UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA

Módulo 6
IA Generativa

Diploma en Inteligencia Artificial

informatica.usm.cl
@informaticausm

Capítulo 3: Transformers, LLM y GPT

- Evolución Histórica de los Modelos de Lenguaje
- Fundamentos de Procesamiento de Lenguaje Natural (NLP)
- La Arquitectura Transformer
- Tipos de Arquitecturas y Modelos Derivados
- BERT (Bidirectional Encoder Representations from Transformers)
- GPT y LLMs (Large Language Models)
- Avances Recientes y Estado del Arte

Algo de Historia

Modelos Noisy Channel

- Previo a 2017, dos tareas se realizaban en modelos de lenguaje:
  - Reconocimiento de habla (speech recognition)
  - Traducción automática (machine translation)
- Definición: un modelo **noisy channel** combina un modelo de transducción (probabilidad de convertir **y** en **x**) con un modelo de lenguaje (probabilidad de **y**)

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}\, p(\mathbf{y} \mid \mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmax}}\, p(\mathbf{x} \mid \mathbf{y}) p(\mathbf{y})$$

transduction model

language model

- Meta: Recuperar **y** a partir de **x**
  - Para habla: **x** es una señal acústica, **y** es la transcripción
  - Para traducción automática: **x** es una sentencia en un lenguaje origen **y** es la sentencia en un lenguaje destino

Modelos grande de lenguaje n-gramas

- Los fundacionales (verdaderos) largos modelos de lenguaje fueron modelos n-gramas
- Google n-gramas
  - 2006: primer reléase, n-gramas en Inglés
    - Entrenado con 1trillón de tokens de textos web (95 billones de sentencias)
    - Incluía 1-grama, 2-gramas, 3-gramas, 4-gramas, 5-gramas
  - 2009-2019: n-gramas en Japonés, Chino, Sueco, Español, Portugués, Polaco, Italiano, Francés, Alemán, Checo

English n-gram model is **~3 billion parameters**

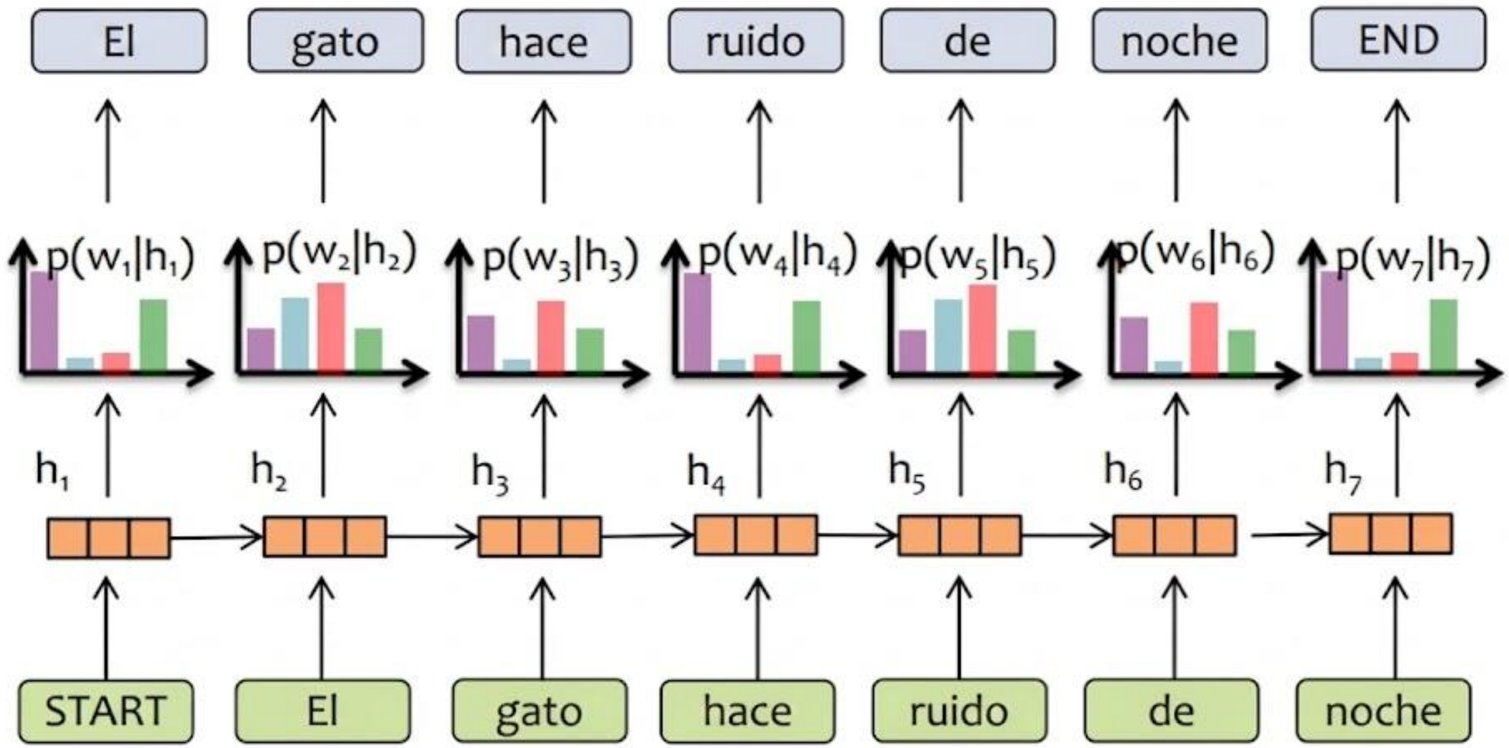
Number of unigrams:	13,588,391
Number of bigrams:	314,843,401
Number of trigrams:	977,069,902
Number of fourgrams:	1,313,818,354
Number of fivegrams:	1,176,470,663

```
serve as the incoming 92
serve as the incubator 99
serve as the independent 794
serve as the index 223
serve as the indication 72
serve as the indicator 120
serve as the indicators 45
serve as the indispensable 111
serve as the indispensible 40
serve as the individual 234
serve as the industrial 52
serve as the industry 607
```

```
accessoire Accessoires </S> 515
accessoire Accord i-CTDi 65
accessoire Accra accu 312
accessoire Acheter cet 1402
accessoire Ajouter au 160
accessoire Amour Beauté 112
accessoire Annuaire LOEIL 49
accessoire Architecture artiste 531
accessoire Attention : 44
```

```
惯例 为 电影 创作 52
惯例 为 的 是 95
惯例 为 目标 职位 49
惯例 为 确保 合作 69
惯例 为 确保 重组 213
惯例 为 科研 和 55
惯例 为 统称 </s> 183
惯例 为 维 和 50
惯例 为 自己 的 43
惯例 为 艺术类 学院 44
惯例 为 避免 侵权 148
```

Redes RNN

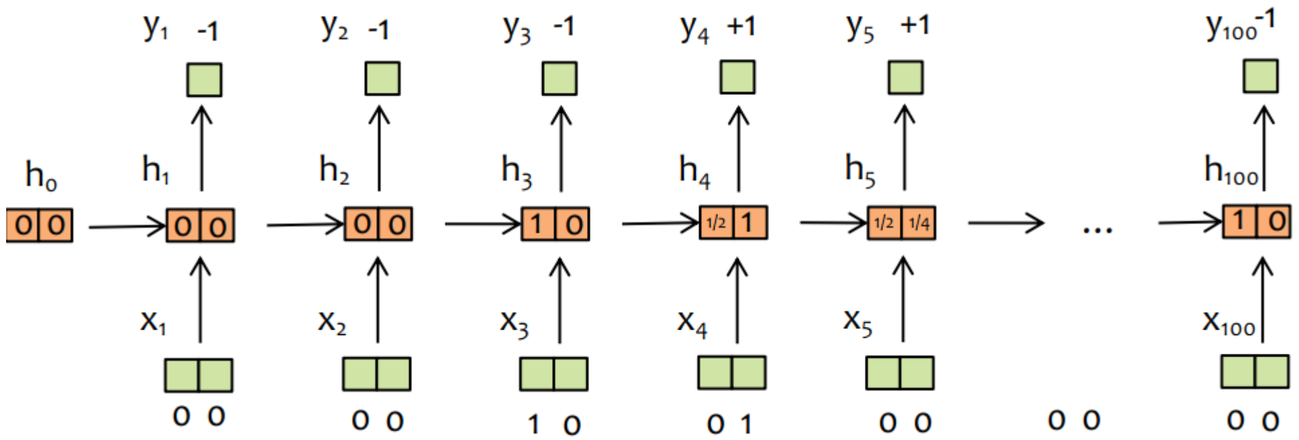


Idea Clave:

(1) convertir todas las palabras anteriores en un **vector de longitud fija**

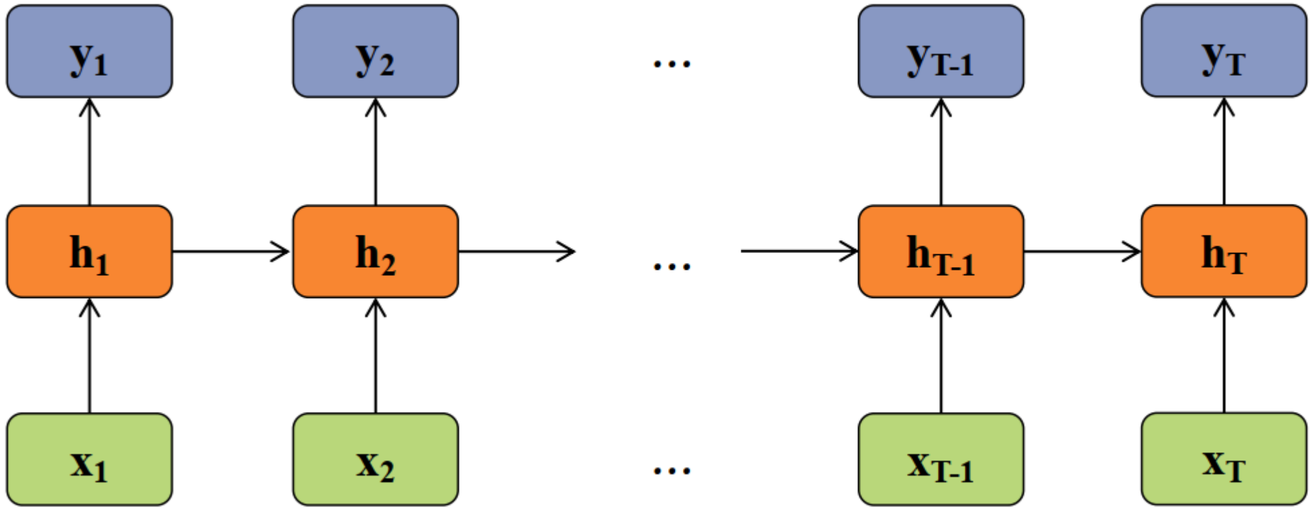
(2) definir la distribución $p(w_t|f_\theta(w_{t-1},\ldots,w_1))$ que condiciona sobre el vector $\mathbf{h}_t = f_\theta(w_{t-1},\ldots,w_1)$

Redes RNN y Olvido



Supongamos que queremos una RNN (Red Neuronal Recurrente) sobre vectores binarios de longitud 2 que pueda recordar si ha visto, o no, un valor de 1 en ambas posiciones de entrada.

