

# MÓDULO 1: FUNDAMENTOS DE LLAMA 3.1 Y TRANSFORMERS

META DAY URUGUAY 2025

# Objetivos del Módulo

1

## Fundamentos de LLMs

Comprender los conceptos fundamentales de los Large Language Models (LLMs) y su evolución histórica hasta la actualidad.

2

## Arquitectura Transformer

Entender la arquitectura Transformer y el revolucionario mecanismo de atención que cambió el procesamiento del lenguaje natural.

3

## LLaMa 3.1 de Meta

Conocer en detalle las características, innovaciones y aplicaciones de LLaMa 3.1, uno de los modelos de código abierto más avanzados.

# Estructura del Módulo

## 1.1 Introducción a los Modelos de Lenguaje

15 minutos

- Definición y evolución de LLMs
- Conceptos fundamentales
- Aplicaciones prácticas

## 1.3 Introducción a LLaMa 3.1 de Meta

25 minutos

- Historia y evolución de LLaMa
- Características distintivas
- Comparación de versiones

1

2

## 1.2 Arquitectura Transformer Simplificada

20 minutos

- Componentes principales
- Mecanismo de atención
- Tipos de arquitecturas

3

# Introducción a los Modelos de Lenguaje

¿Qué son los LLMs y cómo han evolucionado hasta convertirse en una tecnología transformadora?

# ¿Qué son los Large Language Models (LLMs)?

- Los Large Language Models son sistemas de inteligencia artificial entrenados en enormes cantidades de texto que pueden:
  - Generar texto coherente y contextualmente relevante
  - Comprender consultas en lenguaje natural
  - Resumir documentos extensos
  - Crear contenido creativo como poesía o código

# Evolución de los Modelos de Lenguaje

## Modelos Estadísticos (1980–2010)

Basados en probabilidades y n-gramas. Limitados a contextos cortos y sin comprensión semántica profunda.

## Transformers (2017–actualidad)

Arquitectura revolucionaria que permitió procesar contextos más largos y capturar relaciones complejas entre palabras.

## Modelos Neuronales (2010–2017)

Redes neuronales recurrentes (RNN, LSTM) con mejor comprensión del contexto pero limitaciones con secuencias largas.

## LLMs Modernos (2020–actualidad)

Modelos masivos con billones de parámetros capaces de razonamiento avanzado y comprensión del lenguaje natural.

# Jerarquía de la IA para LLM



# Conceptos Fundamentales: Tokens

Los tokens son las unidades básicas de procesamiento en los LLMs:

- Pueden representar palabras, partes de palabras o caracteres
- El texto se divide en tokens antes de ser procesado
- Cada modelo tiene su propio vocabulario de tokens
- La tokenización afecta el rendimiento y la eficiencia



# Ejemplo de Tokenización

## Texto Original

"La inteligencia artificial está transformando el mundo"

## Tokens Posibles

["La", " intel", "igencia", " arti",  
"ficial", " está", " transform",  
"ando", " el", " mundo"]

## ID de Tokens

[245, 1089, 4532, 781, 3302, 456,  
8921, 2290, 125, 1456]

Los tokens pueden variar entre modelos, incluso para el mismo texto. Esto es especialmente relevante en idiomas diferentes al inglés.

# Conceptos Fundamentales: Embeddings

Los embeddings son representaciones vectoriales de tokens en un espacio multidimensional:

- Capturan el significado semántico de las palabras
- Palabras similares tienen embeddings cercanos
- Permiten operaciones matemáticas con significado semántico
- Son la base del "entendimiento" en los LLMs

# Matemática de los Embeddings

- Los embeddings representan palabras como vectores, permitiendo cálculos semánticos:

$$\textit{Rey} - \textit{Hombre} + \textit{Mujer} \approx \textit{Reina}$$

- Esta representación matemática permite a los modelos:
  - Calcular similitudes entre palabras
  - Resolver analogías
  - Comprender relaciones complejas

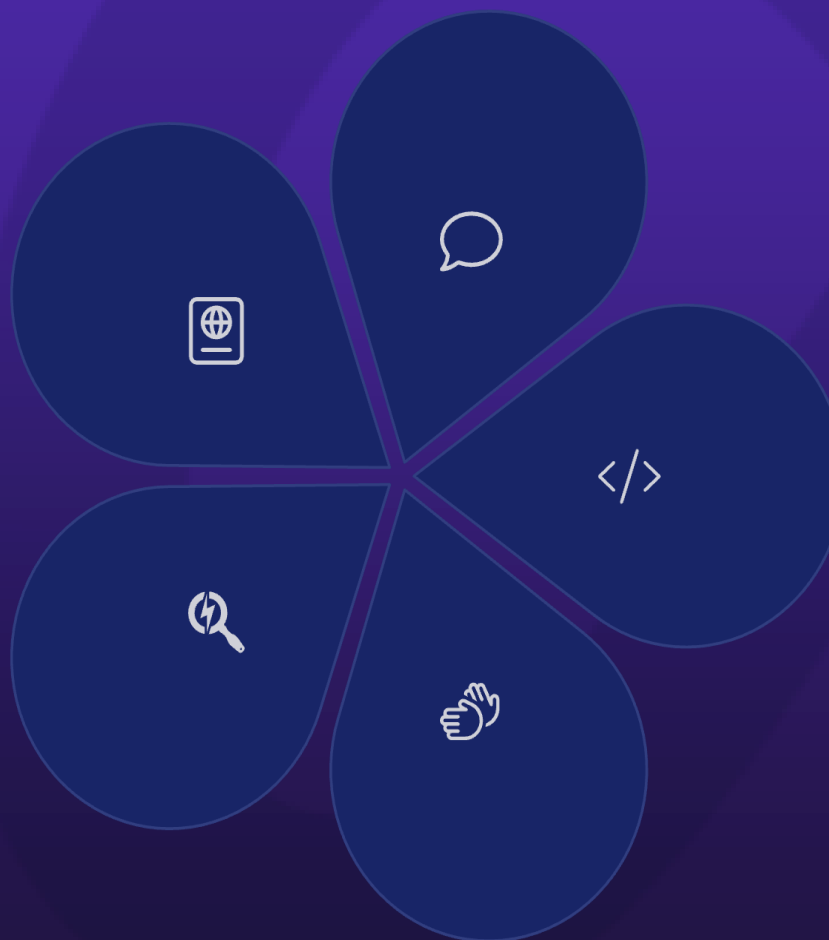
# Aplicaciones Principales de los LLMs

## Generación de Contenido

Creación de artículos, informes, correos electrónicos y cualquier tipo de texto estructurado.

