

# Del Transformer Original al Ecosistema Moderno

Métodos Generativos, curso 2025-2026

---

Guillermo Iglesias, [guillermo.iglesias@upm.es](mailto:guillermo.iglesias@upm.es)

Jorge Dueñas Lerín, [jorge.duenas.lerin@upm.es](mailto:jorge.duenas.lerin@upm.es)

Edgar Talavera Muñoz, [e.talavera@upm.es](mailto:e.talavera@upm.es)

5 de noviembre de 2025

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos | UPM



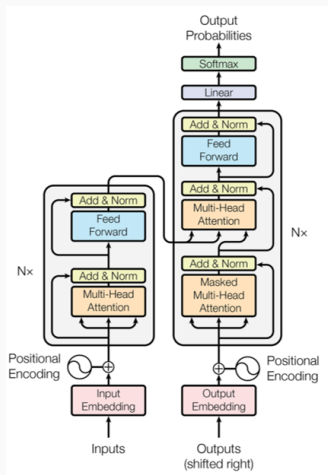
1. Introducción
2. Auto-encoders (AEs)
3. Auto-encoders Variacionales (VAEs)
4. Generative Adversarial Networks (GANs)
5. Transformers
6. **Del Transformer Original al Ecosistema Moderno**
7. Diffusion Models

# Evolución y Variantes de Transformers

---

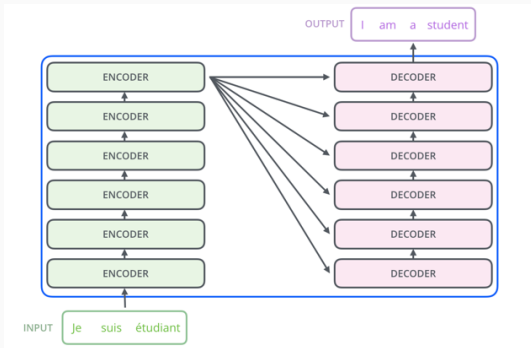
# Recap: ¿Por qué funcionó tan bien el transformer original?

- **Atención:** mecanismo que facilita capturar dependencias a largo plazo sin recurrencia
- **Paralelización:** eliminación de dependencias secuenciales
- **Escalabilidad:** arquitectura capaz de aprovechar grandes cantidades de datos
- **Transferibilidad:** representaciones aprendidas útiles para múltiples tareas



Estos principios han dado lugar a tres familias arquitectónicas principales...

# Las tres familias arquitectónicas



De este diseño original se derivan tres familias: los modelos que entienden (encoder-only), los que generan (decoder-only) y los que traducen de una representación a otra (encoder-decoder).

- **Encoder-only:** solo la parte izquierda (encoders apilados)
- **Decoder-only:** solo la parte derecha (decoders apilados),
  - sin cross-attention pero con masked-attention
- **Encoder-decoder:** la arquitectura completa

## Fases entrenamiento, tareas y modelos de ejemplos

---

# Tareas: Upstream vs Downstream

Pretraining (conocimiento general) → Fine-tuning (tarea específica)

## Upstream Task (Tarea general)

- Objetivo **inicial** del entrenamiento del modelo base
- Grandes volúmenes de datos, principalmente **no etiquetados**
- Aprendizaje **auto-supervisado o con supervisión débil**
- El modelo adquiere patrones lingüísticos y conocimiento general del mundo

## Downstream Task (Tarea específica)

- Tarea **final** o aplicación concreta
- Datos más limitados, normalmente **etiquetados**
- Aprendizaje **supervisado o basado en instrucciones**
- El modelo se adapta al dominio o tarea de interés

# Tareas: otras estrategias

## Continued pretraining / Domain Adaptation

- Corpus de dominio específico sin etiquetar
- Mismo **tipo** de objetivo: aprender más sobre el lenguaje pero con datos especializados
- Adapta vocabulario y conocimiento del dominio
- Ejemplo: textos médicos, legales, etc.

## Aprendizaje sobre las tareas en el contexto (In-context learning)

- Grandes modelos del lenguaje
- Grandes ventanas de contexto
- Aprender de la tarea sin modificar pesos.
- Ejemplo: zero-shot, few-shot, etc.

## Generación Aumentada por Recuperación (RAG)

- Vectorización de conocimiento
- Inyección en el contexto



# Taxonomía de Tareas: Auto-supervisadas (sin etiquetas)

## Sin etiquetas específicas:

### MLM *Masked Language Modeling*

- Predecir tokens enmascarados usando contexto bidireccional.
- **Ejemplo:** "El gato está \_\_\_ la alfombra" → "sobre".

### CLM *Causal Language Modeling*

- Predecir el siguiente token (autoregresivo).
- **Ejemplo:** "El clima está" → "soleado".