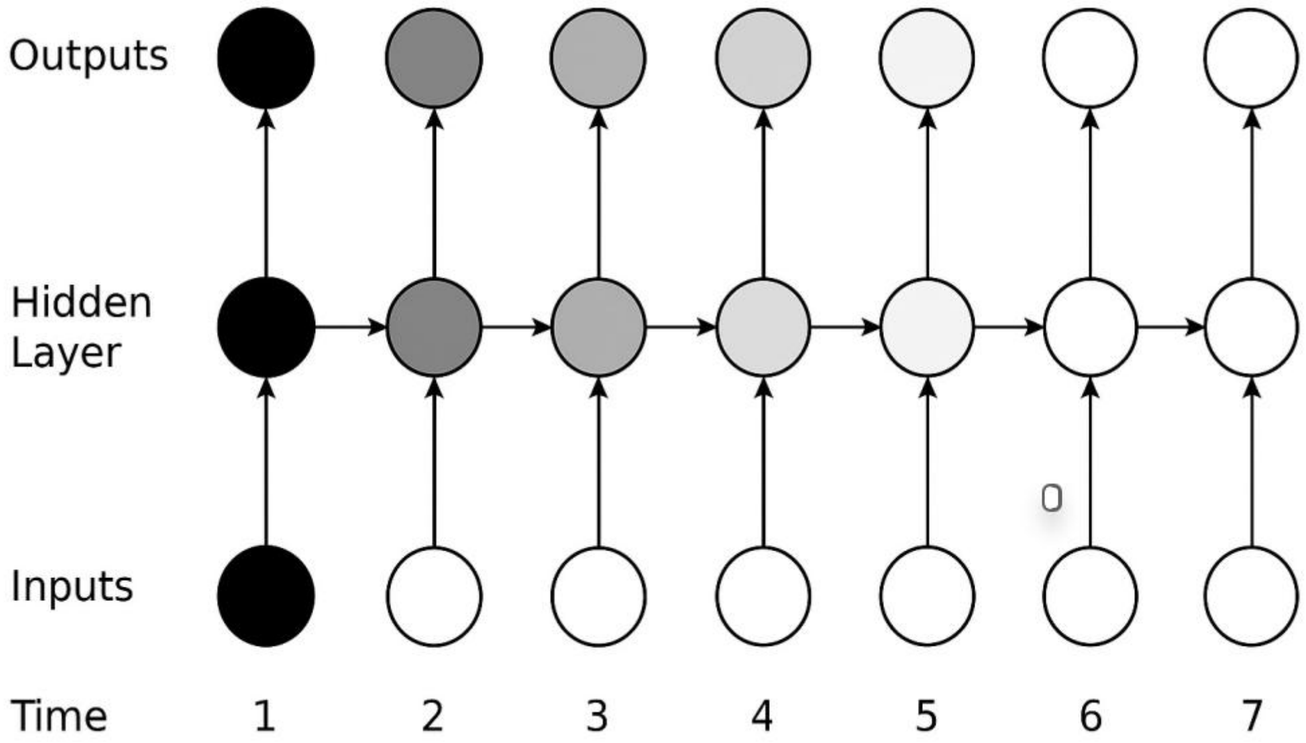
Long Short-Term Memory (LSTM)



Motivación:

- Las redes estándares RNNs tienen problemas para recordar dependencias de larga distancia
- Redes LSTM combaten ese problema

Long Short-Term Memory (LSTM)

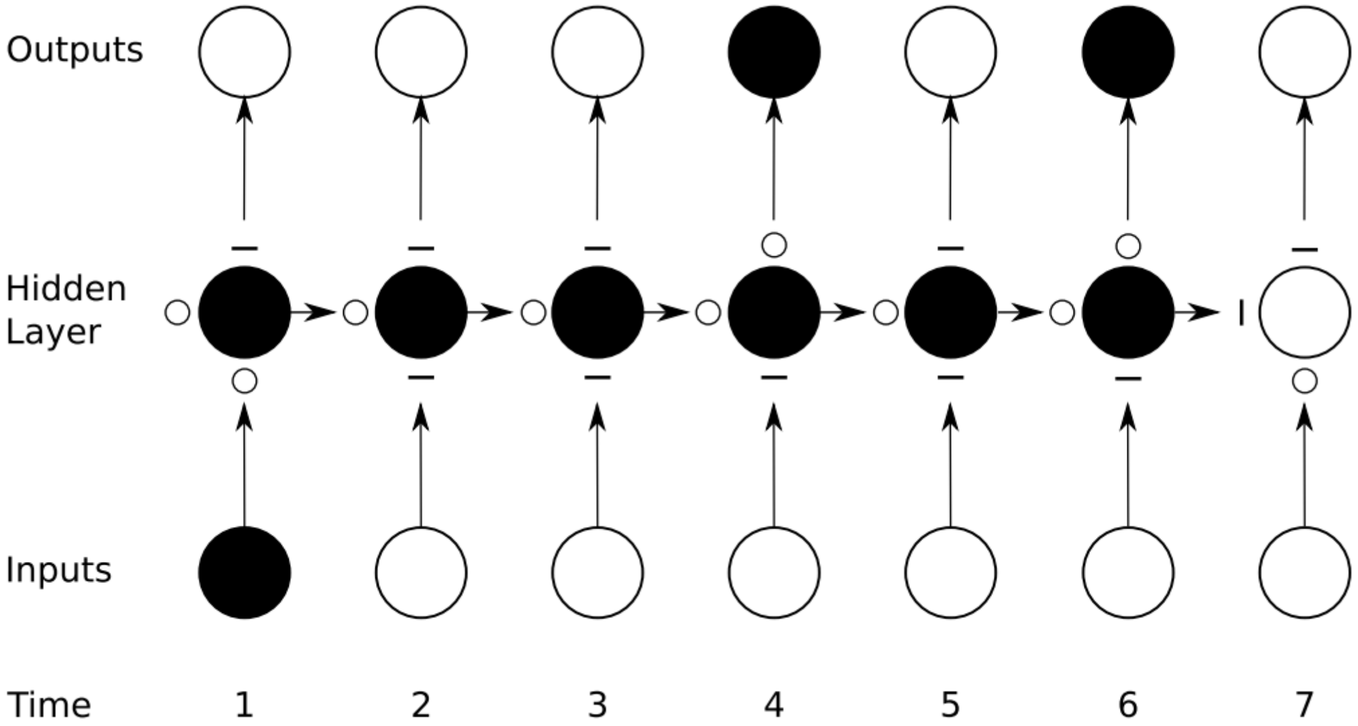


Motivación:

- Problemas del gradiente descendiente de las redes RNN estándar
- La figura muestra la sensibilidad (oscuro = más sensible) a la entrada de tiempo=t1

Figura de (Grave, 2012)

Long Short-Term Memory (LSTM)

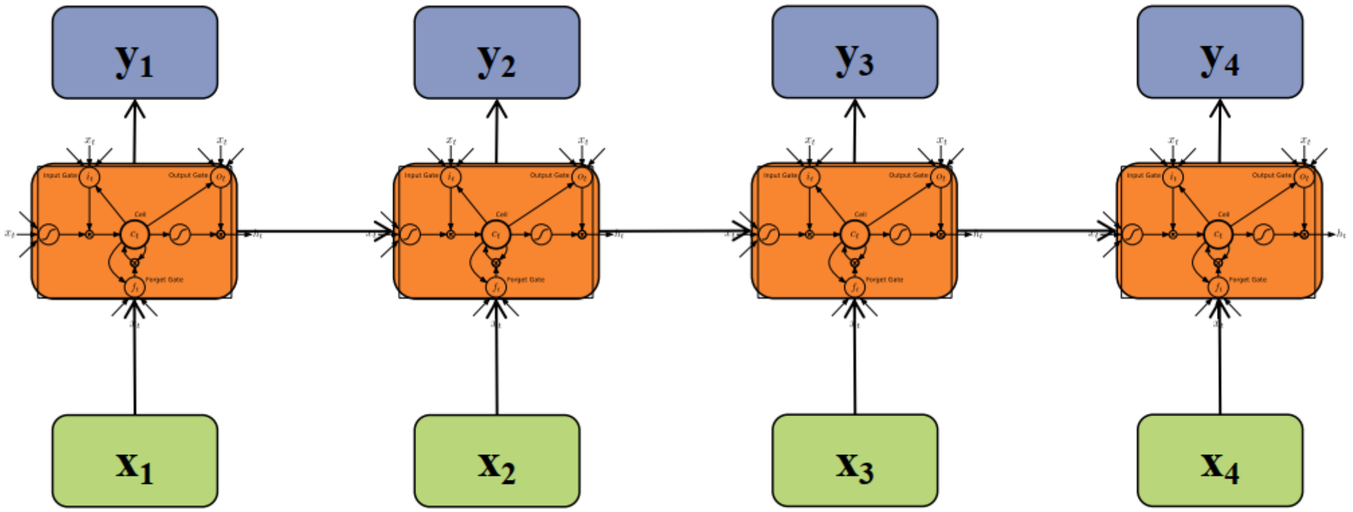


Motivación:

- Redes LSTM tienen una rica estructura interna
- Varias puertas determinan la propagación de la información y eligen si deben recordar u olvidar

Figura de (Grave, 2012)

Long Short-Term Memory (LSTM)



¿Por qué no usar redes LSTM para todo?