## Búsqueda Semántica

Sistemas de búsqueda que entienden el significado en lugar de solo palabras clave.



## Asistentes Conversacionales

Chatbots y asistentes virtuales capaces de mantener conversaciones naturales y útiles.

## Programación Asistida

Generación, explicación y depuración de código en múltiples lenguajes de programación.

## Traducción

Traducción contextual entre idiomas con preservación de significado y estilo.

# Arquitectura Transformer

Comprendiendo el diseño revolucionario que cambió el procesamiento del lenguaje natural.

# Transformers



## 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\*<sup>†</sup>  
University of Toronto  
aidan@cs.toronto.edu

Lukasz Kaiser\*  
Google Brain  
lukaszkaizer@google.com

Illia Polosukhin\*<sup>‡</sup>  
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 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

\*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.

<sup>†</sup>Work performed while at Google Brain.

<sup>‡</sup>Work performed while at Google Research.

# Componentes Principales del Transformer

## Embedding + Encoding Posicional

Convierte tokens en vectores y añade información sobre la posición de cada token en la secuencia.

## Mecanismo de Atención

Permite al modelo "prestar atención" a diferentes partes del texto de entrada según su relevancia.

## Red Feed-Forward

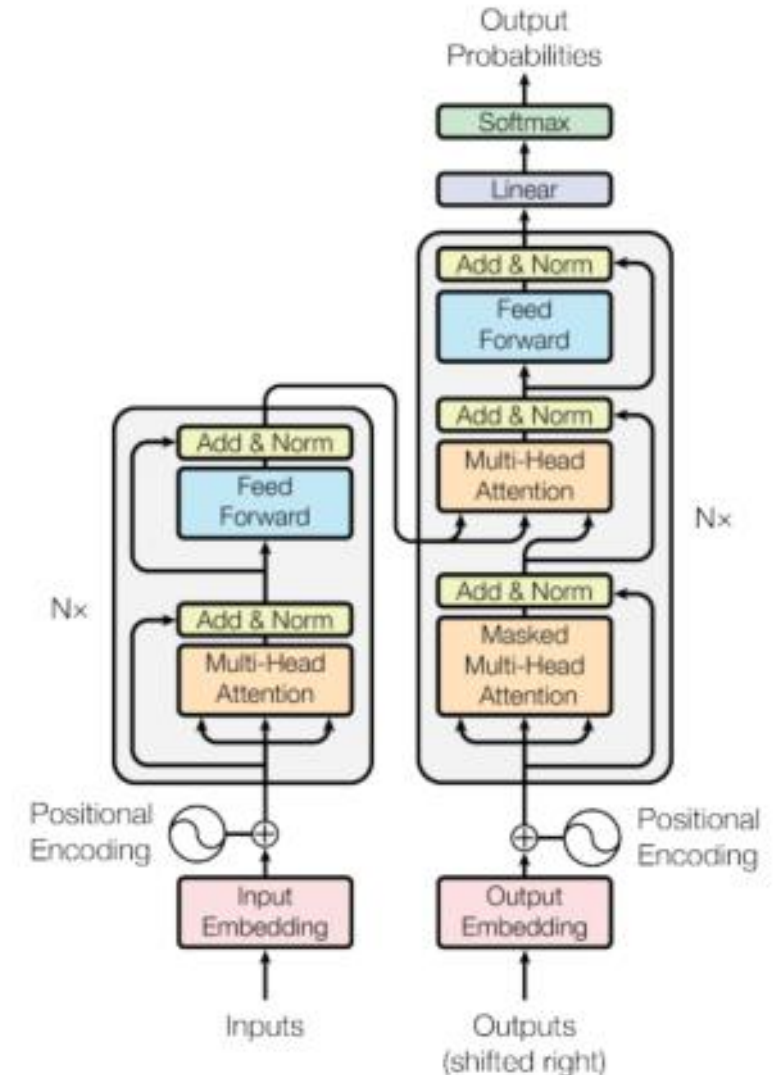
Procesa las representaciones generadas por el mecanismo de atención para cada posición.

## Normalización y Conexiones Residuales

Estabilizan el entrenamiento y permiten que la información fluya a través de capas profundas.