**NSP** Next Sentence Prediction — predecir si una oración sigue a otra en el texto original.

- **Ejemplo:**
  - A: "Juan salió de casa."
  - B: "Luego compró pan."
  - Etiqueta: Verdadero (sí son consecutivas).

# Taxonomía de Tareas: Auto-supervisadas (sin etiquetas)

## SOP *Sentence Order Prediction*

- Determinar el orden correcto de oraciones.
- **Ejemplo:** "Juan salió. Luego compró pan." vs "Luego compró pan. Juan salió."

## RTD *Replaced Token Detection*

- Detectar tokens falsos (ELECTRA).
- **Ejemplo:** "El perro comió pizza" → detectar que "pizza" es improbable.

## Span Corruption

- Predecir secuencias enmascaradas.
- **Ejemplo:** "El \_\_\_ en el jardín \_\_\_ muy feliz."
- seq1: perro corre; seq2: está.

# Taxonomía de Tareas: Supervisadas (Fine-tuning)

Con etiquetas específicas:

**Text Classification** Categorizar texto completo.

- **Ejemplo:** "Este producto es excelente"
- Sentimiento: positivo.

**Token Classification** Etiquetar cada token.

- **Ejemplo NER:** "Barack Obama nació en Hawái"
- [Persona: Barack Obama], [Lugar: Hawái].
- **Ejemplo POS:** Part-of-Speech tagging (sustantivos, verbos, etc.)

**Question Answering** Responder preguntas basadas en contexto.

- **Ejemplo extractivo:**
- "Obama fue el presid... nació en Hawái, en la ... ¿Dónde nació Obama?"
- Respuesta: "Hawái".

# Taxonomía de Tareas: Generación y Transformación

**Summarization** Resumir texto largo.

- **Ejemplo:** Artículo
- "Resumen breve".

**Paraphrasing** Reformular manteniendo significado.

- **Ejemplo:** "Hace frío"
- "La temperatura es baja".

**Dialogue** Responder en conversación.

- **Ejemplo:** Usuario: "¿Cómo estás?"
- Modelo: "Muy bien, gracias".

**Translation** Traducir idiomas.

- **Ejemplo:** "Hello world"
- "Hola mundo".

## Upstream Tasks (Pretraining – aprendizaje general)

- **Auto-supervisadas:** MLM, CLM, NSP, SOP, RTD, Span Corruption
- Aprender representaciones y conocimiento general del lenguaje

## Continued Pretraining / Domain Adaptation

- **Auto-supervisadas:** sobre corpus especializado.
- Ejemplo: GPT -> Generamos LegalGPT o MedicalGPT

## Downstream Tasks (Fine-tuning o uso contextual)

- **Supervisadas:** Text / Token Classification, QA, Summarization, Paraphrasing, Dialogue, Translation
- **Sin actualización de pesos:** In-context Learning (zero-shot, few-shot)

## Otras Estrategias:

- RAG (Retrieval-Augmented Generation)

## Analogía educativa

Pensemos en el modelo como una persona que aprende.

**Pretraining:** la etapa escolar. Aprende a leer, escribir y el uso del lenguaje. No hay tareas concretas, solo comprensión general del lenguaje y del mundo.

**Domain adaptation:** la universidad. Se especializa en un campo (medicina, derecho, programación) Aprende con textos del dominio, pero sigue sin etiquetas.

**Fine-tuning:** el trabajo. Aprende una tarea concreta con ejemplos reales. Ya no estudia teoría: aplica lo aprendido para resolver problemas específicos. **Ejemplos:** un médico que aprende a diagnosticar, o un abogado que redacta demandas

## Ejemplos de modelos

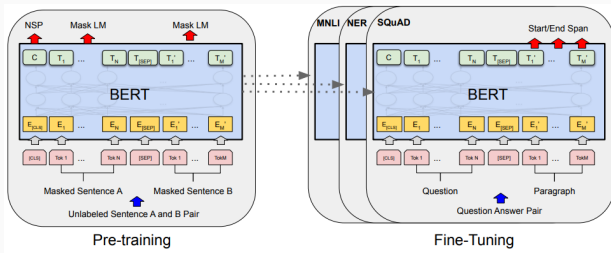
---

# Familia 1: Encoder-only — BERT

BERT: *Bidirectional Encoder Representations from Transformers*

## Características principales

- **Arquitectura:** sólo usa *encoders* del Transformer
- **Bidireccionalidad:** cada token ve todo el contexto (izquierda y derecha)
- **Objetivo de entrenamiento:** *Masked Language Modeling (MLM)* y *Next Sentence Prediction (NSP)*
- Produce **representaciones contextuales** ricas