Se hizo por un tiempo. Pero…..

1. A pesar de todo tienen dificultad para las dependencias de largo alcance
2. Sus cálculos son inherentemente seriales, lo que complica las paralelización en GPU
3. A pesar que resuelve (mayormente) el problema del gradiente descendiente, aun pueden sufrir de Explosión

Conceptos Básicos

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

Conceptos Fundamentales: Tokens

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: Embedding

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

Conceptos Fundamentales: Embedding

Matemática de los Embeddings

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

$$Rey - Hombre + Mujer \approx Reina$$

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

Transformers Language Model

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA

Arquitectura Transformer

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

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.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

Arquitectura Transformer

Componentes Principales

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

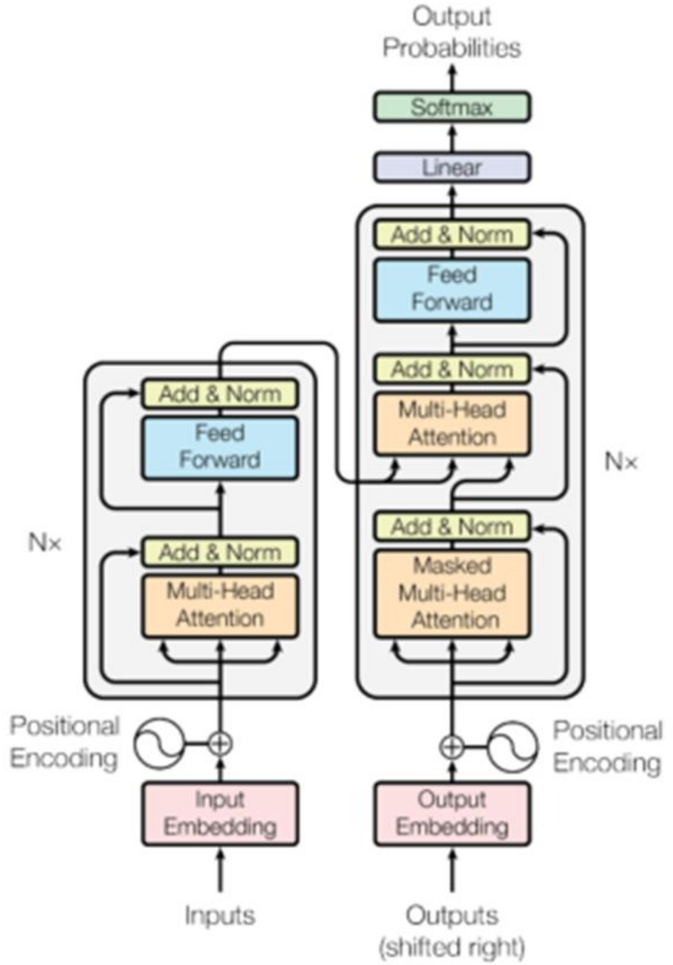
Procesa las representaciones generadas por el mecanismo de atención para cada posición.

Normalización y Conexiones Residuales

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

Arquitectura Transformer

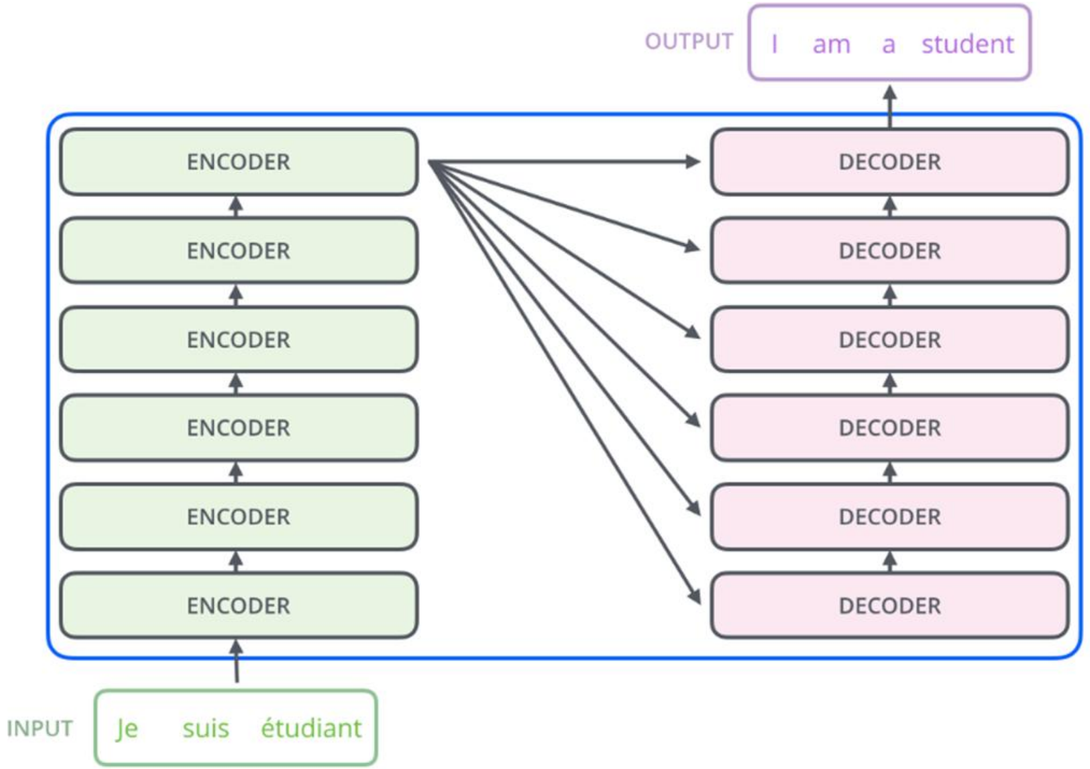
- Arquitectura Codificador Decodificador
- Incluye:
  - Positional Encoding
  - Multi-head attention



Arquitectura Transformer

https://jalammar.github.io/illustrated-transformer/

Arquitectura Transformer



https://jalammar.github.io/illustrated-transformer/

Modelo de Atención

Query (Q)

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

Key (K)

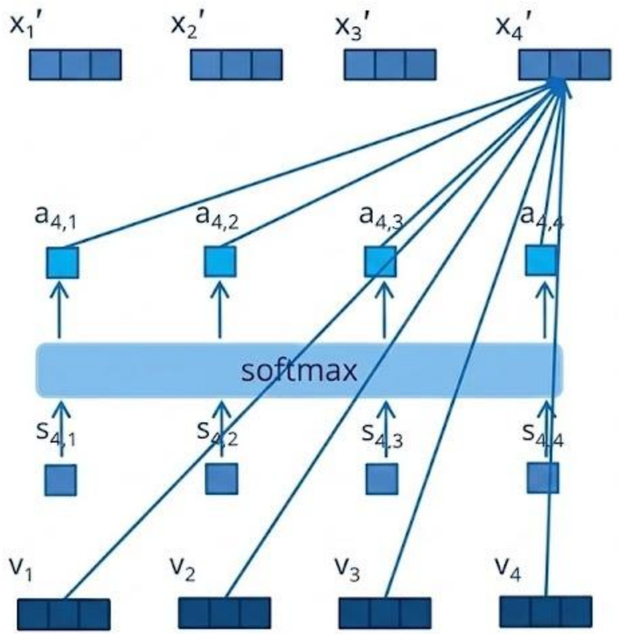
"¿Qué tengo para ofrecer?" – Cada palabra ofrece información que puede ser relevante para otras palabras.

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.

Modelo de Atención



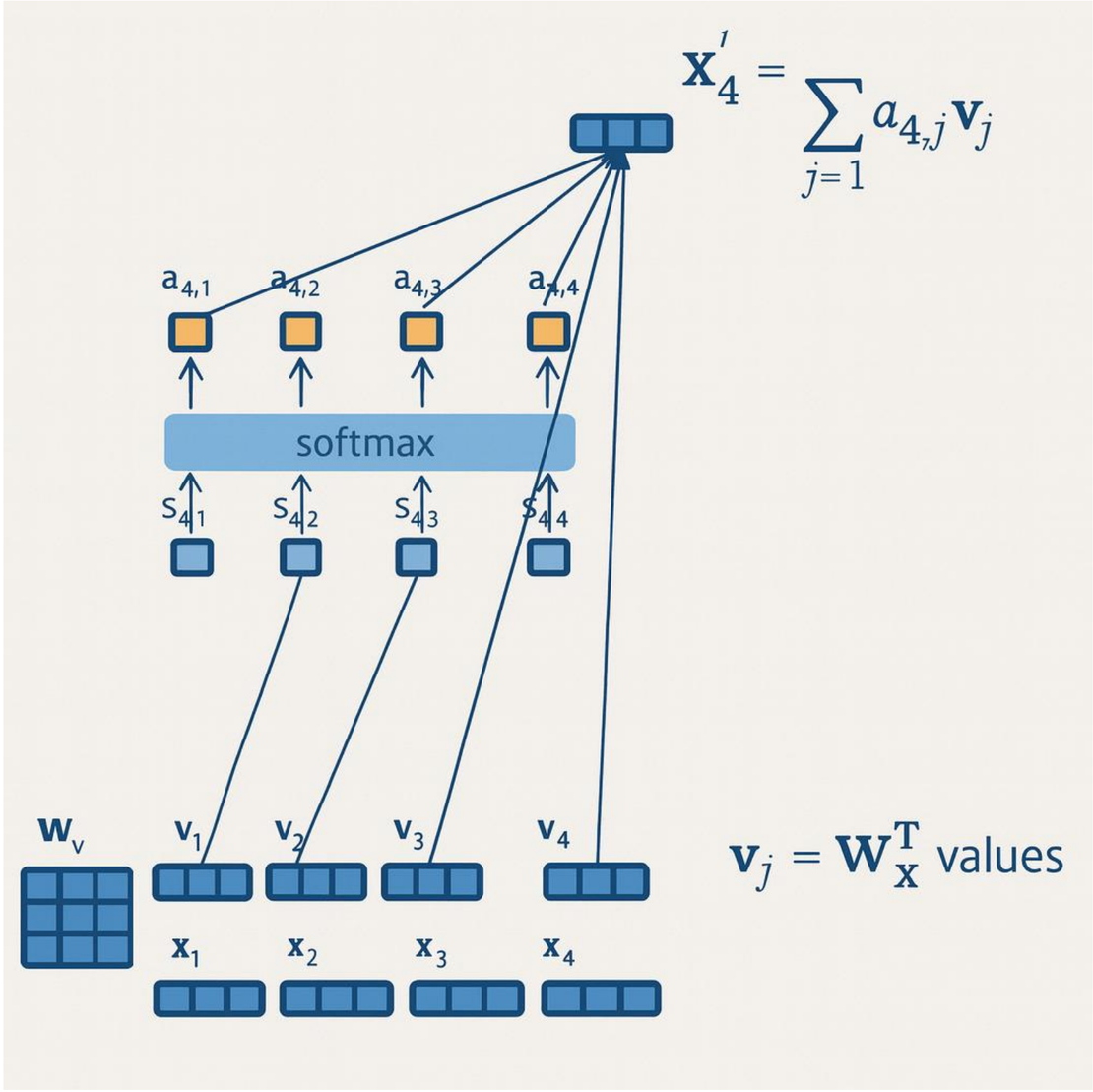
$$\mathbf{x}'_t = \sum_{j=1}^{t} a_{t,j}\mathbf{v}_j$$