# Arquitectura Transformer

- Codificador-Decodificador
- Positional Encoding
- Multi-Head Attention

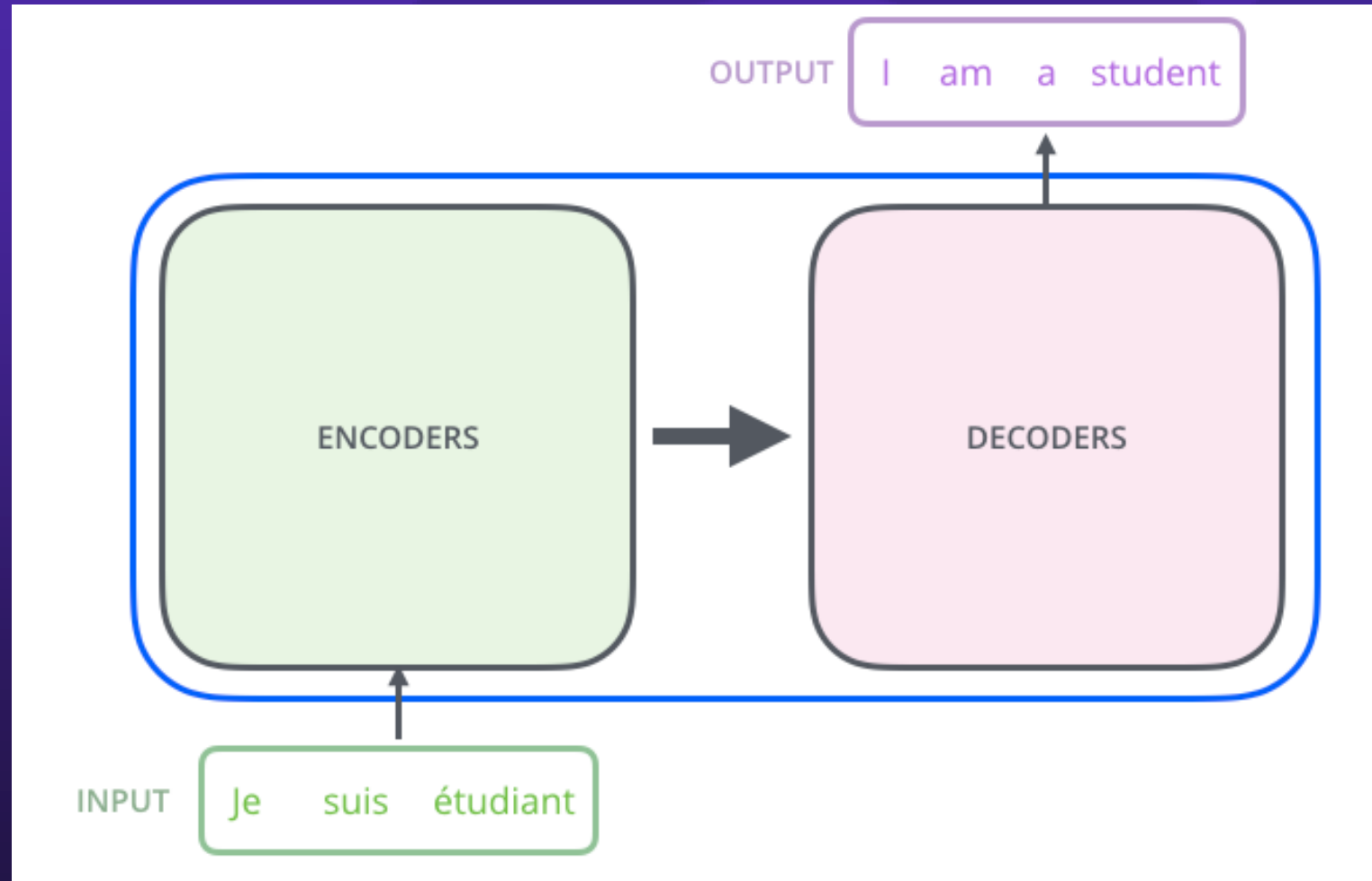




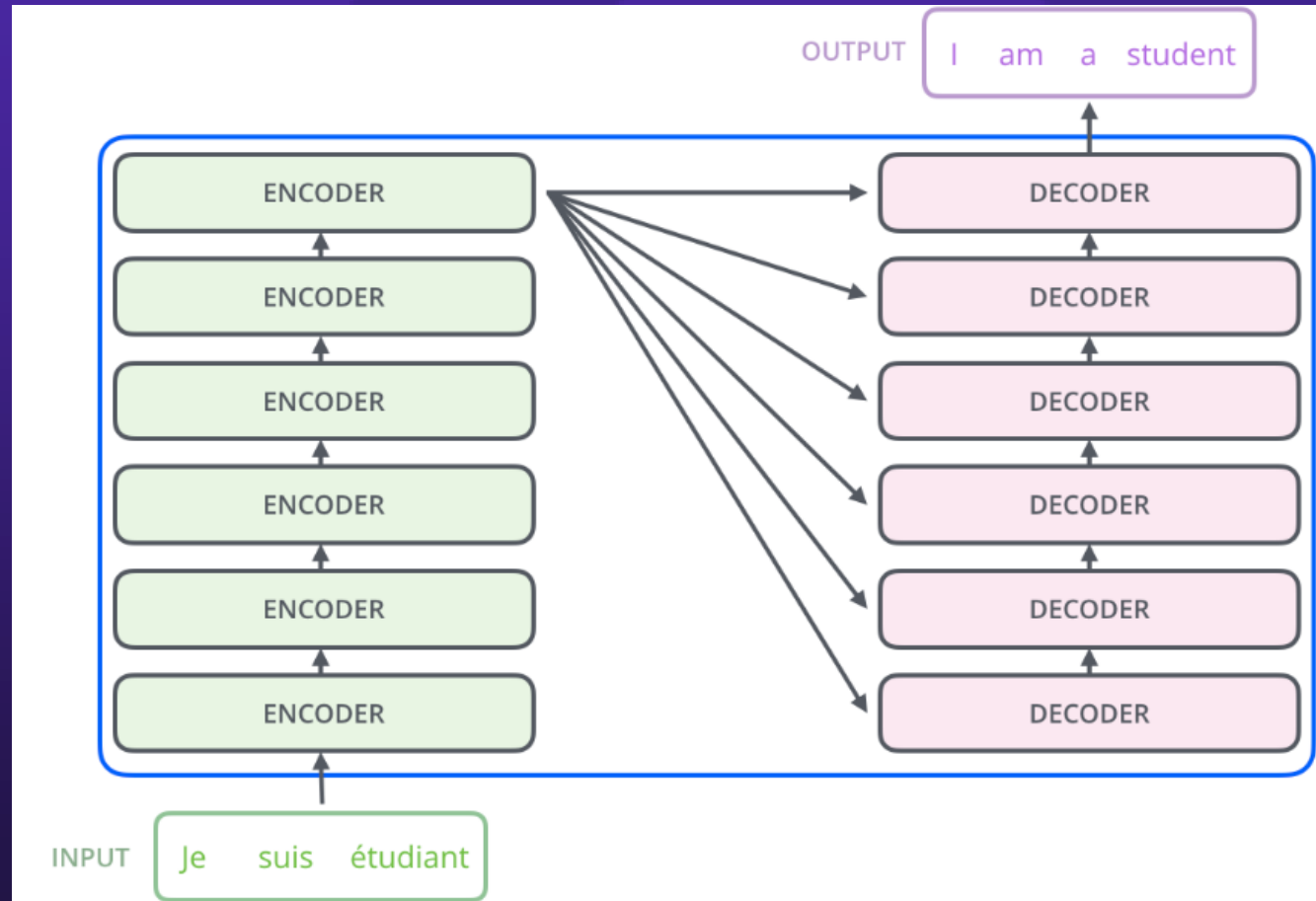
# Arquitectura Transformer



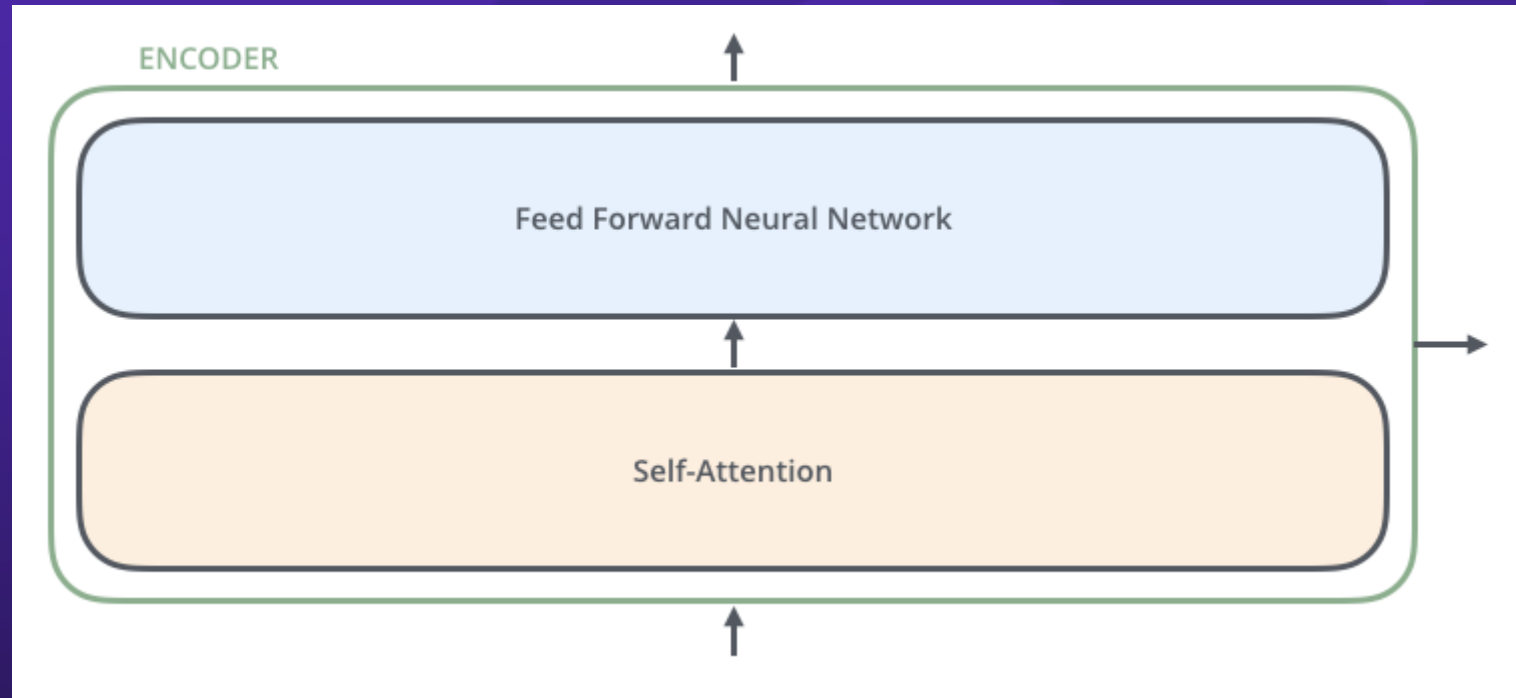
# Arquitectura Transformer

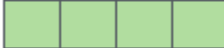


# Arquitectura Transformer

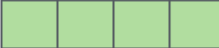


# Arquitectura Transformer

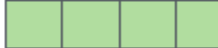


$x_1$  

Je

$x_2$  

suis

$x_3$  

étudiant

# Componentes de la Atención: Query, Key, Value

## Query (Q)

"¿Qué estoy buscando?" –  