# Entrenamiento de BERT (Pre-training)

**Objetivo:** aprender representaciones lingüísticas generales a partir de grandes corpus sin supervisión.

**Datos de entrenamiento:** *BooksCorpus* y *Wikipedia* en inglés

## Tareas de pre-entrenamiento

- Ambas cabezas (MLM y NSP) se entrenan en paralelo

$$L = L_{\text{MLM}} + L_{\text{NSP}}$$

- **Masked Language Modeling (MLM)**

- Se selecciona aleatoriamente el **15% de los tokens** para intentar predecirlos.
- De esos tokens seleccionados:
  - 80% se reemplazan por **[MASK]**
  - 10% se reemplazan por un token aleatorio
  - 10% se dejan sin modificar (eliminar dependencia de **[MASK]**)

- **Next Sentence Prediction (NSP)**

- 50% de los pares son oraciones consecutivas (*IsNext*)
- 50% son oraciones no relacionadas (*NotNext*)

# Representaciones y entrenamiento de BERT

El token [CLS] Se añade al inicio de la secuencia

- Su vector final **resume el significado global** del texto.
- Se usa para tareas de clasificación (NSP, análisis de sentimiento, etc.).

Embeddings por token

- Cada token obtiene una representación contextual completa.
- Útiles para tareas de secuencia (NER, QA, POS tagging).

Ejemplo

*Input: "El [MASK] duerme. Está [MASK]."*

*Objetivo MLM: predecir "gato", "cansado"*

*Objetivo NSP: IsNext*

*Input: "El [MASK] duerme. Los trenes son [MASK]."*

*Objetivo MLM: predecir "gato", "rápidos"*

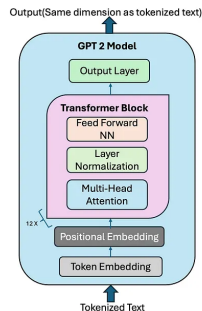
*Objetivo NSP: NotNext*

# Familia 2: Decoder-only — GPT-2

## GPT-2: Generative Pre-trained Transformer 2

### Características principales

- Usa solo la parte **decoder** del Transformer.
- Entrenamiento **autoregresivo**: predice el siguiente token dado el contexto previo.
- Objetivo de entrenamiento: **Causal Language Modeling (CLM)**.
- Atención **unidireccional**: cada token solo ve los anteriores.



Link a fuente de imagen  
¡Ojo! Masked MHA

**GPT-2:** ¡Ojo! se entrena en paralelo. En inferencia se va token a token.

**Objetivo:** generar texto coherente y fluido, un token a la vez.

## Durante la generación

- El modelo calcula la probabilidad de cada token posible:

$$P(x_t|x_{<t})$$

- Se elige el siguiente token según una estrategia de muestreo.
- El proceso continúa hasta alcanzar un token de parada o el límite de longitud.

## Estrategias de muestreo

- **Greedy:** elige siempre el token más probable (poco creativo).
- **Sampling:** elige aleatoriamente según la distribución de probabilidad.
- **Top- $k$  / Nucleus (Top- $p$ ):** limita el muestreo a los tokens más probables.

# Generación con GPT-2

## Ejemplo de generación

*Input: "El gato duerme en" → "el sofá."*

*Input: "La IA transformará" → "la educación en el futuro."*

## Temperatura y creatividad

Controla la aleatoriedad en el muestreo:

$$P'_i = \frac{\exp(\log P_i / T)}{\sum_j \exp(\log P_j / T)}$$

Temperatura baja ( $T < 1$ ) → texto predecible

Temperatura alta ( $T > 1$ ) → texto más variado

## Casos de uso:

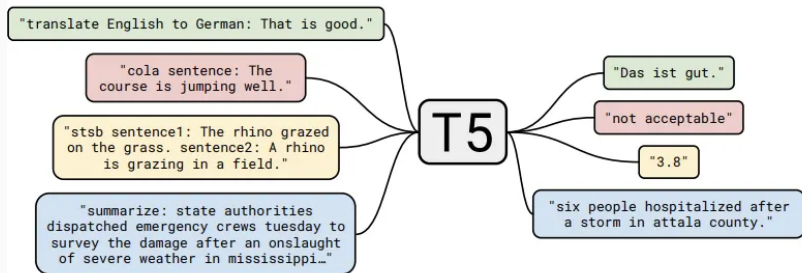
- Resumen de textos
- Paráfrasis
- Question Answering generativo
- Traducción automática (También aunque se usan más enc-dec)

### T5: *Text-To-Text Transfer Transformer*

#### Características principales:

- **Arquitectura completa:** encoder bidireccional + decoder autoregresivo.
- Entrenamiento basado en **denoising**: el modelo aprende a reconstruir texto corrupto.
- Formulación unificada: todo se expresa como una tarea **text-to-text**.
- Máxima **flexibilidad** para tareas de tipo seq2seq.