attention weights

scores

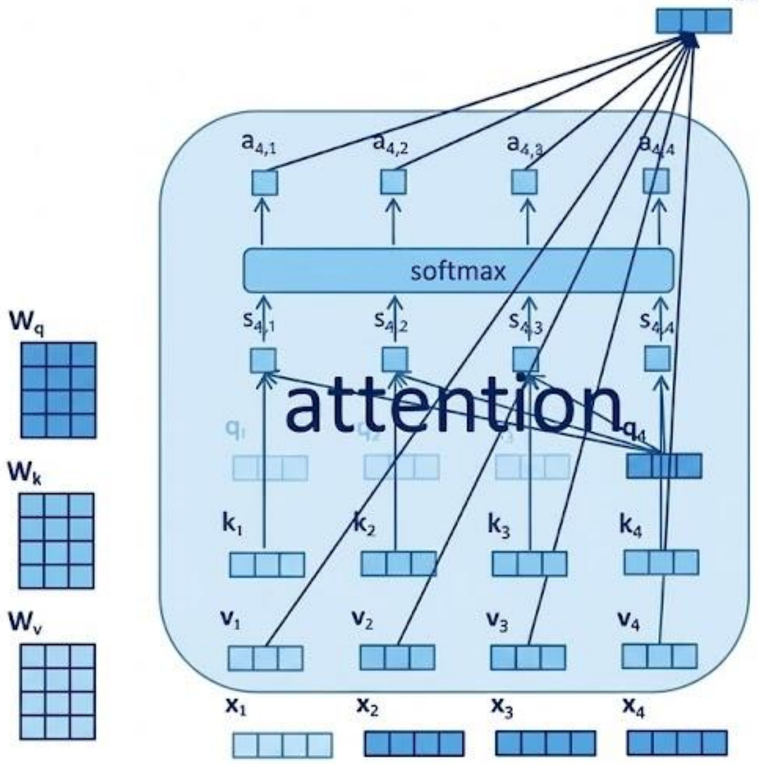
values

Modelo de Atención



Producto Cruz escalado del modelo de Atención

Modelo de Atención



$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

$\mathbf{a}_4 = \text{softmax}(\mathbf{s}_4)$ attention weights

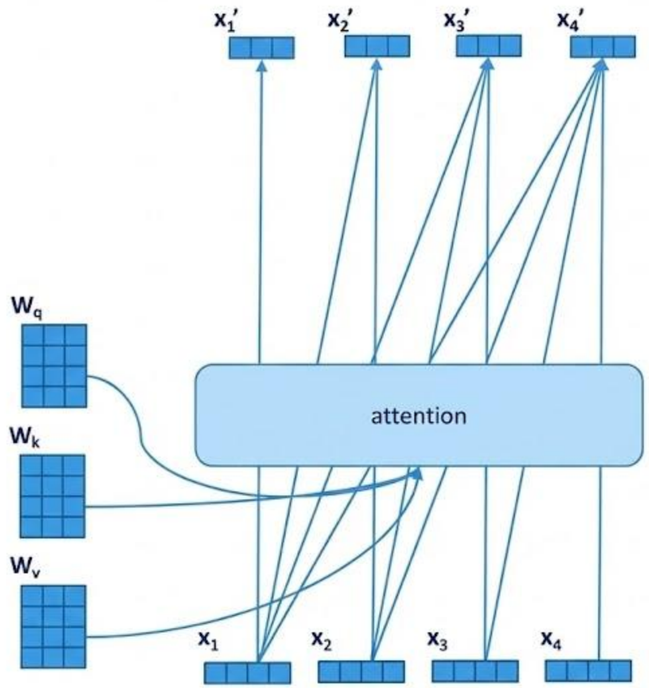
$s_{4,j} = \mathbf{k}_j^T \mathbf{q}_4 / \sqrt{d_k}$ scores

$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j$ queries

$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j$ keys

$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$ values

Modelo de Atención



$$\mathbf{x}'_t = \sum_{j=1}^{t} a_{t,j}\mathbf{v}_j$$

$\mathbf{a}_t = \mathrm{softmax}(\mathbf{s}_t)$ attention weights

$s_{t,j} = \mathbf{k}_j^T \mathbf{q}_t / \sqrt{d_k}$ scores

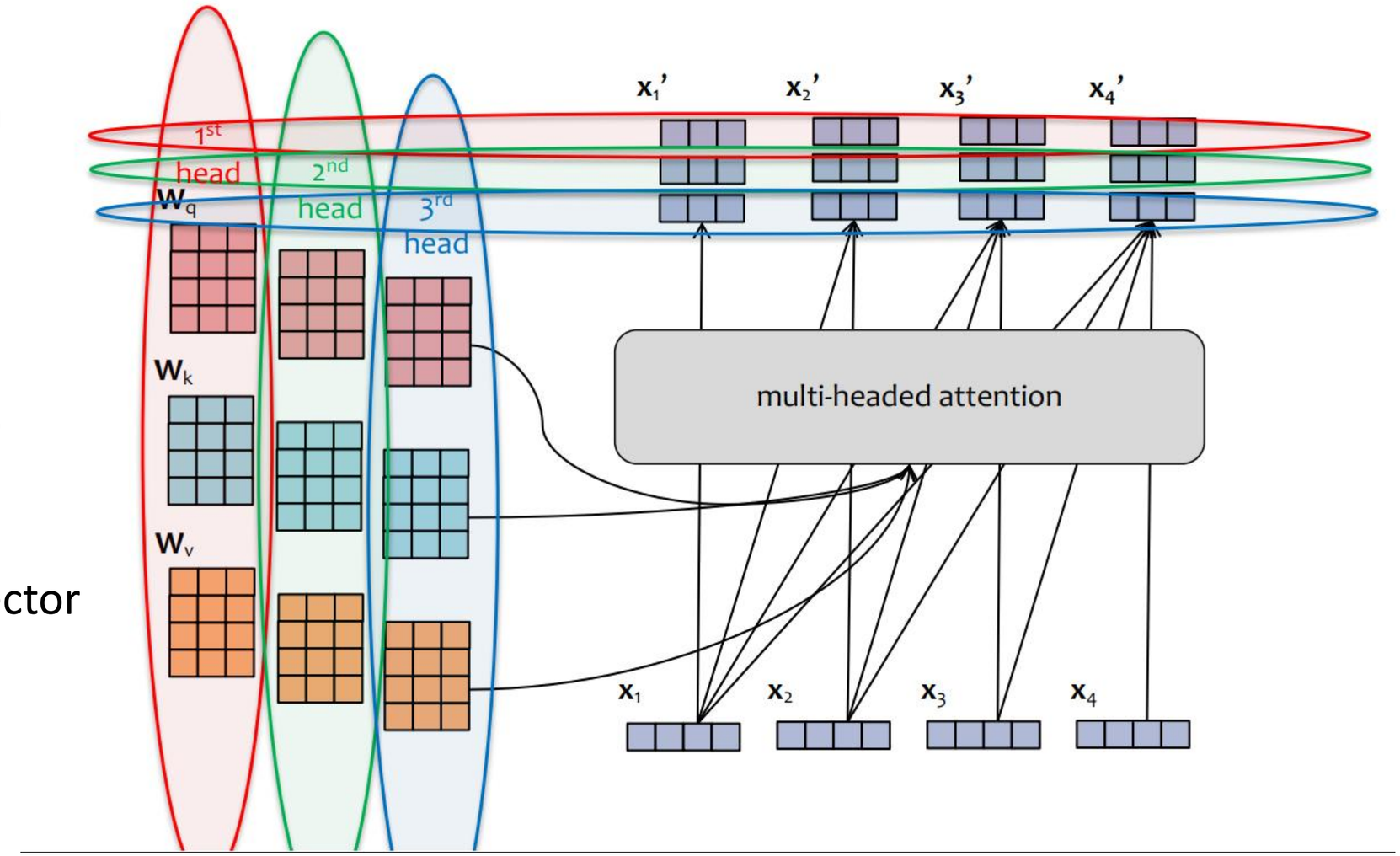
$\mathbf{q}_q = \mathbf{W}_q^T \mathbf{x}_j$ queries