Representa la palabra actual que  
busca información relacionada  
con otras palabras.

## Key (K)

"¿Qué tengo para ofrecer?" –  
Cada palabra ofrece información  
que puede ser relevante para  
otras palabras.

## Value (V)

"¿Cuál es mi contenido?" – El  
contenido real que se transferirá  
si hay una fuerte  
correspondencia entre Q y K.

Para cada token, se calculan Q, K y V mediante proyecciones lineales de su embedding.

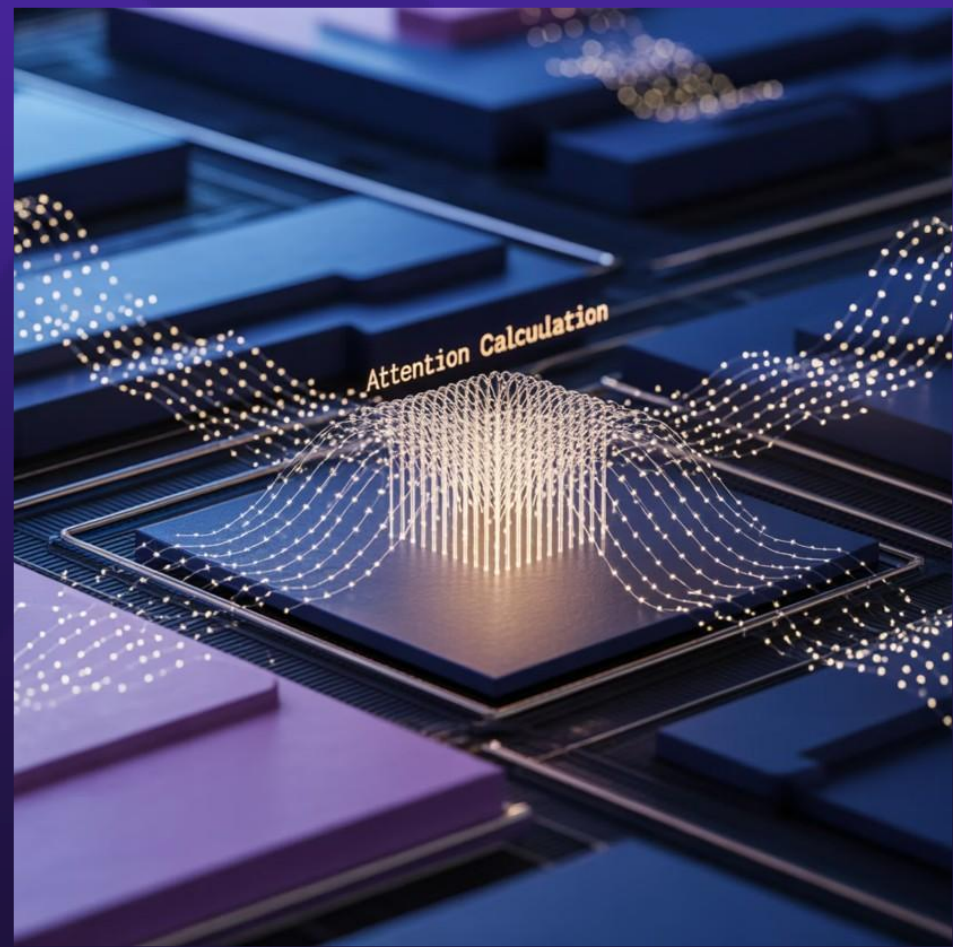
# Cálculo de la Atención

- El mecanismo de atención se calcula matemáticamente como:

