **Escalabilidad**: Se entrena más rápido y con mayor eficiencia en GPUs.
* **Mejor rendimiento**: Supera a modelos anteriores en tareas como traducción automática.

**📈 Resultados**

* El Transformer logró **mejores resultados** que modelos previos en tareas de traducción automática, como en el conjunto de datos WMT 2014 (Inglés ↔ Alemán y Francés).

**🧩 Impacto**

* Este trabajo sentó las bases para modelos como **BERT**, **GPT**, **T5**, entre otros.
* Cambió radicalmente el enfoque del NLP moderno, desplazando a las RNNs y CNNs.

Ejemplos de cómo funcionan distintos **pipelines** (secuencias de pasos) en modelos de aprendizaje profundo, especialmente en el contexto de **NLP**.

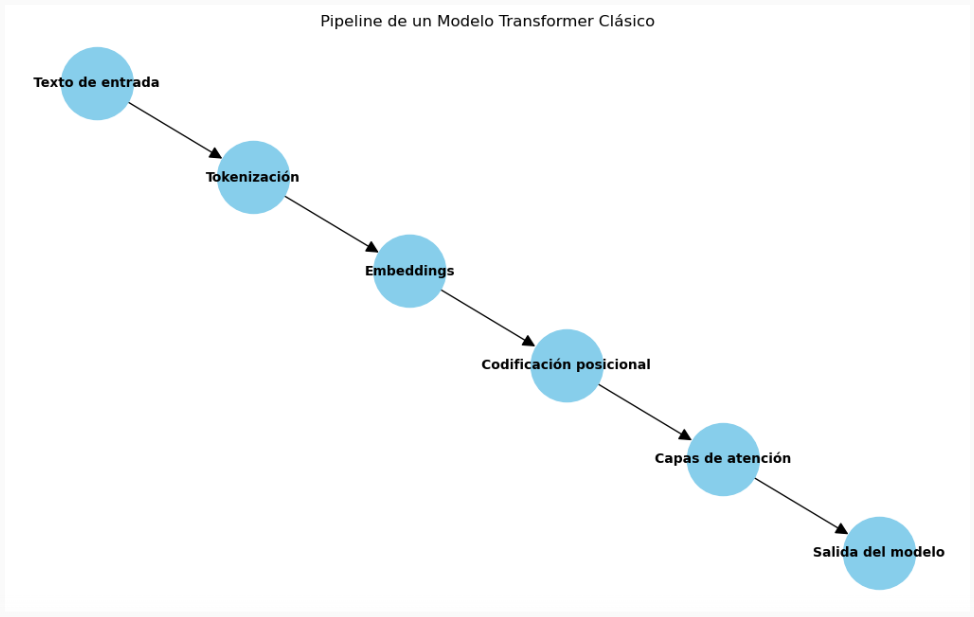
Tres ejemplos representativos:

**1. Pipeline de un modelo Transformer clásico (como BERT o GPT)**

**Objetivo: Comprender o generar texto.**

**Pasos:**

1. **Entrada de texto**  
   → "El gato duerme en el sofá."
2. **Tokenización**  
   → Se divide el texto en unidades (tokens):  
   ["El", "gato", "duerme", "en", "el", "sofá", "."]
3. **Conversión a IDs**  
   → Cada token se convierte en un número según un vocabulario.
4. **Embeddings**  
   → Los IDs se transforman en vectores densos (representaciones numéricas).
5. **Codificación posicional**  
   → Se añade información sobre el orden de las palabras.
6. **Capas del Transformer**  
   → Se aplican múltiples capas de atención y redes feed-forward.
7. **Salida**
   * En BERT: vectores de contexto para cada palabra (útiles para clasificación, QA, etc.)
   * En GPT: predicción del siguiente token (útil para generación de texto)



**Explicación de cada paso:**

1. **Texto de entrada**  
   → La oración original que se quiere procesar.
2. **Tokenización**  
   → Se divide el texto en unidades básicas (tokens).
3. **Embeddings**  
   → Cada token se convierte en un vector numérico que representa su significado.
4. **Codificación posicional**  
   → Se añade información sobre el orden de las palabras.
5. **Capas de atención**  
   → El modelo aprende qué partes del texto son más relevantes para cada palabra.
6. **Salida del modelo**  
   → Puede ser una predicción, una traducción, una respuesta, etc., según la tarea.