# Familia 3: Encoder-Decoder - T5



## Casos de uso:

- Traducción automática
- Resumen de textos
- Paráfrasis y reformulación
- Question Answering generativo

# Comparación de las tres familias

Familia	Atención	Objetivo	Uso principal
Encoder-only	Bidireccional	MLM	Comprensión
Decoder-only	Causal	CLM	Generación
Encoder-decoder	Ambas	Seq2seq	Transformación

## Regla general

- ¿Necesitas **entender**? → Encoder-only
- ¿Necesitas **generar**? → Decoder-only
- ¿Necesitas **transformar**? → Encoder-decoder



# Conceptos Clave del Ecosistema

---

## 1. Transfer Learning: Pretraining + Fine-tuning

- Entrenamiento en dos fases
- Modifica los parámetros del modelo
- Requiere dataset de la tarea objetivo
- Ejemplo: BERT pre-entrenado -> fine-tuned para clasificación de sentimiento
- Ejemplo: BERT pre-entrenado -> Domain adaptation "legal" -> fine-tuned para clasificación de posibles veredictos

## 2. In-Context Learning (emergente en modelos grandes)

- Solo disponible en modelos suficientemente grandes
- Ventana grande, conocimiento amplio
- **No modifica** los parámetros del modelo
- Aprende de los ejemplos en el prompt

# In-Context Learning: Zero-shot y Few-shot

Zero-shot: Sin ejemplos

*Classify the sentiment of this text: "I love this product!"*

*Answer: Positive*

Few-shot: Con pocos ejemplos (2-5 típicamente)

*Classify sentiment:*

*Text: "Great service!" → Positive*

*Text: "Terrible experience." → Negative*

*Text: "I love this!" → ?*

## Diferencia clave

In-context learning NO actualiza parámetros, solo usa el contexto del prompt.

- **Disciplina** que surgió con el auge de los LLMs
- Técnicas principales:
  - **Instruction prompting:** instrucciones claras y específicas
  - **Few-shot prompting:** proporcionar ejemplos
  - **Chain-of-thought:** pedir razonamiento paso a paso
  - **Role prompting:** asignar un rol al modelo
- Arte y ciencia de diseñar instrucciones efectivas
  - CoT -> Modelos razonadores
- El prompt adecuado puede marcar la diferencia entre éxito y fracaso

# Chain of Thought (CoT) y Modelos razonadores

**Chain of Thought:** Pedir al modelo que piense y haga explícito el razonamiento intermedio de una respuesta/tarea.

- **Resultado:** En lugar de dar directamente la respuesta final, el modelo genera una secuencia de pasos lógicos antes de llegar a la solución.
- **Ejemplo:** Para resolver un problema matemático, el modelo escribe los cálculos y justificaciones antes de dar el resultado.
- **Objetivo:** Mejorar la precisión en tareas complejas (matemáticas, lógica, planificación) al permitir que el modelo “razone” de forma estructurada.

**Modelos razonadores:** Modelos diseñados y optimizados específicamente para razonar de forma más profunda y fiable. **Técnicas empleadas:**

- **Entrenamiento con CoT:** aprenden a generar pasos intermedios.
- **Self-Consistency:** generan varias cadenas de razonamiento y eligen la más coherente.
- **Verificación interna:** el modelo revisa sus propios pasos antes de responder.

## Herramientas: Hugging Face

---

# Hugging Face Hub: El GitHub de los modelos

## ¿Qué es Hugging Face Hub?

- Plataforma centralizada para compartir:
  - Modelos pre-entrenados (180,000+)
  - Datasets
  - Spaces (demos interactivas)
- **Model cards:** documentación estandarizada
  - Arquitectura y tamaño
  - Dataset de entrenamiento
  - Métricas de rendimiento
  - Limitaciones conocidas
  - Consideraciones éticas

## Biblioteca transformers

- Abstracción unificada
- Mismo código para todos los modelos
- API simple:

```
from transformers import pipeline  
classifier = pipeline("sentiment-analysis")  
result = classifier("I love this!")
```

## Notebook aplicando conceptos

---



## Práctica: Explorando herramientas y arquitecturas

- Hugging Face Hub: <https://huggingface.co/models>
- Transformers Documentation:  
<https://huggingface.co/docs/transformers>
- Ollama: <https://ollama.ai/>
- The Illustrated Transformer: <http://jalammar.github.io/illustrated-transformer/>
- Prompt Engineering Guide:  
<https://www.promptingguide.ai/>