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

- **Donde:**

- Q, K, V son matrices de Query, Key y Value
- d\_k es la dimensión de las claves
- La división por  $\sqrt{d_k}$  estabiliza los gradientes



# Ejemplo Práctico de Atención

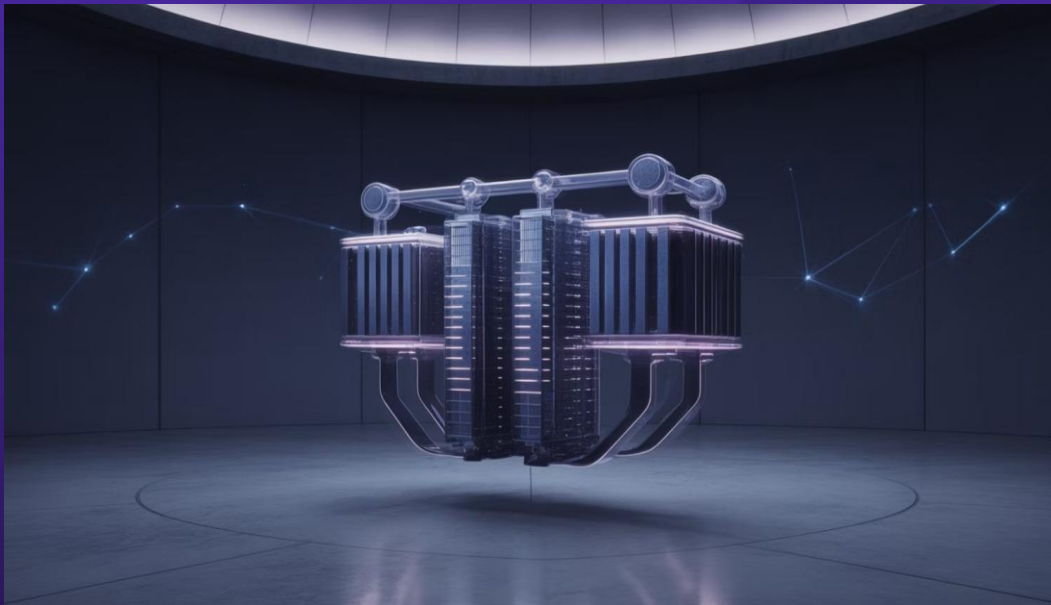
- Visualización de pesos de atención para la oración:
  - "El **perro** que persiguió al gato **cruzó** la calle".
  - Observe cómo "cruzó" presta mayor atención a "perro" (sujeto) que a "gato" (objeto), capturando correctamente la estructura gramatical.



# Arquitectura Encoder-Decoder vs. Decoder-Only

1

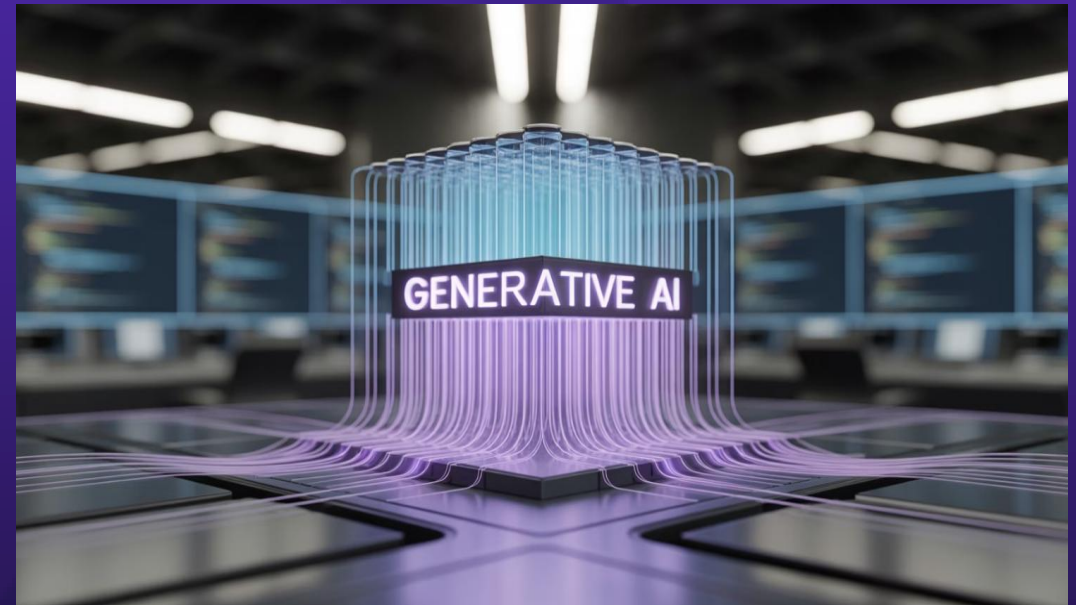
## Encoder-Decoder



- El encoder procesa toda la entrada
- El decoder genera la salida
- Ideal para traducción, resumen
- Ejemplos: T5, BART, mT5

2

## Decoder-Only



- Solo usa componentes de decoder
- Predice tokens autoregresivamente
- Ideal para generación de texto
- Ejemplos: GPT, LLaMa, Claude



# Ventana de Contexto

- La ventana de contexto determina cuánto texto "recuerda" el modelo:
  - Define el número máximo de tokens que el modelo puede considerar
  - Limita la "memoria" del modelo durante la generación
  - A mayor ventana, mayor capacidad de mantener coherencia
  - Evolución: 512 tokens (BERT) → 128K tokens (LLaMa 3.1)

# Ejercicio Práctico: Tokenización Manual

- **Instrucciones:**
  - Dividir la frase en posibles tokens
  - Identificar subpalabras comunes
  - Comparar con la tokenización real
- **Frase: "La inteligencia artificial generativa está revolucionando todas las industrias."**

**Resultado: [ "•", "La", "inteligencia", "artificial", "generativa", "está", "revolucionando", "todas", "las", "industrias", "." ]**

# Limitaciones de los Transformers

## Complejidad Cuadrática

La atención escala cuadráticamente con la longitud de la secuencia ( $O(n^2)$ ), limitando la eficiencia con textos muy largos.

## Consumo de Recursos

Los modelos grandes requieren GPUs potentes y mucha memoria, limitando su accesibilidad.

## "Alucinaciones"

Pueden generar información incorrecta pero presentada con confianza, especialmente fuera de su dominio de entrenamiento.