**¿Qué lo diferencia de modelos como GPT o BERT?**

| **Característica** | **Gemini** | **GPT (OpenAI)** | **BERT (Google)** |
| --- | --- | --- | --- |
| **Multimodalidad** | Sí (texto, imagen, audio, video) | Limitado (GPT-4V: texto + imagen) | No (solo texto) |
| **Entrenamiento** | Desde cero como multimodal | Texto primero, luego visión | Solo texto |
| **Tareas** | Generación, razonamiento, análisis de datos, agentes | Generación de texto | Clasificación, QA, etc. |
| **Velocidad** | Flash: optimizado para baja latencia | GPT-4: más pesado | Rápido pero limitado |

**2. Pipeline de un modelo de clasificación de texto (como RoBERTa + capa final)**

**Objetivo: Clasificar un texto (por ejemplo, detectar si una reseña es positiva o negativa).**

**Pasos:**

1. **Texto de entrada**  
   → "Este producto es excelente."
2. **Tokenización y embeddings**  
   → Igual que en el pipeline anterior.
3. **Modelo base (RoBERTa)**  
   → Produce un vector de salida para cada token.
4. **[CLS] token**  
   → Se toma el vector del token especial [CLS] como representación del texto completo.
5. **Capa densa + softmax**  
   → Se pasa por una capa final que predice la clase (positivo o negativo).
6. **Salida**  
   → "Positivo" con una probabilidad, por ejemplo, 92%.

Diagrama

El contenido generado por IA puede ser incorrecto.

**Explicación de cada paso:**

1. Entrada de texto  
   → Por ejemplo: "Este producto es excelente."
2. Tokenización  
   → El texto se divide en tokens (palabras o subpalabras).
3. Embeddings  
   → Cada token se convierte en un vector numérico que representa su significado.
4. Modelo (BERT, LSTM, etc.)  
   → Procesa los embeddings para capturar el contexto y las relaciones entre palabras.
5. Capa de clasificación (Softmax)  
   → Convierte la salida del modelo en probabilidades para cada clase.
6. Salida de clase  
   → Por ejemplo: "Positivo" con una probabilidad del 92%.

**3. Pipeline de un modelo de traducción automática (como el Transformer original)**

**Objetivo: Traducir una oración de un idioma a otro.**

**Pasos:**

1. **Texto de entrada**  
   → "The cat sleeps on the couch."
2. **Tokenización y embeddings (en inglés)**  
   → Se procesan los tokens de entrada.
3. **Codificador (encoder)**  
   → Produce una representación contextual de la oración.
4. **Decodificador (decoder)**  
   → Genera la traducción palabra por palabra, usando atención sobre la salida del encoder.
5. **Generación de salida**  
   → "El gato duerme en el sofá."

Diagrama

El contenido generado por IA puede ser incorrecto.

**Explicación de cada paso:**

1. **Entrada de texto (idioma origen)**  
   → Por ejemplo: "The cat sleeps on the couch."
2. **Tokenización**  
   → Se divide el texto en tokens (palabras o subpalabras).
3. **Embeddings**  
   → Cada token se convierte en un vector numérico.
4. **Codificador (Encoder)**  
   → Procesa los embeddings para capturar el significado del texto original.
5. **Decodificador (Decoder)**  
   → Genera los tokens del idioma destino, uno por uno, usando la información del encoder.
6. **Generación de tokens (idioma destino)**  
   → Se crean los tokens traducidos, como "El", "gato", "duerme"...
7. **Reconstrucción del texto traducido**  
   → Se unen los tokens para formar la oración final.
8. **Salida de texto (idioma destino)**  
   → Resultado: "El gato duerme en el sofá."

**Pipeline general de Gemini (versión 2.5 Pro o Flash)**

**1. Entrada multimodal**

Gemini puede recibir múltiples tipos de datos:

* Texto (como preguntas o instrucciones)
* Imágenes (fotos, diagramas)
* Audio (voz, sonidos)
* Video
* Código fuente

**2. Preprocesamiento**

Cada tipo de entrada se convierte en una representación numérica adecuada:

* Texto → tokens
* Imágenes → parches o embeddings visuales
* Audio → espectrogramas o embeddings acústicos
* Video → secuencias de frames + audio

**3. Embeddings unificados**

Todos los datos se transforman en un **espacio de representación común**, lo que permite que el modelo entienda relaciones entre modalidades (por ejemplo, texto que describe una imagen).

**4. Modelo central (Transformer multimodal)**

* Utiliza una arquitectura basada en Transformers, pero adaptada para manejar múltiples tipos de datos simultáneamente.
* Aplica mecanismos de atención cruzada entre modalidades (por ejemplo, texto que se refiere a una parte de una imagen).
* Puede razonar, generar código, responder preguntas, resumir videos, etc.

**5. Salida**

Dependiendo de la tarea, Gemini puede generar:

* Texto (respuestas, resúmenes, explicaciones)
* Código (en varios lenguajes)
* Audio (voz generada)
* Imágenes (en modelos con capacidad generativa)
* Combinaciones (por ejemplo, texto + imagen)

Diagrama

El contenido generado por IA puede ser incorrecto.

**Explicación del flujo:**

1. **Entrada Multimodal**  
   → Gemini puede recibir texto, imágenes, audio, video y código como entrada.
2. **Preprocesamiento**  
   → Cada tipo de dato se transforma en una representación numérica adecuada.
3. **Embeddings Unificados**  
   → Todas las modalidades se integran en un espacio común de representación.
4. **Modelo Transformer Multimodal**  
   → Procesa la información combinada usando atención cruzada entre modalidades.
5. **Salida**  
   → Puede generar texto, código, imágenes, o una combinación según la tarea.