$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j$ keys

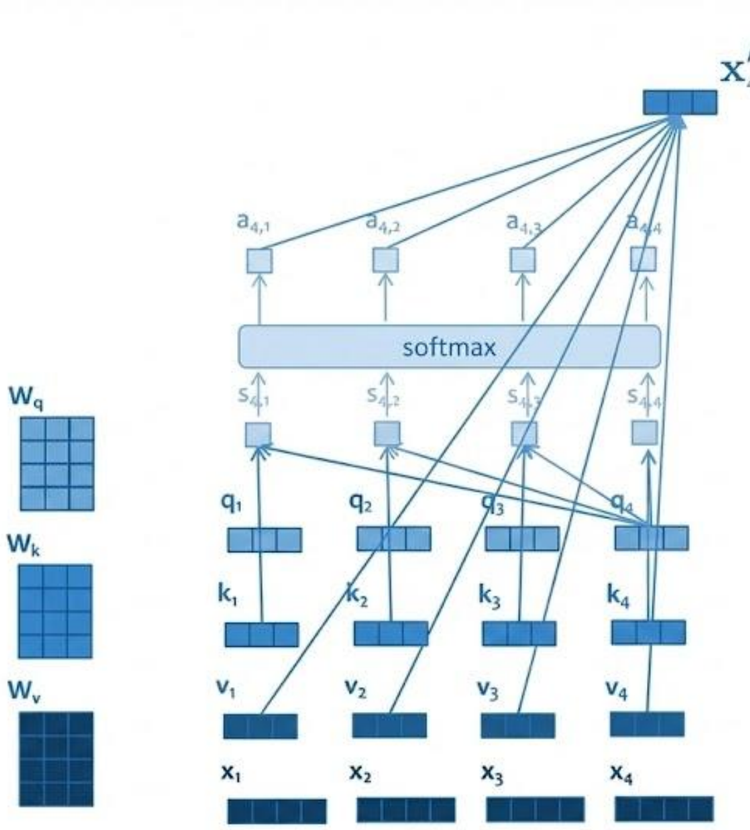
$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$ values

Multiatención

- Al igual que en la capas convolucionales, pueden haber múltiples canales. Así se pueden implementar multi cabeza de atención
- Cada cabeza tiene sus propios parámetros y analiza distintas características de las palabas, ej. Sinónimos, verbos, adjetivos
- Se pueden concatenar todas las salidas para obtener un único vector



Versión Matricial de modelo de Atención



$\mathbf{a}_4 = \text{softmax}(\mathbf{s}_4)$ attention weights

$s_{4,j} = \mathbf{k}_j^T \mathbf{q}_4 / \sqrt{d_k}$ scores

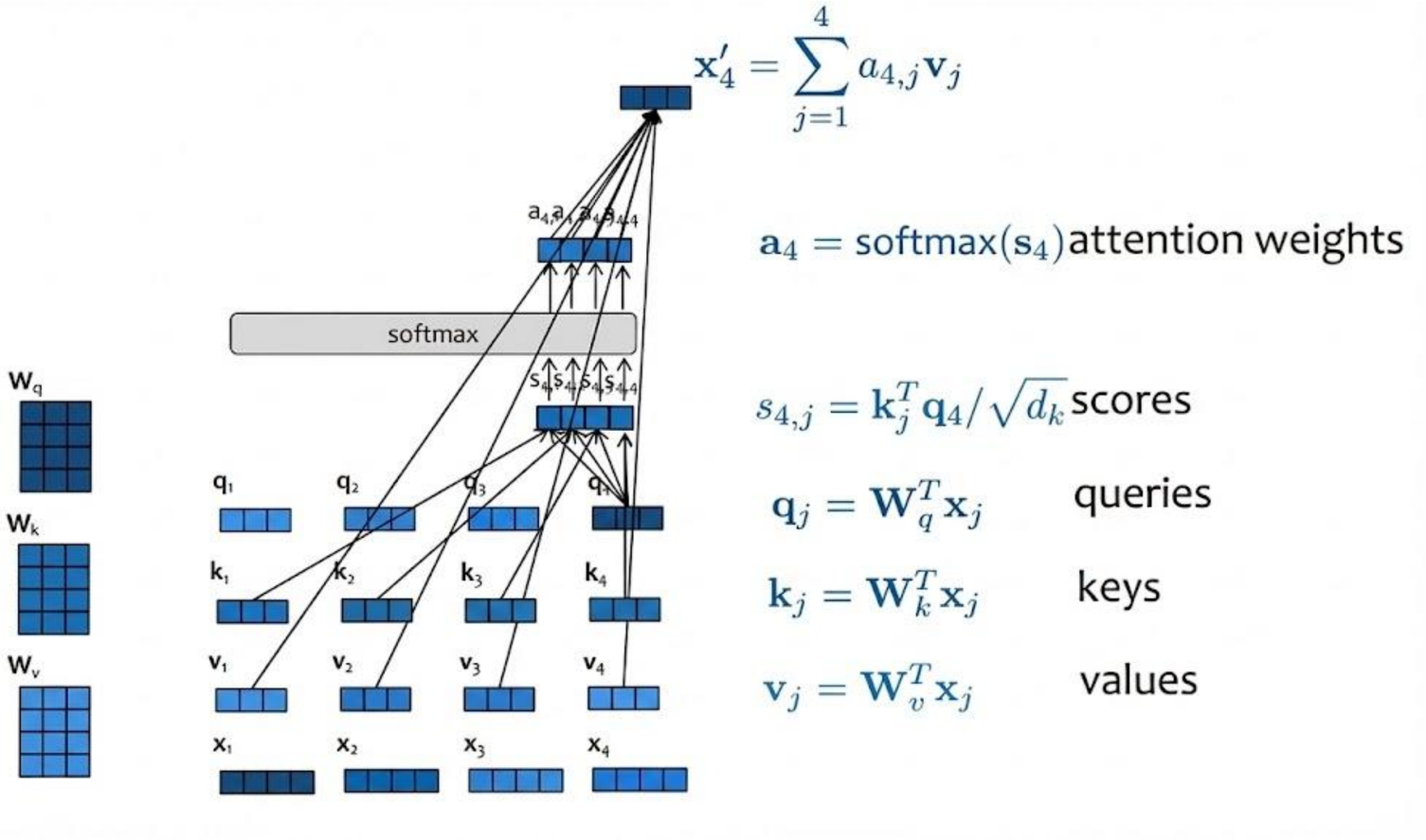
$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j$ queries

$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j$ keys

$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$ values

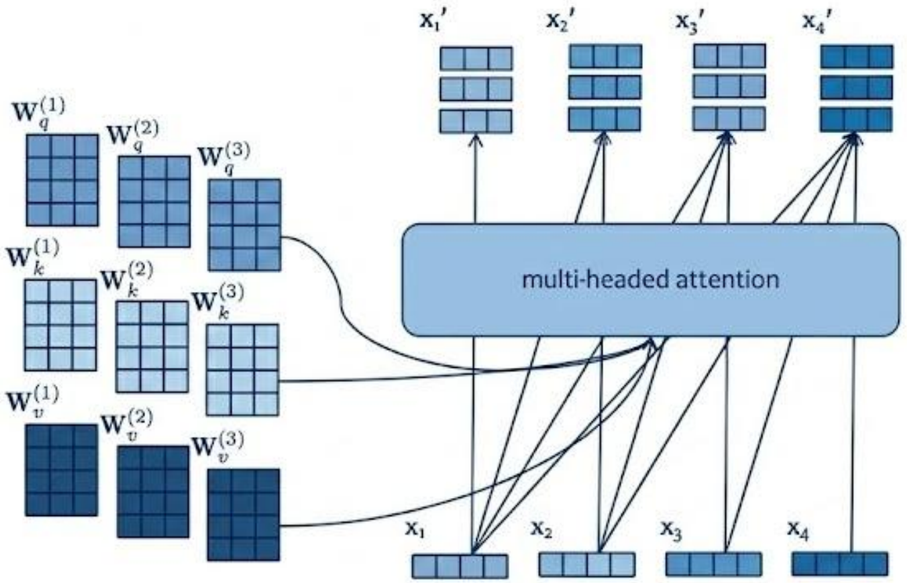
Versión Matricial de modelo de Atención

- Por velocidad, se calculan todos los queries a la misma vez usando operaciones matriciales
- Primero se empaquetan queries, keys y values como matrices
- Luego se opera todo a la vez



Multiatención

- Al igual que en la capas convolucionales, pueden haber múltiples canales. Así se pueden implementar multi cabeza de atención
- Cada cabeza tiene sus propios parámetros y analiza distintas características de las palabas, ej. Sinónimos, verbos, adjetivos
- Se pueden concatenar todas las salidas para obtener un único vector



$$\mathbf{X} = \text{concat}(\mathbf{X}'^{(1)}, \ldots, \mathbf{X}'^{(h)})$$

$$\mathbf{X}'^{(i)} = \text{softmax}\left(\frac{\mathbf{Q}^{(i)}(\mathbf{K}^{(i)})^T}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}^{(i)}$$