# Avances Recientes en Arquitecturas

- **Atención Eficiente**

- Técnicas como Sparse Attention, Flash Attention y Multi-Query Attention reducen los requisitos computacionales.

- **KV Cache**

- Reutiliza cálculos previos para acelerar la generación de secuencias largas.

- **Mixture of Experts (MoE)**

- Activa solo una parte del modelo para cada entrada, permitiendo modelos más grandes con la misma capacidad computacional.

- **Rotary Positional Embeddings (RoPE)**

- Mejor representación de información posicional que facilita la extrapolación a contextos más largos.

# Demostración Práctica

Realizaremos una demostración de los conceptos básicos.



# Introducción a LLaMa 3.1 de Meta

Conociendo uno de los modelos de código abierto más avanzados y sus capacidades revolucionarias.

# La Familia LLaMa: Evolución Histórica

## LLaMa 1 (Febrero 2023)

Primera versión: 7B-65B parámetros

- Contexto: 2K tokens
- Principalmente inglés
- Buen rendimiento en benchmarks académicos

## LLaMa 3 (Abril 2024)

Tercera versión: 8B-70B parámetros

- Contexto: 8K tokens
- Mejor soporte multilingüe
- Avances en tool calling y razonamiento

1

2

## LLaMa 2 (Julio 2023)

Segunda versión: 7B-70B parámetros

- Contexto: 4K tokens
- Versiones base y "chat"
- Mejoras en seguridad y razonamiento

3

4

## LLaMa 3.1 (Agosto 2024)

Versión actual: 8B-405B parámetros

- Contexto: 128K tokens
- Soporte para 8 idiomas
- Razonamiento avanzado

# El Salto de LLaMa 3 a LLaMa 3.1

## Aumento Masivo de Contexto

De 8K a 128K tokens (16x más), permitiendo procesar documentos completos o conversaciones largas.

## Nuevo Tamaño de 405B

Introducción del modelo de 405B parámetros, el más grande de la familia LLaMa y comparable a los modelos comerciales más avanzados.

## Mejoras en Multilingüismo

Soporte mejorado para español, francés, alemán, italiano, portugués, polaco, y filipino, además del inglés.

## Razonamiento Avanzado

Mejoras significativas en matemáticas, coding, y razonamiento general, acercándose a modelos propietarios como GPT-4.



# Versiones de LLaMa 3.1

**8B**

LLaMa 3.1 8B

Modelo más compacto, ideal para aplicaciones con recursos limitados, dispositivos móviles o despliegue en servidores económicos.

**70B**

LLaMa 3.1 70B

Modelo de tamaño medio con excelente balance entre rendimiento y requisitos computacionales. Recomendado para la mayoría de aplicaciones.

**405B**

LLaMa 3.1 405B

Modelo de élite con capacidades de razonamiento superior, ideal para tareas complejas que requieren máximo rendimiento.

# Características Distintivas de LLaMa 3.1



Contexto 128K tokens



Soporte 8 idiomas



Tool calling avanzado



Arquitectura optimizada



Razonamiento  
mejorado

# Arquitectura de LLaMa 3.1

- LLaMa 3.1 utiliza una arquitectura decoder-only optimizada con:
  - RMSNorm para normalización de capas
  - Rotary Positional Embeddings (RoPE)
  - SwiGLU como función de activación
  - Group Query Attention (GQA) para eficiencia
  - KV Cache optimizado
  - Entrenamiento con Objetivo Autorregresivo (Next Token Prediction)

# Contexto Extendido: 128K Tokens

- El contexto de 128K tokens es revolucionario porque permite:
  - Procesar documentos completos de cientos de páginas
  - Mantener conversaciones largas sin "olvidar" el inicio
  - Analizar código extenso con todas sus dependencias
  - Mantener coherencia en generaciones largas

# Soporte Multilingüe en LLaMa 3.1

## Inglés

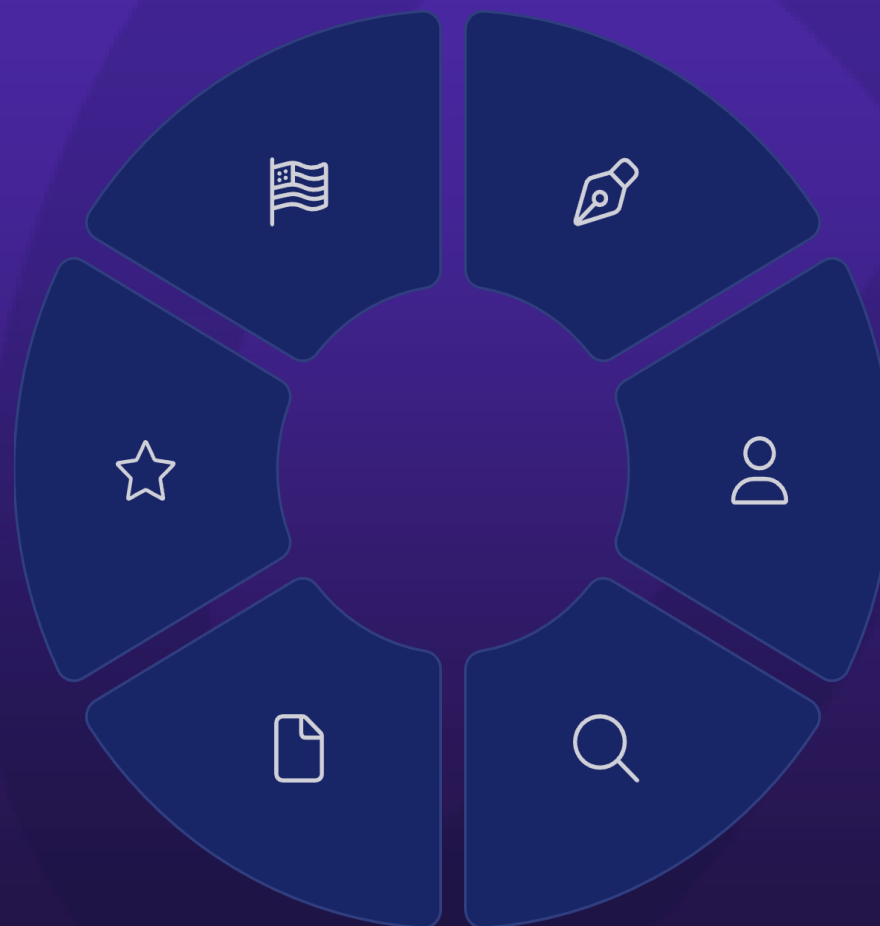
Idioma principal del modelo con máximo rendimiento.

## Portugués

Compatibilidad con variantes brasileña y europea.

## Italiano

Capacidad de procesamiento de texto contextual.



## Español

Soporte robusto para el segundo idioma más hablado del mundo.

## Francés

Buena capacidad de generación y comprensión.

## Alemán

Soporte para estructuras gramaticales complejas.

También incluye soporte para polaco y filipino con rendimiento variable.

# Tool Calling en LLaMa 3.1

- LLaMa 3.1 introduce capacidades avanzadas de tool calling que permiten:
  - Integración con APIs externas
  - Ejecución de funciones definidas por el usuario
  - Formateo correcto de parámetros JSON
  - Uso de múltiples herramientas en secuencia
- Esto facilita la creación de agentes AI que pueden interactuar con sistemas externos.

# Comparativa de Versiones LLaMa 3.1

Característica	LLaMa 3.1 8B	LLaMa 3.1 70B	LLaMa 3.1 405B
Parámetros	8 mil millones	70 mil millones	405 mil millones
VRAM requerida	~16 GB	~140 GB	~800 GB
GPUs recomendadas	1x RTX 4090	2-4x A100	8+ H100
Rendimiento relativo	Bueno	Muy bueno	Excelente
Casos de uso ideales	Aplicaciones móviles, inferencia económica	Uso general, empresas medianas	Investigación avanzada, grandes empresas

# Comparativa de LLMs Populares

Modelo	Parámetros	Contexto	Arquitectura	Fortalezas
GPT-4	~1.8T (estimado)	128K tokens	Transformer decoder-only	Razonamiento avanzado, multimodal
LLaMa 3.1	8B / 70B / 405B	128K tokens	Transformer decoder-only	Código abierto, multilingüe, eficiencia
Claude 3.5	No revelado	200K tokens	No revelado	Instrucciones complejas, documentos largos





# Comparativa Arquitectura: LLaMa 3.1 vs GPT-4

## LLaMa 3.1

### Arquitectura Base

Decoder-only transformer con optimizaciones específicas para eficiencia computacional

### Tamaños de Modelo

8B, 70B y 405B parámetros (versión más grande pública)

### Ventana de Contexto

128K tokens (ampliado desde 8K en versiones anteriores)

### Enfoque de Entrenamiento

Preentrenamiento + SFT + DPO, con generación de datos sintéticos iterativa

### Multimodalidad

Modelo de texto puro, sin soporte multimodal nativo

## GPT-4

### Arquitectura Base

Decoder-only transformer con modificaciones propietarias no públicas

### Tamaños de Modelo

Estimado >1 trillón de parámetros (tamaño exacto no revelado)

### Ventana de Contexto

128K tokens en GPT-4o (versión original: 32K)

### Enfoque de Entrenamiento

Preentrenamiento + RLHF + optimizaciones propietarias no reveladas

### Multimodalidad

Integración completa de texto, visión y audio en GPT-4o

# Requisitos de Hardware



## LLaMa 3.1 8B

Accesible con hardware de consumo:  
1 GPU de gama alta (RTX 4090) o 2  
GPUs de gama media.



## LLaMa 3.1 70B

Requiere hardware profesional: 2-4  
GPUs A100 (80GB) o equivalente  
para inferencia completa.



## LLaMa 3.1 405B

Requiere infraestructura de  
datacenter: mínimo 8 GPUs H100 o  
sistemas especializados.

Los porcentajes representan el nivel de recursos computacionales necesarios en relación al máximo.

# Técnicas de Optimización



## Cuantización

Reducción de precisión de parámetros (FP16, INT8, INT4) para disminuir requisitos de memoria y acelerar inferencia.



## Offloading

Distribución del modelo entre GPU y CPU para ejecutar modelos más grandes que la memoria de la GPU.



## LORA/QLoRA

Técnicas de fine-tuning eficiente que requieren menos recursos que el entrenamiento completo.

# Acceso a LLaMa 3.1

1

## Meta AI

Acceso directo a través del sitio de Meta AI o API de Meta (requiere solicitud).

2

## Hugging Face

Repositorio oficial con pesos del modelo y scripts de inferencia (requiere aceptar términos).

3

## Proveedores Cloud

Disponible en AWS, Azure, y Google Cloud como servicios gestionados.

4

## Ollama

Herramienta para ejecución local simplificada (solo 8B actualmente).

# Desafíos y Limitaciones de LLaMa 3.1



## Sesgo Anglófono

A pesar del soporte para 8 idiomas, el rendimiento sigue siendo superior en inglés, con menor calidad en otros idiomas.