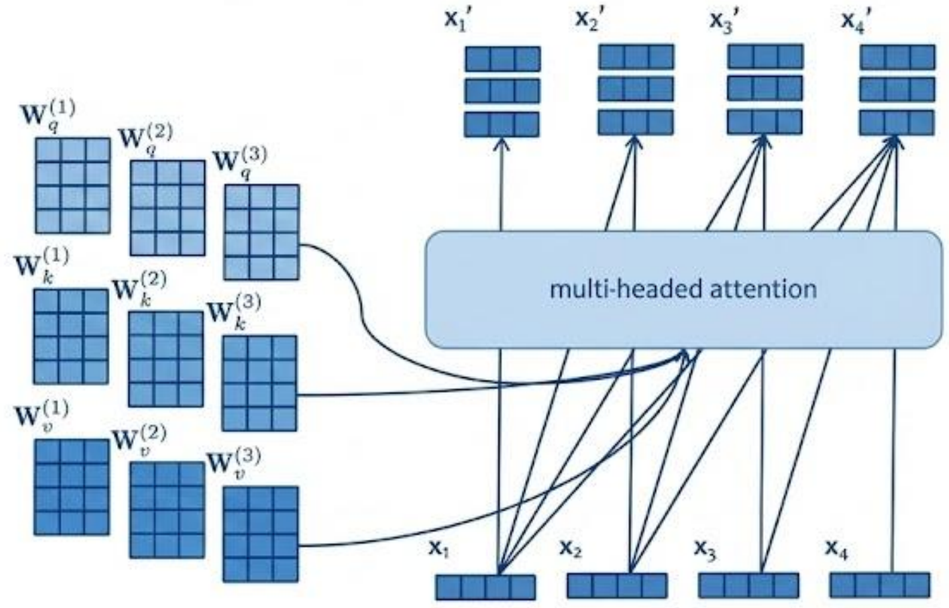
$$\mathbf{Q}^{(i)} = \mathbf{X}\mathbf{W}_q^{(i)}$$

$$\mathbf{K}^{(i)} = \mathbf{X}\mathbf{W}_k^{(i)}$$

$$\mathbf{V}^{(i)} = \mathbf{X}\mathbf{W}_v^{(i)}$$

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_4]^T$$

Versión matricial Multiatención



$$\mathbf{X} = \text{concat}(\mathbf{X}'^{(1)}, \ldots, \mathbf{X}'^{(h)})$$

$$\mathbf{X}'^{(i)} = \text{softmax}\left(\frac{\mathbf{Q}^{(i)}(\mathbf{K}^{(i)})^T}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}^{(i)}$$

$$\mathbf{Q}^{(i)} = \mathbf{X}\mathbf{W}_q^{(i)}$$

$$\mathbf{K}^{(i)} = \mathbf{X}\mathbf{W}_k^{(i)}$$

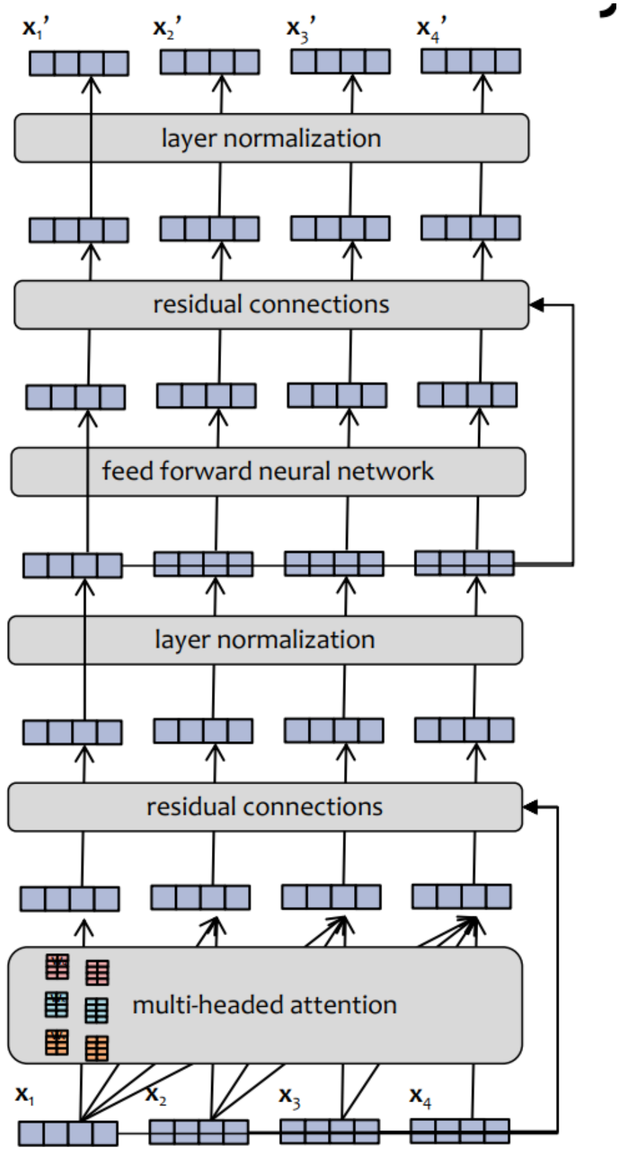
$$\mathbf{V}^{(i)} = \mathbf{X}\mathbf{W}_v^{(i)}$$

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_4]^T$$

Capa Transformer

Cada capa de un modelo de lenguaje Transformers, incluye:

- Multi atención
- Red Feed-forward
- Capa de normalización
- Conexiones residuales

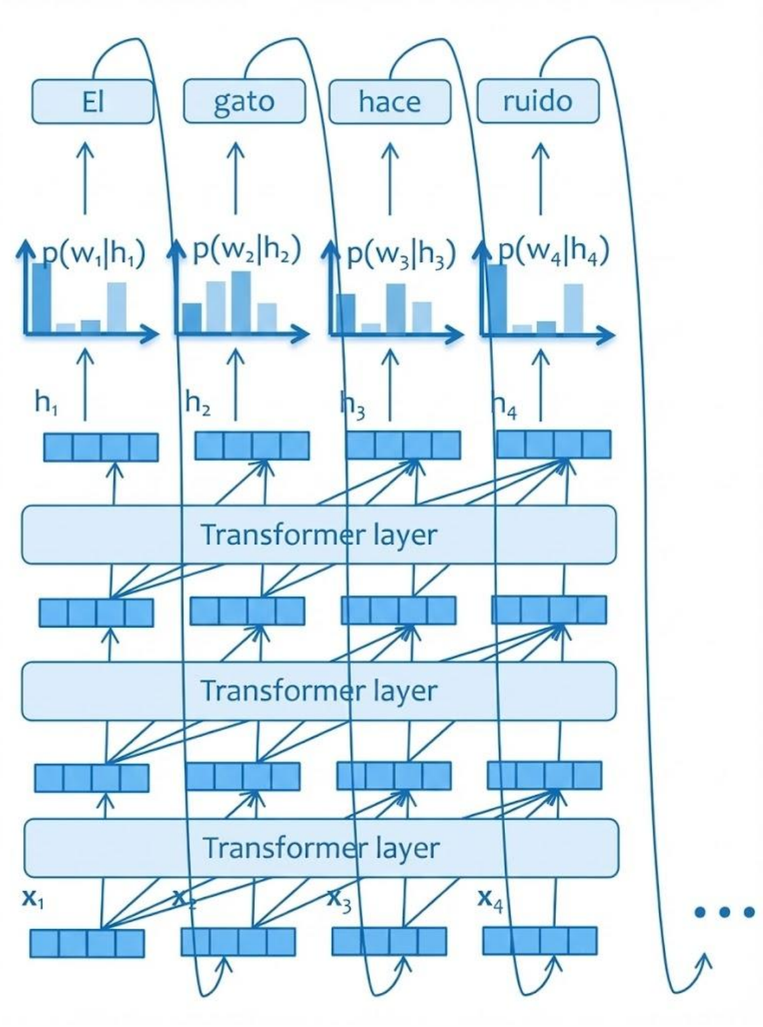


Modelo de Lenguaje Transformers

Cada capa de un modelo de lenguaje Transformers, incluye:

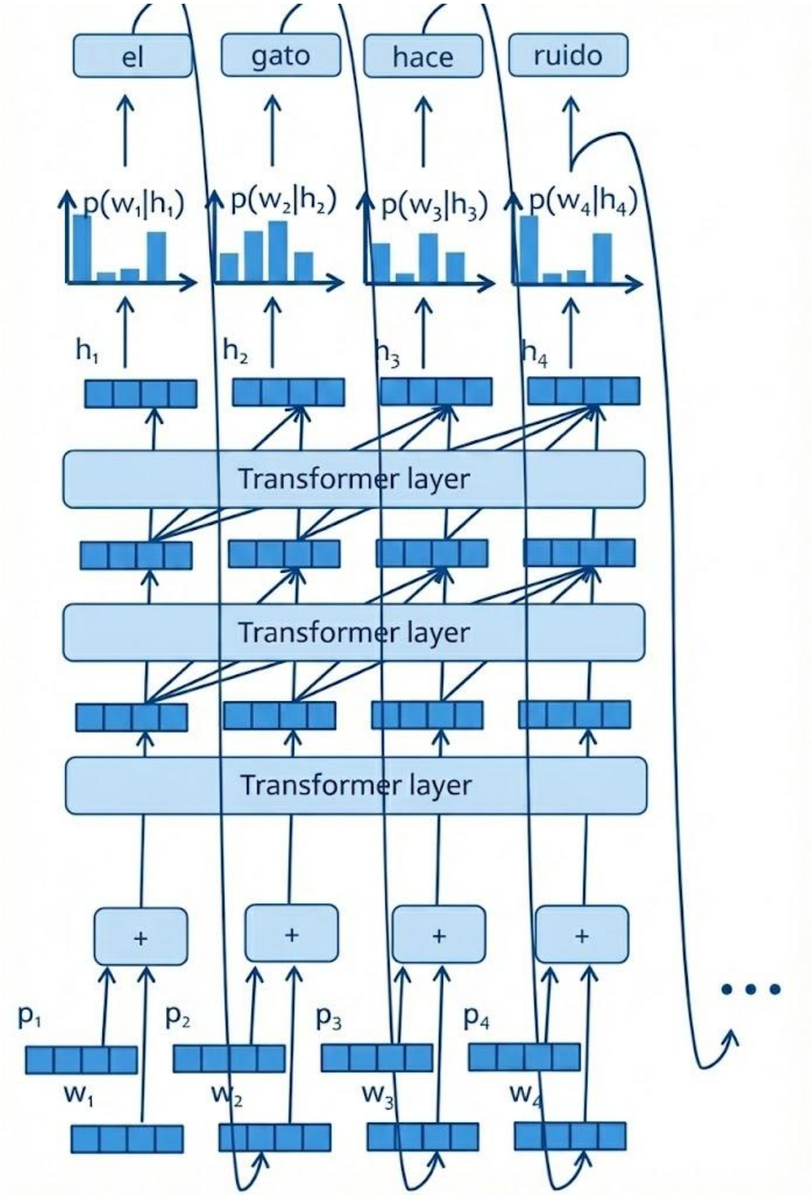
- Multi atención
- Red Feed-forward
- Capa de normalización
- Conexiones residuales

Cada vector oculto revisa los vectores ocultos de la capa actual y de la previa.