## Alucinaciones

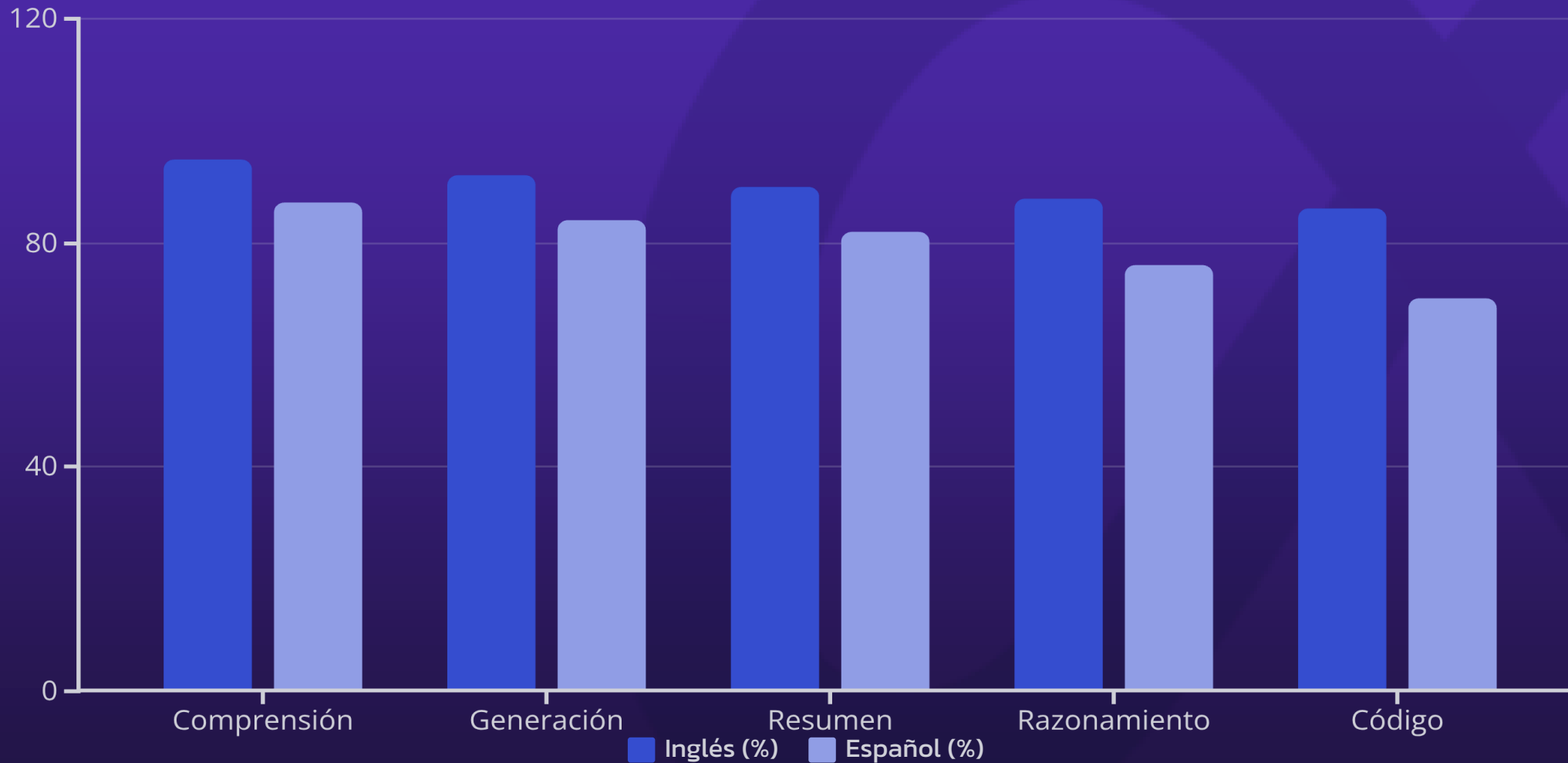
Como todos los LLMs, puede generar información incorrecta presentada con confianza, especialmente en dominios especializados.



## Requisitos Computacionales

El modelo 405B es prácticamente inaccesible para la mayoría de usuarios por sus enormes requisitos de hardware.

# Rendimiento en Español vs. Inglés



LLaMa 3.1 muestra una brecha de rendimiento entre inglés y español, siendo más pronunciada en tareas complejas como razonamiento y programación.

# Integración de LLaMa en Aplicaciones



## Configuración

Elegir versión del modelo, ajustar hiperparámetros (temperatura, top\_p) y definir sistema de prompt.



## Conexión

Establecer conexión mediante API o inferencia local a través de bibliotecas como llamafile, llama.cpp o Transformers.



## Interfaz

Desarrollar interfaz de usuario para entrada de consultas y visualización de respuestas.



## Supervisión

Implementar monitoreo de rendimiento, retroalimentación de usuarios y optimización continua.

# Hiperparámetros de Generación

Parámetro	Función	Valores Típicos	Impacto
Temperature	Controla aleatoriedad	0.0 - 1.5	Mayor valor = más creatividad
Top_p	Muestreo nucleus	0.1 - 1.0	Limita tokens por probabilidad
Max_tokens	Longitud máxima	128 - 4096	Limita extensión de respuesta
Repetition_penalty	Penaliza repeticiones	1.0 - 1.5	Reduce bucles y repeticiones
Presence_penalty	Penaliza temas ya mencionados	0.0 - 1.0	Favorece diversidad temática



# Impacto de la Temperatura

## Temperatura Baja (0.2)

"La capital de España es Madrid. Madrid es la ciudad más poblada del país y su centro político, económico y cultural."

Respuestas deterministas, factuales y predecibles. Ideal para tareas que requieren precisión.

## Temperatura Alta (0.8)

"España tiene como capital la vibrante ciudad de Madrid, un fascinante núcleo urbano que combina tradición e innovación, donde el arte y la gastronomía se entrelazan en sus animadas calles y plazas."

Respuestas más creativas, diversas y elaboradas. Mejor para generación creativa.

# Estrategias de Prompt Engineering



## Prompts Básicos

Instrucciones directas y específicas.

Ejemplo: "Explica qué es la fotosíntesis."



## Prompts con Rol

Asignar un rol específico al modelo.

Ejemplo: "Eres un biólogo experto. Explica la fotosíntesis."



## Prompts Estructurados

Definir formato y estructura deseada.

Ejemplo: "Explica la fotosíntesis en 5 pasos, con un ejemplo para cada paso."



## Chain-of-Thought

Guiar el razonamiento paso a paso.

Ejemplo: "Piensa paso a paso cómo resolver este problema matemático..."

# Ajuste de LLaMa para Casos Específicos

## Fine-tuning Completo

Ajuste de todos los parámetros del modelo.  
Requiere muchos recursos pero ofrece máximo rendimiento para tareas específicas.

## LoRA / QLoRA

Low-Rank Adaptation. Ajusta solo un pequeño conjunto de parámetros adaptadores, reduciendo drásticamente los requisitos computacionales.

## Retrieval Augmented Generation (RAG)

Combina el modelo con una base de conocimiento externa. Ideal para información especializada o actualizada.

## Prompt Engineering Avanzado

Optimización sistemática de prompts sin modificar el modelo. La opción más accesible pero con limitaciones.

# El Futuro de LLaMa



## Capacidades Multimodales

Integración de procesamiento de imágenes, audio y posiblemente video en futuras versiones.



## Mejoras en Razonamiento

Avances en capacidades de razonamiento complejo, matemáticas y resolución de problemas.



## Expansión Multilingüe

Soporte para más idiomas y mejor rendimiento en idiomas no ingleses.

## Optimización de Recursos

Reducción de requisitos computacionales manteniendo o mejorando el rendimiento.

# Recursos Adicionales



## Documentación Oficial

Repositorio oficial en Hugging Face con documentación completa, ejemplos y guías de uso.

[huggingface.co/meta-llama](https://huggingface.co/meta-llama)



## Comunidad

Foros y comunidades donde discutir implementaciones, problemas y soluciones con otros desarrolladores.

[github.com/meta-llama/llama3](https://github.com/meta-llama/llama3)



## Tutoriales

Tutoriales y cursos en línea para profundizar en aspectos específicos de LLaMa 3.1.

[youtube.com/c/MetaAI](https://youtube.com/c/MetaAI)