Positional Encoding

- **Problema**: Porque el modelo de atención es invariante a la posición, se requiere aprender acerca de las posiciones
- **Solución**: Se utilizar un conjunto de embedding posicionales: **pt** que almacena la posición del embedding **wt**.
  La ideal clave es que cada palabra que aparece en la posición **t** usa el mismo embedding posicional **pt**.
- Existen distintos tipos de embeddings posicionales:
  - Absolutos (como lo descrito anterior), pero también puede usar embeddings posicionales relativos (ej. Relativos a la posición del vector del consulta)
  - Algunos son fijos (basados en seno o coseno), como otros (Word embeddings)



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)

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

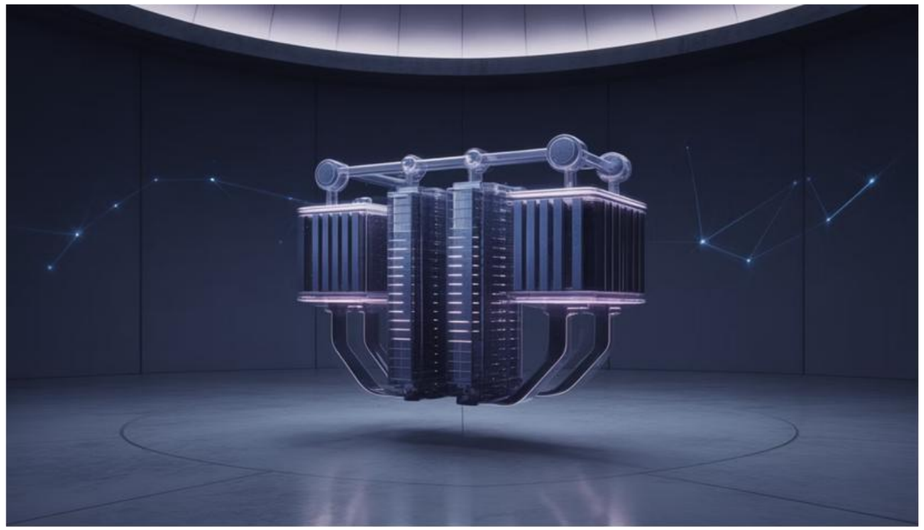
Los modelos grandes requieren GPUs potentes y mucha memoria, limitando su accesibilidad.

"Alucinaciones"

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

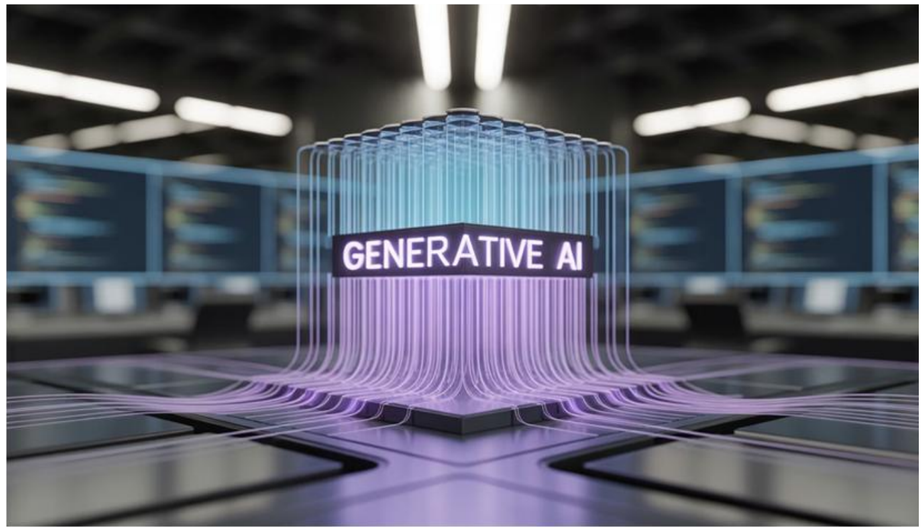
Tipos de Arquitectura

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 autoregressivamente
- Ideal para generación de texto
- Ejemplos: GPT, LLaMa, Claude

¿Qué es BERT?

¿Qué es BERT?

Bidirectional Encoder

Representación a partir de modelo Transformers, desarrollado por Google en 2018

Contexto Bidireccional

Modelo preentrenado que comprende el contexto completo desde ambas direcciones simultáneamente

Revolución NLP

Superó radicalmente los modelos unidireccionales y secuenciales anteriores en múltiples tareas

Arquitectura de BERT

Fundamento Transformer

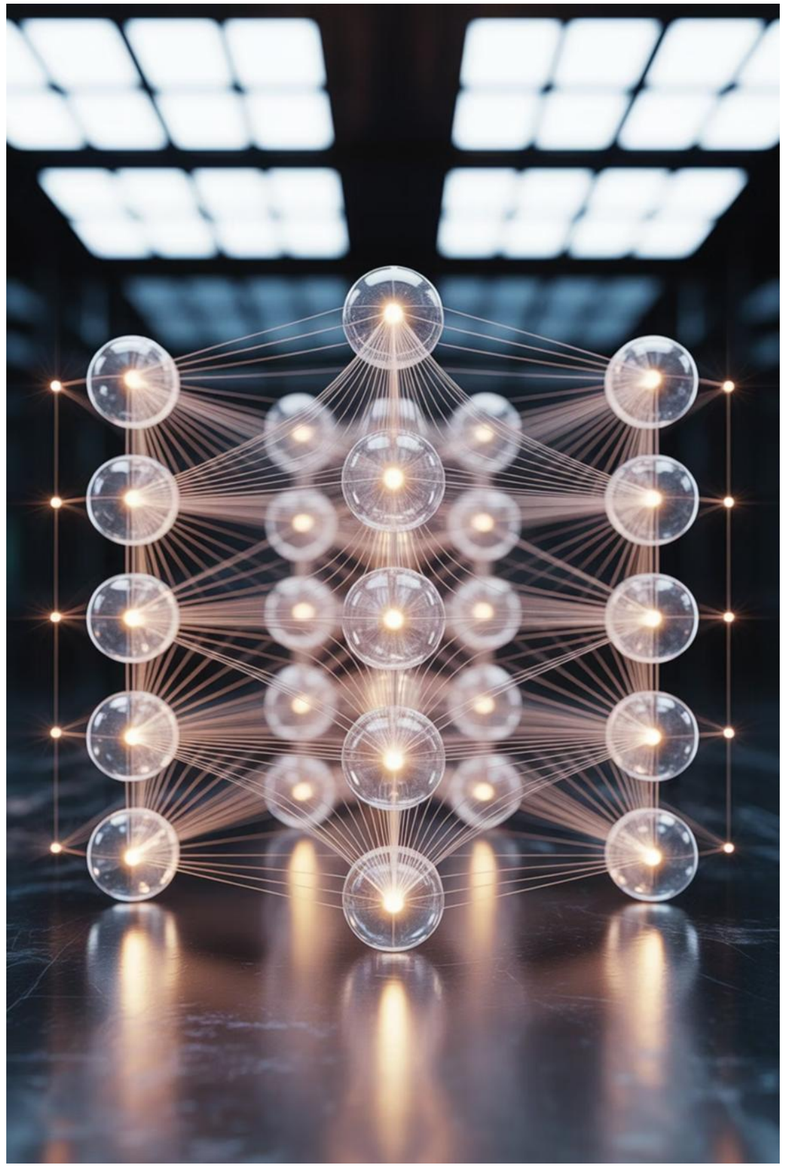
BERT se basa en la arquitectura Transformer original propuesta por Vaswani et al. en 2017, utilizando exclusivamente la parte del codificador (encoder).

- Sin componente decodificador
- Múltiples capas de atención
- Procesamiento paralelo eficiente

Atención Bidireccional

El mecanismo de atención bidireccional permite que BERT analice el contexto completo de una oración simultáneamente, capturando relaciones complejas entre palabras.

- Contexto izquierdo y derecho
- Comprensión profunda del significado
- Representaciones contextuales ricas



Preentrenamiento de BERT

Masked Language Model (MLM)

BERT oculta aleatoriamente el 15% de las palabras y aprende a predecirlas usando el contexto bidireccional completo

Next Sentence Prediction (NSP)

Predice si dos oraciones son consecutivas en el texto original, capturando relaciones entre oraciones

Corpus Masivo

Entrenado con aproximadamente 3.3 mil millones de palabras de Wikipedia en inglés y BookCorpus

Impacto de BERT

Mejoras Significativas

BERT logró resultados estado del arte en 11 tareas de NLP, incluyendo clasificación de texto, respuesta a preguntas, reconocimiento de entidades nombradas y análisis de sentimientos.

Variantes Inspiradas

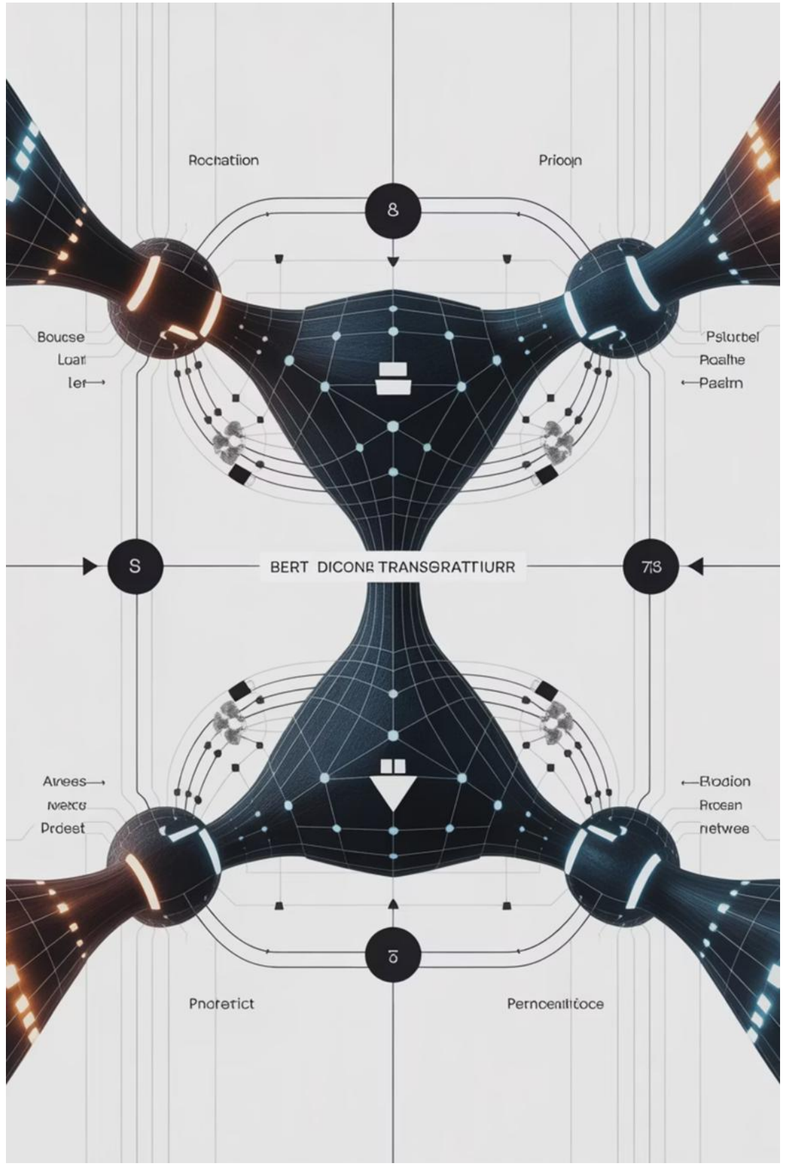
RoBERTa (optimización robusta), ALBERT (versión ligera), DistilBERT (destilación eficiente), y muchas otras adaptaciones especializadas surgieron rápidamente.

Camino a Modelos Mayores

Estableció el paradigma de preentrenamiento-ajuste fino que permitió el desarrollo de modelos cada vez más grandes y capaces.

Arquitectura de BERT

El flujo bidireccional de información permite que cada palabra "vea" todas las demás palabras en la oración simultáneamente, capturando dependencias complejas a largo plazo.



¿Qué son los LLMs?

¿Qué son los LLMs?

Large Language Models

Modelos masivos con miles de millones (o billones) de parámetros entrenables que aprenden patrones lingüísticos complejos.

Corpus Enormes

Entrenados en cantidades masivas de texto de internet, libros, artículos científicos y otras fuentes diversas.

Capacidades Duales

Combinan tareas de comprensión (análisis, clasificación) y generación (escritura, diálogo, creatividad).

Ejemplos destacados: BERT se especializa en comprensión bidireccional, mientras GPT destaca en generación unidireccional de texto coherente y creativo.

GPT: Generative Pre-trained Transforme

Historia y Evolución

- **GPT-1 (2018):** 117M parámetros, primera versión
- **GPT-2 (2019):** 1.5B parámetros, generación impresionante
- **GPT-3 (2020):** 175B parámetros, cambio de paradigma
- **GPT-4 (2023):** Capacidades multimodales avanzadas

Características Clave

Arquitectura basada en el decodificador Transformer unidireccional, optimizada para generar texto coherente y contextualmente relevante palabra por palabra.

BERT vs GPT: Diferencias Fundamentales

BERT: Comprensión

Bidireccional: Analiza contexto completo simultáneamente

Aplicaciones: Clasificación, extracción de información, Q&A

Encoder-only: Optimizado para entender significado

GPT: Generación

Unidireccional: Procesa texto de izquierda a derecha secuencialmente

Aplicaciones: Generación de texto, diálogo, creatividad

Decoder-only: Optimizado para producir texto coherente

Ambos enfoques son complementarios y se utilizan según las necesidades específicas de cada aplicación de NLP.

Evolución y Variantes Destacadas

1

RoBERTa (2019)

Optimización robusta de BERT con más datos de entrenamiento, eliminación de NSP y ajustes en hiperparámetros

2

ALBERT (2019)

Versión ligera y eficiente con factorización de parámetros y compartición de capas, reduciendo tamaño sin perder capacidad

3

T5 (2020)

Text-to-Text Transfer Transformer que unifica todas las tareas NLP bajo un formato común de entrada-salida

4

PaLM, LLaMA, Claude (2022-2024)

Nueva generación de LLM con mejoras significativas en eficiencia, razonamiento y capacidades multimodales

Línea de Tiempo: Evolución de Transform

01

2017: Transformer Original

Vaswani et al. - "Attention is All You Need"

02

2018: BERT y GPT-1

Revolución bidireccional y generativa

03

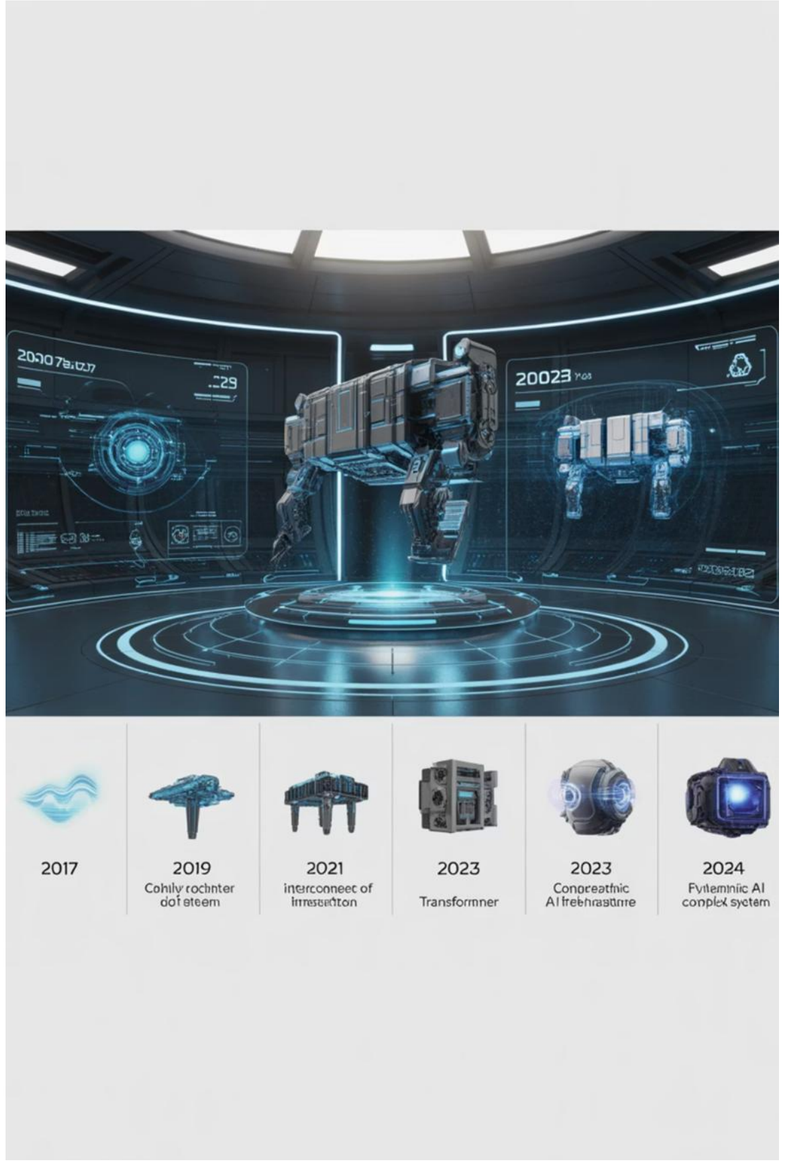
2019-2020: Optimizaciones

RoBERTa, ALBERT, T5, GPT-3

04

2021-2024: Era Multimodal

GPT-4, PaLM, Claude, LLaMA, Gemini

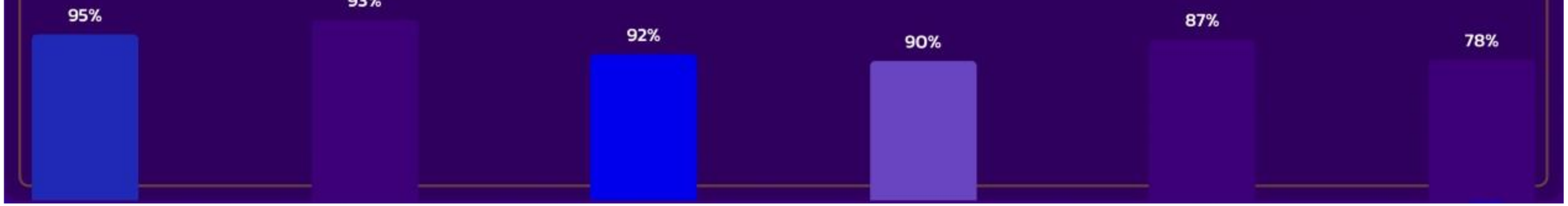


Avances en las Arquitecturas

- 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.

Comparativas de LLMs

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
Gemini	No revelado	32K – 1M tokens	Multimodal Transformer	Multimodal avanzado, razonamiento



Comparativas de LLMs

Model	Creators	Year of release	Training Data (# tokens)	Model Size (# parameters)
GPT-2	OpenAI	2019	~10 billion (40Gb)	1.5 billion
GPT-3	OpenAI	2020	300 billion	175 billion
PaLM	Google	2022	780 billion	540 billion
Chinchilla	DeepMind	2022	1.4 trillion	70 billion
LaMDA (cf. Bard)	Google	2022	1.56 trillion	137 billion
LLaMA	Meta	2023	1.4 trillion	65 billion
LLaMA-2	Meta	2023	2 trillion	70 billion
GPT-4	OpenAI	2023	?	? (1.76 trillion)
Gemini (Ultra)	Google	2023	?	? (1.5 trillion)
LLaMA-3	Meta	2024	15 trillion	405 billion